2011 年度　卒業論文

オープンソース・ソフトウェアの経済分析

慶應義塾大学　経済学部
石橋孝次研究会　第 12 期生

古屋　海斗

はしがき

現実の一見不可解な現象を、経済学のフレームワークを用いて分析する。それが、私が卒業論文の執筆を開始するにあたり、一番の指標としたものであった。理由は単純で、それが面白いと思ったからだ。大学四年間の経済学の勉強の中で、私の印象に一番残っているのが、ペプシコーラ社及びコカコーラ社に対する Monsanto 社の独占供給契約の話だ。Monsanto 社の人工甘味料であるアスパルテームの特許は 1992 年に失効した。通常なら両飲料メーカーは、特許切れによる新規企業の参入により、人工甘味料をより安価に調達できるようになることを期待するはずである。しかしながら、両社は特許切れの直前にも関わらず、この独占供給契約に応じた。一体なぜだろうか？こうした、一見すると不可解なエピソードを、経済学のフレームワークを用いて論理的に分析し、解き明かしていく。それが、私が四年間のうちに掴み取った経済学の醍醐味であり、卒業論文においても、これを最大の主題にしようと考えた。

そうして辿り着いたのが、オープンソース・ソフトウェアであった。題材がオープンソース・ソフトウェアに決まったあと、私は研究目的半分、興味半分で、とあるオープンソース・ソフトウェアのカンファレンスに出席した。そこで目にした光景に、私は大きな衝撃を受けた。ステージには、映画館と見紛うほどの巨大なスクリーンがかけられ、そこには高度なプログラミング技術に関するスライドが投影されている。スライドの両端には縦長の別枠が設けられ、そこを会場の参加者のコメントがものすごい勢いで流れていく。そして、大勢の参加者を前に登壇し、高度なプログラミング技術についてプレゼンテーションをしているのは、十代の若い開発者であった。

オープンソース・ソフトウェアは、コンピュータの歴史の中ではまだ若い存在だが、日に日にその存在感を増している。今後ますますオープンソース・ソフトウェアが注目を浴びていくだろう一方で、不思議に思われる性質も未だ多く残っている。そうした性質を、経済学のフレームワークを用いて分析していきたいと思う。

目次

序章

90 年代後半の登場以来、オープンソース・ソフトウェア（以下OSS）は日に日に存在感を高め、今やソフトウェア産業の中で無視できないものとなった。身近な範囲でも、インターネット・ブラウザのFirefoxやChromium、モバイルオペレーションシステムのAndroidが、競合する企業のクローズド・ソフトウェアとしのぎを削っている。また、LAMPと呼ばれるOSSの組み合わせは、動的なウェブコンテンツの開発環境の定番となり、多くのウェブサイトがOSSの恩恵を受けて運営されている。[1]

現実経済で強い存在感を放つと共に、OSS は経済学においても無視することができない、いくつもの特性を備えている。例えば、Lerner and Tirole (2005) ではオープンソース・ライセンスと呼ばれる OSS 独自の知的所有権の考え方を、理論・実証の両面から分析している。また、コミュニティと呼ばれる集団が金銭報酬ゼロの中でソフトウェアの質を高め、時には企業のつくったソフトウェアのシェアを奪っていく、という OSS の発展過程は、契約理論の問題を孕んでいる。このように OSS は固有の特性を持っているが、その理論的かつ実践的な研究は、未だ多くは為されていないのが現状だ。オープンソースという開発形態が、今後更に普及していく中で、こうした固有の特性を経済学の枠組みで捉え直し、理論・実証の両面から分析する必要がある。

本稿の目的は、OSS 固有の二つの特性を取り上げ、経済学の枠組みで捉え直し、理論・実証の両面から分析を行うことである。1 つ目のトピックはオープンソース・ライセンスであり、2 つ目は、オープンソース・コミュニティである。

本章に続く本稿の構成は次の通りである。第 1 章では、OSS の概要や歴史に触れ、OSS 固有の特性を洗い出す。第 2 章では、固有の特性の中から、オープンソース・ライセンスを取り上げる。このトピックは前述の通り Lerner and Tirole (2005) が取り上げているが、当時に比べて、近年では企業が OSS に介入することがより盛んとなっている。そうした変化を受け、商業的な圧力と折り合いをつけながら、OSS の知的所有権をいかに取り扱うべきかを、理論・実証の両面から分析する。第 3 章では、オープンソースコミュニティの動機の分析を行う。このトピックは未だ理論的な分析が行われていないから、まず契約理論のフレームワークを応用して理論を導入し、その上で、独自の手法による定量的な分析を行う。そして第 4 章において、本稿の簡単なまとめを行う。

---

[1] LAMP は Linux, Apache, MySQL, Perl(PHP,Python)という OSS の頭文字をとっている。

第 1 章　オープンソース・ソフトウェアの概要

オープンソース・ソフトウェアは、様々な経済学上の特性を孕んでいる。それは、オープンソース・ライセンスという知的財産に対する独自の考え方であるし、オープンソース・コミュニティという存在そのものが提起する、契約理論にまつわる問題でもある。OSS がこうした経済学上の特性を獲得したのは、主にその定義と成り立ちに原因があると言える。本章では、OSS の定義・誕生の背景を踏まえた上で、上述した OSS の経済学上の性質を洗い出す。

1.1　オープンソース・ソフトウェアの定義

そもそもオープンソース・ソフトウェアとは、一体何を指す言葉なのか。そこに、共通の明確な定義は存在しないが、一般にはオープンソース・イニシアティブが提唱する「The Open Source Definition」というオープンソースの決まりごとが普及している。オープンソース・ソフトウェアというと、ソースコードが開示されているソフトウェアという印象を抱きがちだが、OSS はソースコードの開示のみに規定されるものではない。The Open Source Definition には、導入の但し書きとして、ソースコードの開示を含めて OSS は頒布に際し、10 の基準を満たさなければならないとされている。表 1-1 に挙げたものがその 10 項目であり、そのどれもが OSS を規定する条項だが、中でも OSS の一般的なイメージを形作っているのは 1，2 の自由な再配布やソースコードの開示といったものであろう。しかし、一見見慣れないライセンスという存在も、OSS を語る上でははずせない。とりわけ、ライセンスに横たわる『コピーレフト』という独特な知的財産権の考え方は、OSS を理解するために極めて重要なものである。そこで、以下では、『ソースコードの開示』と『コピーレフトとライセンス』、そしてもう一つの重要な要素である『オープンソース・コミュニティ』という三つの項目にわけ、OSS の特徴を解説する。

表 1-1 The Open Source Definition の 10 項目

1	自由な再配布
2	ソースコードの開示
3	派生著作物に関するライセンス条件
4	著作者のソースコードの完全性
5	個人やグループに対する差別の禁止
6	利用分野に対する差別の禁止
7	ライセンスの継承
8	特定製品に特化したライセンスの禁止
9	他のソフトウェアを制限するライセンスの禁止
10	テクノロジーニュートラルなライセンス

出所：秋本・岡田(2004)

1.1.1　ソースコードの開示

オープンソース・ソフトウェアという言葉を聞いて、一般的に想起されやすいのがこの性質であろう。The Open Source Definitionの第二項には、ソースコードの開示が明言されている。一般的にソフトウェアというものは、プログラマーが人間の理解できるプログラミング言語で設計・記述したものを、コンピュータの理解できる機械語に変換して（この操作をコンパイルと呼ぶ）運用する。この機械語に変換する前の、人間に理解できる状態のコードをソースコードと呼ぶ。端的に言えば、ソースコードとは、そのソフトウェアがどうつくられたのかを記録したレシピであり、どのように動作するかを記述した設計図にあたる。非オープンソースの商業的ソフトウェアでは、ソースコードは秘匿され、それを梃子にソフトウェアを販売し、収益を得るのが通常のビジネスモデルである。ソースコードを秘匿されてしまえば、複雑なソフトウェアの機能からソースコードを割り出すのは難しいし、そもそもソフトウェアの中には利用規約によりそうした操作を禁止するものもある。[2]

ソースコードが公開されていると、一体どのようなメリットがあるのだろうか。ソースコードを閲覧することにより、OSS のユーザーはソフトウェアの改変を行うことが可能となる。ソースコードを読めば、ソフトウェアがどのような仕組み・アルゴリズムで動いているのかがわかり、そこに自分の使いやすいように新たな機能を付け加えたり、アルゴリズムを改善して動作速度を改善したりすることができるからだ。そうして生まれた新機能は、時にウェブ等で他のユーザーと共有されていくから、OSS は多くのユーザーの手によって改変され、洗練されていく。一方で、ソースコードを秘匿している商業的ソフトウェアでは、基本的にユーザー自身の手で新機能の追加や高速化を行うことは出来ず、メーカーによる更新を待つほかない。たとえば Microsoft のインターネットブラウザである Internet Explorer は、基本的に Microsoft 自身が行うアップデートによってのみ新機能が追加されていく。一方で、OSS のインターネットブラウザである Mozilla の Firefox は、Mozilla によるアップデートだけでなく、ユーザー自身が自分の欲しい機能を産み出すことができ、そうした新機能の多くは、アドオンという形でウェブ上に公開され共有されている。

1.1.2　コピーレフトとライセンス

前述のように、OSSはレシピであるソースコードが公開されているため、ユーザー

2) リバースエンジニアリングの禁止など。

自身が自由に改変を行うことができるし、Open Source Definitionにある通り、自由に再配布することができる。[3]こうした性質を一見すると、著作権が放棄されたかのように思えるが、OSSの著作権は放棄されていない。[4]OSSやその前身となったフリーソフトウェア（後述）では、知的所有権の考え方で「コピーレフト」という価値観が普及している。コピーレフトとはフリーソフトウェア財団を設立したリチャード・ストールマンによって提唱された概念で「制作者の著作権を保持したまま、二次著作物を含め、すべてのユーザーがOSSを利用・再配布・改変できなければならない」という考え方である。このコピーレフトのユニークなところは、「二次著作物を含めて」この価値観を維持しなくてはならないところにある。たとえば、Aという名前のオープンソースのソフトウェアを改変し、A2 というソフトウェアを新たに作製したとする。すると、A2 にもAに採用されていたコピーレフトの考え方が保持され、A2 の制作者はA2 を自由に利用・再配布・改変できるようにソースコードを公開しなければならない。同様に、A2 を改変したソフトウェアA3 にもコピーレフトの考え方が保持され、以降、コピーレフトの価値観は親から子へ、子から孫へと引き継がれていく。この二次著作物に対する制約は、OSSを改変したり、他のソフトウェアと組み合わせて生み出した成果物のソースコードを秘匿し、そのソフトウェアを販売することによって収益をあげようとするといった、ただ乗りを防ぐものとして機能する。

こうしたコピーレフトの概念は、その厳密さに差こそあれど、オープンソース・ライセンスを通じて OSS の頒布手段の中心概念として採用されている。オープンソース・ライセンスとは OSS の利用・複製・改変・頒布等に対するポリシーを記述した一連の契約書である。この書類は、OSS の開発者自身が自ら作製することも可能だが、いくつかの団体が発表している出来合いのものを採用することが多い。出来合いのライセンスを採用することで、開発者にとって複雑な書類作製のコストを削減し、さらによく知られたライセンスを採用すれば、利用者の間でも煩雑な契約書を読む手間が省ける。多くの OSS のライセンスは、オープンソース・イニシアティブの「The Open Source Definition」に準拠して、利用や頒布についての規約を定める。次ページの表 1-2 は、オープンソース・イニシアティブが「The Open Source Definition」に準拠

---

3) The Open Source Definition の第二項、自由な再配布は次のように規定されている。*『「オープンソース」であるライセンス(以下「ライセンス」と略)は、出自の様々なプログラムを集めたソフトウェア頒布物(ディストリビューション)の 一部として、ソフトウェアを販売あるいは無料で頒布することを制限してはなりません。ライセンスは、このような販売に関して印税その他の報酬を要求してはなりません。』*

4) 著作権を放棄したソフトウェアの頒布手段はパブリックドメインと呼ばれる。

表 1-2　代表的なオープンソース・ライセンスと採用 OSS

ライセンス	作成者	代表的なソフトウェア
GNU General Public License(GPL)	フリーソフトウェア財団	Linux, gcc, GNU Emacs, Firefox, Ruby (v1.9.2 迄)
LGPL	フリーソフトウェア財団	Firefox, openoffice.org
BSD License	カリフォルニア大学 バークレー校	Ruby (v1.9.3 から) Free BSD
MIT (X11) License	マサチューセッツ工科大学	X Window System(X11)
Mozilla Public License	Mozilla Foundation	Firefox
Apache License	Apache ソフトウェア財団	Apache HTTP Server
Common Public License	IBM	(Eclipse) [5]

