オンライン ISSN 1347-4448
印刷版 ISSN 1348-5504

赤門マネジメント・レビュー 4 巻 2 号 (2005 年 2 月)

# 脱パッケージ化したソフトウェアの開発
## ―ソフトウェア開発におけるユーザの組織化―

藤田　英樹
東洋大学経営学部
E-mail: fujita-h@toyonet.toyo.ac.jp

生稲　史彦
一橋大学イノベーション研究センター
E-mail: fikuine@iir.hit-u.ac.jp

**要約：フリーウェア、シェアウェアの開発事例を分析する。これらオンラインソフトの開発では、開発者とユーザがより緊密に直接的な結びつきを持ち、あたかもユーザが開発活動の一端を担っているような現象が観察された。この現象をユーザの組織化と呼ぶことにするが、パッケージング販売を最初から放棄し、流通経路をおもにインターネットに求める「脱パッケージ化」というオンラインソフトにおける意思決定が、単なる流通経路の選択にとどまらず、ユーザの組織化と開発への貢献を引き出し、また、ソフトウェアの完成度がパッケージソフトほどは求められないため、開発・流通のコストが比較的小さくなることが、開発スタイルに劇的な変化をもたらすことを明らかにする。このため、オンラインソフトではバージョンアップのペースとして観察される「開発サイクル」の回転を速めることができ、ソフトウェアの品質やパフォーマンスが一般的なパッケージソフトと比べて格段に速く向上していくのである。**

**キーワード：脱パッケージソフト、開発サイクル、ユーザの組織化**

## 1. はじめに

Linux の大成功を受けて、そのライセンス形態、開発の進め方であるオープンソース・ソフトウェア（以下、OSS）に、社会的のみならず、学問的にも注目が集まっている。*Research Policy* での von Krogh and von Hippel (2003) を中心とする特集号はその好例であろう。OSS に関する研究は Raymond (1997) がひとつの契機になったと思われるが、その後の少なからぬ研究は OSS と Raymond (1997) に関する誤りを含む解釈に基づくことになり、OSS の成功

要因として「無償」の「オープンソース」であることをことさら重要視し、そのライセンス形態の意義、およびそれを支える「コミュニティ」に研究の関心が集められる傾向があった。

しかし高橋・高松 (2002) が指摘したように、Linux の成功は「無償」で「オープンソース」であったためであるとは言い難く、OSS はソフトウェア開発の成功のための必要条件ではあっても、決定的な成功要因とは言えない。また、オープンソース運動に関係した人々の意見や証言をまとめた Dibona, Stone and Ockman (1999) を参照すれば、OSS がそれ以前のフリーソフトウェア運動の延長線上にあり、開発コミュニティと企業とのアライアンス (佐々木, 北山, 2000; West, 2003) などへの配慮から、半ば「人工的に」創り出されたものであるという歴史的事実もある。

OSS に関するより正確な認識と評価に基づくこれらの研究をみれば、OSS が具体化した「無償」、「オープンソース」[1] であることは、ソフトウェアの普及の成功要因として一定の評価ができる (佐々木, 北山, 2000; West, 2003) ものの、ソフトウェアの開発の成功要因であるとは言い難い。したがって Linux および OSS の成功のみをことさら重要視して、ソフトウェア開発の正否について論じることには問題があると思われる。ソフトウェア開発の成功要因は、「無償性」や「オープンソース」にあるとは必ずしも言えず、むしろ高橋・高松 (2002) が指摘した開発者のモチベーションなど、他の要因も視野に入れて考察していくべきであると考えられるのである。

このような OSS に縛られない視点[2] を持って、ソフトウェアとその開発活動の歴史、現状を再確認すると、OSS の発生以前から、Richard Stallman が主導したフリーソフトウェア運動、あるいは国内外においてフリーウェア、シェアウェアの開発、配布が行われてきた事実に気付かされる。それらは、「無償性」「オープンソース」を伴っている場合もあるが、特に日本のフリーウェア、シェアウェアと呼ばれるソフトウェアでは、それらの条件を伴わないにもかかわらず、多数のユーザに支持され、中にはビジネスとして成功しているソフトウェアすらも存在している。

---

[1] ただし、オープンソースの定義（OSD; 原文 http://www.opensource.org/docs/definition_plain.html、邦訳 http://osdn.jp/article.pl?sid=01/08/17/0849221）によれば、無料での配布を制限しないことが定められているだけなので、実質的に配布時に無料／有料を選択できる。

