オープンソース～一人勝ち市場型開発～

96C2122008G 榊祐介

序章

1990年代後半は、インターネットという新しいメディアが普及し、ポータルサイトのYahoo!や書籍等販売のアマゾンと言った「ドットコム企業」がアメリカの株価を牽引した。インターネットは企業間のグローバル競争を促進し、あちこちでウィナーテイクオール現象が起きている。その典型とされるPC-OS市場ではMicrosoftがWindowsで市場を制覇し、次の狙いをインターネットに定めている。これに対し、対抗馬として注目されるようになったのがLinuxである。Linuxがマスコミ受けしたのはWindows vs Linuxという構図によるものだが、それは本質ではない。Linuxは、世界各地のプログラマーが、本職の合間に趣味でハッキングすることによって開発されている。そういう「片手間」OSがなぜMicrosoftのような大企業のOSと対抗できているのか？Linuxコミュニティはなぜ混乱のうちに分裂し、消滅してしまわないのか？そうしたソフトウェア工学、組織論上の疑問がでてくるのは当然である。この論文では一人勝ち市場における新たなソリューションについてオープンソース勢の代表であるLinuxを中心に考察する。

1. オープンソースとは何か

オープンソース・ソフトウェアとはライセンスにGNU GPLなどを使用するソフトウェアのことを指す。このライセンスによってソース公開を義務づけるとともに、そのソフトウェアの派生物に対しても同じライセンスを継承させることによって(CopyLeft)、ソフトウェア資産を必ずコミュニティに還元させることを目的としている。

1. 1 ソフトウェアライセンスによる分類

Linuxに代表されるオープンソース・ソフトウェアは、ソースとバイナリの両方が配布または入手可能な形になっているようなソフトであり、通常は無料である。オープンソース・ソフトウェアはしばしば「シェアウェア」や「フリーウェア」と間違われるが、これらのライセンスモデルには大きな違いがある。また、「フリーソフト」と「フリーウェア」もまた別物であるが、その話題については後ほど触れる。

オープンソースなライセンスの定義について正確な定義は存在しない。従来は慣習的に行われていたからである。ここではオープンソース・ソフトウェアの半数で採用されている事実上の標準ライセンスであるGNU General Public License(GNU GPL)を見ることにする。

GPLの序文に目的が書かれている(*1)。

- フリーソフトの複製物を自由に頒布できること(そして望むならあなたのこのサービスに対して対価を請求できること)。
- ソースコードを実際に受け取るか、あるいは、希望しさえすればそれを入手することが可能であること。
- 入手したソフトウェアを変更したり、新しいフリー・プログラムの一部として使用できること。
- 以上の各内容を行うことができるということをユーザー自身が知っていること。

GNU GPLはソースコードの公開を義務づけている点が最大の特徴である。しかもそのソースを自由に変更してよいとしている。さらに本文の方でライセンスの継承義務をうたっている。ソースを改良したりそれを元に新しいソフトウェアを開発してもかまわないが、その場合新しいソフトウェアにもGNU GPLを適用しなければならない。「ソースコードの公開」と「ライセンス継承」の2点がGNU GPLの根幹をなし、GPLによって公開されたソフトウェアとその子孫は永久に独占されないことが保証されているのである。その恩恵として、GPLを採用したオープンソース・ソフトウェアは、仮に開発企業、グループが解散してもソースが手に入れられるために将来に対する長期的な信用が生じることである。

このように、ソフトウェアがユーザー間で継承されるだけでなく、そのソフトウェアから派生した新しいソフトにもGNU GPLが継承される仕組みを「CopyLeft」と呼ぶ。

1. 2 オープンソースという用語

「オープンソース」という用語は1998年に生まれたばかりである。それまでは「フリーソフト」という用語が使われていた。フリーソフト運動を主導していたのはリチャード・ストールマン率いるFSF(Free Software Foundation)である。なぜ「オープンソース」という用語が新たに打ち出されたのか。それには2つの理由がある。

リチャード・ストールマンはGNU GPLの発明を含めUNIX界に大きな貢献をした人物である。しかし彼は共産主義的と評される厳格なハッカーで、ソースコードが公開されていないソフトウェアを「独占ソフト」と呼び、激しく糾弾する人物でもある。こうした理念ははビジネス界から感情的反発を受け、オープンソース・ソフトウェアがビジネス界でメインストリームになることを阻害していた。それに対し「フリーソフト」のように商用ソフトに対立する概念としての用語ではなく、ビジネス界に受け入れられるよう中立的な概念を使おうというプラグマティスト達が「オープンソース」という用語を提唱した。

もうひとつ、「フリー」という単語の紛らわしさも問題とされた。英語の「free」にはリチャード・ストールマンが主張する独占からの「自由」という意味の他に「無料」という意味がある。同じ「free」でも「フリーウェア」では「無料」、「フリーソフト」では「自由」を意味だ。そして、実際多くのフリーソフトウェアは無料で配布されている。「free」を強調することは、世間的には「無料」のソフトウェアとの誤解を、ビジネス的には反商業的だと反発を受けることになる。

こうした経緯から「フリーソフト」は「オープンソース」という新しい用語によってマーケティングしなおされた。

1. 3 オープンソース運動の歴史

オープンソースの歴史は、主にUNIXの歴史であり、ハッカーの復権と言うことができる。初期のUNIXはソースが公開され、ハッカー達は自由に内部へアクセスし、開発することができた。しかし商用UNIXと、Windowsの支配する市場では占有ライセンスによってソースが非公開となり、ハッカーはその基盤を失った。そして現在、オープンソースによって再びハッカーは内部へアクセスすることが可能となった。最初交換経済であったものが一人勝ち市場として生まれ変わったのである。

