オープンソースの存在意義

平成 16 年度　卒業論文

日本大学　法学部　政治経済学科 4 年

0120335　田中　裕

はじめに

近年、オープンソースソフトウェア及びフリーソフトウェアという言葉を聞く機会が増えてきている。有名なところでは Linux、+Lhaca などである。しかし、これらがどういった性質・機能を持っているのか、またどのような歴史的背景を持って生まれたのかを知っている人はそう多くはないだろう。

しかし、オープンソースソフトウェアやフリーソフトウェア化されている技術は今日、我々の生活とは切っても切れない関係にまで浸透している。携帯電話や冷蔵庫、電子レンジといった各種家電製品である。これら多くの製品には TRON という、1984 年に東京大学の坂村博士によって生み出されたオープンソースが利用されている。また、携帯電話も 2003 年の 12 月、NTT DoCoMo が 2004 年秋に発売する FOMA シリーズの OS に Linux を採用すると発表した。

このように、利用者の知らない間にオープンソースソフトウェアやフリーソフトウェアは世の中に浸透してきているのである。このような書き方をするとこれらの技術、また考え方は近年になって生まれたものであると考えるかもしれない。確かに“ オープンソース ”という言葉ができたのは 1998 年のことである。しかし、勘違いしてはならない。これらの基本的な技術・考えは数十年前のコンピュータ創成期から存在していた。いや、むしろ元をただせば知識・環境の共有をするためにソースコードの公開は基本中の基本であり、世界中の研究者たちは公開されたソースコードをネットワークを介して手に入れ、それを改良、もしくはそのソースコードを元にして新たなソフトウェアを開発していた。つまり、オープンソースという考えはコンピュータが今日のような発展を迎えるにあたって非常に重要なファクタであった。

この論文では上記のように古くからあった考えが近年になってから急速に浸透してきた理由、また商用ソフトウェアと比べてどのような役回りを持つのか、という点に焦点を当てて考察を進めていこうと思う。

目　次

はじめに・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・p1

第一章　オープンソースとは何か・・・・・・・・・・・・・・・・・・・・・・・・・p3
　　　　第一節　オープンソースとは？・・・・・・・・・・・・・・・・・・・・・・p3
　　　　第二節　オープンソースを定義する・・・・・・・・・・・・・・・・・・・p4

第二章　ソフトウェアの知的所有権と分類・・・・・・・・・・・・・・・・・・・・・p8
　　　　第一節　ソフトウェアと知的所有権・・・・・・・・・・・・・・・・・・・p8
　　　　第二節　ソフトウェアの分類・・・・・・・・・・・・・・・・・・・・・・p9

第三章　オープンソースという概念の誕生・・・・・・・・・・・・・・・・・・・・・p12
　　　　第一節　オープンソースソフトウェアの原点：フリーソフトウェア・・p12
　　　　第二節　オープンソースが生まれた理由・・・・・・・・・・・・・・・・p14
　　　　第三節　オープンソースとハッカの関係・・・・・・・・・・・・・・・・p16

第四章　オープンソースのメリット・・・・・・・・・・・・・・・・・・・・・・・・p20
　　　　第一節　開発者側のメリット・・・・・・・・・・・・・・・・・・・・・・p20
　　　　第二節　ユーザ側のメリット・・・・・・・・・・・・・・・・・・・・・・p22

第五章　オープンソースの現状・・・・・・・・・・・・・・・・・・・・・・・・・・p24
　　　　第一節　オープンソースソフトウェアの種類・・・・・・・・・・・・・p24
　　　　第二節　商用ソフトウェアとの比較・・・・・・・・・・・・・・・・・・p25

第六章　オープンソースの抱える課題・・・・・・・・・・・・・・・・・・・・・・・p29
　　　　第一節　オープンソースプロジェクトの現状と課題・・・・・・・・・p29
　　　　第二節　不正コード混入問題と知的所有権問題・・・・・・・・・・・p30
　　　　第三節　商用ソフトウェアのオープンソース化・・・・・・・・・・・p31

おわりに・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・p34

第一章　オープンソースとは何か

第一節　オープンソースとは？

オープンソースソフトウェア、と聞いてその特徴、性質を正しく捉え、他人に対して詳細な説明をすることができる人はそう多くないだろう。だが、それは無理もない。このオープンソースソフトウェアという言葉がこの世に誕生したのはわずか数年前、1998年のことである。しかし、名前の響きこそ新しいものだが、オープンソースソフトウェアの持つ性質、概念といったものは遥か昔、それこそコンピュータ創世記から脈々と流れてきているものである。字面を見れば「ソースがオープンになっているソフトウェア」であるということは漠然と理解できるだろうが、その先の段階に理解が進む人は稀であろう。

では、そもそもソースとは何か。これはプログラミング言語と呼ばれる高水準の言葉で書かれている。これはあらゆるソフトウェアを作るにあたり、その設計図とでも言うべきものである。ソフトウェアを作るにはそのプログラムを作らなければならない。しかし、人間とコンピュータでは理解できる言語が異なり、例えば英語や日本語でプログラムを入力してもそれをコンピュータに理解させることは不可能である。では、どうすればいいのかというと、ここでプログラミング言語が使われる。プログラミング言語とは人間の理解できる言語とコンピュータが理解できる言語の中間に位置する言葉であり、主なものとして C 言語や JavaScript といったものがある。プログラマはこのプログラミング言語を使いプログラムを作成する。この段階ででき上がる物がソースコードと呼ばれるものである。このソースコードはプログラミング言語を読むことができない人には何が書いてあるのか分からないだろうが、その言語を学んだ人であれば手間と時間をかければ解読することができる。つまり、このソースコードは一見コンピュータに読ませるための言語に見えるが、まだ人間向けの言語であり、コンピュータがそれを理解することはできない。そこで、このソースコードをコンパイルすることでやっとプログラムとして完成するのである。

コンパイルとは翻訳することを指し、人間向けの言語で書かれたソースコードをコンピュータが理解できる言語に変換することである。このコンパイルを経てソースコードはバイナリコード呼ばれるものに変化する。バイナリコードとは実行プログラムのことである。ソフトウェアを購入するなりダウンロードするなりして手に入れた場合、大概のソフトにはセットアッププログラムがあり、それを起動することでソフトウェアはコンピュータにインストールされる。この際に使われるセットアッププログラム、これがバイナリコードである。このバイナリコードは通常、「0」と「1」の羅列、つまり２進法により記述されており、この段階になりやっとコンピュータに理解できる状態になる。

これらを踏まえた上でオープンソースソフトウェアの説明をすると、オープンソースソフトウェアとは「設計図たるソースコードが公開されたソフトウェア」ということに

なる。ここで注目する点はソースコードを公開する、という点である。前述のとおり、ソースコードとはソフトウェアの設計図にあたる。このソースコードを手に入れた人は技術さえあればそのコードを改変することで元の機能を持った類似製品を低コスト、短時間で作成したり、またはそれを元にしたまったく新しいソフトウェアを作成したりすることができる。つまり、このソースコードは開発者側から見れば本来死守すべきものなのである。

では、なぜこのソースコードを公開するのだろうか。詳しい説明は第四章で述べるためここでは避けるが、ソースコードを公開することには開発者側、ユーザ側双方にメリットがある。ユーザ側のメリットをあげるとおおよそ三つのメリットがある。プログラム保守、プログラム移植、機能追加の三点である。第一にプログラム保守とは、バグ等を発見した際、開発者の修正を待たずに自身の手でその修正をすることができる。第二にプログラム移植とは同じソフトウェアを他の OS 環境にあるコンピュータに移植する場合、例えば Windows 環境から Mac 環境へ移植する場合に通常ならそれぞれの機種に対応しているソフトウェアを別途購入しなければいけないが、ソースコードがあれば必要な部分のみ訂正すれば新たに購入する必要はない。第三に機能追加はユーザによる自由なカスタマイズを可能にするためである。

第二節　オープンソースを定義する

オープンソースという言葉を最も広義に使う場合、これはソースコードの公開の有無のみを指し、それ以外の条件付けはされていない。逆に、最も狭義の意味で使われる場合にはオープンソースの定義が正しく守られるように管理しているオープンソースイニシアティブ(Open Source Initiative 以下、文中では OSI と表記)[注1)]による定義(Open Source Definition 以下、文中では OSD と表記）[注2)] による「OSI 認定マーク」の付いたソフトウェア、または OSI 認定のライセンスを使用しているソフトウェアのことを言う。また、これ以外にも OSI の認定は受けていないがこれら「オープンソースの定義」に準拠しているソフトウェアもある。

このように、様々な解釈のされているオープンソースだが、本論文中で「オープンソース」、または「オープンソースソフトウェア」という言葉を使った際にはこの最も狭義の意味で使うものとする。

では、OSI による OSD とはどのようなものなのか。これは現在では全十か条の定義となっており、

1. Free Redistribution（自由な再頒布）
2. Source Code（ソースコードの開示）
3. Derived Works（派生著作物に関するライセンス）
4. Integrity of The Author's Source Code（著作者のソースコードの完全性）
5. No Discrimination Against persons or Groups（個人やグループに対する差別禁

止）
6. No Discrimination Against Fields of Endeavor（利用分野に対する差別禁止）
7. Distribution of License（ライセンスの継承）
8. License Must Not Be Specific to a Product（特定製品に特化したライセンスの禁止）
9. License Must Not Restrict Other Software（他のソフトウェアを制限するライセンスの禁止）
10. License Must Be Technology-Neutral（テクノロジーニュートラルなライセンス）

となっている。以下では上記の各項目について説明をしていく。

1. Free Redistribution（自由な再頒布）

この項目では完全に自由な再頒布を許諾することで、オープンソースソフトウェアの普及を目指している。そのため、販売することも許諾しなければならないが、手元を離れたオープンソースソフトウェアに対して更なる使用料や報酬を求めてはならない。