[2] *Research Policy* の特集号、例えば von Krogh, Spaeth and Lakhani (2003) は、OSS に対象を限定したが故に、その研究成果が一般性を欠くことになってしまったのではないだろうか。具体的には、OSS であるということ、すなわちソースコードが公開されているということは、その開発に参加してきた人々がソースコードの理解、問題点の発見、改変、コンパイルなどができるコンピュータとソフトウェアに一定以上の知識を持つ人々に限定される可能性が高い。そうした暗黙の限定条件が存在する諸事例のみにもとづいて研究を進めたことにより、コミュニティの形成、あるいは組織化の対象が上記のようなコンピュータとソフトウェアに関して一定以上の知識を有する人々に限定され、より多くの人々がコミュニティに参加する可能性を考察できなくなるという研究の限界を当初から内包していたと考えられる。

では、OSS も含め、フリーソフトウェア（運動）やフリーウェア、シェアウェアなどに広く共通することがらとは何であろうか。それは、インターネットに代表される全世界的な広がりを持ちつつ、容易に利用でき、しかも情報を伝達するコストが安い情報交換手段を、非常に積極的に活用し、開発活動、配布活動を行っていることであると考えられる。

そこで我々は、インターネットを開発、配布に利用することを前提にしたソフトウェアを「脱パッケージソフト（Non-Packaged Software）」[3] と定義し、これらの開発活動を記述、分析することを通じて、OSS 研究のみを対象とした既存研究よりもより一般性のある知見を獲得しようと考えた。その際、企業が商用目的で開発しているパッケージソフトやシステム、カスタマイズド・ソフトウェアを対象とした既存研究を念頭に置くことにより、脱パッケージソフトの特徴の抽出、とりわけなぜ「脱パッケージ化」で開発スタイルまでが変わるのかを考察する。そのための研究対象はメールソフトであり、比較的成功していると考えられるシェアウェア、フリーウェア各 1 本について、開発者に対するインタビュー調査を実施し、その開発活動のあり方、とくに開発組織や開発過程について詳細な事例研究を行った。その結果、どちらもその開発スタイルが従来のパッケージソフトとはまったく異なるものであることが明らかになる。本研究の発見は、パッケージング販売という流通経路を完全に放棄することこそが、開発スタイルの変化に決定的な影響を与えているというものである。

確かにパッケージソフトであっても、例えば Microsoft 社の Windows Update などのように、インターネットを利用すれば、不具合修正やセキュリティ強化などの修正パッチをより頻繁に提供することが可能になる。それはインターネットの恩恵と言えるだろう。しかし本稿で明らかにされるメールソフトの進化は、それとは量的にも質的にも異なるレベルのものである。早ければ毎週・毎月行われるバージョンアップで、不具合修正だけではなく新機能追加・仕様変更といったバージョンアップまでが行われる。つまりインターネットを利用していても、パッケージング販売にこだわるか否かで、進化のレベルが異なるのである。

「ユーザの組織化」を前提にしたソフトウェア開発は、実はパッケージング販売を放棄したことではじめて可能になった。それによって「開発サイクル」[4] が劇的に変化し、結果として、インターネット利用が開発サイクルを格段に速めるように作用するようになり、ソフトウェアの品質やパフォーマンスが急速に向上していくのである。

[3] 脱パッケージソフトの語義に厳密に従えば、OSS やフリーソフトウェアの多くはもちろんのこと、一部のパッケージソフトのオンライン販売版も含まれることになるが、この論文では便宜上、シェアウェアとフリーウェアを合わせて脱パッケージソフトと呼ぶことにする。なお、シェアウェアが有料、フリーウェアが無料のソフトウェアである。

[4] ソフトウェア開発に関する既存研究がおもな考察の対象としてきた「開発プロセス」を、連続的に循環させる過程を、この論文では「開発サイクル」と呼ぶことにする。

このように、本稿ではソフトウェア進化のレベルの違いを説明するために開発サイクルという概念を提示して用いる。実は、従来、ソフトウェア開発は一回性の高いプロセスだと暗黙の内に仮定されていたのである。Cusumano (1991) を嚆矢とする製品開発論としてのソフトウェア開発研究では、[5] Cusumano and Yoffie (1998) においてMicrosoft社とNetscape社の事例にもとづいて、Cusumano and Selby (1995) で見出した同期安定化プロセスあるいはその発展型が、インターネット用ソフトウェアの開発にも有用であり、とくにインターネットを利用したβテストと結びついて、迅速かつ良質なソフトウェア開発に繋がっているとした。Iansiti も一連の研究の中で、「フレキシブル製品開発システム」(Iansiti & MacCormack, 1997)、「技術統合」(Iansiti, 1998; 邦訳, 第9章) などの概念を提示し、こうした主張を支持している。ところが、これらの研究はいずれも、ソフトウェアの開発を、コンセプト創造（目標設定）に始まり、開発終了・公開で完結する一回性の高い「プロセス」と認識していた。これは、それ以前のソフトウェア・エンジニアリングあるいは製品開発研究の影響であるが、これでは開発プロセス間の連続的運動を認識することができない。ソフトウェアだけでなく製品開発全般においても、前のバージョン／ある製品の開発プロセスで蓄積された知識や、ユーザの使用体験のフィードバックが、次のバージョン／別の製品の開発プロセスに重要な役割を果たすことは、既に指摘されている (例えば、延岡, 1996 など)。[6] そこで本稿では開発サイクルという概念を提示することで、開発プロセス間の連続的運動を分析する。[7]