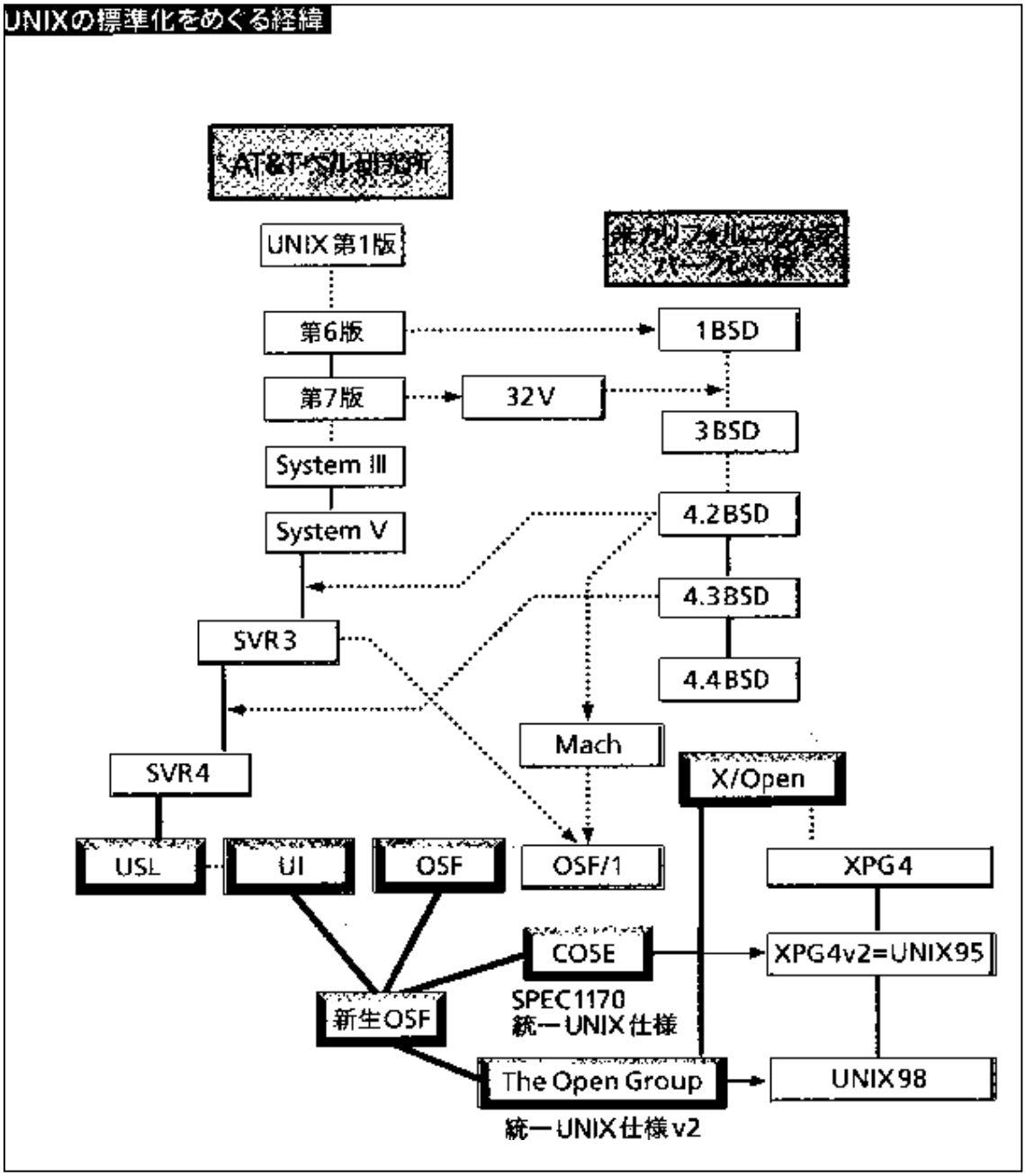
(1) UNIXの歴史

UNIXは1969年にAT&Tベル研究所のケン・トンプソンとデニス・リッチーによって書かれたOSだ。当初UNIXは実費程度でソースが配布され、大学へと広まっていった。ソース公開、改変自由といったオープンソースの文化はこの時代にさかのぼることができる。1977年にカリフォルニア大学バークレー校が開発配布を始めたBSD(Berkeley Software Distribution)は、UNIXの評価を大きく高めた。BSDはその後AT&Tのライセンスに抵触する部分をオリジナルなソースに書き換え、1990年にはほとんどの部分がオリジナルなソース

となった。そのBSDからさらに386BSD、FreeBSD、NetBSDといった完全にフリーなBSDが枝分かれして現在に至っている。

また各企業もバイナリだけを販売するAIX、HP-UNIX、Solarisといった商用UNIXを開発したが、これは各社の独自仕様を取り込んだもので、内部抗争を繰り広げていたUNIX陣営はMicrosoftのWindows、WindowsNTに結果として市場独占を許すこととなった。

UNIXの標準化を巡る経緯



(出典：日経BP デジタル大事典 1999-2000年版)

(2) GNUプロジェクト

1980年代に現在のオープンソース運動の原点が生まれる。リチャード・ストールマンによるFSF(Free Software Foundation)の設立であり、GNUプロジェクトの開始であり、GNU GPLの発明である。

リチャード・ストールマンはハッカーの復権を求めた。彼が1970年代に働いていたMITでは仕事の現場ではフリーソフトが持ちいられていて、プログラマーは自由にソースにアクセスしてソフトウェアを書き換えることができていた。しかし1980年代になるとほとんどのソフトウェアがソースを公開しない占有ライセンスの商業ソフトウェアになってしまった。彼はこれをユーザー間を分断し、お互いが協力し合おうという連帯意識を破壊するものだと考えた。