2. Source Code（ソースコードの開示）

OSI ライセンスではすべての製品に対して適切な価格での提供を定めている。これに加え、オープンソースソフトウェアを頒布する際には必ずソースコードを含まなければならないということと、ソースコードでの頒布の許可を認めることが必要である。また、この他にもソースコードを意図的にわかりにくくすることは許されていない。これはオープンソースソフトウェアの普及という考えが背景にある。改変を困難にすることでこの普及が妨げられることを防ぐためである。

3. Derived Works（派生著作物に関するライセンス）

OSI ライセンスでは以下の項目を遵守することを前提にソフトウェアの改変と派生ソフトウェアの再頒布を認めている。それはソフトウェアに対する自由な改変の許諾、及び派生ソフトウェアに対してもとのソフトウェアと同条件での頒布を許諾しなければならない、というものである。これはソースコードの公開のみではオープンソースとは言えず、自由改変、自由再頒布を認めなければならない、という前提があるためである。

4. Integrity of The Author's Source Code（著作者のソースコードの完全性）

OSI ライセンスでは著作者はオリジナルのソースコードと派生版との見分けをつけられるように、「ソースコード+パッチファイル」という形で頒布を認める場合はオリジナルのソースコードと派生版との相違点を明確にし、製作者の名前を記述し、

元のソフトウェアが何かということを分からせなければならない、というものである。これはオリジナルと同名のソフトウェアが出回ると区別がつきにくくなってしまうからである。これにより改悪版などが作られた場合、著作者の評判を落とすことを防ぐことができる。

5. No Discrimination Against persons or Groups（個人やグループに対する差別禁止）

国家によっては外国為替法によりソフトウェアの輸出制限を行っている場合があるが、OSI ライセンスではそのような制限をかけることは許されない。これはオープンソースの進化のためには多くの人が触れる必要があり、それを締め出すことは禁止されている。

6. No Discrimination Against Fields of Endeavor（利用分野に対する差別禁止）

オープンソースコミュニティでは学術・商業利用のためのユーザの参加も奨励している。このため、ライセンスで使用分野に制限をかけることを禁止している。

7. Distribution of License（ライセンスの継承）

OSI ライセンスによるソフトウェアに付随する権利は再頒布された人すべてに対して適用されなければならない。

8. License Must Not Be Specific to a Product（特定製品に特化したライセンスの禁止）

ここでは特定の頒布物に対して特化した特長をつけることを禁止している。これは Linux ベースのマシンではオープンソースになるが、その他の OS マシンでは使えない、ということを防ぐためである。これは、オープンソースはあらゆる環境においてオープンソースでなければならないからである。

9. License Must Not Restrict Other Software（他のソフトウェアを制限するライセンスの禁止）

これはオープンソースとその他ソフトウェアを同時に利用した場合、その他ソフトウェアの機能を制限するようなことがあってはならない、ということである。

10. License Must Be Technology-Neutral（テクノロジーニュートラルなライセンス）

OSI ライセンスではいかなる項目もその技術に対して制限を設けることはできない。これは FTP ダウンロードや CD-ROM での頒布、ミラーサイトからの頒布といったクリック・ラッピングとの相性の悪い頒布方法あり、またこのライセンス方法を共用することで再頒布を足止めしてしまう結果を生む危惧が発生してしまう。そ

のため、OSI に準拠するライセンスは特定の技術に固執することなく、技術的に中立なライセンスを保持する必要があるのである。

さて、ここまで OSD の十項目のライセンスを提示してきたが、ここで注目すべき点が一つある。これは、Linux カーネルなどを筆頭に一般に思われている「オープンソース」像を崩すものである。それは、この OSD は「頒布に際してそのソフトウェアは無料でなければならない」という項目を持たない、という点である。オープンソースソフトウェアの定義の中には有償か無償かという考えは含まれてはいないのである。

事実、Linux カーネルをはじめとした多くのオープンソースソフトウェアは無償で頒布されている。しかし、これは「無償で頒布しなければならないもの」だからではなく、頒布元や再頒布元の判断で無償での頒布を行う、と決めているためである。つまり、逆に言えば頒布元、再頒布元が「有償での頒布を行う」と判断を下せば有償頒布を行うことができ、また、今現在では無償頒布されているソフトウェアが将来的に有償になる可能性も常に秘めているのである。

また、この Linux カーネルにしてもそのベースとなるオリジナルのものは無償頒布されているが、Red Hat 社をはじめとする多くの企業がディストリビュータとして有償による頒布を行っている。これらの企業から頒布を受ける場合には Windows OS とさして変わらない料金を提示されることになる。これは、そのディストリビュータはあくまでもビジネスとして頒布を行い、本来は自らで行う必要があるバージョンアップや修正パッチの確認を企業側が行い、それに付随して各種ソフトウェアとの組み合わせ販売や独自のサポートを行うことで付加価値をつけ、有償での頒布を行える道を作ったのである。

以上のことから、ユーザはオープンソースソフトウェアを無償で手に入れる確証を得ているわけではない、ということになるのが分かるだろう。

参考文献／Web サイト

注 1）　オープンソースイニシアティブ（Open Source Initiative）
http://www.opensource.org/

注 2）　オープンソースの定義（Open Source Definition）
http://www.opensource.org/docs/definition.php

第二章　ソフトウェアの知的所有権と分類

第一節　ソフトウェアと知的所有権

前述のとおり、オープンソースソフトウェアとはソースコードを公開したソフトウェアである。そして、このオープンソースソフトウェアと商用ソフトウェアとの相違を作り出している権利が「知的所有権」である。ソフトウェアに知的所有権を認めることは開発者の知的財産を保護するために必要なことであるが、同時にソフトウェアの普及を妨げる大きな要因にもなる。では、この知的所有権をソフトウェアに認めるようになった時期、理由、つまり「オープン」を信じる人たちにとっての忌まわしき経緯はどのようなものなのだろうか。

1970 年代後半、コンピュータは多くの国々に浸透をし始め、それに付随する形でソフトウェア産業も発展を見せていた。しかし、それは同時に開発したソフトウェアの盗作やコピーを多く生み出す原因にもなっていた。そのため企業や開発者は自らが作り出したソフトウェアを保護する法律を求めたのである。ここで法的保護の対象となる「ソフトウェア」とは単にプログラムのみならず、そのマニュアルや開発ノウハウといった開発環境周辺にも及んでいる。

そしてまず世界に先駆けてアメリカで認められた権利が著作権である。ソフトウェアも芸術や音楽と同じようにその創作性を認められたのである。これが 1980 年のことであり、これに続く形でドイツやフランス、そして日本も 1985 年にソフトウェアを著作物と認めるように法改正が行われた。

上記の 1985 年に改正された著作権法では「電気計算機を機能させて一の結果を得ることができるようにする指令を組み合わせたものとして表現したものをいう」と定義し、プログラムが著作物のなかに入るようになった注 1)。しかし、近年になり著作権よりも強く保護できる特許権により開発側の権利を守ろうという動きが現れた。従来の著作権ではソフトウェアのコピーを防ぐことはできてもその内容に関する「アイディア」を守ることは不可能であった。これに対して特許権はいわゆる「発明品」を保護するものであり、これがソフトウェアに適用されることによりそのアイディアを長期間独占することができるようになったのである。

プログラムの複製に関する国内の判例をひとつ挙げてみると、重要判例として「スペース・インベーダー・パート２」注 2 というゲームに関するものがある。これは 1982 年に行われた裁判であり、原告（株式会社タイトー）が製造・販売・賃貸を行っていた同マイクロコンピュータ内臓テレビゲームについての裁判である。被告（株式会社アイ・エヌ・ジ・エンタープライゼス）はスペース・インベーダー・パート２の基盤を取り外し、別のゲーム機に取り付け、同じゲームができるように改造し、これを販売していた。これに対し、原告側が著作権侵害を理由に損害賠償請求を起こしたのである。

この裁判における判旨では「本件プログラムは作成者独自の学術的思想の創作的表現

であり、著作権法上保護される著作物に当たる」とし、原告側の主張を全面的に支持した。この判決を受け、それまでは何の制約も受けていなかった他人のプログラムを使用すること関して、「他人の製作したプログラムは著作物に当たる」という考えがコンピュータ業界に浸透することになった。そして国内ではその 3 年後、1985 年に正式な法改正が行われたのである。

このように、「プログラムは著作物である」という考えは近年になってから生まれたものである。この考えが浸透したことにより企業側は他社に自らの技術を無断で利用されることはなくなったが、同時に他社が開発した技術を利用できないことで業界全体としての技術発展はその歩みを大きく遅らせる結果となった。似たような機能を持つ製品を開発するにも一から開発を始めなければならなくなり、これは結果として製品コストを引き上げ、最終的にはユーザの負担を大きくすることになった。

現在、そして今後のコンピュータ業界にとってもっとも必要なことは各ソフトウェアの技術を守りつつ、ユーザの視点で開発を進めていくことである。

第二節　ソフトウェアの分類

現在、世の中には数多くのソフトウェアがあふれている。例えば Microsoft 社の文書作成ソフトである「Microsoft Word」や同社の表計算ソフト「Microsoft Excel」、Apple Computer 社の音楽ソフトである「QuickTime」などである。これらは企業ごと、利用目的によって実に様々である。