同時にこの開発サイクルは、企業の境界内に閉じこめて理解しないようにしたい。とくにソフトウェア開発に関しては、優れた能力を持つユーザが数多く存在する。フリーウェア、シェアウェアでは、ユーザ・イノベーション (von Hippel, 1988) のようにユーザが原作者の開発活動に積極的に関与することで、その機能・性能が予想外の方向に進化することが多々あるという (宮垣, 佐々木, 1998)。したがって、ソフトウェアの開発サイクルは、企業を含めた元来の開発主体の境界を超えて自由に広がりうるネットワークとしての組織 (高橋, 2003) を前提に理解することが重要である。

[5] コンピュータの誕生と発展過程、ソフトウェア開発研究の成立経緯の詳細については、立本 (2002) を参照のこと。また初期のソフトウェア・エンジニアリングの代表的な著作としては、Brooks (1995) を参照のこと。

[6] ただし、延岡 (1996)、青島 (1997)、青島・延岡 (1997) などが焦点を当てているのは、企業（開発主体）内部のプロセス間の連携・連続性、知識移転である。しかしソフトウェアの場合、バージョンアップにはユーザからのフィードバックが重要であるので、企業外部に存在するユーザの活動を媒介したプロセス間の連続性にも注目する必要があると考えられる。

[7] Iansiti (1998) はユーザからのフィードバックに言及しているものの、それが開発に及ぼす影響を実証してはいない。

## 2. ケース紹介—「鶴亀メール」と「電信八号」—[8]

### ケースの選択と調査方法

既存のソフトウェア開発研究のほとんどはパッケージソフトのみを対象としてきた。しかし冒頭で述べたように、今日では脱パッケージソフトも普及しつつあり、その中にはパッケージソフトより優れたものや、小規模ながらビジネスとして充分成り立っているものもある。同じソフトウェアでありながら、まったく別の市場を形成していることにはどのような要因が影響しているのだろうか。このような問題意識から、この研究では脱パッケージソフトを取り上げ、その開発活動のあり方について調査・分析を進めていくことにした。

一口に脱パッケージソフトと言っても、非常に多くのソフトウェアが存在し、種類・機能も多岐に渡っている。そこで、脱パッケージソフトに関する研究の第 1 歩として、その中でも代表的なメールソフトの事例を取り上げることにする。メールソフトを取り上げるのは、コンピュータをインターネットに接続して利用することが当然のこととなった現代において、メールソフトは Web ブラウザに次いで利用頻度が高いソフトウェアになっているからである。また、脱パッケージソフトの中でもメールソフトはソフトウェアの規模が大きい部類に入るので、本研究で比較対象として念頭に置くパッケージソフトに近いからでもある。

そこで今回は、著名なオンラインソフトの配布サイトに登録されているメールソフトの中から、シェアウェア、フリーウェア各 1 本を選択した。このような配布サイトに登録されていることは、そのソフトウェアの利用者が多いことを示しており、成功事例とみなせるからである。

調査方法としてはインタビュー調査を主体に実施し、電子メールによる追加質問で適宜補った。インタビューは 2002 年 10～11 月に行われた。インタビューでは、事前に質問項目をリストアップしてインタビュイーに送付し、必要に応じて筆者が質問を補いながら質問票に沿って話してもらった。質問項目には、ソフトウェアの概要、ソフトウェア開発の契機、ソフトウェアの開発とバージョンアップのプロセス、開発組織などに関するものがある。

### 鶴亀メール—シェアウェアの事例—

鶴亀メールは、斉藤秀夫氏が開発した Windows 対応の高機能メールソフトである。メール本文を編集するエディタ部分には、同じく斉藤氏が開発し、Windows 用エディタの定番とも言える「秀丸エディタ」のコンポーネントが利用されている。