リチャード・ストールマンは1984年、占有ライセンスの商業ソフトをいっさい使わなくてもよいフリーなUNIX互換OSの開発に乗り出した。FSFによるGNUプロジェクトである。GNUは「GNU is Not UNIX」の略とされている。FSFはオープンソース運動の精神的支柱であると同時にLinuxに代表されるオープンソース運動の注目が高まるまでの潜伏期間においてオープンソース運動の最大の支援者であった。

GNUプロジェクトは結局フリーなUNIX互換OS＝Hurdの開発には成功しなかったがむしろUNIX文化の基礎というべき有名なフリーソフトウェア群を開発することで、UNIXに共通開発環境を提供した意義が大きい。

GNUプロジェクトによって開発されたアプリケーション群

・GNU Emacs
・GNU Compiler Collection (GCC)
・GNU awk
・GNU GhostScript
・他多数(*2)

(3)インターネットの普及

インターネットの普及に足並みを揃えるようにLinuxを代表とするオープンソースが注目を集めだしたのは偶然ではない。インターネットはオープンソース運動に次の効果を与えた。

・オープンソース・ソフトウェアへの需要

WWWサーバーのApache、DNSサーバーのBind、CGIで用いられる開発言語Perl、MAILサーバーのSendmail、これらはみなオープンソース・ソフトウェアの一種である。WWWサーバーとして世界一のシェアを誇っているApacheを始め、インターネットを利用する人は必ずお世話になっている(*3)。つまり、インターネット市場が拡大することは、これらのオープンソース・ソフトウェアへの需要が増大していくことを意味している。

・無料文化、共有文化への関心

ホームページを始め、インターネット上のコンテンツは増大していく一方であるが、そのほとんどは無料で公開されている。こうしたインターネットの文化はソースを共有するというオープンソース・ソフトウェアの運動と合致する物である。

・オープンソース・ソフトウェアの開発機能

LinuxといったOS開発はプログラム開発プロジェクトとして最大級のものであり、当然それに必要な人的、物的資源も大量に必要である。インターネットによる開発者資源の統合、メーリングリストやWWWによる情報共有と言った開発環境の整備によってはじめてOS開発という大プロジェクトが可能になった。

オープンソース・ソフトウェアがインターネットのミッションクリティカルな部分を支えているという事実は、そのままオープンソース・ソフトウェアの実績と、信頼性の評価につながっている。

(4)ビジネス界への進出

インターネットの市場拡大と、それに伴うオープンソース・ソフトウェアの実績は、大手のソフトウェア企業にも影響を与え始めている。これは、オープンソース運動が無視できない大きさになったことの証明である。

事例1：1998年1月、ネットスケープ・コミュニケーション社がネットスケープ・コミュニケーターのソースコード公開を決定

マイクロソフト社のIEとのブラウザ戦争で窮地に陥っていたネットスケープ・コミュニケーション社は1998年1月、自社ブラウザのソースコード公開を決定した。この決定にエリック・レイモンドの論文が影響を与えたことは有名である。

しかし、現実としてMozzilaプロジェクトは2つの目標のうち1つしか達成できていない。1つ目の目標はオープンソース・コミュニティの力を借りてブラウザ開発を進めることだ。プロジェクト開発開始から2年たっても未だ製品と呼べるものをリリースできていない。2つ目の目標はマイクロソフトによるブラウザ市場独占の阻止である。こちらは成功していて、Linux、UNIXと言ったワークステーション市場とPC市場においてネットスケープ社のブラウザは一定のシェアを確保している。

事例2：1998年6月、IBMが自社製品にApacheの採用とサポートを発表

IBMのような大企業が自社製品にApacheを組み込んだことはオープンソース運動にとって大きな転換点となった。Apacheはシェア60%を占める市場No。1のWWWサーバーであり、IBMの採用は現実の追認に過ぎないが、オープンソース運動が社会的に認知されるきっかけの1つとなった。

2. Linux

2. 1 Linuxとは何か

Linuxはオープンソース運動の最大の成果である。Linuxは1998年のサーバーOS市場で純粋な形でシェアを延ばしているただ2つのOS(WindowsNTとLinux)でもある(*4)。

LinuxはMINIXという教育用PC-UNIXを基にフィンランドの学生だったリーナス・トーバルズが開発した。1991年の発表以来彼を中心としたコミュニティによって開発が続けられている。

狭義のLinuxはカーネル部分だけを指し、それを支えるOS全てのことではない。一方広義のLinuxはカーネルやドライバ、アプリケーションなど、完全なUNIX/GUIを提供するに必要なコンポーネントをまとめたパッケージのことを指す。

Linuxは完全にゼロから書き起こされたOSであり、AT&Tなどのソースコードが一切含まれていない。従って無料で再配布が可能である。実際Linuxはインターネット上の数百ものFTPサイトから、あるいはたくさんのベンダが出しているフロッピーやCD-ROMから入手することができる。

LinuxカーネルはGNU GPLによってライセンスされている。従ってLinuxカーネルのソースコードは常にフリーに入手可能でなくてはならず、望めばLinuxに対して金銭をやり取りすることも可能であるが、その場合もLinuxの再配布を制限することはできない。

2. 2 Linuxの開発手法(伽藍vsバザール)