しかし、一口に「ソフトウェア」といってもそれらはそれぞれの性質によっていくつかのグループに分けることができる。有名なものでは Microsoft 社の文書作成ソフトや表計算ソフトが入れられている「Microsoft Office」が挙げられる。このような企業が商用目的で製作・販売を行っているソフトウェアを「商用ソフトウェア」、また個人によって開発された圧縮・解凍ソフトである「+Lhaca」といった無償であり、バイナリコードのみを利用可能なソフトウェアを「フリーウェア」、同条件でいくらかの金銭を支払う必要があるものを「シェアウェア」と呼ぶ。そして、基本的に無償で入手することができ、バイナリコードのみならず、ソースコードを開示して自由な改変や再頒布を許可しているものを「オープンソースソフトウェア」、またオープンソースソフトウェアの元となった「フリーソフトウェア」というものがある。では、これらがそれぞれどのようなものかを説明していく。

まず、フリーウェアとは文字通り「無償（Free）で利用可能なソフトウェア」である。フリーウェアは多くの場合は開発者が自らのために製作したソフトウェアを他人にも利用してもらうために公開している場合が多い。ユーザはソフトウェアの利用のみを認められており、公開されるものはバイナリコードのみである。そのため、ソフトウェアにバグや欠陥が見つかった場合は開発者がその修正を行うが、開発者にその意思がなければ更新されることはない。また、再頒布も自由に行うことはできず、開発者に許可を取る必要がある。

つぎに、シェアウェアである。これは「開発費の一部を負担する（Share）ことで利用可能なソフトウェア」であり、ユーザは開発者に対していくらかの料金を支払うことでそのソフトウェアを利用することができるようになる。その性質はフリーウェアと同様であり、改変や修正は開発者が行う。しかし、フリーウェアと同様の性質を持っているが故に開発者にその義務はない。

近年ではこのシェアウェアの価格が上がり、逆に商用ソフトウェアの価格が下がるという傾向が見られるため、シェアウェアの意義が薄れつつある。似たような金額を支払うのであればサポートのしっかりした商用ソフトウェアのほうがユーザにとってはメリットが大きいからである。つまり、シェアウェアの課題は開発者・ユーザ相補にメリットがあるような環境を作ることにある。これができなければ近い未来、シェアウェアはその姿を消すことになるかもしれない。

フリーウェアは無償で、シェアウェアは有償で頒布されるソフトウェアである。オープンソースソフトウェアもまた無償で提供される場合が多いため、フリーウェアと混同されがちである。しかし、これらには大きな相違点を三つ挙げることができる。以下にその相違点を挙げる。

まず一つ目にソースコードの公開・非公開である。ソースコードの公開がなされていないフリーウェアやシェアウェアを利用するユーザに認められているのはあくまでも使用権のみであるのに対し、オープンソースソフトウェアのユーザは開発者と同様にあらゆる権利を与えられることになる。このため、ユーザは自由にそのソフトウェアを改変し、使用することができる。

二つ目は改変の自由の有無である。これは著作権の中にある「同一性保持権」に関するものであり、著作者以外のものは無断でソフトウェアを改変することはできない。このため、フリーウェアやシェアウェアのユーザはそのソフトウェアに対して改変を加えることは認められていない。しかし、オープンソースソフトウェアはソースコードが公開されているため、ユーザは自分の好みに合わせて自由な改変を加えることが可能である。

そして三つ目が自由な再頒布権の有無である。著作物の複製を行うにはその著作者の許可を得る必要がある。フリーウェアはその一部にはユーザによる複製や再頒布を認めているものもあるが、シェアウェアの多くは商用ソフトウェアと同様にその複製を認めていない。一方、オープンソースソフトウェアの場合は著作者と同様にユーザの判断で自由に複製・再頒布を行うことが可能である。

これらを踏まえた上でオープンソースソフトウェアを定義し直すと、「ユーザにあらゆる権利が開発者同様に与えられているソフトウェア」となる。

では、このオープンソースソフトウェアという概念・考え方が生まれるにあたって元となった「フリーソフトウェア」とはどういったものなのか。これを次章で述べようと思う。

参考文献／Web サイト

注1）　著作権　『第 1 章総則・第 1 節通則・第 2 条定義　10－2・プログラム』　より

注 2）　「スペース・インベーダー・パート 2 事件」
『コンピュータ・プログラム・スペース・インベーダー・パート 2 事件』
著作権判例百選〔第二版〕「別冊ジュリスト No.128」

第三章　オープンソースという概念の誕生

第一節　オープンソースソフトウェアの原点：フリーソフトウェア

「フリー」には「自由」と「無償」という二つの意味がある。フリーウェアの持つフリーとは多くの場合は無償のことを指す。名称が似ているため混同されがちであるが、フリーソフトウェアの持つフリーの意味は自由であり、「自由なソフトウェア」という意味である。オープンソースソフトウェアはこのフリーソフトウェアを起源とし、独自の発展をしてきた。では、このフリーソフトウェアとはどのようなものなのか、また、オープンソースソフトウェアとの違いは何なのだろうか。

そもそも、1970 年代まではソフトウェアに関する法制度が進んでおらず、大学や研究所間でのソフトウェアの共有は当たり前のことだった。特にコンピュータ同士がネットワークで繋がれるようになるとプログラムの共有はますます進み、これが起因となってインターネットが普及した、といっても過言ではないだろう。このような時期においてはそもそもフリーソフトウェアという考えすら生まれていなかったのである。

しかし、1980 年代に入ると状況は変化の兆しを見せた。ソフトウェアに関する法整備が進み、プログラムの共有が認められなくなったからである。ソフトウェアの専有化が行われ、ソースコードの公開の自由や改変の自由、頒布の自由がなくなった。これは、それまで当たり前のようにソフトウェアを共有していたハッカにとっては大きな痛手であった。このため、再びソフトウェアの共有を復活させるために生まれたのがフリーソフトウェアという考えである。

このフリーソフトウェアという考えは Richard M Stallman の「私有ソフトウェア制度（ユーザにソフトウェアの共有や変更を認めない制度）が反社会的で、非倫理的で、単純に間違っている」注 1) というリチャード・ストールマンの思想を基盤としている。また、このフリーソフトウェアという思想を打ち出したのもこのストールマンである。

ストールマンが MIT（マサチューセッツ工科大学）の人工知能（AI）研究所で勤務している最中にソフトウェアの専有化が行われ、彼の所属していたソフトウェア共有のコミュニティは崩壊を迎えた。このため、彼は再びソフトウェアを共有するコミュニティを作るため、MIT を辞め GNU ソフトウェアの製作を開始したのである。

彼の開発しようとしたソフトウェアとは UNIX ライクなフリーの OS、「GNU（グニュー）」注 2) であった。この「GNU」という名称はハッカの伝統的な命名法である回帰法を使用したもので、「GNU’s　Not Unix」の頭文字からつけたものである。

では、このストールマンが考案したフリーソフトウェアとはどのような性質を持ったソフトウェアなのか。前提として知っておかなければならないこととして、フリーソフトウェアとフリーウェアはまったくの別物である、ということである。フリーウェアは二章で説明したとおり、ユーザに認められている権利はバイナリコードの利用のみである。これに対し、フリーソフトウェアとは 1985 年にストールマンが設立した非営利団

体であるフリーソフトウェア財団（Free Software Foundation 以下、文中では FSF と表記）注 3) によって定められた定義に準じているソフトウェアのことを指す。

この FSF はフリーソフトウェアの総本山であり、GNU プロジェクトに必要な資金集めを担っている。FSF の収益事業は GNU ソフトウェアの販売だけでなく、GNU 以外のフリーソフトウェアを収録したソフトウェアやマニュアルの販売、及びその関連サポートがある。これは、現在のフリーソフトウェアを利用した代表的なビジネスモデルとなっている。

しかし、FSF の役回りは開発されたソフトウェアの営業・販売以外に GNU ソフトウェアの開発自体も行っている。FSF は他のオープンソースソフトウェアプロジェクトとは異なり、ボランティア以外に多くの有能なハッカを自ら集め、専任のスタッフによる有用なソフトウェアの開発やきめ細かいサポートを行っている。彼らによって開発されたソフトウェアの中でも「GNU Emacs」や「GNU C Library」、「BASH（Bourne Again Shell）」、「GCC（GNU C Compiler）」である。これらは GNU／Linux システムにおいて、なくてはならないソフトウェアとなっている。

また、フリーソフトウェア独自の考えとして、「コピーレフト（Copy left）」という考えがある。これは本来「コピーライト（Copy right）」の名称で使われている著作権をあらわす用語であり、対象のソフトウェアに永続的なフリーソフトウェアとしての自由を与えることを保証したものである。そして、コピーレフト宣言するために必要なライセンスとして「GNU General Public License（GNU 一般公衆利用許諾契約書、または一般公共使用許諾契約書）」（以下、文中では GNU GPL と表記）注 4) がある。これは、

1. 著作権者がソフトウェアに対して著作権を主張する
2. 著作権者が利用者に対してソフトウェアの複製や頒布、または改変についての法的な許可を与える

という二点を踏まえた上で「フリー」であることを守っている。この GNU GPL は著作権を放棄するのではなく、権利を明確にすることでユーザによる自由な利用を許可しているのである。

上記のように、フリーソフトウェアはオープンソースソフトウェアと同様に自由な改変と再頒布の権利がユーザに与えられる。しかし、フリーソフトウェアにはひとつの制限が設けられている。それは「フリーソフトウェアの派生物もフリーソフトウェアでなければならない」というものである。これはフリーソフトウェアとして公開されたソフトウェアが人から人へ頒布される間に改変や頒布の自由のないソフトウェアに変更されることを防ぐことが目的であり、ユーザ保護のためである。

また、フリーソフトウェアはユーザにあらゆる権利を与えるために、第三者が手に入れたフリーソフトウェアを販売することも認めている。ただし、「フリーソフトウェアの派生物もフリーソフトウェアでなければならない」という制限があるためにそのソフトウェアを購入した人がこれをコピーし、他人に対して譲渡・販売することもまた認め