していると認定した著名なライセンスと、そのライセンスを採用している代表的ソフトウェアをまとめたものである。ライセンスにはここに上げた以外にも様々な種類のものが存在し、細かな違いまで網羅すれば様々なバリエーションがあるが、最も注視される差異はコピーレフトをどれほど厳密に採用しているか、である。具体的に言えば、前述した二次著作物に対しての制約の程度が、ライセンスにより異なっている。たとえばGNU General Public License (GPL) は極めて厳密にコピーレフトの考えを採用しているライセンスで、OSSを改変したら必ずその二次著作物（先ほどの例におけるA2）にも同様のGPLライセンスを継承しなくてはならない。これはGPLを採用したOSSの二次著作物A2 も、必然的にソースコードを公開しなければならないことを意味する。表中のLGPLやCommon Public LicenseはGPLをベースにつくられた、GPLよりわずかにゆるいライセンスである。他方、BSD Licenseは制限のゆるいとされるライセンスで、いくつかの規定さえクリアすれば、二次著作物のソースコードを非公開にすることができる。[6]そのため、他のソフトウェアに組み込んだり、商業化のしやすいライセンスであるといえよう。また、表中のMIT (X11) LicenseやApache LicenseはこのBSDを元にしてつくられたライセンスとされている。

また、一つのソフトウェアに対して複数のライセンスを採用することが可能で、こ

5) 今は Common Public License を元にした Eclipse Public License を採用している。
6) 規定とは、著作権表示、ライセンス条文、無保証の旨の三点をドキュメント等に記載することである。

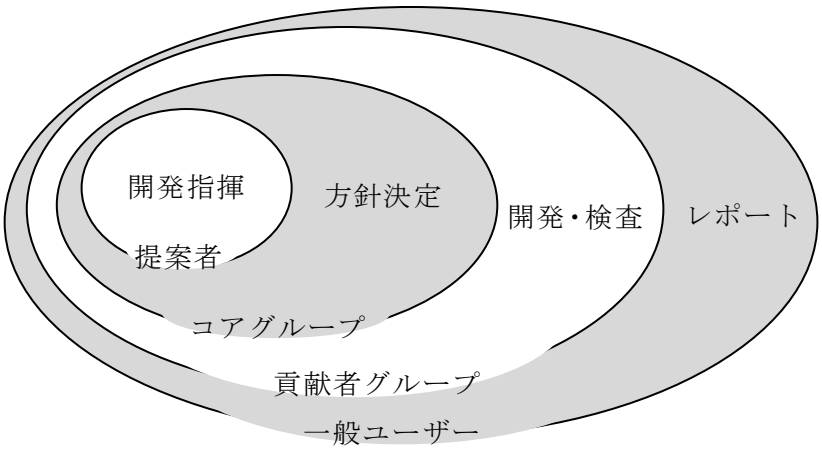
れをデュアルライセンスやトリプルライセンスという。たとえば Ruby(v1.9.3 以降)は BSD ライセンスと Ruby ライセンスのデュアルライセンスであり、Firefox は表にもある通り、GPL, LGPL, Mozilla Public License のトリプルライセンスである。ユーザーはこれらのライセンスのうちから、自らの使用目的に最も合致したライセンスを選択することができる。

1.1.3　オープンソース・コミュニティ

OSS をの概要を説明する上で、オープンソース・コミュニティは極めて重要なものである。オープンソース・コミュニティは一言で言ってしまえば OSS 利用者の集まりであり、交流の場である。ただし、オープンソース・コミュニティはただの利用者の集まりにはとどまらない。オープンソース・コミュニティは OSS を利用するだけでなく、OSS の開発をも行うのだ。

オープンソース・コミュニティをオープンソース開発プロジェクト（以下プロジェクト）のための組織としてみるとき、所属するメンバーは、貢献の度合いや決定権の度合いによって主に三つの層にわけることができる。コミュニティの最も中心に位置し、オープンソースの方針を決定し、新バージョンの仕様を策定するメンバーをコアグループと呼ぶ。コアグループはプロジェクトに参加し、OSS に多大な貢献をした人の中から選ばれ、プロジェクトに対する決定権を持つ。コアグループの外側に位置するメンバーを貢献者グループと呼び、プロジェクトの開発や OSS の検査を行う。そして､貢献者グループのさらに外側には一般ユーザーが存在し､バグレポートを行う。こうしたグループが互いに連絡を取り合いながら、自身の能力を提供し、ネットワーク型組織を形成しているのがオープンソース開発の特徴といえよう(図 1-1)。参加は

図 1-1　オープンソース・コミュニティの組織構造



出所：秋本・岡田（2004）

義務として行われるのではなく、任意の貢献という形で行われる。

オープンソース・コミュニティのもう一つの特徴は、参加者のほとんどがボランティアであるということだろう。企業の業務で派遣されている等といった一部のメンバーを除いて、彼らはコミュニティへの参加自体からは収入を得ることはない。そのため、ほとんどのメンバーが日々の業務の合間を縫って OSS 開発に勤しんでいる。

1.2 オープンソース・ソフトウェア誕生の背景

独特の性質をもつ OSS だが、その成り立ちはどのようなものであったのだろうか。OSS の誕生の背景を語るうえで欠かすことができないのが、フリーソフトウェアの存在である。コンピュータがごく一握りの人々にしか使われていなかった 1970 年代までは、ソフトウェアの権利は至極曖昧なものであった。当時は、ソフトウェアは共有し、相互利用されるのが一般的で、特に、研究者の間では相互利用が盛んに行われていた。しかし 1980 年代に入ると、ソフトウェアを専有化し、そこから収益を得ようという流れが誕生する。そうした商業ソフトウェアの登場により、ソフトウェアの所有権は明確なものとなり、ソースコードも秘匿され、自由な配布や改変を行うことは不可能なものとなった。そうした状況で誕生したのが、先に挙げたフリーソフトウェアというものだ。フリーソフトウェアは、ソフトウェアの専有化の潮流とは逆に、コンピュータ黎明期のような相互利用を促すためにつくられた概念である。フリーソフトウェアは、ソースコードを公開し、自由に配布や改変を行うことができる。さらに、フリーソフトウェアを改変して生まれたソフトウェアも、フリーソフトウェアでなければならない、というコピーレフトの性質をもっている。このフリーソフトウェアの概念をつくったのが、フリーソフトウェア財団の代表であるリチャード・ストールマンである。ストールマンは The Open Source Definition を提唱しているオープンソース・イニシアティブの創設者でもある。フリーソフトウェアは派生ソフトウェアも必ずフリーソフトウェアとしなければならないという、厳密なコピーレフトの性質を持っていたから、当時の商業ソフトウェアのビジネスモデルとの間に大きな軋轢を生んでいた。そこでストールマンは、フリーソフトウェアの定義をゆるめ、権限の移譲の方式に多様性を認めた。そうして新たにうまれた概念がオープンソース・ソフトウェアである。オープンソース・ソフトウェアは、1.1 節のとおり、著作者の考え方に合ったライセンスを選ぶことで、コピーレフトをどこまで厳密に採用するかを決定することができる。このように、OSS は、商業ソフトウェアとフリーソフトウェアとの折り合いをつけるために生まれたのだ。

1.3　ソフトウェア産業におけるオープンソース

これまでは OSS の定義と歴史を洗い、その概要に触れてきた。では、その実態はどのようなものであるのだろうか。1.2 節においても触れたように、OSS はそもそも個人が作成し、公開していたフリーソフトウェアを起源としているから、一般的に企業との関係性は薄かった。しかしながら、OSS の発展が進むにつれ、OSS を自らのビジネスに取り入れようとする企業が登場する。その先駆けとなったのが Red Hat である。Red Hat は Linux ディストリビューションである Red Hat Linux の提供を行なった。Linux ディストリビューションとは、オープンソースで開発されている OS である Linux を、一般的な利用者が導入・利用しやすいように様々なソフトウェアを組み合わせ、パッケージングしたもののことを言う。Linux ディストリビューションには様々なものがあるが、Red Hat Linux は、ネットワークサーバやワークステーションといった、主に企業を対象とした構成になっている。Red Hat Linux の最も特徴的なところは、そのビジネスモデルにある。構成するソフトウェアのほとんどが OSS である Red Hat Linux は、基本的に無料での頒布が行われた。その一方で、サポートを有料にし、そこから収益を生み出していた。90 年代の後半では、こうした、OSS の関連商品・サービスを販売して収益を得るというビジネスモデルが多く存在した。

その一方で、2000 年代中頃に入ると、オープンソースの開発形態を自社製品に取り込むことで、イノベーションコストを削減することを目的とする企業が登場した。その最も顕著な例が Google である。Google は検索事業からの広告収入を主な収益源としているが、その主幹事業を間接的に支えるために、インターネットブラウザである Google Chrome やモバイルオペレーションシステムである Android といったソフトウェアの開発・提供を行っている。ブラウザや OS といった汎用的なソフトウェアは、多様なユーザーニーズに逐一応えていると非常にコストがかさんでしまう。そこで Google は、この 2 つのソフトウェアの一部をオープンソースとすることで、ユーザー自身が追加機能の開発や、カスタマイズを行えるようにし、多様なユーザーニーズに対応した。

1.4　オープンソース・ソフトウェアの経済学上の問題

ここまで触れてきた OSS の概要を元に、OSS の経済学上の問題を考えると、次の二つが挙げられる。一つは、オープンソース・ライセンスにまつわる知的所有権の問題であり、より具体的には、最適なライセンス選択の問題である。どのようにライセンスを選択すれば、OSS の発展にとって最適であろうか。また、OSS が普及するに

つれ､企業の介入が増えてきたのは 1.3 節で述べたとおりだが､そうした環境の中で、コピーレフトをどこまで厳密に採用すべきか。OSS 製作者にとって、ライセンスの選択は極めて重要な課題である。もう一つは、オープンソース・コミュニティの動機にまつわる問題である。無給のボランティアであるコミュニティが、一体どのようなモチベーションに従って OSS の開発に従事しているのか。これは契約理論上の課題であると同時に、コミュニティは OSS のエンジンと言える存在であるから、その動機の構造は、OSS の様々な問題を分析する下地として必要となるだろう。

第 2 章 最適なオープンソース・ライセンスの選択

前章末において、オープンソース・ソフトウェアには二つの経済学上の特性、そして問題があることを述べた。一つは知的所有権にまつわるライセンシングの問題、もう一つは契約理論にまつわるオープンソース・コミュニティの問題である。この章では、そのうちライセンスについての分析を行う。

2.1 最適ライセンシングの二つの理論：理論分析

OSS の製作者にとって、ひとつの大きな課題はライセンスをどのように選ぶかというライセンス問題である。ここで注意しておきたいのは、OSS のライセンス問題は、一般的な経済学におけるライセンス問題とは異なり、タイミングや排他性、課金構造などの基本的課題は考慮するものとならないことである。OSS ライセンスは主に二次著作物に対する制約の度合いが異なるため(1.1.2 項参照)、ライセンサー（ここでは OSS 製作者や OSS 開発企業）は二次著作物に対する制約の度合いを考慮しライセンスを選択する。Lerner and Tirole (2005) はコミュニティの存在に重きを置き、Wang (2008) は企業の存在に重きを置いて、ライセンサーの選択問題をモデル化している。今節では、この二つのモデルを紹介する。

2.1.1 コミュニティモデル

Lerner and Tirole (2005) は、コミュニティの存在に重きをおいて、ライセンスの選択問題をモデル化した。便宜上、本稿ではこれをコミュニティモデルと呼ぶ。

ライセンサーはライセンスの制約の厳しさを考慮してライセンスを選択する。ライセンスの二次著作物に対する制約の厳しさにおいて、特に重要な基準は次の二点である。

(i) OSS を改変して二次著作物を作成し、それを配布する際に、二次著作物のソースコードを公開しなければいけないことを、ライセンスが求めているか否か。これはコピーレフトの考え方に言及されているものを示す。

(ii) ライセンスが、OSS と同様のライセンスを採用していない他のソフトウェア(たとえばクローズドソースのソフトウェア)とを組み合わせて新たなソフトウェアをつくることを制限しているかいないか。

2.2 節の実証分析では、(i)の性質を持つライセンスを「制約的な」ライセンス、(i)に加えて(ii)の性質も持つライセンスを「非常に制約的な」ライセンスと表現している。