古参のハッカーで、オープンソースの広報役として知られるエリック・レイモンドは1997年に発表した論文『伽藍とバザール』の中でLinuxの開発手法を分析し、それが従来の開発手法と大きく異なる性質を持っていることを明らかにした。まず彼は開発手法をLinuxで行われているインターネットを介して、中心部だけで数百人、周辺部を含めると数千人のプログラマーがそれぞれ自発的に参加して、文字通りよってたかって作る「バザール方式」とBSDやWindowsNTで採用されている中央集権的で、整然と統制のとれた「伽藍方式」に分類した。そして、前者は性能の信頼性、長期信頼、速い開発速度において優れているとした。

(1)「人月の神話」

Linuxはインターネットによるコミュニケーション・コストの低減により世界的規模での共同開発を実際に行っている点で、「人月の神話」を超越できることを示した点で注目されている。

1975年に出版された『人月の神話』における著者ブルックスの主張は、25年たった現代でも色あせていない。彼はIBMのシステム/360の父として知られている人物である。

彼の主張は主に2つである。

- 「人月の神話」
  ブルックスの法則：「遅延しているソフトウェアプロジェクトに要員を追加するとさらに遅れる。」(*5)
- 「銀の弾などない」
  「技術においても管理手法においても、それだけで十年間に生産性や信頼性と容易性での飛躍的な改善をひとつでも約束できるような開発は一つとしてない。」(*6)

「人月の神話」とは、ソフトウェア開発における見積単位である「人月」について、「人」と「月」が相互に交換可能だという仮定が作業者間のコミュニケーションといったマネージメントコストを無視した前提に成り立っていることを指摘したものだ。

マネンジメントコストが発生しない場合「人月」は交換可能で、コストは人数と月数の積に比例する。しかし実際にはどうなるかを示したのが図1、図2である。コミュニケーションを図ることによるコストは、教育・訓練と、相互コミュニケーションの2つからなる。教育・訓練コストは線形に変化するが、相互コミュニケーションによるコストは人がn人いればn(n-1)/2に比例する。

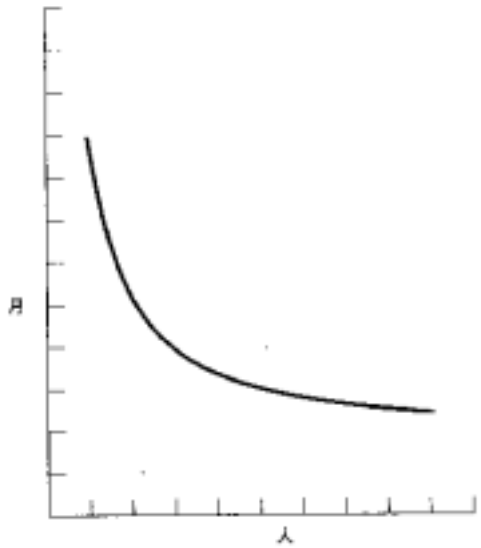


図1 コミュニケーションが必要となる分担可能な仕事の場合

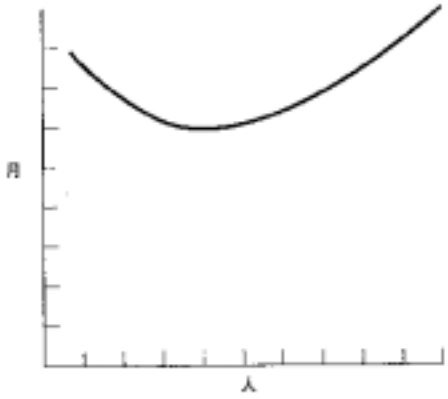


図2 複雑な相互関連を持つ仕事の場合

そこでブルックスは「ブルックスの法則」として「遅延しているソフトウェアプロジェクトに要員を追加するとさらに遅れる。」(遅れたプロジェクトマネージャーがやりがちである。)を主張したのである。

「銀の弾などない」とは、「技術においても管理手法においても、それだけで十年間に生産性や信頼性と容易性での飛躍的な改善をひとつでも約束できるような開発は一つとしてない。」というブルックスの主張を分かりやすく言い換えたものだ。

ブルックスはソフトウェアが本質として複雑であり、ソフトウェアの規模に従って複雑性が非線形的に増大すると述べている。

ブルックスの主張に従い、OSのような大規模で従って複雑性が非線形的に増大するようなソフトウェア開発は、「一番大事なソフトは伽藍のように組み立てられなきゃダメで、一人のウィザードか魔術師の小集団が、まったく孤立して慎重に組み立てるべきもので、完成するまでベータ版も出さないようにしなければダメだと思っていた。」(エリック・レイモンド)(*7)と考えられていた。

一方Linuxはどうか。OSという大規模なソフトウェア開発。主要な開発者だけで数百人、全体では数千人の開発コミュニティ。頻繁なリリース。一見すると様々な作業やアプローチが混在する、バザールのようであり、そこから一貫した安定したシステムが開発されるという従来の常識を覆す現象が起きている。

(2) Linuxの開発プロセス

Linuxカーネルは安定系と開発系を循環させることにより、発散と収束を繰り返して開発される。これにより信頼できるバージョンを使いたい人は安定系、最新版を利用したい人は開発系という使い分けが可能となっている。

Linux カーネルは、

1. 開発系カーネルに対して新機能の追加や抜本的な改善に取り組み
2. やがてフィーチャーフリーズ※で大幅な変更を終了し
3. しばらくはバグ修正に専念して、システムに磨きをかけ
4. その後コードフリーズ※によって安定カーネルとしてリリースされ
5. 一般ユーザを巻き込んで一連のバグ修正を施し、さらに磨きをかけ
6. 再び開発系カーネルとして分岐させ大幅な変更を加えていく