られているのである。このことからも分かるように、フリーソフトウェアはオープンソースソフトウェア同様、必ずしも無償でなければならない、ということはない。このように、フリーソフトウェアを利用する際には無償であるという誤った先入観を取り除いてから使用しなければならない。

以上のことからフリーソフトウェアの定義づけをすると、五つの項目から成っている。それは

1. ソースコード公開の自由
2. コピーの自由
3. 改変の自由
4. 頒布の自由
5. 派生ソフトウェアはフリーソフトウェアでなければならない

というものである。これらの条件を踏まえたソフトウェアがフリーソフトウェアと呼ばれるのである。

このフリーソフトウェアはソースコードを公開していることからも分かるよう、オープンソースソフトウェアの一種である。しかし、派生物もフリーソフトウェアでなければならない、という極めて厳格な条件があるためにフリーソフトウェアは受け入れられない原因となった。この条件があるために企業はフリーソフトウェアを商用利用することができず、ビジネスの世界からは敬遠され、またあらゆる分野にフリーソフトウェアを浸透させようとしたため、外部からは一種のコミュニティを形成しているように見えたことも普及を妨げる大きな一因となったのである。

しかし、この状況を打開したものが Linux の誕生である。Linux の広がりはビジネス界に大きな衝撃を与え、それまでのライセンス重視のビジネスモデルのみならず、フリーソフトウェアを利用したビジネスの形態が模索されるようになった。この結果、生まれたのが「オープンソース」という考えなのである。オープンソースソフトウェアはフリーソフトウェアと違い、派生ソフトウェアを自社の製品としてコピーライトで保護することが出来るため、ビジネスへの応用に対して存在していた垣根を取り払うことになったのである。

第二節　オープンソースが生まれた理由

第一節において過去、ソフトウェアは共有されていることが一般的であったと述べたが、あくまでもそれは研究所や教育機関、またソフトウェアの利用方法を学ぶためのユーザ会においてである。この頃でもメーカにとってはビジネスであるため、利益追求のためにソフトウェア開発を行っていた。その結果として開発者の権利として認められたものが著作権であり、これにより商用ソフトウェアは法的な保護を受けることが出来るようになったのである。この様な利益追及のために開発されるものに対し、社会貢献の

手目に開発されるものがオープンソースソフトウェアである。フリーソフトウェアやオープンソースソフトウェアは「ソースコードの開示」や「改変の自由」、「再頒布の自由」を謳っている。つまり、ユーザに対してすべての権利を与えることによってより良いソフトウェア環境を提供するとともに、更なる発展を期待し、ひいてはそれは社会貢献につながる、という考えである。では、何をきっかけとしてこのような動きが生まれたのか。それは、ソフトウェアの専有化からすべてが始まっているのである。

コンピュータの創世記、コンピュータ技術者が少なかったことも幸いしてソフトウェアの取り扱いは非常におおらかであった。開発者は自らのプログラムを他のプログラマにテストを依頼し、そのプログラマがプログラムをコピーし、第三者に譲り渡すといったことも珍しくないことであった。しかし、第一節で述べたように 1980 年にアメリカにおいて著作権法が改正され、ソフトウェアに著作権が認められるようになると状況は一変することになった。これにより各国においてもソフトウェアに著作権を認める動きが活発となり、ソフトウェアの利用に際してライセンス契約が必要となったからである。この結果、それまでは暗黙の了解の下で行われていたプログラマ同士による相互協力のコミュニティは崩壊し、それぞれの利益追求のためのソフトウェア開発へ変化していったのである。

世の中にライセンス重視のソフトウェアが浸透を始めた中、この動きに待ったをかけたのがストールマンである。1983 年、彼は「Free Unix！」で始まる電子メールを発信したのである。彼はこのメールの中で「完全な Unix 互換ソフトウェアシステム」を開発し、誰もが「Free」で使うことのできるようにする、と謳っている。このメールの内容は 1985 年に「GNU Manifesto（GNU 宣言）」注 5) として FSF から公式に発表された。この宣言により、ソフトウェアの専有化に対抗した完全に「Free」である OS の開発が始まったのである。

GNU プロジェクトとしてストールマンが最初に取り組んだのは「GNU Emacs」というエディタソフトである。これにより GNU プロジェクトはその存在を表すこととなり、このソフトウェアは多くの人によって改良が加えられ、GNU プロジェクトは参加者を着々と増やしていった。そして、1990 年に入るまでにカーネル以外の OS を構成する主要なプログラムが完成したのである。

しかし、フリーソフトウェアが一般のユーザにまで広がることはなかった。その大きな要因がストールマンのフリーソフトウェアに対する考え方であった。彼は商用ソフトウェアを開発している企業を「諸悪の根源」と評し、商用ソフトウェアを完全に排除した、「フリーソフトウェアのみで構成された OS」としての GNU を目標にしていた。このため、利益追求を目的とした企業との間に大きな隔たりが生まれることになったのである。これが 1995 年頃になると、ストールマンの強硬姿勢に反発し、使用制限を緩やかにしたソフトウェアの使い方が模索されるようになった。その結果生まれたのが「オープンソースソフトウェア」なのである。フリーソフトウェアとオープンソースソフトウェアを名前を変えただけとし、同一視する動きもあるが、名称を変え、その定義を緩やかにしたことでフリーソフトウェアを代表とした共有を目的とするソフトウェア群

は脚光を浴びることなり、実際にオープンソースソフトウェアを利用したソフトウェア開発も数多く行われることになったのである。

自社のソフトウェアをオープンソースソフトウェア化することに抵抗を感じる企業は今なお多いが、開発の手順にオープンソースプロジェクトを利用することは増えつつある。多くのボランティアを集め、彼らにベータ版を試験的に利用してもらうことによってデバッグの変わりとすることによって開発期間の短縮と開発コストの削減が図れるためである。この方式は今では多くの企業によって行われており、特に PC ゲームの開発を行っている企業においてはほぼすべての企業によって行われている。また、この方式を行うことによりユーザの要望をすばやくソフトウェアに反映させることが可能になったのである。

第三節　オープンソースとハッカの関係

まず、大前提として言っておかなければならないこととして、「ハッカ」とは決して不正アクセスやウィルスの製作など、ネットワークをりようした悪事を働く者の通称ではない、ということである。ハッカとは高度なコンピュータ技術を持つソフトウェア技術者のことを指し、利益よりも興味の追求を目的としたソフトウェア開発を行っている。「ハッカ＝ネットワーク犯罪者」というのは誤った認識なのである。

本来、ネットワーク犯罪者を指す言葉として妥当なものは「クラッカ」である。そのため、ネットワークを介した犯罪は「クラッキング」と呼ばれるべきなのだが、ハッカの一部にも自らの開発したプログラムの実証実験のために他者のコンピュータに不正アクセスをしていた者がいたこともまた事実であり、このため「ハッカ＝クラッカ」という図式ができてしまったのである。しかし、コンピュータ技術者からすればハッカとは最上級の尊敬を意味し、実際にずば抜けた技術を持つ者に与えられる一種の称号なのである。本論文中では「ハッカ」とは本来の意味である「上級ソフトウェア技術者」の意味で使うこととし、同様に「ハッキング」も「問題やプロジェクトに関して創造的な才能を発揮すること」という意味で使用する。

それでは、このハッカとオープンソースソフトウェアにはどのような関係があるのだろうか。一般に商用ソフトウェアは企業の利益追求のために開発されるため、技術者の思惑とは異なる製品になることも珍しくない。一方、オープンソースソフトウェアはハッカの個人的な趣味や意向により開発されることが多く、ビジネスを目的に開発するのではないので商用ソフトウェアとは一線を画した、利用目的が極めて狭いマニアックなソフトウェアが開発されることもある。

では、実際にオープンソースソフトウェアを開発する場合、ハッカはどのような手順を取るのか。いかにオープンソースソフトウェアといっても個人で開発するにはその規模に限界がある。商用ソフトウェアならば後に開発資金を回収できるため、多くの技術者を雇用することが可能だが、オープンソースソフトウェアでは収入を見込むことは難しく、そのためオープンソースプロジェクトを利用し、その開発プロジェクトに賛同し

たハッカに協力を仰ぐのである。また、実際に開発に関わらなかったハッカも実際にそのソフトウェアを利用し、バグやセキュリティホールなどを見つけ、自ら修正する、といった後発的な協力も行われている。このように、多くのはハッカの協力の元に出来あがるオープンソースソフトウェアのライセンスを取り決めているのが一章で触れたOSI、オープンソースイニシアティブである。

OSIは1998年に非営利団体として設立され、このOSIにより「共有ソフトウェア」は「オープンソースソフトウェア」という名称に統一された。実際に「オープンソース」という言葉が使われたのは1998年、2月3日に行われた会議によってである。この会議は同年1月22日にNetscape社が行った

「NETSCAPE ANNOUNCES PLANS TO MAKE NEXT-GENERATION COMMUNICATOR SOURCE CODE AVAILABLE FREE ON THE NET（ネットスケープ社は次世代コミュニケータのソースコードをネット上で無償で入手可能とする計画を発表）」注6)

というタイトルのニュースリリースであった。このネットスケープ社の発表に共感した多くのハッカがカリフォルニア州パロアルトで行われた会議に集まり、フリーソフトウェア時代から続いていた共有ソフトウェアと商用ソフトウェアの対立に終止符を打ち、ネットスケープ社の後押しをするために現実的なビジネスにも対応できる共有ソフトウェアを開発することを取り決めたのである。そして、フリーソフトウェアに変わる新たな名称として決められたのが、この会議に参加していたChris Peterson（クリス・ピーターソン）によって提案された「オープンソース」だったのである。