[7]たとえば、GPLのようなライセンスは「非常に制約的なライセンス＝非常に厳しいライセンス」となり、LGPLは「制約的なライセンス＝厳しいライセンス」、BSDは「非制約的なライセンス＝ゆるいライセンス」といった具合に分類される。

ライセンサーは(a)ソフトウェアをオープンソース・ライセンスの下で公開するのかどうか、(b)もしオープンソース・ライセンスを採用するのならば、どのタイプのライセンスを用いるのかを決断する。OSS の開発に携わるコミュニティの便益はライセンスのタイプに依存するから、ライセンサーはコミュニティに所属するプログラマー達の複雑なモチベーションを維持するために、ソフトウェアの性質や環境などの要因を考慮に入れる必要がある。

ライセンサーとコミュニティは、非商業的な便益をオープンソース開発プロジェクト（以下プロジェクト）から得る。非商業的な便益とは、仲間からの承認・キャリアの改善・達成感・ソフトウェアを自分の業務に使いやすいように改変し、その利用自体から生まれる便益などが想定される。こられの便益は、ライセンサーにとって $a+c$ で表現され、コミュニティにとっては $a+b+c$ で表現される。

a はプロジェクトの魅力を示す、外生的なパラメーターである。 c はライセンスの制約の度合いである。ライセンスの制約が厳しいほど c の値は大きくなる。これが正のパラメーターとして効用に組み込まれている理論的な根拠は、もし制約のゆるいライセンスが採用された場合、商業ソフトウェア企業などによる乗っ取りが生じる恐れがあるからである。より詳しく言えば、OSS にいくつかのクローズドなソフトウェアを組み合わせ、そのソースコードを秘匿することで、OSS の派生物を容易に創造することができ、場合によっては、その派生物で市場を独占することすら可能になるかもしれない。こうした乗っ取り行為が社会的に有害かどうかははっきりとしないものの、少なくともオープンソースの開発者たちからはいくつかの非商業的な便益を奪うこととなるだろう。ゆるいライセンスはそうした危険性を保有しているため、 c は正のパラメーターとして組み込まれている。コミュニティの効用のみに含まれている b はコミュニティの OSS プロジェクトに対する熱狂度を反映したランダムパラメーターである。これはコミュニティによるプロジェクトの将来性に対する評価を示唆する。また、その他の、コミュニティにあってライセンサーにはない便益（前述した自分の業

---

[7] Lerner and Tirole (2005) の論文が上梓された当時はオープンソース・ライセンスに関わる裁判の判例は存在せず、彼らもライセンスの制約については、解釈の曖昧さの残るところであると記述している。しかし、同時に彼らは理論のモデル化及び実証分析の上で重要なのは序数的に制約が厳しいか否かであると説明している。それは、米国の連邦巡回高等裁判所（CAFC）などによりオープンソース・ライセンスの下での著作権が認められ、いくつかの判例が出ている現在も変わらない。

務に合うようにコーディングすることなど）は、bの分散に組み込まれる。

プロジェクトから得られる商業的なインセンティブは、1.3 節において述べたRedHat のように、OSS の成功から、副次的に生じるサービスや商品などから得られる金銭・収益として描かれる。ライセンサーが企業ならば、OSS と競合するソフトウェアを開発している競合他社に対し、競争の圧力をかけることによって、間接的に収益を得ることも可能かもしれない。いずれにせよ、そうした金銭上のインセンティブを$\pi(c)$と表現する。$\pi(c)$は期待収益であり、$\pi'(c)<0$である。これは、より厳しいライセンスほど商業的な収益を得る機会が少ないことを反映している。

ライセンサーは、

$$a+c+\beta\ \pi(c) \tag{2.1}$$

を最大にするライセンス制約の度合いを選択する。βはライセンサーがどれだけ商業的便益に重きをおいているかを表す指標である。次に、コミュニティのプロジェクト参加への熱心さを、コミュニティが実際にプロジェクトに参加する意志があるかを示す確率$p(c)$で表現する。さらに、ソフトウェアをオープンソースとせず、ソースコードを秘匿することによってライセンサーが得られる収益を$\overline{V}$とすると、ライセンサーは

$$\max_{\{c\}} p(c)[a+c+\beta\ \pi(c)] \geq \overline{V} \tag{2.2}$$

の時ソフトウェアをオープンソースとして開発する。$\overline{V}$はもちろんcから独立している。しかし、その一方で、$\overline{V}$はaと正の相関関係にある可能性がある。たとえば、競合するソフトウェアに市場を独占されている場合、$\overline{V}$は低くなるが、そういった場合、コミュニティはその OSS にあまり魅力を感じないはずだ。なぜなら、成功の見込みのないソフトウェアに貢献することは彼らの気が進まないからである。言い換えれば、出遅れたプロジェクトはオープンソースとなりやすいものの、強力な競合との戦いにさらされるため、コミュニティからの同調は得られないだろう、ということになる。

次に、コミュニティの選好をみていく。コミュニティの商業的便益に対する重み付けをαとすると、コミュニティがプロジェクトに参加することによって得られる便益は

$$a+b+c+\alpha\ \pi(c) \tag{2.3}$$

と表現される。ここで、次の二つの仮定を設ける。

①変数bの分布は累積分布$F(b)$に従い、その密度は$f(b)$である。この分布は単調な危険率を持つ。即ち、$f(b)/(1-F(b))$はbに対して単調増加する。
②ライセンサーの商業的便益に対する重み付けは、コミュニティの重み付けよりも大きいか等しい。即ち、$\beta \geqq \alpha$である。

ここで、$\overline{U}$をコミュニティのプロジェクトに参加する際の機会費用だとすると、コミュニティの参加制約は

$$a+b+c+\alpha\ \pi(c) \geq \overline{U} \tag{2.4}$$

と表現される。そのため、コミュニティがプロジェクトに参加する可能性は

$$p(c) = 1-F(\overline{U}-[a+c+\alpha\ \pi(c)]) \tag{2.5}$$

となる。

(2.4)と(2.5)からライセンサーの解くべき最大化問題は、

$$\max_{\{c\}}\left\{\left[1-F(\overline{U}-(a+c+\alpha\ \pi(c)))\right][a+c+\beta\ \pi(c)]\right\} \geq \overline{V} \tag{2.6}$$

となる。ライセンサーはこの最大化問題を$c \in [\underline{c}, \overline{c}]$について解く。[8]

ここで、仮にコミュニティが制約のゆるいライセンスを好むとすると、$1+\alpha\ \pi' < 0$となる。$\beta \geqq \alpha$が仮定されているから、ライセンサーについても同様である。よって、この時にライセンサーが選択するライセンスの制約は、様々な要因から独立して$c = \underline{c}$の端点解となるのは自明である。

問題は、コミュニティがより厳しいライセンスを好む場合、即ち$1+\alpha\ \pi' > 0$の時である。[9]ここで、

$$\hat{a} \equiv a+c+\alpha\ \pi(c),\ \frac{\partial \hat{a}}{\partial c} \in (0,1)$$

とおく。さらに、この式をcについて解いたものを$c(\hat{a})$とする。さらに、

$$\hat{\pi}(\hat{a}) \equiv (\beta-\alpha)\pi(c(\hat{a}))$$

とおく。これらと(2.6)により、ライセンサーの目的関数を次のように書きなおすこと

---

8) 例えば、$\underline{c}$はBSD、$\overline{c}$はGPLのようなライセンスである。

9) 現実的には、こちらの方がより一般的である。

ができる。

$$V = \left[1 - F(\overline{U} - \hat{a})\right]\left[\hat{a} + \hat{\pi}(\hat{a})\right] \tag{2.7}$$

ただし、$(\hat{a} + \hat{\pi}(\hat{a}))' = 1 + \beta\ \pi' / 1 + \alpha\ \pi'$である。この設定では、ライセンスの制約$c$を選択することは$\hat{a}$を選択することと等しい。そこで、(2.7)の両辺の log をとり、$\hat{a}$について微分すると次の関係式が得られる。

$$\frac{\partial(\log V)}{\partial \hat{a}} = \frac{f(\overline{U} - \hat{a})}{1 - F(\overline{U} - \hat{a})} + \frac{1 + \hat{\pi}'(\hat{a})}{\hat{a} + \hat{\pi}(\hat{a})} \tag{2.8}$$

もし$1 + \hat{\pi}' > 0$、即ち$1 + \beta\ \pi' > 0$であるとき（ライセンサーが制約的なライセンスを好む場合）は、他の要因から独立して$c = \overline{c}$の端点解が選択される。続いて、内点解が得られる場合について考察する。内点解においては FOC、即ち$\partial(\log V)/\partial \hat{a} = 0$を満たす。ここで、$\overline{U}$について比較静学を行う。(2.8)を$\overline{U}$について微分すると

$$\frac{\partial^2(\log V)}{\partial \overline{U} \partial \hat{a}} = \left(\frac{f}{1 - F}\right)' > 0$$

が得られる。ゆえに、$\hat{a}$はコミュニティの機会費用$\overline{U}$が増加すれば、同様に増加しなければならない。$\hat{a}$が増加する時は、ライセンスの制約の度合いcが増加する時であるから、結局コミュニティの機会費用$\overline{U}$が増加する時、ライセンサーはより制約の厳しいライセンスを採用することとなる。

ここで、仮に$\hat{\pi}$が凹関数だったとすると、ある追加的な示唆を得ることができる。FOC を微分すると$0 < \partial \hat{a} / \partial \overline{U} < 1$がわかる。これにより、$\hat{a}$の増分は$\overline{U}$の増分より少ないから、コミュニティの参加可能性は機会費用の増大に伴い減少していくということがわかる。

この結果に対し、さらにいくつかの追加的な示唆を得ることができる。まず、OSS プロジェクトが充分に多く市場に存在する場合、機会費用$\overline{U}$は他のプロジェクトの相対的な魅力としてみなすことができる。ライセンサーへの魅力を一定とすると、プロジェクトの魅力が向上する時、ライセンスの制約はよりゆるくなり、コミュニティの参加率は上昇する。これを、bの分布の移動で表現する。今、θをコミュニティに対してのみ機能するプロジェクトの魅力を変動させる要素だとする。そして、bの分布を$F(b - \theta)$とすると、危険率の仮定から$f(b - \theta)/(1 - F(b - \theta))$は$\theta$に対して減少関数となる。より魅力的なプロジェクトはθの値が高いものである。

θ の値が高いプロジェクトは、例えば①コミュニティがライセンサーを信用しているプロジェクト②プログラムを自分好みに書き換えることで便益が得られやすいプロジェクトなどがある。

OSS の中には、それ単体では成功を収めることができず、機能を補う他のソフトウェアと組み合わせなければ成功しないといった性質を、そもそも持ったものも存在する。こうした性質を持った OSS であれば、GPL のような厳しいライセンスを採用することによって、潜在的な商業ユーザーを失望させてしまうことになる。こうした性質は、$\pi(c)$ の中に制約的なライセンスが潜在的な商業営利を減少させるという形で組み込まれている。

このことに関連して、プロジェクトの潜在的な商業可能性について考察する。例えば、プロプリエタリなOSや営利企業をユーザーに持つような場合である。[10]直感的には、こうした場合、プロジェクトはゆるいライセンスを持つと予想できる。しかし、こうしたケースにおいては、ライセンサーは大きな収益を得ることができるから、まずはプロジェクトの成功を確実なものとするために、より制約の厳しいライセンスを採用するかもしれない。

この状況を分析するために、$\pi(c)$ を $\pi(c)=\pi_0+h(1-c)$ と改める。ここで、π_0 は OSS を提供することによって得られる収益をあらわす。一方、$h(1-c)$ は、他の相補的なソフトウェアと組み合わせることによって得られる潜在的な収益をあらわす。ここで、$c\in[0,1]$ とする。$c=1$ の時、ライセンサーの目的関数 V は h から独立である。c が 1 より小さい時、h が増加することにより、コミュニティの参加に伴ってライセンサーの便益が向上する効果と、コミュニティの参加自体がもたらす効果を通じて、V は増加する。よって、h が増加する場合、GPL のような制約の厳しいライセンスは、それより制約のゆるいライセンスよりも魅力が小さくなる。結局、プロプリエタリな OS で動作するものや、商業的な環境において機能する OSS においては、非常に制約の厳しいライセンスは採用されにくいということが言える。