というサイクルを繰り返すことによって開発が進められている。

フィーチャーフリーズとは開発系Linuxカーネルに新機能を追加することを終了すること。また、その宣言。その後は、コードフリーズを目指して、バグ修正やドライバ更新など、システムの他の部分へ影響をほとんど与えないような変更のみが施される。

コードフリーズとは開発系Linuxカーネルに修正を加えることを終了すること。また、その宣言。フィーチャーフリーズの後、コードフリーズがなされるまでは、新機能の追加を差し控え、バグ修正（やドライバ更新）などに専念し、開発者でない一般ユーザが安心して使えるように、 システムの安定化に重点が置かれた開発が行われる。コードフリーズが宣言されると、開発系カーネルがメジャーバージョンアップした安定系カーネルとしてリリースされる。

(3) 開発コミュニティ

Linux開発コミュニティには次の様な特徴がある。

・地理的に極度に広がっている。Linuxの主要開発者は、ヨーロッパ、アメリカ、アジア中に一様に散らばっている。(開発者リストを参照)

・核となる個人の集団をもとにした、巨大な数の貢献者集団。Linuxの場合、パッチやバグ修正を送る人々は1,000人以上、そしてカーネルに直接コードを提供する人は200人以上にのぼる。(開発者リストを参照)

・金銭的な動機では動いていない。こうした個人は、ほかのフルタイムの正業を持ったうえで、余暇や余力をオープンソース・ソフトウェア開発に注いでいる。プログラマーとして素人とプロという目で見ればプロが書いている点に注意。また、Cygnus社のように企業としてLinuxに成果を供給している企業もある。

(4) 開発者コミュニティ運営

開発者コミュニティの運営は、インターネット特有のツールに依存している。

・メーリングリスト　　・ニュースグループ

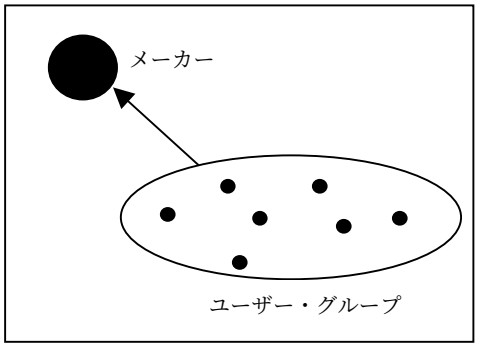
・Web サイト　　・国際的な購読者による一日24時間年365日のモニタリング

24時間世界のどこかで開発が続けられていると言って過言ではない。

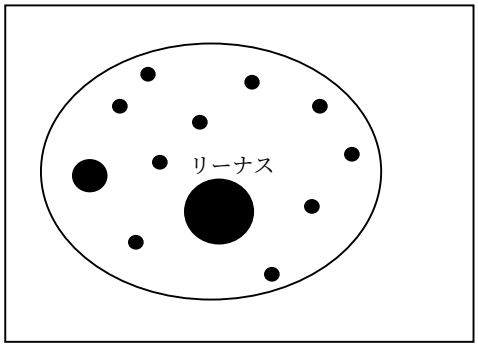
Linux規模のプロジェクトは、非常のスキルの高い開発者が集団である問題に取り組めるようになって初めて成立可能となる。インターネットの発展と開発可能なプロジェクト規模には相関関係があると言える。

(5) 低い参入障壁(ユーザー＝開発者)

従来の開発者とユーザーの関係



Linux開発コミュニティ



上図は開発者とユーザーとの距離感、開発への参入障壁を示したものである。

LinuxがUNIXであることの最大の恩恵は20年余のUNIXの歴史と遺産を受け継いでいることだ。UNIXの操作系とGNUプロジェクトが提供する開発環境は、ユーザーがバグを発見したり改良を思い立ったら即開発に取りかかることを可能にしている。ユーザーはコンパイラも、デバッガも、何ひとつ買う必要がない。

従来開発者とメーカーは個々に独立した存在で、ユーザーがフィードバックする手段はほとんどなく、それが反映される保証もなかったことに比べると大きな違いである。

(6) 並列開発

Linuxの設計はコンポーネント化されているために、それぞれの機能の開発は独立した小グループごとに行われている。新たに開発された機能がカーネルに組み込まれるか否かの最終判断はリーナスが決定権を持っている。企業が開発する場合同じ機能を複数のグループが開発することはコスト的に見合わないが、Linuxは商用ベースではないので、同じ機能を複数のグループが開発した場合、最終的に優れていた方を選択することが可能となっている。

(7) ピア・レビュー

ピア・レビューはオープンソース型開発の中で最も革新的な部分である。エリック・レイモンドは、『伽藍とバザール』の中でピア・レビューを「リーナスの法則」として紹介している。

「ベータ・テスターと共同開発者の基盤さえ十分大きければ、ほとんどすべての問題はすぐに見つけだされて、その直し方もだれかにはすぐわかるはず。」(*8)

この法則は「目玉の数さえ十分あれば、どんなバグも深刻ではない。」という表現をされることもある。ユーザーが開発者を兼ねるメリットと、インターネットによる情報共有によって、Linuxのバグフィックスは非常に早いという評価を受けている。

従来の開発手法では、バグを全て潰せたと確信できるまで少数の人間が何ヶ月も専念してチェックしなければならないと考えられていた。当然リリースの間隔はあかざるを得ない。しかし伽藍方式で開発されたOSであるWindowsNTを見れば、続々とバグが発見され、しかも対応が遅いということで信頼性がないという評価を受けている。これは、伽藍方式のバグフィックスに限界があることを示している。