また、続く2月23日には同じくネットスケープ社から

「NETSCAPE ANNOUNCES MOZILLA.ORG, A DEDICATED TEAM AND WEB SITE SUPPORTING DEVELOPMENT OF FREE CLIENT SOURCE CODE（ネットスケープ社はMOZILLA.ORGというフリーのクライアントソースコードの開発をサポートするチームとWebサイトを発表）」注7)

というニュースリリースがあった。このとき発表されたMOZILLA.ORGが、最初のオープンソースとなったのである。

OSIの最大の功績は共有ソフトウェアの名称を「オープンソース」としたことであろう。名称を変えたことでそれまでフリーソフトウェアの持っていた商用ソフトウェアに対する敵対意識を和らげ、企業が共有ソフトウェアを元に新たなソフトウェアを開発しやすい環境を作り上げたのである。実際、Oracle社のOracleやIBM社のDB2など多くのソフトウェアがLinux上に移植され、開発者、ユーザ双方に共有ソフトウェアは浸透していったのである。

では、このオープンソースソフトウェアはどのような手段をとって開発されるのだろうか。従来のソフトウェアは「ウォーターフォールモデル」といわれる上位から下位へと作業を進めていく方式や開発初期段階においてプロトタイプをつくり、開発方針を確認する「プロトタイプモデル」などが一般的である。これに対し、オープンソースの利点を生かすことによって行われる開発手段を「バザール（伽藍）方式」注8)という。こ

れは著名なハッカである Eric S. Raymond が命名したもので、レイモンドはそれまでの開発手段を「The Cathedral（大聖堂）」にたとえ、この新たな開発手段をさまざまな人が参加して行う「The Bazaar（雑貨市）」とたとえた。バザール方式とはソフトウェアの開発にあたって、多くのハッカがネットワークを通じて各プログラムを作り、それを適宜公開することで開発とベータテスタによるデバッグを平行して行う方式のことである。

オープンソースプロジェクトはこのバザール方式によるプロジェクトであり、このプロジェクトマネジメントに関する重要な教訓を以下に挙げる。

1. よいソフトウェアは、開発者の個人的な問題解決や興味から生まれる
2. 再利用技術が最重要技術
3. 自分で継続できなくなったら、有能な後継者に引き継ぐ
4. 大勢のベータテスタと共同開発者がいれば、問題の発見と解決が容易
5. 早めに公開、頻繁に公開。ユーザの意見やアイディアの収集が大切
6. ベータテスタやユーザは価値あるリソースであり、共同開発者
7. 開発コーディネータが多くの協力者と協働するためには、インターネットなどの通信手段と強制なしにリードする方法を身につける必要がある

特に「早めに公開、頻繁に公開」という考えは開発段階から公開し、ベータテスタやユーザを共同開発者とみなすという、従来のソフトウェア開発では考えられなかった手段である。

この章ではオープンソースが生まれた理由から、実際に開発されるに当たってどのような手段がとられるのか、ということを説明してきたがこの開発手段であるオープンソースプロジェクトも絶対的なものではない。事実、最初のオープンソースソフトウェアとして計画された「MOZILLA.ORG」は 1998 年 2 月に計画が発表されたが、協力するハッカが集まらず、約 100 人の Netscape 社の開発者、外部ハッカ 30 人という状況で開発が行われた。そして、正規版が発表されたのは 2002 年 6 月という、実に 4 年以上の月日が経過していた。このように、「バザール方式」による開発はボランティアであるハッカが集まらない場合、その開発は大きく遅れることになる。つまり、決してオープンソースプロジェクトは決して全能なものではないのである。このことは、Mozilla プロジェクトに関わった Jamie Zawinski が「resignation and postmortem（辞職そして追悼）」注 9) のなかで「オープンソースは特効薬ではない」と述べていることからも窺い知ることができる。

参考文献／Web サイト

注 1） 『フリーソフトウェアと自由な世界』
Richard M Stallman 著／長尾高弘 訳／アスキー 刊

注 2） GNU’s Not Unix ／ 注 3 Free Software Foundation
http://www.gnu.org/

注 4） GNU GPL
http://www.gnu.org/licenses/gpl.html

注 5） GNU Manifesto（GNU 宣言）
『フリーソフトウェアと自由な世界』
Richard M Stallman 著／長尾高弘 訳／アスキー 刊

注 6） Netscape 社
http://wp.netscape.com/newsref/pr/newsrelease558.html

注 7） Netscape 社
http://wp.netscape.com/newsref/pr/newsrelease577.html

注 8） 『The Cathedral and The Bazaar（伽藍とバザール）』
http://cruel.org/freeware/cathedral.html Eric S. Raymond 言／山形浩生 訳

注 9） 『resignation and postmortem（辞職そして追悼）』
http://www.jwz.org/gruntle/nomo.html Jamie Zawinski 著

第四章　オープンソースのメリット

第一節　開発者側のメリット

開発者にとってオープンソースのメリットはソフトウェアを自由にカスタマイズできることや、目的にあった環境に移植できることにある。これにより、システム構築やカスタマイズソフトウェアの開発などのビジネスも可能となる。このオープンソースを使ったソフトウェア開発を行うに当たって有効な点が二点ある。まず一点目がオープンソースソフトウェアを使ったソフトウェア開発が行えること、二点目に既存のソフトウェアを二次利用できる、という点である。

既存のソフトウェアを利用したいと思った場合、そのソフトウェアが自分の持っている環境に合わなければ使うことは不可能であった。Macintosh と Windows の互換性の問題などがこれにあたる。Macintosh は Windows が普及する以前から GUI に適応した環境を提供していたため、DTP（Desk Top Publishing）やデザインなどの業界で多く利用されていた。こういったソフトウェアは多くが Macintosh 用として開発されたため、Windows では利用できない、ということが数多くあった。逆に、ワープロや表計算ソフトといったオフィスソフトは Windows が主流であったため、Macintosh では利用できなかった。このため、コンピュータ技術者はこれらのソフトウェアを双方に利用できるものに作り変えたいと思っても、商用ソフトウェアは使用が認められているのはバイナリコードのみであり、ソースコードは非公開であったため、改変をすることは非常に困難であった。

さらにもうひとつの大きな障害が著作権の問題である。1980 年代に各国でソフトウェアに著作権が認められたため、特定の条件以外では第三者は勝手に改変や再頒布を行うことはできなくなっていたのである。

しかし、これがオープンソースソフトウェアの場合、この問題は解決する。オープンソースソフトウェアはソースコードが公開され、最初から移植を含むあらゆる改変の自由が認められている。商用ソフトウェアでは開発会社以外にできない移植もオープンソースソフトウェアでは決して簡単な作業ではないが、技術と時間があれば誰でも自由に行うことができるのである。

また、まったく新しいアプリケーションソフトウェアを開発したとして、そのソフトウェアを Macintosh ないし Windows 上で開発した場合、そのソフトウェアを提供したとしてもユーザはそのソフトウェアに対応した OS を購入する必要がある。しかし、元から Linux 上で開発をしていれば、そのソフトウェアと Linux をセットにして頒布することが可能であるため、ユーザに対して余計な費用負担を強いることはない。開発者、ユーザ双方にとって思い通りの環境が低コストで用意することが可能となる。

このように、自由な改変やバグの修正が自らで行えるといった利点がコンピュータ技術者に受け入れられ、利用者が広がる中で商用ソフトウェアを抜きディファクトスタン

ダードとなったソフトウェアも少なくない。特に、Linux に代表されるサーバコンピュータ用のソフトウェアはオープンソースソフトウェア抜きには考えることができない規模にまで浸透している。特にシェアの大きなソフトウェアとしては Web サーバソフトウェアである「Apache」は全体の 66.04%（2002 年)、電子メール配信ソフトウェアである「Sendmail」は全体の 42%（2001 年)、DNS（Domain Name System）用ソフトウェアである「BIND」にいたっては全体の 95%（2000 年）と、他の追随を許さないほど普及している。また、Linux 自体も Windows の 49.6%に対して 29.6%（2002 年）のシェアを持っている注 1)。

これらのソフトウェアは現在では Linux のディストリビューションによって Linux のカーネルとともに頒布されているが、元をただせばこれらのソフトウェアは Linux 用に開発されたものではない。元々は UNIX 用ソフトウェアとして開発されたものが Linux 用に移植され、Linux とともに普及をしていったのである。では、この Linux が普及していった敬意はどのようなものなのか。Linux は現在普及している OS の中では特に歴史の浅い OS である。この歴史の浅い OS が現在の地位に上り詰めた主要な要因は二つあげることが出来る。

一つは多くのボランティアの協力がある。プロジェクトに協力したハッカは様々なドライバを開発し、Linux は多くの環境に適応可能な OS となった。この結果、Linux は FreeBSD などそれまで主流であった UNIX 系 OS を押しのけ、UNIX 系 OS のディファクトスタンダードとなった。二つはそれまでの GNU や UNIX の資産と相乗効果が挙げられる。これはすでに多くのユーザの支持を得ていた GCC や Apache、Sendmail や BIND といったソフトウェアを利用できたため、サーバ用 OS としての地位を得た。また、Linux が普及したことによって付随する形でこれらのソフトウェアもディファクトスタンダードとして広がっていったのである。現在、FSF では Linux のことを「GNU／Linux」と呼んでいる。これは、Linux のカーネル部分と Linux パッケージを区別するための名称であり、「GNU／Linux」とはツール群を含むパッケージ全体のことを指している。