最後に、ライセンサーの参加制約（$V\geq\overline{V}$）によって生じるサンプルセレクションバイアスについて考察する。もし $\overline{V}$ がオープンソースのプロジェクトのあらゆるパラメーターから独立していれば、このバイアスは生じない。しかし、前述したとおり、$\overline{V}$ は a と正の相関関係にある可能性がある。このことにより、充分に高い魅力 a をもつプロジェクトは、オープンソースの領域から一掃されてしまう危険性がある。もし

10) プロプリエタリなソフトウェアとは、使用、改変、複製を法的・技術的な手法を用いて制限しているソフトウェアのこと。その性質から、オープンソースと対比されることが多い。

これらが OSS であれば、制約のゆるいライセンスを選択したはずである。

こうしたバイアスは、ライセンサーが企業である時に発生しやすい。企業は競合相手に遅れをとっている時、ソフトウェアをオープンソース化し、コミュニティを巻き込んで競合相手に追いつこうとする傾向があるからだ。このことは、企業ライセンサーの制約の厳しいライセンスを提供する傾向を強化してしまっているかもしれない。

2.1.2 企業モデル

Wang (2008) では、時間の流れと共に、OSSが商業との結びつきを強めていく中で、商業との関わり合いの下で、ライセンサーがいかにライセンスを選択するべきかをモデル化している。このモデルでは、ライセンサーはプロジェクトに投入される貢献量の総和を最大化することを目的とすると仮定する。オープンソース・ソフトウェアの品質はコミュニティの貢献量、努力量の総和に大きく依存する。[11]逆に言えば、継続的に多くの努力量を投入することに成功すれば、OSSの品質は向上し続けていくといえよう。そのため、この仮定は、現実的に妥当といえる。

今、ライセンスの制約のゆるさを$\gamma \in [0,1]$とする。コミュニティモデルでは制約の厳しさをcとおいたが、このモデルでは厳しいほどγの値が小さくなるところに注意をしていただきたい。即ち、$\gamma=1$は BSD、$\gamma=0$は GPL にそれぞれ該当する。

また、OSS の開発段階を二期にわける。第一期はプロジェクトの開始初期段階であり、第二期はプロジェクトの開発が進んだ成熟期である。各期において、オープンソース・コミュニティと営利企業の二つのプレイヤーがそれぞれの貢献量を選択する。コミュニティの貢献はコードを書く努力であり、企業の貢献は有給のプログラマーを派遣することの他に、プロジェクト自体に開発資金を提供すること等が挙げられるが、いずれにせよ何らかの投資行動である。各プレイヤーの選択した貢献量は、その期が終わるまでお互いに観察することはできない。営利企業は純粋に利潤を最大化することを目的とするが、コミュニティはこれまで繰り返したように、様々なインセンティブを持っている。そうしたインセンティブは、一般的に企業の乗っ取り行為により危険に晒されるから、コミュニティはより制約の強いライセンスを好む。企業は、その逆である。

ここで、第一期におけるコミュニティの効用関数を次のように表現する。

---

[11] Linux の開発者リーナス・トーバルズの言葉に"Given enough eyeballs, all bugs are shallow."（目玉の数さえ十分あれば、どんなバグも深刻ではない）というものがある。コミュニティによる貢献が、改善のための主なエンジンである OSS は、大勢のコミュニティメンバーがプロジェクトに参加することによって、その品質が保たれる。

$$U_C^1 = a - \left[e_0^1 - \left(1 - \frac{\gamma}{2} \right) \right]^2 \tag{2.9}$$

さらに、第二期におけるコミュニティの効用関数を次のように表現する。

$$U_C^2 = \begin{cases} a - \left(e_0^2 - \dfrac{1}{2} \right)^2, & if \quad \gamma \neq 0 \quad and \quad e_1^1 = 0 \\ a - \left(e_0^2 - 1 \right)^2, & otherwise \end{cases} \tag{2.10}$$

$a > 0$である。$e_0^t \in [0,1]$は第t期におけるコミュニティのプロジェクトへの貢献量であり、同様に、$e_1^t \in [0,1]$は第t期における企業の貢献量である。(2.9)式の通り、コミュニティの貢献はγの値が大きくなると減少する。第二期においては、ライセンスがGPL($\gamma = 0$)でない時は貢献量は少なくなる。しかし、第一期において企業がプロジェクトに対して貢献を行った場合、これを観察したコミュニティは第二期においてより多くの貢献を行う。これを“お返し”効果と呼ぶ。

次に、第t期における企業の利得は次のとおりである。

$$\begin{aligned} & U_F^t = U_H^t + U_S^t = \left(\gamma \, e_0^t \right) + \left(\alpha_t e_1^t - e_1^t \right) \\ & where \quad \alpha_t = \alpha \sum_{i=0}^{1} \sum_{r=1}^{t} (e_i^r) \quad and \quad \alpha > 0 \end{aligned} \tag{2.11}$$

企業の利得は、二つの部分に分解される。一つは乗っ取りによるもので、もう一つが関連商品・サービスの販売によるものである。それぞれがU_H^t、U_S^tにあたる。乗っ取りによる利得は、当然γの上昇にともない増加する。一方、関連商品・サービスによる利得は、次の二つの要因によって決定される。それは、①潜在的な商業的価値と②ソフトウェアの品質である。①の潜在的な商業的価値はそのプロジェクトの機能分野に依存する。たとえばオペレーティング・システムの分野などは、高い商業的価値があると見做されている。(2.11)式においては、α_tの中のαがこの潜在的な商業的価値に相当する。また、ソフトウェア産業は極めて競争的な産業であるから、該当分野において他を圧倒できるような品質がなければ、市場において成功することはできず、関連サービスによる利得を稼ぎ出すことはできない。さらに、ソフトウェアには互換性の問題があるため、高いネットワーク外部性が存在することも無視できない。[12]そ

[12] 特にOSやオフィスソフトなどの分野では、互換性の問題が顕在化しやすい。

のため、高い品質のソフトウェアは極めて広く採用されるが、その一方で品質の低いソフトウェアは淘汰される。これらのことから、②ソフトウェアの品質は重要な要因と言えるだろう。(2.11)式においては、α_tの中の、Σ以降がこのソフトウェアの品質に相当する。この和の式はコミュニティ及び企業の貢献量の総和を表現している。OSSの品質は前述の通り、貢献量の合計に大きく依存するから、このように表現することができる。

以上のU_S^tの性質を鑑みれば、企業にとって、第一期から投資を行うことが高いリスクを伴うことが言える。なぜなら、OSS の開発初期段階では、ソフトウェアの品質に未だ多くの改善の余地があるのが一般的だからだ。

以上を踏まえて、企業は次の４つの戦略を取りうるものとする。

①全期投資(full investment)：$e_1^1 = e_1^2 = 1$

②無投資(staying out)：$e_1^1 = e_1^2 = 0$

③早期投資(early investment)：$e_1^1 = 1, \quad e_1^2 = 0$

④成熟期投資(late investment)：$e_1^1 = 0, \quad e_1^2 = 1$

この時、③早期投資と④成熟期投資は被支配戦略である。以下ではそれを証明する。企業が全期投資戦略を選択した時、コミュニティの貢献量は$e_0^1 = 1-\gamma/2, \quad e_0^2 = 1$となる。この時企業の全期の利得は(2.11)に各値を代入して

$$\begin{aligned} U^F &= \left[\gamma(1-\gamma)+\alpha\left(2-\frac{\gamma}{2}\right)-1\right]+\left[\gamma+\alpha\left(4-\frac{\gamma}{2}\right)-1\right] \\ &= 2\gamma-\frac{\gamma^2}{2}+\alpha(6-\gamma)-2 \end{aligned} \tag{2.12}$$

となる。同様の手順で、企業の各戦略における利得を求めると

$$\begin{cases} U^S = \dfrac{3}{2}\gamma-\dfrac{\gamma^2}{2}, & \text{無投資} \\ U^E = 2\gamma-\dfrac{\gamma^2}{2}+\alpha\left(2-\dfrac{\gamma}{2}\right)-1, & \text{早期投資} \\ U^L = \dfrac{3}{2}\gamma-\dfrac{\gamma^2}{2}+\alpha\left(\dfrac{5}{2}-\dfrac{\gamma}{2}\right)-1, & \text{成熟期投資} \end{cases}$$

となる。もし早期投資が支配戦略である場合、$U^E \geq \max\left\{U^F, U^S, U^L\right\}$を満たさなくてはならない。即ち、

$$\begin{cases}\gamma(1-\alpha)\geq 2-4\alpha \\ \alpha\ \gamma\geq 8\alpha-2 \\ \gamma\geq\alpha\end{cases}$$

を満たさなくてはならないが、$\alpha\in(0,+\infty)$，$\gamma\in[0,1]$において、この三つの不等式を同時に満たす（α,γ）の組み合わせは存在しない。よって、早期投資はあらゆる組み合わせにおいて支配される、即ち被支配戦略である。同様に、成熟期投資についても被支配戦略であることが証明できる。

結局、企業は全期投資か無投資かのいずれかの戦略しか取りえないことがわかる。企業は両戦略の利得を比較し、どちらか一方の戦略を選択する。

以上の状況において、ライセンサーはコミュニティと企業の貢献量の和を最大化するγを選択する。この時、次の命題が得られる。

命題：ライセンサーは次のようにγを選択する。

$$\begin{cases}\gamma=0, \quad if \quad \alpha<\dfrac{3}{10} \quad and \quad \alpha>\dfrac{1}{3} \\ \gamma=\dfrac{4-12\alpha}{1-2\alpha}, \quad if \quad \alpha\in\left[\dfrac{3}{10},\dfrac{1}{3}\right]\end{cases}$$

企業は、$\alpha<3/10$の時には無投資戦略を選択し、$\alpha>1/3$の時には全期投資戦略を選択する。

(証明)

企業は前に証明したとおり無投資戦略か全期投資戦略かしか選択しないので、この両者の利得を比較してやれば良い。全期投資が企業の最適戦略である時、次の条件を満たさなくてはならない。

$$\begin{cases}U^F\geq U^S \\ 0\leq\gamma\leq 1\end{cases}$$

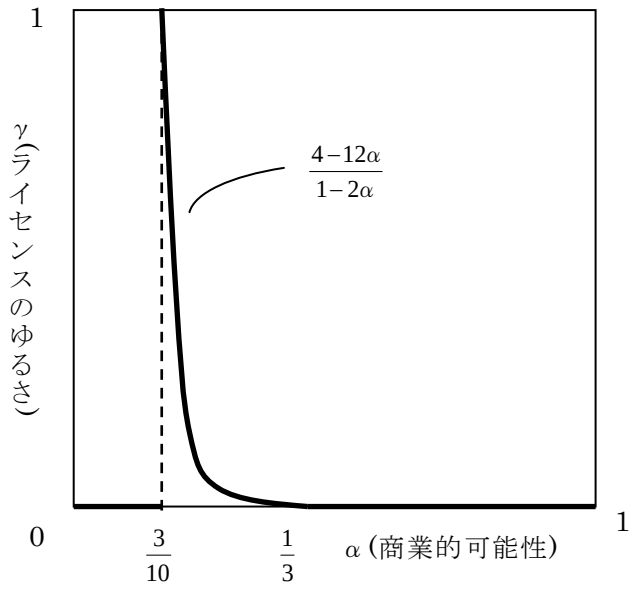
このような場合、企業の貢献量は$e_1^1=e_1^2=1$であるから、ライセンサーは上記の条件を満たしつつ、コミュニティが最も好むライセンスを選択してやれば良い。即ち、ライセンサーの選択は、上記の条件を満たす最も低いγということになる。これを解くと、ライセンサーの選択は、

$$\begin{cases} \gamma = \dfrac{4-12\alpha}{1-2\alpha} \quad when \quad \alpha \in \left[\dfrac{3}{10}, \dfrac{1}{3}\right] \\ \gamma = 0 \quad when \quad \alpha > \dfrac{1}{3} \end{cases} \tag{2.13}$$

となる。同様に、企業の最適戦略が無投資である時、即ち$U^F \le U^S$かつ$0 \le \gamma \le 1$の場合を考える。これは$\alpha < 3/10$である時に成立する。このような場合、企業からの貢献を得られないから、ライセンサーはコミュニティからの貢献を最大にすることのみを考えれば良い。よって、ライセンサーは$\gamma = 0$を選択する。以上によって、命題が証明された。