バグはプログラムに一定の割合で必然的に発生する。そして、テストユーザーが多ければ多いほどバグが発見される確率は高い。この2つの事実から導かれる一番効率的かつ低マネンジメントコストなバグフィックスは、テストユーザーを増やしてバグを発見してもらい、しかもテストユーザー自身にバグフィックスしてもらうことである。これによってLinuxはブルックスが指摘したコミュニケーションの増大によるマネンジメントコストの爆発から逃れているのである。

このサイクルをうまく動かすためには、フィードバックを反映させる構造が必要だ。Linuxはインターネット上から無料で配布され、次に述べる頻繁なリリースによってこのサイクルを有効に機能させている。その結果、Linuxが安定版に達する頃には他のソフトに比べ多数のバグが潰されていることになり、なぜオープンソース・ソフトウェアがインターネットのインフラ部分を担う信頼性を得ているかの答えとなる。

(8) 頻繁なリリース

(5)の参入障壁、(6)の並列開発、(7)のピア・レビュー、これらは開発者コミュニティの協力が欠かせない。彼らの協力を得るためにリーナスがとった方法は、頻繁にリリースを繰り返し、開発結果、フィードバック結果を反映させることである。例えば日本語では http://www.Changelog.net/kernel.html でカーネルリリース情報が得られるが、最近でも週に1回、開発初期には1日に数回リリースされることすらあった。

頻繁なリリースはフィードバック済のバグフィックスを素早く広めることで作業の重複をなくすという実際的な目的の他に、開発者コミュニティに「褒美」をあげる目的もある。

例えばフィードバックの場合を考えてみよう。せっかく自分が改善方法や、プログラムを提供したのにそれが採用されたかどうかわかるのが1年後で、しかも採用されなかったとしたらどうか。その開発者は2度とフィードバックしてくれなくなるだろう。

2. 3 プログラマーにとっての参加動機

この節では、Linuxの開発者たちが開発プロジェクトに貢献したいと思う主要な理由について検証する。

(1) 目の前の問題の解決

これは基本的に、『伽藍とバザール』におけるレイモンドの最初の経験則を言い換えたものである。「よいソフトはすべて、開発者の個人的な悩み解決から始まる。」(*9)

多くのオープンソース・プロジェクト、例えばApacheは、目先の問題を解決しようとして出発したWebmasterの小集団から始まった。個人が自分のシナリオにあわせてコードを加えるにつれて、コードがだんだん改良されていく。

(2) 教育目的

Linuxカーネルは、ヘルシンキ大学での教育プロジェクトから育ってきたものだ。同じように、GNU Linuxシステムの多くのコンポーネント（X windows GUI、シェルユーティリティ、クラスタリング、ネットワークなど）は大学等教育機関にいる個人によって拡張されてきた。たとえば東アジアでは、Linuxはインターネット接続よりも急速に成長していると報告されている(*10)。これは主に教育現場で採用されているからだ。大学は、オープンソースを教育用ツールとして採用している。

Linux上の研究教育プロジェクトは、ソースコードが広く入手可能であるため、簡単に広められる。具体的にはこれは、新しい研究上のアイデアはまずLinux上で実装されて入手可能となり、その後でほかのプラットホームで提供されたり組み込まれたりするようになる、ということだ。

(3) エゴの満足

オープンソース開発の動機づけとしてもっとも実体がなく、しかし最も重要なものは、純粋なエゴの満足である。『伽藍とバザール』でエリック・レイモンドはこう述べている。

> Linuxハッカーたちが最大化している「効用関数」は、古典経済的なものではなく、自分のエゴの満足とハッカー社会での評判という無形のものだ。(*11)

そしてもちろん、「ほかのひとからハッカーと呼ばれないうちはハッカーではない」。

(4) ノウアスフィアの開墾

エリック・レイモンドが公開した2番目の論文『ノウアスフィアの開墾』では、経済的に動機づけられたやりとり（例：商業ソフト開発）と「贈り物の交換」（例：名誉を求めてのオープンソース・ソフトウェア開発）との差について論じている。

「ノウアスフィア」とはだいたい「あらゆる作業の空間」と定義される。「開墾」は、ある所有物を最初に「発見」したり、それに対して大きな貢献をいちばん最近に行った人間になることによって取得することを指す。従ってレイモンドの主張では、オープンソース開発に参加するハッカーの動機づけは、作業の総体のなかで最大の領域を獲得することである。この競争は『ウィナー・テイク・オール』で論じられている一人勝ち市場における競争である。「誰が一番貢献したか」は相対的に決まるが、ハッカー達はインターネットというグローバルな市場で競争することを強いられているのだ。

こうして検討すると、オープンソース・ハッカー達の社会がまさに贈与文化であるのは明らかだ。そのなかでは「生存に関わる必需品」、つまりディスク領域、ネットワーク帯域、計算能力、が深刻に不足するようなことはない。Yahoo!にアクセスするコストも、自分の猫を紹介するページへアクセスするコストも同じである。この贈与文化が産み出すのは、競争的な成功の尺度として唯一ありえるのが仲間内の評判だという状況だ。

(5) 楽観主義

『ウィナー・テイク・オール』では一人勝ち市場の問題として費用―便益の関係から正当化される数より競争者が多すぎることを指摘している。この地位を巡る競争が実際に最も生産性の高い働き手を引きつけるという点も指摘しているが、これはLinuxの生産性の高さを支持するものと言える。

2. 4 コードの分裂

あらゆる大規模開発チームにおける大きな脅威は、コード分裂のリスクである。特にLinuxにおいてはインターネット規模の開発チームということで生じるプロセスの混乱により一層拍車がかかるのではないかと予想される。オープンソースはこれにどう対処しているのか。