次に既存ソフトウェアの二次利用であるが、代表的なものではやはり Linux や Apache、またデータベースソフトウェアの MySQL や PostgreSQL などがある。これらのソフトウェアは商用ソフトウェアと比べて決して見劣りするものではなく、また市場シェアからみても大きなシェアを持っている。このような安定性の高いソフトウェアを使えるシステムでは初期コストを大きく削減することができるというメリットがある。また、特定の目的に適合したカスタムソフトウェアを開発する際にも同様の機能を持ったオープンソースソフトウェアを利用することによって開発期間、及びコストを大幅に削減できるのである。この開発期間の削減とはシステム設計、デバッグ、そしてメンテナンスの期間を短縮することが出来るのである。

しかし、開発者が気を付けなければならない点としてオープンソースソフトウェアには動作保証がない、ということである。商用ソフトウェアであれば当然指定された環境であれば動作保証があるが、オープンソースソフトウェアにはこういった保証は一切な

い。オープンソースソフトウェアはユーザが自由に利用できる代わりに一切の動作保証はなく、ソフトウェアのメンテナンスはすべてユーザが負うことになる。また、中にはいかなる環境においても動作保証自体をまったくしていない、というものもある。このため、開発者は完成したソフトウェアに何らかの問題が起きた場合はその責任はすべて開発者に覆いかぶさるのである。

第二節　ユーザ側のメリット

オープンソースソフトウェアに対して一般ユーザが感じるメリットは開発者が感じるメリットと大きくかけ離れていることが多い。これは基本的に一般ユーザはソフトウェアを元のまま利用し自ら手を加える、といったことを行わないからである。彼らがオープンソースソフトウェアを利用する理由はソースコードが公開されているからでもなく、また改変の自由が認められているからでもない。商用ソフトウェアに匹敵する高性能ソフトウェアを無償で利用できるからである。

これまでにも述べたようにオープンソースソフトウェアは必ずしも無償で頒布されているわけではないため、「オープンソース＝無償のソフトウェア」という考えは誤りである。しかし、多くのユーザから見ればオープンソースソフトウェアは無償頒布されていて当たり前、と認識されているのである。確かに入手からセッティングまでの肯定を考えればそれまでにかかるコストはディストリビュータに支払うソフトウェアの代金のみであり、企業単位での使用の際もひとつ分の料金を支払えば後は複製の自由があるため商用ソフトウェアに比べはるかに安価ですむ。しかし、後々の保守・運営のことまで考えるとそのコストはユーザの利用目的や技術者の技術によって異なる。ユーザによってはオープンソースソフトウェアを利用することによって余計な手間とコスト負担が増える可能性があるのである。このため、オープンソースソフトウェアを利用するに当たっては初期導入コストのみではなく、その後のサポートコストを含む調査の必要がある。では、ユーザがオープンソースソフトウェアを利用する際にある無償以外のメリットは何があるだろうか。

それでは以下にメリット、そしてデメリットを挙げる注 2)。

メリット

1. コストを節約できる。特にデータベースでは大きく節約できる
2. 安定性がある（メジャソフトウェアに限る）
3. セキュリティ・パッチ対応が早い。ソースコードが公開されているため、いざというときは自分で対応できる
4. 機能の追加を自由に行える

デメリット

1. 自分でサポートを行う場合、技術が伴わない場合がある
2. マイナソフトウェアの場合、開発者がサポートを行わなくなると自分で修正を行わなければならなくなる
3. **ディストリビュータが販売している製品まで無償と誤解されることがある**
4. Linux などに詳しいエンジニアが必要

これを見ると、メリットとしてはコストの削減、安定性の確保、ソースコードの公開と多方面から見たメリットが挙げられる。しかし、逆にデメリットを見ると 3.のオープンソースに対する誤解を除けば残りはサポートに関するものばかりである。このことから、目先のコスト削減やソースコードの公開といった自由度の高さを理由に採用し、安易に採用した結果サポート作業に苦心する、といったことが多いようである。

また、個人ユーザに関していえば一般的にソースコードを使用することはない。彼らがオープンソースソフトウェアを使用する最大の理由はひとえに「無償」という点にあり、自らソフトウェアを改変し、カスタムソフトウェアを作る、といった高い知識を必要とすることを行うユーザは極めてまれである。そんななか、個人ユーザがソースコードを利用せざるを得ない機会がひとつだけある。それは、ソフトウェアのアップデートの際である。

ソフトウェアをアップデートするには通常、パッチとして頒布されるバイナリコードを利用する。しかし、場合によってはこのバイナリコードの使用ができないことがある。Linux を例にすると、GNU／Linux ではオープンソースプロジェクトが推奨しているディレクトリ構造とディストリビューションのディレクトリ構造が異なる場合がある。このような場合、バイナリコードを使ったアップデートは行えず、ユーザがソースコードを入手し、自らそのソースコードをコンパイルしなければならないのである。しかし、通常は頒布元のディストリビュータが元のソースコードをコンパイルしたプログラムを提供しているので、やはり個人ユーザがソースコードを触る機会は滅多にないと考えて構わないだろう。

参考文献／Web サイト

注 1／注 2）　『オープンソース・ソフトウェアの現状と今後の課題について』　経済産業省
http://www.meti.go.jp/kohosys/press/0004397/1/030815opensoft.pdf

第五章　オープンソースの現状

第一節　オープンソースソフトウェアの種類

現在、オープンソースソフトウェアの数は増加の一途にある。しかし、オープンソースソフトウェアの開発者は自らの興味や目的によって開発が行われるため、ソフトウェアの種類や分野には多くの偏りがある。そのため、ユーザは自らの判断でニーズにあったソフトウェアを見つけなければならない。この節ではこの数あるソフトウェアの中からメジャなソフトウェアをいくつか紹介する注 1)。

1. Linux

Linux は 1991 年、フィンランドの学生であった Linus Torvalds を中心に開発されたオープンソースの UNIX 系カーネルである。この Linux はインターネットサーバ用 OS として大きなシェアを占めており、頒布条件は GNU GPL によって公開されている。Linux の特徴はバザール方式を始めて採用して開発されたオープンソースソフトウェアである、ということである。Linux はこの開発方式によって多くのハッカを集めることができ、またそのハッカから提供されたドライバソフト（周辺機器を利用できるようにするのに必要なソフトウェア）を組み込むことができたため、急速に普及していったのである。

2. FreeBSD

FreeBSD はオープンソース UNIX 系 OS として 1993 年に発表された。カリフォルニア大学（UC）の Berkeley 校で開発された長い歴史を持つ UNIX の BSD（Berkeley software distribution）版を Intel 社のマイクロプロセッサ搭載パソコンで動作するように開発された OS である。日本では当初、IBM 互換機でしか動作しない Linux よりも BSD 系 OS のほうが普及していた。この BSD はソフトウェアの商用利用に関する制限が最も少ないライセンスの一つであり、カスタムソフトウェアを商用ソフトウェアとして販売することも可能である。また、この FreeBSD は Apple 社の最新 OS である Mac OS X のベースになったことも知られている。

3. Apache

Apache は現在世界で最も高いシェアを持っている Web サーバである。このオリジナルは NCSA（National Center for Supercomputing Applications）注 2) が開発したフリーソフトウェアである。1995 年にこの主要な開発スタッフが Netscape 社に移籍後、オリジナル版のユーザであったハッカが開発を続け、1999 年に Apache Software Foundation 注 3) を設立。オープンソースソフトウェアの成功例として現在に至っている。

4. BIND

BIND は現在 95%という驚異的なシェアを誇る DNS サーバである。1984 年に最初のリリースがあり、1993 年からは ISC（Internet Systems Consortium）注 4) がその管理を行っている。この ISC には Sun Microsystems 社や HP 社、IBM 社といった主要ベンダが参加している。

5. Sendmail

電子メールのメールサーバとして長期間にわたって高いシェアを持っているのが Sendmail である。1981 年に UC Berkeley 校の Eric Allman が開発したものである。アールマンは今も主要な開発者として活動しており、1997 年に Sendmail 社を設立。オープンソースソフトウェアとしてディストリビュートを行うとともに、商用版の販売も行っている。

6. MySQL

MySQL はスウェーデンの TcX dataconsult AB 社が開発したリレーショナルデータベース管理システムである。 マルチユーザ、マルチスレッドで動作し、高速性と安定性が高いことで知られている。1994 年に最初のリリースがあり、2000 年に GNU GPL によってオープンソース化されている。Yahoo！社や Motorola 社などが採用している。

7. Mozilla

Mozilla は Web ブラウザである Netscape のオープンソースソフトウェア版である（三章・注 7 を参照)。1997 年に同社からソースコードが公開されたが、オープンソースコミュニティの賛同を得ることができず、開発は大きく遅れることとなった。1998 年の Mozilla プロジェクトの発表から 4 年がたった 2002 年になって Mozilla 1.0 がリリースされた。

第二節　商用ソフトウェアとの比較

オープンソースソフトウェアと商用ソフトウェアという選択肢がユーザに用意されている。これらの違いは開発工程やソースコードの公開の有無だけではない。使用に当たっての責任の所在や導入に当たってのコストなど、双方の間には大きな違いがある。ここではその相違点を挙げる。

1. 無保証の提供条件

オープンソースソフトウェアは基本的に無保証である。著作権者のみならず、頒布を行っている関係者も含め、ソフトウェアの利用に関して一切の保証を付け

ずに提供している。

2. 環境への適合性への保証

オープンソースソフトウェアは開発者側で一定の環境の元、動作チェックは行っているが、それはユーザに対して保証するものではない。

3. 品質や性能に関する責任

開発者はオープンソースソフトウェアの品質や性能に関する責任を負わない。ソフトウェアに欠陥があったとしても著作権者は修正の義務を負わず、ユーザが自ら修正や訂正を行うことが求められている。また、そのためにかかるコストはすべてユーザの負担である。