この命題の中でより注目すべきは、OSS の潜在的な商業的可能性αとライセンス制約のゆるさγとの関係性であろう。この関係性をグラフにまとめたものが図 2-1 である。このグラフが示唆することを直観的に解釈すると次のとおりである。まず、OSS の潜在的な商業的可能性が充分に大きい時は、ライセンスの制約の度合いに関わらず企業は投資行動を行うとライセンサーは予測できる。そのため、ライセンサーはコミュニティの好みだけを考慮すればよいということとなり、コミュニティの選好に合致した GPL のような制約の厳しいライセンスを選択する。また、OSS の商業的可能性が充分に小さい時は、企業にとって旨みが小さすぎるため、今度はライセンスの制約の度合いにかかわらず企業が投資を行わないと予想できる。この時も同様に、ライセンサーはコミュニティの嗜好だけを反映すればよいということとなり GPL を選択する。最も興味深いのが、中間的な商業的可能性をもつ場合である。この場合、ライセ

図 2-1　ライセンスの制約と商業的可能性との関係



ンサーは企業の機嫌をうかがいながら、ライセンスの制約をある水準までゆるめていくこととなる。仮に、ライセンサーが GPL を採用したとする。この場合の商業的可能性は、企業にとって、早期と成熟期の両方の期間に亘って投資するのに充分なほど高いものではない。早期では、その低い商業的価値によって、関連商品・サービスから得られる利潤は負である。しかしながら、もし企業が両期にわたって投資を続けたなら、成熟期における関連商品・サービスから得られる利潤は正となる。そこで、ライセンサーは企業の投資を後押しするために、ライセンスの制約をゆるめ、乗っ取りによる利潤を発生させる。結局ライセンサーは、企業が投資を決断する最低水準までライセンスの制約をゆるめていくこととなる。

2.2 実証分析

ライセンシング問題の理論としては、Lerner and Tirole (2005) によるコミュニティモデル、Wang (2008) による企業モデルの二つを紹介した。コミュニティモデルは、オープンソース・コミュニティの機会費用がライセンスの制約の厳しさと正の相関関係をもつということを示した。企業モデルでは、商業的可能性が中間的な場合においては商業的可能性とライセンス制約が正の相関関係を持つということ、並びに中間的な場合以外では、GPL のような非常に制約の厳しいライセンスが選択されることを示した。これは図 2-1 に示したとおりのことである。

この二つのモデルを比較すると、商業的可能性に関する考察において、異なる帰結を提示している。コミュニティモデルでは、オープンソースプロジェクトの商業的可能性が向上すればコミュニティの機会費用は減少する。[13]これは、商業的可能性の上昇によって、ライセンスの制約がゆるくなっていくということを示している。それに対して、企業モデルの方では、商業的可能性が極めて大きい場合においても、最も制約の厳しいライセンスが選択されることを示し、中間的な場合には、商業的可能性が高くなると、ライセンスの制約が厳しくなっていくことを示している。これら二つのモデルの商業的可能性に対する観点をまとめると、次のとおりである。

コミュニティモデル：商業的可能性の高いソフトウェアは、BSD などのような制約のゆるいライセンスを選択しやすい傾向にある。
企業モデル：商業的可能性が高い場合、低い場合は、GPL のような制約の厳しいライセンスを選択する。商業的可能性が中間的なソフトウェアにおいて、商業的可能性が

13) (2.4)式参照。

向上するほどに制約の厳しいライセンスを選択しやすい傾向にある。
この二つの帰結のうち、どちらがより現実に妥当かということを、実証分析を通じて確認する。実証分析は、Lerner and Tirole (2005) を参考にして行う。Lerner and Tirole (2005) では 2.1.1 項で紹介したコミュニティモデルの実証分析を、SourceForge.net という OSS の開発とダウンロードの場を提供するウェブサイトのデータを用いて行なっている。このサイトには、2011 年現在において 30 万以上の OSS が登録されており、開発及び配布が行われている。本稿では、同ウェブサイトの最新のデータを用いて、独自の手法で分析を行う。

2.2.1 データセット

データは、前述の通り SourceForge.net という OSS のウェブサイトから取得した。まず、サンプルについて説明する。サンプルは、SourceForge.net により定義された十の機能分野ごとに収集した。SourceForge.net では、Audio＆Video や Business＆Enterprise といった機能分野が定義されており、それぞれの OSS は該当の機能分野に割り振られていく。データを機能分野ごとに収集したのは、企業モデルの理論において説明したように、今回焦点をあてる商業的可能性 α が、ソフトウェアの機能分野に依存していることによる。具体的には、十の機能分野のそれぞれにおいて、ランキング上位 112 個ずつの OSS をサンプルとして選択した。ランキングは SourceForge.net 内でつけられているもので、OSS のダウンロード数やユーザーレビュー数を考慮してつけられている。ランキング上位をサンプルとして選択したのは、活発でない OSS だと情報が登録されていないことが多く、データが充分に得られない、というデータの入手可能性を勘案した結果である。こうして得られた 1120 のサンプルのうち、複数の機能分野にまたがって登録されているデータの重複を取り除き、データの欠損したサンプルを省いた合計 775 のソフトウェアをサンプルとした。

次に、被説明変数を紹介する。被説明変数はライセンスの制約の度合いであるが、OSS はデュアルライセンスやマルチライセンスと言った具合に、複数のライセンスを採用している場合があるから、単純なダミー変数による数値化を行うことはできない。そこで、Lerner and Tirole (2005) の考案した手法を改良して、ライセンスの制約の度合いを数値化する。まず、ライセンスを「非常に制約的なライセンス」、「制約的なライセンス」、「制約的でないライセンス」の三種類にふるい分ける。このふるい分けの基準は、コピーレフトの価値観の有無、及び、同様のライセンスを採用していないソフトウェアと組み合わせることを制限しているかいないか、という二つの基準によ

る。両方の性質を持つものを「非常に制約的なライセンス」と呼び、前者のみの性質を持つものを「制約的なライセンス」、どちらの性質も持たないものを「制約的でないライセンス」とする。今、説明のために「非常に制約的なライセンス」を A、「制約的なライセンス」を B、「制約的でないライセンス」を C とする。そして表 2-1 の基準に基づいて、各 OSS のライセンスの制約の度合いを数値化する。数値が大きいほど制約が厳しいことを意味する。

表 2-1　ライセンス制約の数値化基準 [14]

ライセンスの制約の度合い	採用しているライセンス
4	全て A
3	A のライセンス＋B
2	全て B
1	A あるいは B のライセンス＋C のライセンス
0	全て C

次に、説明変数を紹介する。説明変数は、SourceForge.net の各 OSS のページに掲載されているデータを用いる。サイトには、動作環境や対象とするユーザー層が掲載されている。以下では、その一つずつを紹介する。

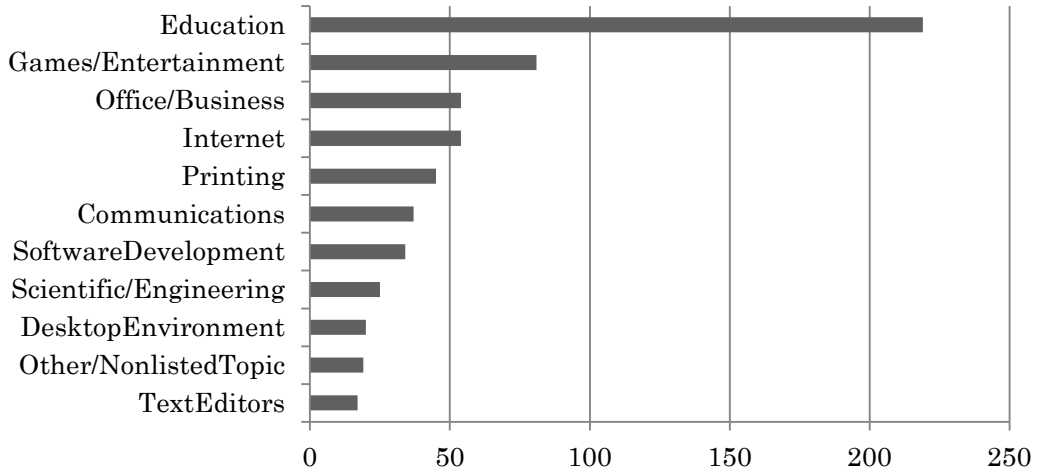
- Directory：SourceForge.net によるソフトウェア分類。本稿では機能分野として用いる。Audio&Video, Business&Enterprise, Communications, Development, Home&Education, Games, Graphics, Science & Engineering, Security&Utilities, System Administration の十の分野に分類されている。各分野ごとにダミー変数を設け、該当するものを 1 とし、それ以外を 0 とする。
- Intended Audience：対象とするユーザー層。End Users/Desktop, Developers, System Administrators などをはじめとして、いくつかのカテゴリが存在する。コミュニティモデルの議論において、仲間への承認をコミュニティのインセンティブとして挙げた。ここでは、次のような仮定を立てる。Developers や System Administrataors 向けのソフトウェアは仲間への承認が得られやすく、ゆるいライセンスを採用しやすい傾向がある。一方で Users/Desktop などのライトユーザ

[14] Lerner and Tirole (2005) では、3 を「A のライセンス＋B あるいは C ライセンス」、1 を「B のライセンス+C のライセンス」としていた。しかしながら、デュアルライセンスの「二次著作物の配布者は、配布にあたり、採用されているライセンスのうち、より目的に適った方を用いることができる」という性質を考慮し、今回のような変更を加えた。なぜなら、一つでも C のライセンスが採用されていれば、そちらのライセンスを採用することによって、コピーレフトの価値観を踏襲せずに配布を行うことが可能となるからだ。

一向けのソフトウェアは厳しいライセンスを採用しやすい傾向がある。

- Topic：各 OSS に対して、用途や関連する分野がタグ付されている。これは機能分野である directory よりも細かい分類であり、一つのソフトウェアに対して複数のタグがついていることが多く、その種類も多岐に渡る。図 2-2 はその中でも代表的なものと、そのサンプル中の数である。Education が最大数で 219 個であり、それに続く Games/Entertainment が 81 個であることから、非常に多様なタグが存在することがわかる。この変数に対しても Intended Audience と同様に考えることができる。Software Development のようなタグを付けられたソフトウェアでは、仲間への承認が得られやすく、ゆるいライセンスを選択しやすい。一方で、Education や Games/Entertainment, Office/Business といったものは厳しいライセンスを選択しやすい。その他のカテゴリについては、何れの分類に所属するかを判断するのは難しい。

図 2-2　代表的 Topic とサンプル中の数



- Platform： Platform は、利用可能なオペレーションシステムを指す。これはソフトウェアの技術的な特性を捉えるための代理変数として用いられ、コミュニティモデルにおける追加的な示唆、プロプリエタリな OS や商業的な環境ではゆるいライセンスが採用されやすいという結論に対応している。
- Not English：OSS の開発をしていく上で英語以外の言語が用いられているものを 1,そうでないものを 0 とするダミー変数としておいた。英語以外の言語の場合はコミュニティの機会費用が高まり、ライセンスが厳しくなる傾向があると予想できる。

- Month：SourceForge.net に登録されてから現在までにどれほどの月数が経過したかを示す。これは Lerner and Tirole (2005) において開発段階という指標が挙げられていたが、サイトを観察した結果、開発段階の精度に問題があったため、こちらの変数を代理として用いることとした。この変数によって、通時的な効果を制御する。OSS は開発が進むことにより、思想の変化やネットワーク効果による機会費用の変動が生じる。そうした通時的な変化が考慮され、採用ライセンスが変更されることが度々あるため、これを制御する。

以上の変数の他に、ユーザーレビューによるソフトウェアの評価（良い評価の比率を、パーセンテージで表したもの）と、一週間あたりのダウンロード数をそれぞれ「Review」及び「DL」として、説明変数に組み込んでいる。

2.2.2 推計結果

次ページの表 2-2 は、推計結果をまとめたものである。まず Directory についてであるが、これを見ると、Wang (2008) の帰結が妥当であることがわかる。この表では、最も制約の厳しいと推計された Development が排除され、それとの比較として数値が出ている。それを踏まえ、他の分野についてみていくと、Development と有意な差が存在するのは Communications, Home＆Education, Graphics, Security＆Utilities の 4 分野のみである。他の 5 分野については Development と有意な差はなく、Development と同程度に制約の厳しいライセンスを選択しやすい。企業モデルの理論に則れば、これら五分野に Development を加えた六つの分野は、商業的可能性の高い分野、あるいは商業的可能性の低い分野である。一方で、有意な差がみられた四つの分野は商業的可能性が中間的なものだと考えられる。しかしながら、これらは企業モデルに則るという前提付きの結論に過ぎないから、この結果をより詳細に検討するため、次のような NASDAQ 株式市場を用いた調査を行った。