コード分裂とは、開発プロジェクトのいろいろなやりとりの中で、プロジェクトのコードベースに複数の共存できないバージョンが生じてしまうことである。

たとえば商業ソフトの世界では、Windows NTコードベースの単一の管理は、商業UNIXの実装(Solaris,IRIX,HP-UXなど)などで見られる分裂したコードベースに対する最大の優位性の一つと考えられている。

UNIXの歴史の中で触れたようにUNIXの歴史は分裂の歴史でもある。例えば、当初BSD UNIXは、カリフォルニア大バークレー校で開発され、実費程度で配布されていたが、後にAT&T等の権利関係で、ライセンスに様々な制限が加えられるようになった。これに対し完全にフリーなBSD UNIXをつくるため有志による（閉鎖的な）開発チームがFreeBSDをつくった。さまざまな理由から FreeBSD チームと対立した開発者たちが、更に別の変種をつくりだした (OpenBSD, NetBSD, BSDI)。

BSDツリーが分裂していった原因として大きなものは2つある。

だれもがBSD のコードベースに貢献できるわけではない。BSDは閉鎖的な開発チームによって開発が進められた。これは実質的に「ノウアスフィア」の規模を制限してしまい、従ってだれか別の人が、分裂した自分たちのコードのほうがコアのBSD コードよりもっと優勢になるぞ、と説得力をもって主張することが可能になる。

GPLとは異なり、BSD 方式のライセンスは派生コードに何の制限もつけていない。従って、自分の加えた改良に付加価値があると思ったら、コードを分裂させても、それで金をとっても、名前を変えても、なにをしてもいい。

これらの動機はどちらも、開発者がちょっとした付加価値を付けることでコードを分裂させて、BSD社会全体を犠牲にしてでも報酬（お金でもエゴでも）を集めようとする状況を作り出す。

BSDの例とは対照的に、Linux カーネルのコードベースは分裂していない。なぜLinuxコードベースの一貫性が保たれているのか、その理由としては以下のようなものが挙げられる。

・万人が認めるリーダーシップ

リーナス・トーバルスはLinux 界における有名人であり、彼の決定は最終的なものとされている。対照的に、BSD派生の活動では、このような有名人リーダーは存在しなかった。

リーナスは開発チームから、公平で筋の通ったコード管理者と思われており、Linuxコミュニティにおける彼の評判はかなり強力である。しかしリーナスもあらゆる判断に関与

するわけではない。通常はサブグループが、自分たちの相違点を自分たち同士で解決して、コード分裂を防いでいる。

・参加はオープンで、長期的な貢献ができる可能性もある。
BSDの閉鎖的なメンバーシップとは対照的に、だれでもLinuxには貢献できるし、その人の「地位」はその人のそれまでの貢献によって決まってくる。
間接的ではあるが、これはコード分裂をおさえるもう一つの理由になっている。分裂した少数派コードベースが、メインのLinuxコードベースの技術革新速度を維持できると保証できるような、信頼性の高いメカニズムはほとんどないと言っていい。

・GNU GPLライセンス方式によって、経済的な動機でのコード分裂はあり得ない
GPL Linuxの派生物は「CopyLeft」に従い必ずなんらかのフリーな形で入手可能でなくてはならないので、Linux ツリーを分裂させた少数派にとっての長期的な経済的メリットは低くなる。GNU GPLはゲームの構造を変えることを許さない。

・コードベースを分裂させると「ノウアスフィア」も分裂
エゴ的な動機づけのため、オープンソース開発者は最大のノウアスフィアにいちばん大きくかけるようになる。コードを分裂させれば、どうしてもその後の新しいコードツリーの開発者にとって達成できる空間は縮小してしまう。

3. 一人勝ち市場を利用するLinux

1章、2章を通じて、私はオープンソース・ソフトウェアとは何か、数千人の開発コミュニティでよってたかって作る「バザール方式」とは何かを理解しようとしてきた。「伽藍方式」で開発されるソフトウェアに比べ「バザール形式」で開発されるオープンソース・ソフトウェアが技術的な信頼性、長期信頼、開発速度において優れているのはなぜか。Linuxに代表されるオープンソース・ソフトウェアは、一人勝ち市場の原理を利用して開発されるからだ、というのがその結論だ。

(1) オープンソースにおける一人勝ち市場の発生

GNU GPLライセンスの採用が、オープンソース開発の一人勝ち市場成立に決定的な役割を果たした。GNU GPLライセンスは、そのCopyLeft機能によってあるオープンソース・ソフトウェアから派生した全ての子孫＝ソフトウェアとそのソースがオープンソース市場に還元されることを強要する。そして、GNU GPLライセンスは例外を認めないが故に市場が分割される危険を未然に防ぐ。これによって全てのオープンソース・ソフトウェアは、インターネットで全世界に広がるオープンソース市場において評価されることとなった。

(2) オープンソース市場での評価

一人勝ち市場では、報酬が相対的な成績で決まる特徴がある。オープンソース市場においては技術が勝敗を決める。例えばあるソフトウェアのバグを修正するパッチを書き、それをオープンソース市場に提供した人の評価は、金銭的価値では図れない。周りのプログラマーがそれをどう評価するかにかかっている。オープンソース開発に参加するプログラマー達は、己の技術だけを武器に評判ゲームの勝者となるべく競争を繰り広げている。

(3) オープンソース市場の勝者

オープンソース市場における評判ゲームで勝者は誰か。Linux開発における勝者を考えてみよう。Linuxの開発には中心部だけで200人、周辺部で数千人の開発者がいる。しかし「Linuxを開発したのは誰か？」「リーナスだ」とみな答えるであろう。彼は最初にノウアスフィアの開拓を始めたことで最大の名誉を得ている。彼及び開発トップの数人だけが株式公開による収入や、講演会に招かれる名誉を受けている。我々はリーナス以外の数千人がLinuxを開発していることを知っていながらリーナス以外の名前を知らない。最初に勝者となったものが、名誉を一人占めしているのは、オープンソース市場が一人勝ち市場であるからだ。