4. 使用結果の責任

ソフトウェアを利用した結果、ユーザに何らかの損害があったとしてもそのことに対する保証は一切行われない。この場合の損害とはデータの損失や他プログラムとの相性問題などを指す。

これに対し、商用ソフトウェアはライセンシーをとっての販売を行う代わりにマニュアルに沿った行動に対しての品質保証を行っている。その保証内容は以下のとおりである。

1. マニュアルに記載のある内容に従って動作することを保証
2. マニュアルに記載のある内容のサポートを行うことの保証
3. ソフトウェアが保証規定を満たさない場合は製品の交換、保証、代金の交換のいずれかを行う

これらの内容は記載のある事柄に関しての保証を行うことを確約するものではあるが、それは裏を返せばマニュアルに記載のないことに関しては一切の保証を行うものではない、ということを意味している。これは商用ソフトウェアに関する問題のひとつとなっている。また、そのソフトウェアを利用して事業を行った結果いかなる損失を企業が負ったとしてもその補償は一切ない。しかし、例外条件として法律にのっとり損失の一部を補填するシステムは存在する。

オープンソースソフトウェアと所要ソフトウェアを比較する場合、真っ先に出てくるのはコストの問題である。オープンソースソフトウェアには無償でなければならないという規定はないが、多くのソフトウェアは無償で行われている。例えば雑誌の付録やボランティアのディストリビュータによるダウンロード頒布などである。これに比べ、商用ソフトウェアは有償での販売を行っている。

しかし、TCO（Total Cost of Ownership　購入から保守・運用までにかかる総コスト）

から見た場合、オープンソースソフトウェアは本当に割安なのだろうか。オープンソースソフトウェアを使用する上で発生するコストにはどのようなものがあるのだろうか。ここではサーバシステムの構築を例に挙げる

1. システム構築にかかるコスト
   サーバを構築するに当たってはアクセス権の限定や不要ソフトウェアのインストールを行わないなどの注意が必要であり、これらを行うためには初期段階での厳密なシステムの設計が必要となる。ここでそのためのコストがかかってくるのである。

2. インストールコスト
   Linux は Windows に比べてメーカによる動作確認情報が不足している。このため、Linux をインストールする場合、使用するパソコンにインストールできるのか、また何かしらの設定を行う必要があるのか、といったことを自ら調べないとならないのである。さらにいえば Linux 関係の技術者は雇用コストが一般の技術者に比べ高くつくことが多い。

3. バージョンアップ時のコスト
   オープンソースソフトウェアの基本は「早めに公開、頻繁に公開」にある。そのため、ユーザにも頻繁なバージョンアップが要求される。これはつまり、その度に技術者に対する人件費がかかる、ということになる。また、バージョンアップ後にしっかりとソフトウェアが起動するかを確かめる動作確認にもコストがかかる。

また、これらの費用以外にも外部サポートを採用している場合にはそのための費用もかかり、さらに言えば企業では社員が Linux 環境になれていない場合には社員教育のための費用もかかることになる。では、これらの点を踏まえた上でユーザはオープンソースソフトウェアと商用ソフトウェアのどちらを選ぶのが有益なのだろうか。

FSF の Richard M Stallman を始めとした「フリーソフトウェア」信者は「商用ソフトウェア＝悪」、「フリーソフトウェア＝正義」という信念を持っている。彼らはソースコードの公開を行わない商用ソフトウェアを販売している企業を「悪の権化」とし、OS 環境をすべてフリーソフトウェアで統一しようとしている。彼らはあらゆる商用ソフトウェアの開発手段を批判している。しかし、この手の批判は穴が多く、的確なものではない。

例えば商用ソフトウェアにはバグが多く、ソースコードが複雑である、という批判がある。しかし、まずバグに関しては現在のコンピュータ工学の技術ではバグを完全になくすことは不可能とされている。またソースコードが複雑である、という点に関しては商用ソフトウェアにはバックワードコンパチビリティ（前バージョンとの互換性を持たせること）の必要があるため、ソースコードが複雑にならざるを得ない。つまり、ユー

ザ本位の開発を行えば、必然的にソースコードは複雑になってしまうのである。

このように「反商用ソフトウェア」派の意見は容易に反論ができる。また、一般的なユーザにとってはソースコード公開の有無はソフトウェアの有益性を語る上で重要視されることはなく、Windows 環境における動作保証がなされ、購入当初の契約でサポートがしっかりと約束されている商用ソフトウェアのほうが安心して利用できる、という面がある。このため、現実的な側面から見ればオープンソースソフトウェアやフリーソフトウェアよりも、商用ソフトウェアを選んだほうが有益であるだろう。

参考文献／Web サイト

注 1） 『オープンソース・ソフトウェアの現状と今後の課題について』 経済産業省
http://www.meti.go.jp/kohosys/press/0004397/1/030815opensoft.pdf

注 2） NCSA（National Center for Supercomputing Applications）
http://www.ncsa.uiuc.edu/

注 3） Apache Software Foundation
http://www.apache.org/

注 4） ISC（Internet Systems Consortium）
http://www.isc.org/

第六章　オープンソースの抱える課題

第一節　オープンソースプロジェクトの現状と課題

オープンソースソフトウェアが抱える課題はいくつかあり、その中でももっとも大きな問題はオープンソースプロジェクトの運営である。そこで、この節ではオープンソースプロジェクトの現状と、人材や資金の調達方法といった問題を採り上げる。

そもそも、「オープンソースプロジェクト」とは何か。オープンソースプロジェクトとはボランティアを募り、ソフトウェアの開発やデバッグ、フィールドテストを行う開発方式の事を指す。しかし、オープンソースソフトウェアは誰にも義務や責任があるわけではないので、参加者が興味をなくしてしまえばその段階でプロジェクトは破綻してしまう。そもそも、魅力のないプロジェクトには人は集まらないため、活動を始めることすらできないのである。

オープンソースプロジェクトは 2005 年 1 月現在で SourceForge.net 注 1) だけでも 8 万件以上、SourceForge.jp 注 2) でも 1300 件以上が登録されている。さらに、この他の方法で登録をしているプロジェクトを含めればその数はさらに膨れ上がる。しかし、このすべてが活動しているわけではない。むしろ活動しているプロジェクトのほうが圧倒的に少なく、全体の 20%、日本国内では 10%に満たないといわれている。このような活動数の少なさには理由がある。

第一に、プロジェクトの提案者がいかにすばらしいアイディアだと思っていいても、他の人が興味を持たなければプロジェクトをスタートすることすらできない、ということである。

第二には、オープンソースプロジェクトの性質に原因がある。オープンソースプロジェクトは参加も脱退もボランティアの自由のため、テーマに賛同していたとしても、プロジェクトの進行に参加メンバが不満を感じ、脱退してしまう人も多いので途中でプロジェクトが破綻することが多いのである。このように、オープンソースプロジェクトは開始も継続も非常に困難なのである。

では、実際に活動を始めたプロジェクトはどのような道筋をたどるのだろうか。オープンソースプロジェクトはフラットな構造のネットワーク型組織を築いて行われる。そして、この組織を維持するために重要な役割を持つのがチームリーダである。通常、チームリーダはプロジェクトの提案者が担当し、そのプロジェクトの最終決定権を持つ責任者である。さらに大規模のプロジェクトになると提案者とともにプロジェクトの方向性に関与する「コア」と呼ばれるグループができ上がる。このコアメンバはプロジェクトに対して大きな貢献をするか、チームリーダに選ばれたハッカがなる。ハッカにとってオープンソースプロジェクトのコアメンバに選ばれることは一種の名誉であり、またコアメンバになることでプロジェクトの方向性決定に参加できるため、多くのハッカはコアメンバを目指して貢献をしている。

このコアメンバと一般のメンバの違いは決定権の有無である。このため、コアメンバはプロジェクトに対して多くの貢献をしたメンバ、つまりプロジェクトの方向性をしっかり理解している者が担当することによってよりよいソフトウェアの完成を目指して作業を進めていくのである。

また、オープンソースプロジェクトの中にはボランティアによって運営される通常の活動とは異なり、企業や国家の支援や資金の提供を受けて活動しているプロジェクトも存在している。先に挙げた Netscape 社の Mozilla などがその代表例である。このようなプロジェクトは開発元の企業が資金援助を行っている。このような活動を行っている団体の中でも大きなものが、Linux の成長と浸透を目的としている非営利団体の「OSDL（Open Source Development Labs）」注 3) がある。この OSDL には日本の富士通や日立製作所、三菱電機、日本電気、アメリカの Alcatel 社、Cisco Systems 社、Dell 社 HP 社、IBM 社、Intel 社、フィンランドの NOKIA 社などのハードウェア企業と Linux ディストリビューションを行っている多くの企業がスポンサとなり、最先端のテスト環境を提供している。

第二節　不正コード混入問題と知的所有権問題

ソフトウェアの開発で最も問題となるのはバグの存在である。セキュリティ問題の多くもこのバグが原因で起きている。また、バグと同様に深刻な問題を引き起こすのが不正コードの混入や第三者による著作権や特許権の侵害に抵触するコード混入の問題である。これらは開発者、ユーザの双方に多くの損害を与えかねないため、非常に大きな問題となっている。

クラッカはこのようなバグが原因でできるセキュリティホールを狙ってクラッキングを仕掛けている。これは、企業や政府の Web サイトだけでなく、オープンソースプロジェクトのサイトも攻撃を受けているのである。このような攻撃によってクラッカは侵入先のパソコンやサーバに不正コードを混入させ、そのコードによって情報の流出や改ざん、破壊などの被害が巻き起こされるのである。Windows が不正コードの標的にされると大々的に報道されるため、このような攻撃はメジャな商用ソフトウェア丹対してのみ行われると思われがちだが、オープンソースソフトウェアに対する攻撃も多く報告されている。実際に世界各地の Linux サーバが攻撃を受け、データが流出したり、破壊されたのである。