NASDAQに上場している企業の中で、ソフトウェア産業に属しているもののうち上位五十社について主力製品（ソフトウェア）を調査し、その製品がSourceForge.netにおける十分野のいずれに該当するのかを調査した。具体的には、SourceForge.net上に掲載されているOSSの中で、機能が競合しているソフトウェアを照合し、そのソフトウェアが属している機能分野に該当するとした。[15]ソフトウェア産業に所属しているかどうかの判断基準は、NASDAQによる産業分類(Industry)において、

15) 例えば、Adobe 社であれば、画像編集ソフトウェアの Photoshop や Illustrator と機能が競合している Inkscape を選び、その所属している機能分野である Graphics に該当していると判断する。

表 2-2　推計結果

Directory	Coefficient	Standard Error
Audio/Video	−0.171	0.189
Business＆Enterprise	−0.228	0.195
Communications	−0.455	**0.195
Development	(omitted)	
Home＆Education	−0.527	***0.195
Games	−0.270	0.194
Graphics	−0.798	***0.199
Science＆Engineering	−0.219	0.195
Security＆Utilities	−0.339	*0.194
System Adminisitration	−0.131	0.196
Intended Audience		
End Users/Desktop	0.538	***0.199
Developers	−0.463	***0.152
System Administrators	0.154	0.263
Advanced End Users	0.379	**0.173
Other Audiences	(omitted)	
Topic		
Education	0.236	0.263
Game/Entertainment	−0.065	0.314
Office Business	0.305	0.345
Internet	0.198	0.353
Printing	−1.73	0.41
Communications	0.648	*0.38
Software Development	−0.23	0.402
Sientific/Engineering	−0.214	0.464
Desktop Environment	−0.812	0.52
Text Editors	−1.065	*0.547

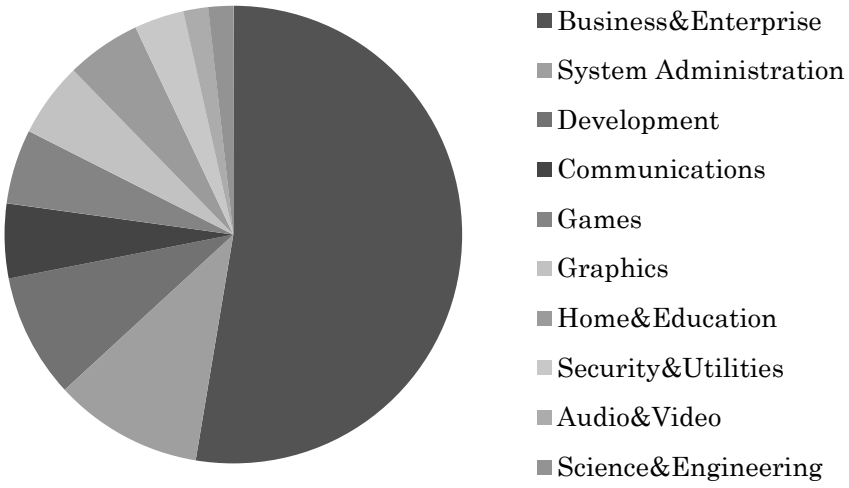
Platform		
Windows	−0.5505	*0.289
Linux	−0.22	0.415
BSD	−0.16	0.188
MacOSX	−0.67	***0.190
Other OS	(omitted)	
Not English	0.336	**0.141
Month	−0.003	0.002
Review	0.003	0.006
DL	−2.08E−08	5.68E−07

p−Value			0.000
Number of Observations			775
Adj R−squared			0.308
Breusch−Pagan test	chi2(30)	=	23.24
	Prob > chi2	=	0.8054
VIF(test for Multicollinearity)	Max = 3.07 (business & Enterprise)		
			Mean = 1.84

有意水準は、* p<0.1, ** p<0.05, *** p<0.01 で示している。

「Computer Software: Prepackaged Software」に分類されているものとした。[16]一部の企業（50 社中 5 社）は主力製品を異なる分野に持っていたので、その場合の重複を認めている。また、十分野のうち、Business&Enterpriseは定義が曖昧なものに思われたので、これはSourceForge.netに実際に登録されているソフトウェアの内容を調べ、経営資源の管理のためのソフトウェア(ERPパッケージ)が該当するものとした。より具体的には、財務管理や部署横断的データ管理ソフト、そして人的資本の管理(Human Resource Management)などのソフトウェアが所属している。以上の調査に基づいて得られたのが図 2-3 と次ページの図 2-4 である。

図 2-3　機能分野別企業数のシェア(NASDAQ 上位 50 社)



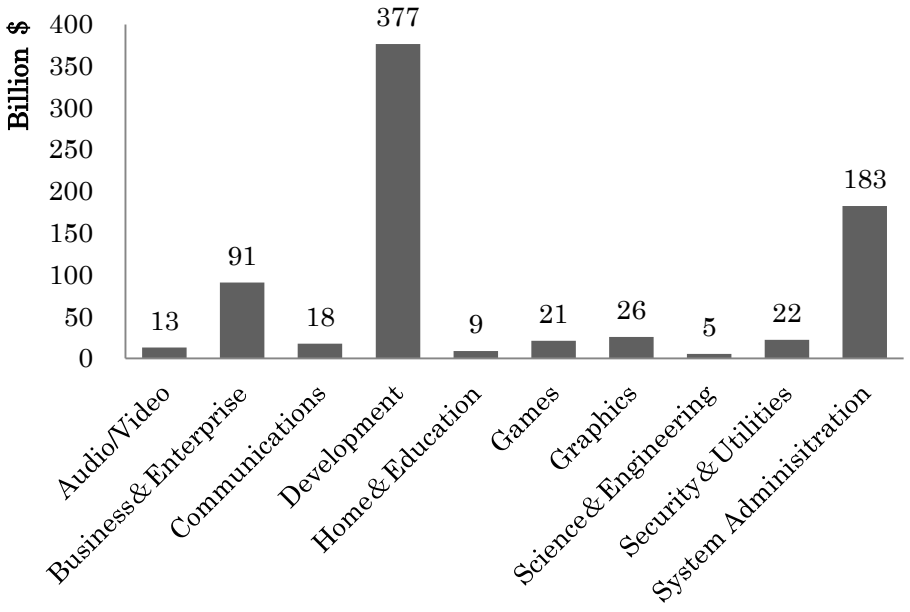
備考）機能分野はシェアの降順で記載

図 2-3 は、上位 50 社における、分野別の企業数シェアをあらわすグラフである。図の右側に示された機能分野の凡例はシェアの降順に記載している。例えば、最も企業数の多い事業ドメインは、Business&Enterprise である。そして、二番目に企業数の多い事業ドメインは System Administration、三番目は Development というように続いていく。ただし、このグラフだけでは商業的可能性を判断することはできない。なぜなら、このグラフは単純な企業数の分布を示したものであるから、競争状態について考慮する必要があるからだ。例えば、企業数が少なくても、市場が寡占状態であり、その数社で莫大な売上を上げているのであれば、商業的可能性は高いと言えるだろう。この問題に対処するため、図 2-4 を用いる。この図は、各分野に所属する企業

16) 似た分類として Computer Software: Programming, Data Processing があるが、こちらは主として Google や Baidu といったインターネットサービスを主体とした企業が所属しており、厳密にはソフトウェア産業とは呼びづらいため、今回の調査対象からははずした。

の NASDAQ における時価総額を合算したものとなっている。時価総額がそのまま商業的可能性に直結するわけではないが、一つの指標としては有益である。

図 2-3 機能分野別の時価総額合算（NASDAQ 上位 50 社）



この結果によると、Development のソフトウェアを主力とする企業は極めて高い市場価値を持ち、以降 System Administration、Business&Enterprise と続く。

以上の二つの調査を、実証分析の結果と照合したのが表 2-3 である。

表 2-3 推計結果と NASDAQ を用いた調査の照合

機能分野	企業数	市場価値(単位は十億ドル)
Development	5(3)	376.5(1)
System Administration	6(2)	182.5(2)
Audio/Video	1(9)	13.1(8)
Science&Engineering	1(9)	5.4(10)
Business&Enterprise	30(1)	90.9(3)
Games	3(4)	21.3(5)
Security&Utilities	2(8)	22.3(6)
Communications	3(4)	17.6(7)
Home&Education	3(7)	8.7(9)
Graphics	3(4)	25.6(4)

備考）括弧内の数字は順位。上位三位は網掛け、下位三位は黒地に白抜きで表示。

この表は、実証分析の結果に基づいて、各分野を、ライセンスの制約が厳しい（係数の絶対値が小さい）順序で並べている。また、太線より下は Development と有意な差が生じていた四つの機能分野である。NASDAQ の調査に基づく二つのそれぞれの指標については、上位三位を網掛け、下位三位を黒地に白抜きで表示している。この表によると、企業数と市場価値の双方が上位三位、または下位三位に入っている機能分野（両方網掛けか両方白抜き）は五つある。これらの五つはいずれも Development と有意な差のない機能分野(太線の上側)であるから、制約の厳しいライセンスを採用しやすい機能分野には、商業的可能性の高いものと低いものが集まっていると考えることができる。その一方で、Development と比較して有意に差のない四つの機能分野(太線の下側)は全て中間的な商業的可能性を保持しているものと考えられる。この結果は、企業モデルの帰結に合致していると言えるだろう。こうした結果となったのは、Lerner and Tirole (2005) のコミュニティモデルが考案された当時と比較して、営利企業がオープンソースに関わる事例が増えたためだと考えられる。そのため、企業の存在を重要視した Wang (2008) の企業モデルの方が、現実的に妥当なモデルを形成しているのではないか。

他の変数についても確認しておくと、有意となっているものは概ねコミュニティモデルの結論と合致する。Intended Audience では、仲間への承認が得られやすい Developers がゆるいライセンスを選択しやすいのに対し、End Users/Desktop は厳しいライセンスを選択しやすい。Platform では、商業的な Windows や MacOSX は、非商業的な BSD や Linux と比べてゆるいライセンスを選択しやすいということがわかる。また、Not English の係数が正であることから、英語を採用していないソフトウェアは厳しいライセンスを選択しやすいということがわかる。また、頑健性の確認として、不均一分散の確認(BP test)と多重共線性の確認を行ったが、どちらも異常はみられなかった。これらの検定の詳細は表 2-2 の末尾に掲載しているので、そちらを参照してもらいたい。

第 3 章　オープンソース・コミュニティの動機の分析

OSS を取り巻く環境において、ライセンスと同様に特徴的なのがオープンソース・コミュニティの存在であることを第 1 章で述べた。オープンソース・コミュニティは OSS の利用者であると同時に、その開発をもてがける、OSS のエンジンと言える存在である。オープンソース・コミュニティは様々な特質を持つが、その中でもとりわけ独特なのが、無報酬であるということだ。基本的に、彼らは OSS に対する貢献から報酬を得ることはない。では、彼らはなぜ OSS の開発に参加するのだろうか。何を動機にしてコミュニティは OSS の開発に取り組んでいるのだろうか。この章では、コミュニティの OSS 開発に焦点をあて、契約理論の観点から分析する。

3.1　シグナリング・インセンティブの理論

オープンソース・コミュニティが基本的に無報酬である点について、Lerner and Tirole (2002) は、コミュニティのメンバーは「シグナリング・インセンティブ」という動機に基づいて開発に取り組んでいるのではないかと指摘した。シグナリング・インセンティブとは、オープンソース・ソフトウェアの開発に携わることで自らの能力を示し、将来のキャリアをよりよくしようとすることから生じるインセンティブのことである。OSS の開発に参加し、自分の開発した機能が優れているということになれば、当然労働市場から高い評価を受けることができ、結果、それが将来のキャリアにつながる、という図式だ。

3.1.1 基本モデル

は、将来のキャリアがインセンティブに影響をもたらすという理論は、Holmstrom (1999) において構築されている。[17]以下では、Holmstrom (1999) の理論を紹介し、シグナリング・インセンティブを経済学の文脈で捉える。

まず、ηを労働者の才能の指標とする。この才能は固定であり、不完全に労働者や労働市場に知られているものとする。労働者と労働市場はηについての信念を共有しており、この信念は平均m_1、精度（分散の逆関数）h_1の正規分布に従っている。時間