(4) オープンソース・ソフトウェアはなぜ信頼性、開発速度で優れているのか？

オープンソース･ソフトウェアの信頼性と言ったときに2つの意味がある｡1つ目はGNU GPLライセンスの採用に起因し、ソースが手にはいるため将来に渡って入手可能であるという長期信頼性である。2つ目はこれから述べる技術的な信頼性で、これから理由を述べる。

オープンソース・ソフトウェアが、インターネットのインフラを支えていて、事実として技術的信頼性が高い理由についてはいろいろな説明がされてきた。ソースが公開されているからユーザー自身がバグを直せるからだとか、ピア・レビューによってバグ発見率が高いからであるとか。それもあるだろう。

しかしその理由を『ウィナー・テイク・オール』的に言えばこうなる。

「今や、人間に与えられた才能の各分野でおよそ一ダースもの
チャンピオンがいれば地球全体がうまくやっていくことができる。」(*12)

インターネットで優秀なプログラムを配布するコストと、素人が書いたプログラムを配布するコストは同じである。地球規模で開発が進むオープンソース開発と一企業が書くプログラムのどちらが優秀かは明白であろう。

これを裏付ける話として『人月の神話』のプログラマーの生産性に関する研究結果をあげよう。その結果によれば、最高のプログラマーと最低のプログラマーの実績比率は生産性にして平均10:1、プログラム開発の速度と量では5:1であったという。この生産性の差で開発速度についても説明できるだろう。

伽藍方式の開発をとる場合、最高なプログラマーと最低なプログラマーもソフトウェアに取り込まなければならない。そして開発マネージャーは、最低なプログラマー達が人並みに働くよう仕事をしてきた。

一方バザール方式で開発されるオープンソースの場合どうか。開発責任者は、最低なプログラマーのソフトウェアを採用する必要はない。幾つかある候補の中で最高の性能を示したものだけを採用すればよいのだ。

終章

1998年から始まったLinuxのブレークは、未だに流行に終わることなく現在も続いている。私は、リーナス・トーヴァルズの最大の功績は、Linuxを開発したことではないと思う。OSという最大規模のソフトウェア開発で、本能的に一人勝ち市場原理を取り入れ、デバッグと開発に投入されるプログラマーと時間を最大化すべく市場を拡大し続けたことにあると思う。彼はその実績によって、バザール方式の開発手法がOSレベルではない小規模プロジェクト(ほとんど全てだ！)で適用できるだろうという可能性を開いたのだ。

参考文献一覧

1.エリック・スティーブン・レイモンド(著)山形浩生(訳)、
『伽藍とバザール』(1999年9月30日)、光芒社
2.Frederick P. Brooks Jr.(1996)*The Mythical Man-Month* U.S.A
(滝沢徹、牧野祐子、富澤昇 訳)、『人月の神話』1996年2月15日、星雲社
3.Chris DiBona、Sam Ockman、Mark Stone(著) 倉骨彰(訳)、
『オープンソースソフトウェア』1999年7月24日、オーム社
4.川崎和哉(編著)、「オープンソース・ソフトウェア・ワールド」1999年12月20日、翔泳社
5.金子郁容(著)、「コミュニティ・ソリューション」1999年5月25日、岩波書店
6.ロバート・H・フランク、フィリップ・J・クック(著)香西泰(監訳)
『ウィナー・テイク・オール』1998年6月12日、日本経済新聞社

引用一覧

(*1)http://www.gnu.org/japan/gpl-2j-plain.txt(英語版)
http://www.gnu.org/copyleft/gpl.html(日本語版)
(*2)http://www.gnu.org/software/software.html#DescriptionsOfGNUSoftware
GNUによって開発されたソフトウェア一覧。
(*3)http://www.netcraft.com/survey/
ネットクラフト社の調査によるWWWサーバーのシェア。Apacheは55%を占める。
(*5)米調査会社IDCによると、1998年の世界のOS市場動向で、Linuxは全サーバOS中17%超シェアを獲得。ライセンス数では212.5%増という劇的な成長(1999/2/2　日刊工業新聞)
(*5)Frederick P. Brooks Jr.(1996)*The Mythical Man-Month* U.S.A
(滝沢徹、牧野祐子、富澤昇 訳)、『人月の神話』1996年2月15日、星雲社、220頁
(*6) Frederick P. Brooks Jr.(1996)*The Mythical Man-Month* U.S.A
(滝沢徹、牧野祐子、富澤昇 訳)、『人月の神話』1996年2月15日、星雲社、165頁
(*7)エリック・スティーブン・レイモンド(著)山形浩生(訳)、『伽藍とバザール』(1999年9月30日)、光芒社、9頁
(*8)エリック・スティーブン・レイモンド(著)山形浩生(訳)、『伽藍とバザール』(1999年9月30日)、光芒社、24頁
(*9)エリック・スティーブン・レイモンド(著)山形浩生(訳)、『伽藍とバザール』(1999年9月30日)、光芒社、13頁
(*10)http://www.zdnet.co.jp/news/0001/17/turbolinux.html
TurboLinuxの中国での販売実績がWindows 98を上回る
(*11)エリック・スティーブン・レイモンド(著)山形浩生(訳)、『伽藍とバザール』(1999年9月30日)、光芒社、53頁
(*12)ロバート・H・フランク、フィリップ・J・クック(著)香西泰(監訳)
『ウィナー・テイク・オール』1998年6月12日、日本経済新聞社、2頁