これに対し、クラッキングとはネットワークを通じて標的となるコンピュータに侵入し、システムの停止や HDD の破壊を引き起こすことである。このようなクラッキングは不正コードの混入に比べればオープンソースプロジェクトの Web サイトの被害は比較的少ないが、近年では UNIX や Linux のディストリビューションサイトが標的になっている。

このような不正コードやクラッキングでオープンソースソフトウェアが被害を受けた例を以下に二つ挙げる。

1. Apache や Linux の開発サイトへの不正コード混入事件

2001 年 5 月、Apache を開発している ASF（五章・注 3 を参照）の公開サーバがクラッキングの標的となり、不正アクセスを受けたことが分かった。このことは「Apache.org compromise report, May 30th,2001」注 4) の中で報告されている。このときは早期発見がなされ、すばやい復旧が行われたため、被害の拡大は防がれた、と報告書にある。

また、2003 年 11 月には Linux カーネルのベータベースにクラッカが侵入し、新バージョンの Linux にトロイの木馬を組み込もうとした事件も起きている。

2. オープンソースソフトウェアの配布におけるウィルス混入

2002 年 8 月、「ftp.openssh.com」や「ftp.openbsd.org」などのサーバで頒布されていた暗号化ソフトウェア「OpenSSH パッケージ」の中にトロイの木馬が組み込まれた。この事件では、ミラーサーバにもトロイの木馬入りのプログラムがコピーされた可能性が示唆されていた。

このように、外部から不正な手段を用いて侵入し、プログラムに不正コードを混入させるほか、ソフトウェアの設計段階で内部のプログラマがわざと混入させることもある。これは、クラッカが外部から侵入するよりも関係者が内部から混入させるほうが遥かに容易に行えるからである。特にオープンソースプロジェクトでは、その開発者の多くはボランティアのハッカであるため、その中にいる悪意を持ったクラッカを特定するのは事実上不可能といえるだろう。また、当初は純粋なボランティアとして参加していたハッカがある日突然クラックを行うことも無いとは言えないため、完全な不正コードの混入を防ぐことは非常に困難である。

また、このようなクラッキングを行うことを目的にしたコードの混入のほかにも、著作権や特許権の侵害を引き起こす可能性があるコードが混入されることがある。これは「財産的な権利」を保護する著作権、及び「発明・アイディア」を保護する特許権を侵害するソースコードが故意、もしくは偶然ソフトウェアに組み込まれたときに起きる問題である。

この問題も商用ソフトのみならず、オープンソースソフトウェアの開発に際してももちろん注意しなければならない問題である。何故ならば、このような権利を侵害するソースコードが混入されたソフトウェアを公開した場合、権利者から訴えを起こされ、賠償金の支払いやソフトウェアの使用を停止しなければならないからである。

第三節　商用ソフトウェアのオープンソース化

近年、企業が開発したソフトウェアをオープンソースソフトウェア化する動きが出ている。有名なものでは三章で取り上げた Netscape 社の「Mozilla」や Sun Microsystems 社の「OpenOffice.org」注 5) などがある。また、2005 年 1 月、IBM 社はパソコン事業

からの撤退に伴い自社の保有する約500個の特許を大学や研究所向けにオープンソースとして公開すると発表し、またこれからも自社のパソコン関連の特許を随時オープンソースとして公開していく、と発表した。

このように、企業が保有するソフトウェアや特許をオープンソースとして公開する目的は大きく二つの理由に分けることが出来る。一つはソフトウェアのビジネス携帯を販売重視からサポート重視への切り替え、そして対照的な理由としてソフトウェアのサポートを中止するためである。自社によるサポートが困難になったため、ソフトウェアをオープンソース化することでユーザの自己責任に任せるためである。では、実際に商用ソフトウェアをオープンソース化するには、どういった工程を踏む必要があるのだろうか。商用ソフトウェアをオープンソースソフトウェアにするためには、公開にあたって必要となる作業と、公開後のオープンソースプロジェクトの形成という、二つの工程が必要である。

まず、公開するために必要な作業には自社が著作権を持つソースコードと第三者からライセンスを受けたソースコードを判断し、第三者のソースコードを削除する必要がある。次に、オープンソース化した後のライセンス方式を決める必要がある。ここでは著作権や特許権に関する声明を明確にしておくことが重要となる。これは自社の立場を明確にしておかないとソフトウェアが公開された後、第三者が権利を持つソースコードが混入された場合、訴訟の対象になりかねないからである。また、完全にフリーソフトウェア、もしくはユーザの使用に制限をかけないのであれば既存の「GNU GPL」や「MIT License」を流用すれば問題無いが、自社独自のライセンスを適用するのであれば、どのような条件を規すかを決める必要もある。このような工程を経て、オープンソースと公開しただけではオープンソースプロジェクトの終了ではない。プロジェクト開始当初はボランティアが満足な人数集まる訳ではないため、充分な人数が集まるまでは公開元の企業が自ら人材を用意し、プロジェクトを運営しなければならないのである。

このような工程を踏むことで商用ソフトウェアはようやくオープンソースソフトウェアとなる。ここまでの作業は非常に煩雑であり、しかもオープンソースプロジェクトを発足させた後も一定期間は自社にとって直接利益につながらない運営を行わなければならない。このため、実際に商用ソフトウェアをオープンソースソフトウェア化する企業は少数であり、先に挙げた Mozilla や IBM 社のような例は非常にまれである。

参考文献／Web サイト

注 1） SourceForge.net
http://sourceforge.net/

注 2） SourceForge.jp
http://sourceforge.jp/

注 3） OSDL　Open Source Development Labs
http://www.osdl.org/

注 4） 「Apache.org compromise report, May 30th,2001」
http://www.apache.org/info/20010519-hack.html

注5） OpenOffice.org
http://www.openoffice.org/

おわりに

この論文では商用ソフトウェアとオープンソースソフトウェア、そしてその原点であるフリーソフトウェアについてそれぞれどのような性質を持っているかを説明してきた。特にオープンソースの歴史や現状、抱えている課題などに焦点を当て、それぞれ問題点や優れている面を紹介した。

しかし、注意しなければならないのはオープンソースソフトウェアだから良い、商用ソフトウェアだから悪い、といった考えをしてはならない、ということである。ソフトウェアはそれぞれが持っている特有の性質・特徴によってその良し悪しを判断すべきであって、頒布形態によって判断しては物の本質を捉えることは出来ないだろう。しかし、オープンソースを開発するにあたって行われるバザール方式や GNU GPL といったオープンソースならではの方針はプログラマやハッカの欲求から生まれたといっても過言ではないだろう。開発責任者からではなく、ユーザに近い立場にある技術者からこのような発想が生まれたということは、今後のコンピュータ業界にとってこのオープンソースという考え方は一定の流れを保ちつつ、少しずつ浸透していくのではないだろうか。

事実、Linux を例に挙げればサーバでこそ大きなシェアを持っているがパソコンでのシェアは 2002 年現在でわずか 1.7%しかない。しかし、近年中国において「OS その他コンピュータの主要ソフトウェアは公共物であり、共有すべきである」という FSF に共感した政策を採用し、国内のパソコン、サーバ双方において Linux を普及させることを国家政策とした。このような流れはオープンソースソフトウェアが普及するにあたって大きな手助けとなるだろう。では、このような動きを受け、オープンソースソフトウェアが商用ソフトウェアに取って代わる日が来るのだろうか。

私の考えでは、答えは「否」である。確かにオープンソースの考えに賛同し、協力する人々は増えている。また、下火になりつつあるとはいえ、「IT ブーム」といわれる動きも後押しになるだろう。しかし、オープンソースプロジェクトに参加しているボランティアの多くがメジャソフトに対する反感、一種の判官びいきからこのプロジェクトに賛同している、という人が多いため、彼らはオープンソースが主流になったらこのプロジェクトから脱退する可能性が高い。また、フリーソフトウェアに関していえば、Richard M Stallman のような狂信的なまでに共有ソフトウェアを信奉し、商用ソフトウェアに対して露骨な敵愾心を持っている者が第一人者である限り、一般の人々の共感を得ることは難しいと思われる。

そもそも、オープンソースという動きは商用ソフトウェアに不満を持ち、満足のいかない人々が自らの欲求を満たすためにその活動は始まった。このような起源を持つオープンソースは限られた分野において一定のシェアを保ち、一般の商用ソフトウェアとの住み分けを行うことでさらなる発展をしていくだろう。

参考文献

『フリーソフトウェアと自由な世界』
Richard M Stallman　著／長尾高弘　訳／アスキー　刊

『オープンソースワールド』　川崎和哉　著／翔泳社　刊

『それがぼくには楽しかったから』　Linus Torvalds・David Diamond　著
／風見潤　訳／中島洋　監修／小学館プロダクション　刊
『オープンソースがビジネスになる理由』　米持幸寿　著／日経 BP 社　刊

『ソースコードの反逆』　GLYM MOODY　著／小山裕司　監訳／ASCII　刊

参考 Web サイト

GNU's Not Unix!　http://www.gnu.org/home.ja.html

Masayuki Hatta's Home
Pagehttp://www.geocities.co.jp/SiliconValley-PaloAlto/9803/index.en.html

Open Source Initiative (OSI)　http://www.opensource.org/

Open Source Development Labs（OSDL）　http://www.osdl.jp/about_osdl

The Ja-Jakarta Site　http://www.jajakarta.org/

『オープンソース・ソフトウェアの現状と今後の課題について』　経済産業省
http://www.meti.go.jp/kohosys/press/0004397/1/030815opensoft.pdf