17)　この理論は、経営者と株主との情報の非対称性のもとで生じるモラルハザードが、通時的に考えれば抑制されるのではないか、という議論に端を発している。Fama (1980) では、通時的に見れば、労働市場の力だけでこのモラルハザードを取り除くことができると述べられている。その理由は、経営者は将来のキャリアを気にかけ、労働市場の評判を維持するために努力を行おうとするからである。

の経過と共に、ηに対する学習が労働者の成果を通して発生する。第t期の成果は

$$y_t = \eta + a_t + \varepsilon_t, \quad t = 1,2,... \tag{3.1}$$

と表される。a_tは労働者の努力投入量であり、ε_tは統計的なノイズである。ε_tは独立しており、平均０、精度h_εの正規分布に従う。労働者は次のような効用関数を持つ。

$$U(c,a) = \sum_{t=1}^{\infty} \beta^{t-1}\left[c_t - g(a_t)\right] \tag{3.2}$$

$g(a_t)$は努力投入による不効用をあらわしており、$g(\cdot)$は凹型の増加関数である。また、関数$U(\cdot,\cdot)$は公に知られている。

労働者はどれだけ努力量を投入するかを決めるために、現在の成果が将来の給料にどれだけの影響を及ぼすかを計算する必要がある。またその一方で、将来の給料がどれだけ現在の成果に依存しているかは､労働者の意思決定ルールの関数である｡結果、この意思決定ルールと賃金の関数は均衡により定まる。一般的に、この相互関係は極めて複雑なものであるから、ここでは単純化する。

まず、$y^t = (y_1,...,y_t)$を第t期における成果の履歴だとする。この情報は、労働市場に知られており、賃金支払いの判断基準となっている。ここで、$w_t\left(y^{t-1}\right)$を第t期の賃金、$a_t\left(y^{t-1}\right)$を第t期における労働者の努力投入量とする。労働市場が競争的である場合、賃金は次のように設定される。

$$w_t\left(y^{t-1}\right) = E\left[y_t \middle| y^{t-1}\right] = E\left[\eta \middle| y^{t-1}\right] + a_t\left(y^{t-1}\right) \tag{3.3}$$

この式により、労働者の意思決定ルールがわかっている場合の第 t 期の賃金を決定する。一方で、労働者はこの賃金の関数を考慮しながら、次の最大化問題を解く。

$$\max_{\{a_t(\cdot)\}} = \sum_{t=1}^{\infty} \beta^{t-1}\left[Ew_t\left(y^{t-1}\right) - Eg\left(a_t\left(y^{t-1}\right)\right)\right] \tag{3.4}$$

(3.4) 式の解と (3.3) 式とで均衡が求まる。

労働市場は労働者の行動を直接観察することはできないが、(3.4) 式を解くことで推測することはできる。このことから、均衡においてy_tを観察することは、次に示す$\mathbf{z}_t$を観察することと一致する。

$$z_t \equiv \eta + \varepsilon_t = y_t - a_t^*\left(y^{t-1}\right) \tag{3.5}$$

ここで、$a_t^*\left(y^{t-1}\right)$は均衡における労働者の努力投入量を示す。$z_t$を観察することにより、市場は$\eta$を学習する。この学習により、$\eta$に対する信念の平均と精度は、それぞれ次に示す$m_{t+1}$と$h_{t+1}$に変化する。

$$m_{t+1} = \frac{h_t m_t + h_\varepsilon z_t}{h_t + h_\varepsilon} = \frac{h_1 m_1 + h_\varepsilon \sum_{S=1}^{t} z_S}{h_1 + t h_\varepsilon} \tag{3.6}$$

$$h_{t+1} = h_t + h_\varepsilon = h_1 + t h_\varepsilon \tag{3.7}$$

ここで、$\{m_t\}$をm_tのたどる履歴だとすると、$\{m_t\}$はランダムウォークの軌跡を辿り、その変化の分散はゼロに向かって減少していく。即ち、期間tを無限大に飛ばすと、労働者の才能であるηは完全に知られることとなる。

(3.6)により、賃金の決定ルールである(3.3)式は次のように書き換えることができる。

$$w_t\left(y^{t-1}\right) = m_t\left(z^{t-1}\right) + a_t^*\left(y^{t-1}\right) \tag{3.8}$$

$z^t = \left(z_1, \ldots, z_t\right)$である。さらに、(3.8)式の期待値を取ると、

$$Ew_t\left(y^{t-1}\right) = \frac{h_1 m_1}{h_t} + \frac{h_\varepsilon}{h_t}\sum_{s=1}^{t-1}\left(m_1 + a_s - Ea_s^*\left(y^{s-1}\right)\right) + Ea_t^*\left(y^{t-1}\right) \tag{3.9}$$

が得られる。この式から、t期における努力投入量a_tの限界収入は$\alpha_t = h_\varepsilon / h_t$となることがわかり、これは、過去の期間から独立している。以上の条件から、 労働者の最大化問題である(3.4)の解を求める。一階の条件から、

$$\gamma_t \equiv \sum_{s=t}^{\infty} \beta^{s-t} \alpha_s = g'\left(a_t^*\right) \tag{3.10}$$

が導き出せる。ここで、γ_tは明らかに減少の経路をたどり、極限においては、$\alpha_s \to 0$から、$\gamma_t \to 0$となる。$g(\cdot)$は凹型の増加関数だから、均衡の経路において努力投入量は減少し続け、最終的に、$t \to \infty$の極限においてゼロの漸近線をたどる。

この結果の解釈は、次に示す通りである。労働者の能力が市場にとって未知数であるかぎり、努力の投入に対して何らかの見返りが得られる。なぜなら、この段階にお

いては、成果を示すことにより、能力に対する市場の認識に影響をもたらすことが可能であるからだ。この時、労働者は努力投入量を必要以上に増やすことによって、潜在的には、彼の能力に対する学習のプロセスを歪めることができる。しかしながら、均衡において労働者がこうした行動をとることはない。なぜなら、市場は均衡において、どのような労働投入量が期待されるかを知っているからだ。言い換えると、労働者は労働市場を騙すことはできない。結局、労働者は均衡における労働投入量を投入し続けることを余儀なくされる。

さらに、労働投入による見返りは、能力に対する不確実性が高いほど大きくなる。初期の情報が少ない頃は、市場は能力ηに対する信念を改める際に、より新しい成果に重きをおく。労働市場が学習行動を重ねるにつれて、ηへの信念は精度を高めていき、最終的には労働者の能力はほとんど完璧に市場に知られることとなる。すると、新しい成果がもたらす情報の価値が無意味なものとなり、最終的には、成果をよりよくしようとするインセンティブが失われ、労働投入量はゼロとなる。

3.1.2 応用モデル

基本モデルの結論では労働市場における評判形成は一時的な価値しかもたないことが示された。それは、労働者の能力ηが労働市場に完全に知られてしまうことが原因であった。この問題を解決するため、Holmstrom（1999）は、ηが一定ではなく、変動するものだという仮定を導入した。この仮定を組み込んだ発展的なモデルを、ここでは便宜上応用モデルと呼び、このモデルによって、シグナリング・インセンティブが永続的に機能することを説明する。

労働者の能力は可変で、$\eta_{t+1}=\eta_t+\delta_t$と表されるとする。$\delta_t$は平均ゼロ、精度$h_\delta$の正規分布に従い、独立である。これに伴い、市場の学習効果もわずかに変化する。記述の簡略化のために$\mu_t=h_t/(h_t+h_\varepsilon)$とすると、平均は基本モデルの(3.6)式と同様に

$$m_{t+1}=\mu_t m_t+(1-\mu_t)z_t \tag{3.11}$$

となる。一方、精度に関する学習を示すh_{t+1}は基本モデルと異なる。$\hat{h}_t$をy_{t+1}を観察する前のη_{t+1}の精度とすると、これは基本モデルの精度と同様に$\hat{h}_t=h_t+h_\varepsilon$である。ここで、$\eta_{t+1}=\eta_t+\delta_t$であるから、$1/h_{t+1}=1/\hat{h}_t+1/h_\delta$が得られ、$h_{t+1}$は次に示す通りとなる。

$$h_{t+1} = \frac{(h_t + h_\varepsilon)h_\delta}{h_t + h_\varepsilon + h_\delta} \tag{3.12}$$

この式によると、h_tは増加していくものの、$t \to \infty$の極限において$h_t \to \infty$とはならない。なぜなら、δの存在が、ηに対して不確実性を与え続けるからだ。無限大に発散する代わりに、h_tは安定状態であるh^*に収束していく。この安定状態は、長年積み重ねられた学習効果が、δの存在によって生じる不確実性を相殺するのに十分となった際に生じるものだ。この安定状態h^*を求めるために、まず安定状態におけるμ_tを求める。簡単な計算により、次のような再帰式が得られる。

$$\mu_{t+1} = \frac{1}{2 + r - \mu_t} \quad , \quad where \quad r \equiv \frac{h_\varepsilon}{h_\delta} = \frac{\sigma_\delta^2}{\sigma_\varepsilon^2} \tag{3.12}$$

安定状態においては$\mu_{t+1} = \mu_t = \mu^*$だから、(3.12)の再帰式を解くと、

$$\mu^* = 1 + \frac{1}{2}r - \sqrt{\frac{1}{4}r^2 + r} \tag{3.13}$$

が得られる。ここで$0 \le \mu^* \le 1$である。もし$r = 0$ならば（即ちεの分散がδの分散に比べて大きい時)、$\mu^* = 1$である。この場合、m_tの学習による更新は滞る。その一方で、$r = 1$の時はその逆が言える。

(3.13)式と$\mu_t = h_t / (h_t + h_\varepsilon)$から、安定状態における精度$h^*$が次のように求まる。

$$h^* = \frac{h_\varepsilon \mu^*}{1 - \mu^*} \tag{3.14}$$

次に、インセンティブの細分化を行う。基本モデルと同様に、最適労働投入量のa_t^*は次の式で与えられる。

$$\gamma_t \equiv (1 - \mu_t) \sum_{s=t+1}^{\infty} \beta^{s-t} \left[\prod_{i=t+1}^{s} \mu_i \right] = g'(a_t^*) \tag{3.15}$$

安定状態においては、$\mu_s = \mu^*$である。これを(3.15)式の中に代入すると、安定状態における労働投入量a^*は

$$\frac{\beta(1-\mu^*)}{1-\mu^*\beta} = g'(a^*) \tag{3.16}$$

を満たす。ここで、効率性の面で最適な労働投入量を$\bar{a}$とする。$\bar{a}$は$g'(\bar{a})=1$によって定義される。(3.16)式の左辺は0と1の間の値を取るから、$a^* \leq \bar{a}$である。もし$\beta=1$であれば、μ^*の値と関係なく、安定状態の労働投入量は効率的なもの、即ち最大値となる。しかしながら、$\beta<1$の場合は、μ^*の値を考慮しなくてはならない。μ^*は、δ_tの分散がε_tより比較的小さい時、1に近づき、安定状態における労働投入量は0に近づいていく。

以上のことをまとめると、安定状態における労働投入量a^*は常に最適投入量$\bar{a}$を下回り、$\beta=1$の時$\bar{a}$と一致する。そして、βが大きいほど、σ_δ^2が大きいほど、σ_ε^2が小さいほど、a^*は$\bar{a}$に近づいていく。これを直感的に解釈すると、能力がより確率的に変動する場合や、成果の観察がより精密に行える場合、より成果がみられやすい場合において、シグナリング・インセンティブはより強く効果を発揮する、ということが言える。

3.2 実証分析

前節のシグナリング・インセンティブの理論を元にして、オープンソース・ソフトウェアの開発に従事するコミュニティのモチベーションの源泉がどのようなものであるのかを定量的に分析する。Lerner and Tirole (2002) は、Holmstrom (1999) において示されたシグナリング・インセンティブの強い環境を読み替え、OSSにおいては、自身のパフォーマンスが、より他の開発者にみられやすい場合シグナリング・インセンティブが効果を発揮すると指摘した。プログラマーの能力は、プログラマーでない者にはわかりづらく、精密な成果の観察はできないであろうから、この指摘は妥当なものであるといえるだろう。本実証分析では、この指摘が妥当なものであるのか検証しつつ、様々な側面からコミュニティの動機の源泉について独自の手法で分析する。

3.2.1 データセット

データセットは、Filehippo.netというウェブサイトをベースに取得した。このサイトは、人気のある無料ソフトウェアの過去数年間にリリースされた全てのバージョンを網羅的に掲載しているサイトである。このサイトに掲載されているソフトウェアの

うち、OSS であるもの 64 個をサンプルとした。

被説明変数はコミュニティの努力水準である。本稿では、その代理変数としてソフトウェアのバージョン更新（バージョンアップ）頻度を用いる。Fershtman and Gandal (2007) では、努力水準の代理変数として、ソースコードの行数(SLOC)を用いていた。SLOCは企業のソフトウェア開発の現場においても、工数の確認に用いられるほど一般的な指標であるが、これを使うには長期にわたる定点観測が必要不可欠なため、データの入手可能性を考慮し、本稿ではバージョンアップの頻度を用いることとした。一般的に、ソフトウェアのバージョンアップには二つの種類がある。一つは、新たな機能を追加し、全体の仕様を更新するメジャーアップデート、もう一つが、細かな互換性の調整や、小さなバグの除去などを行うマイナーアップデートである。[18]一般的にはマイナーアップデートがソフトウェアのバージョンアップの大部分を占めており、もちろんOSSの開発においても同様のことが言える。今回の実証分析においては、このマイナーアップデートとメジャーアップデートの両方を纏めてバージョンアップとして取り扱っている。メジャーアップデートの規模は各ソフトウェアによりまちまちであるが、マイナーアップデートには大きな差はないから、コミュニティの努力水準の高いソフトウェアであるほどバージョンアップ頻度は高くなるといえる。バージョンアップ頻度は、Filehippo.comに掲載されている各ソフトウェアの過去のバージョンの総数を足し上げ、その値を、最も古いバージョンのリリース月から最も新しいバージョンのリリース月までの月数で除したものを用いる。端的に言うと、バージョンアップ回数を、その期間で割ったものである。

続いて、説明変数を紹介する。説明変数は Filehippo.com から得られるものをベースに、64 個のソフトウェアの一つ一つを調査することによって得た。以下、その 1 つずつの中身や、数値化手法について紹介する。

- Developers：開発者向けソフトウェアであることを示すダミー変数である。Filehippo.comではソフトウェアが機能により 16 種類に分類されている。[19]その機能分類の中で、ソフトウェア開発者向けの分類に該当する場合 1 とした。具体的には、「ファイル転送」、「開発者ツール」、「システム最適化」、「ネットワークと

18) たとえば、Microsoft 社の Internet Explorer の場合、Internet Explorer8 から Internet Explorer9 への更新はメジャーアップデートであり、8 から 8.1 や 8.11 などへの更新はマイナーアップデートとされる。

19) ブラウザとプラグイン、ファイル共有、メッセージとチャット、ファイル転送、オフィスとニュース、開発者ツール、マルウェア対策、ファイアウォールとセキュリティ、システム最適化、圧縮とバックアップ、ネットワークと管理、オーディオとビデオ、CD・DVD ツール、デスクトップ、写真と画像、ドライバの 16 種。

管理」の四種類を開発者向けソフトウェアとした。この振り分けの際の判断基準として、SourceForge.net（第 2 章参照)を用いた。Filehippo.comに掲載されたソフトウェアには、SourceForge.netに登録されているものも少なくなかったので、対象とするユーザー層(Intended Audience)が開発者(Developers)と表示されているソフトウェアの多い 4 つの分野を選択した。今回の分析におけるメインの説明変数であり、もしシグナリング・インセンティブが働いているのであれば、係数が正となるはずである。

- Firm licensor：ライセンサーが企業や営利団体であることを示すダミー変数。Lener and Tirole (2002) においても指摘されていた、ライセンサーの信用性に関する評価と、社員の OSS への参加を制御するために説明変数に組み込んだ。
- License restrictiveness：ライセンス制約のこと。第 2 章で示したように、ライセンス制約はコミュニティの効用を左右するものであるから、当然これも制御しなくてはならない。数値化は表 2-1 に従って行った。第 2 章の理論と実証結果に依れば、制約の厳しいライセンスをコミュニティは好むはずなので、係数は正となる。
- Cross platform：クロスプラットフォームであるか否か。クロスプラットフォームとは、複数のプラットフォームで動作するように仕様が策定されたプログラムのことである。クロスプラットフォームであれば、環境に関わらずより多くの開発者にソフトウェアを発信できるから、係数は正となるはずである。

3.2.2 推計結果

関数形を考慮し、被説明変数は対数をとり、回帰を行った。次ページの表 3-1 が、推計結果である。結果は、概ね理論に整合的なものとなった。まず、メインの変数である Developers の係数は正で有意であり、理論に整合的である。このことから、他の開発者に自身の能力のシグナルを発することが、OSS 開発の一つのモチベーションとなっていることが確認できた。他の変数についても確認すると、Firm licensor が正で有意となった。この原因としては、企業がライセンサーである OSS は、社員自身の貢献が大きく寄与していることや、企業がライセンサーであることにより、より広く認知されること等が考えられる。また、License restrictiveness は正で有意であり理論に整合的となったが、Cross platform は負で有意となり理論と非整合的となった。この原因としては、対応しているプラットフォームが多い分、バグの修正作業が長期に亘りやすい可能性などが考えられる。また、頑健性の確認として、不均一分散

の確認(BP test)と多重共線性の確認をしたが、どちらも異常はみられなかった。これらの検定の詳細は表 3-1 の末尾に掲載しているので、そちらを参照してもらいたい。

表 3-1 推計結果

Variables	Developers	Firm licensor	License restrictiveness	Cross platform
Coefficient	0.393	1.24	0.135	-0.672
(t-Value)	(1.71)*	(4.67)***	(1.93)**	(-1.86)*
p-Value = 0.000	Number of observation = 64		Adj R-squared = 0.459	
Breusch-Pagan test	chi2(4) = 3.29		Prob > chi2 = 0.5111	
VIF(test for multicollinearity)	Max = 1.18 (Firm licensor) Mean = 1.10			

備考)有意水準は、* p<0.1, ** p<0.05, *** p<0.01 で示している。

3.2.3 考察

シグナリング・インセンティブの実証分析を行い、結果としては、OSS 開発のモチベーションの一つがシグナリング・インセンティブであることを示すことができた。しかしながら、未だ改善の余地は多くある。まず、被説明変数の努力水準は、データの入手可能性から今回はバージョンアップ頻度を用いたが、前述のとおり、より一般的な指標である SLOC で試す必要がある。また、Developers の分類も経験的な判断に依らざるを得なかったため、より客観性の高いデータの方が望ましい。

もう一つ今回の実証分析を通じて言及しておきたいのが、実証分析をするにあたっての OSS という分野の優位性である。契約理論のような、動機を主題に取り扱った理論は、一般的に定量的な分析は難しいとされる。しかしながら、様々なものがインターネットを介して可視化されているという特性から、OSS は、比較的データの入手がしやすく、実証分析を行いやすいと言える。当然、OSS には様々な特殊性が存在するから、それを考慮に入れなければならないこともまた事実だが、その便益を考えれば、充分一考に値するだろう。

第 4 章　結論

本稿ではオープンソース・ソフトウェアの経済学上の二つの特性について、理論と実証の両面から分析を行った。第 2 章ではライセンシング問題について、Lerner and Tirole (2005) と Wang (2008) の理論を比較した上で、定量的手法によって分析を行った。両理論は、OSS の商業的価値とライセンスの関係において意見を異にしていたが、実証分析の結果は、Wang (2008) と整合的となった。その帰結は、商業的価値が高いソフトウェアと低いソフトウェアは、制約の厳しいライセンスを選択し、一方、商業的価値が中間的なソフトウェアは、制約のゆるいライセンスを選択するというものである。こうした結果となったのは、企業の OSS への介入がより盛んになった近年においては、企業の存在に重きをおいた Wang (2008) の理論の方が、商業的価値に関する視点においては、より現実に沿ったからだと考えられる。第 3 章では、オープンソース・コミュニティの動機について分析した。Holmstrom (1999) の理論を紹介することで、シグナリング・インセンティブがコミュニティの動機の一つであるという仮説を示し、これを定量的に分析した。その結果、この仮説は支持された。

参考文献

秋本芳伸・岡田泰子（2004),「オープンソースを理解する」株式会社ディー・アート.

野田哲夫・丹生晃隆（2009),「オープンソース・ソフトウェアの開発モチベーションと労働時間に関する考察」島根大学法文学部紀要法経学科編『経済科学論集』35 号, pp. 71-93.

吉田智子（2007),「オープンソースの逆襲」出版文化社.

Fama, E., (1980), "Agency Problems and the Theory of the Firm," *Journal of Political Economy,* **88**, 288-307.

Fershtman, C. and N. Gandal, (2007), "Open Source Software: Instrict Motivation and Restrictive Licensing," *International Economics and Economic Policy*, **4**, 209-225.

Fershtman, C. and N. Gandal, (2011), "A Brief Survey of the Economics of Open Source Software," in: L. E. Blume and S. N. Durlauf (eds.),' *New Palgrave Dictionary of Economics*,' Online Edition, Palgrave Macmillan.

Holmstrom, B., (1999), "Managerial Incentive Problems: A Dynamic Perspective," *Review of Economics Studies*, **66**, 169-182.

Lerner, J. and J. Tirole, (2002), “Some Simple Economics of Open Source,” *Journal of Industrial Economics*, **50**, 197-234.

Lerner, J. and J. Tirole, (2005), "The Scope of Open Source Licensing," *Journal of Law, Economics, and Organization*, **21**, 20-56.

Lerner, J., P. A. Pathak and J. Tirole, (2006), “The Dynamics of Open-Source Contributors,” *American Economic Review*, **96**, 114-118.

Schiff, Aaron., (2002), "The Economics of Open Source Software; A Survey of the Early Literature," *Review of Network Economics,* **1**, 66-74.

Wang, Y., (2008), “Economic Analysis of Open Source Software,” *PhD thesis, Aarhus University Economics and Management department.*

オブジェクト指向スクリプト言語 Ruby ホームページ　http://www.ruby-lang.org/ja

Filehippo　http://www.filehippo.com/jp

Open Source Initiative ホームページ　http://www.opensource.org

SourceForge.net　http://sourceforge.net

あとがき

一見不可解な現実の事象を、経済学を用いて分析することが私の卒業論文における最大の主題であったことは、はしがきにも記した。それに次ぐ指標としてテーマ選びの際に意識したのが、未開の分野である、ということだ。これも理由は単純で、私自身が新しいものが好きだからだ。実際にオープンソース・ソフトウェアの研究を進めてみると、やはり性に合ったのか、未開の分野で組み上げられていく理論は刺激的で、知的好奇心をくすぐられるものであった。それと同時に感じたのは、長年に亘って積み重ねられてきた経済学というツールの懐の広さだ。未開の分野とはいえ、既存のモデルを組み合わせたり、一部を改変したりすることで全く新たな事象を分析することができる。こうして枝葉をつけて発展していく経済学の形態は、よくよく考えると、オープンソース・ソフトウェアのそれに少し似ている。

もちろん、未開の分野というのは、いいことばかりではない。均されていない土地を進むことは、それ相応の苦労も伴う。そうした苦労を乗り越えることが出来たのは、偏に周囲に助けられてきたからだ。まず、研究会の同期に感謝を述べたい。共に卒業論文という目標に向かっていく仲間として、時に励まされ、それと同じくらいに、良きライバルとして切磋琢磨し合えた。そうした環境であったから、二年間の研究会の活動は非常に濃く刺激的なものとなったし、夜中の九時までゼミが続いた日も、充実した時間を送ることができた。掛け値なしに、この同期とともに二年間勉強できてよかったと思う。そして、二年間熱心に指導をしてくださった石橋孝次先生に感謝の意を表したい。特に夏期休暇中には、休み中であるにも関わらず、週に一回のペースでしつこく教授棟に通った私を、一切邪険に扱うことなく、相談に乗ってくださった。当時は、論文の方向性もまだよくまとまっておらず、手探りで先行研究を読み漁っていた時期であったから、先生の助言には、非常に助けられた。

OSS の発展形態は、今やソフトウェアの枠組みを越えて様々な分野に飛び火している。インターネット上の動画サイトでは、毎日全く新しいコンテンツが生み出されるのと同じくらい、優れた作品を改変したものが誕生しているし、誰もが編集できる百科事典サイトなども、まさに「オープンソース」的手法によって発展してきたものだ。「オープンソース」という考え方の可能性は、未だ計り知れない。次に生まれてくる「オープンソース」的な現象は、全く思いもよらないものかもしれないし、想像通りのありふれたものに過ぎないかもしれない。いずれにせよ、今この時も何処かで生まれ始めている新たな「オープンソース」に思いを馳せつつ、筆を置くこととする。