[8] 詳細なケースは、オンライン・ソフトウェア研究会のホームページを参照のこと (http://www.gbrc.jp/onlinesoftware/)。

### 鶴亀メールの概要

斉藤氏の運営する「サイトー企画」の主力ソフトウェアは、秀丸エディタ、鶴亀メール、秀 Term である。各々のソフトウェアの登録ユーザ数は、秀丸エディタで約 15 万人、鶴亀メールで約 1000 人、秀 Term で約 9 万人である。ただし、秀丸エディタの登録ユーザは鶴亀メールを無料で使用できるようになっている。このため鶴亀メールの実際の利用者数は、登録者数よりもずっと多いと考えられる。

鶴亀メールは、内蔵されている秀丸エディタの部分を除いた本体プログラムの行数が 117,582 行である。したがって、秀丸エディタの行数 91,795 行を加えると、鶴亀メールの本体プログラムの総行数は、209,377 行となる。

鶴亀メールでは、ユーザによる独自機能の追加はマクロ[9] によって実現されている。斉藤氏が基本的に開発者向けキットやソフトウェアのインターフェイスを公開していないためであるが、これは鶴亀メールへの独自機能の追加が制限されていることを意味しない。むしろ、非常に多数のマクロが作成・提供されており、しかも内蔵エディタである秀丸エディタについてはさらに多くのマクロが提供されている。

### 鶴亀メールの開発経緯

斉藤氏が鶴亀メールの開発を手がけるようになったのは、プロバイダ会社の Xaxon（ザクソン）社のメールソフト「NetMail」に、モジュールとして秀丸エディタを提供していたのがきっかけである。この NetMail のユーザグループのひとつが、秀丸エディタを提供していたサイトー企画に「サイトー企画さんで（メールソフトを）何とかできないか」「秀丸エディタ・ベースのメールソフトで、動作するものであればいいから提供して欲しい」と、斉藤氏にメールソフト開発を要望した。斉藤氏はこうした要望を受けて、2000 年 4 月からフルスクラッチでコードを書き起こしてメールソフト開発を開始し、同年 8 月に Windows 用メールソフトである鶴亀メールを公開した。

### ソフトウェアのバージョンアップについて

鶴亀メールのバージョンアップは、基本的にユーザからの機能追加の要望から始まる。機能追加のうち、マクロによって対応可能なものはマクロ開発や関数の追加で要望に応え、マクロで対応不可能なものに関してはバージョンアップで応じる。

斉藤氏によれば、マクロによる対応を優先するのは、バージョンアップで対応すると鶴亀

[9] マクロとは、ユーザが行う決まり切った操作などを、自動化するために作成する簡単なプログラムである。

**表 1**　鶴亀メールのバージョンアップ履歴と頻度

| | 公開日 | 一日あたり問題解決数 | 1 バージョンあたり開発日数 | 1 バージョンあたり問題解決数 | バージョン数 | 日数 | 問題数 |
|---|---|---|---|---|---|---|---|
| ベータ版 | 2000.8.21～ | 5.310 | 2.598 | 13.793 | 82 | 213 | 1131 |
| Ver.1 | 2001.3.27～ | 3.017 | 4.724 | 14.255 | 98 | 463 | 1397 |
| Ver.2 | 2002.6.28～ | 2.290 | 4.056 | 9.289 | 90 | 365 | 836 |
| 全体 | | 3.232 | 3.856 | 12.459 | 270 | 1041 | 3364 |

注）2003 年 6 月 28 日現在。公式サイト「改訂履歴」をもとに筆者が集計。

メール本体において新たなバグが発生する可能性があるためだという。実際、これまでのバージョンアップの際にもバグが発生し、「ほら、要望に応えたらバグが出ちゃったじゃないか…どうしてくれるんだ」という思いを抱いたことがあるという。

鶴亀メールの場合、バージョンアップに際してテストやデバッグをサイトー企画で行うことはない。すなわち、開発したものはすぐに公開し、サポートフォーラムを通じてバグ報告や機能追加の要望を受け付けて、次回バージョンアップ時にバグ修正、機能追加として反映させる。

バージョンアップに対してはこのような姿勢をとっているため、結果的に表 1 のように、非常に頻繁なバージョンアップが行われることとなっている。

### ユーザとの接点とその役割

ユーザからの意見・要望が寄せられるのが、サイトー企画が運営する「コミュニテックス」に設置されたサポートフォーラムである。このフォーラムで発言するためには会員登録が必要だが、閲覧は自由である。また、サポートフォーラムへは、サイトー企画のホームページである「秀まるおのホームページ」から簡単に行けるようになっている。

### 電信八号―フリーウェアの事例―

電信八号は石岡隆光氏が開発し、現在は任意団体「電八倶楽部」および「電八開発倶楽部」が開発を引き継いでいるメールソフトである。その機能性、完成度は非常に高く評価され、

財団法人インターネット協会（Internet Association Japan）が主催した、第 5 回フリーソフトウェア大賞（FSP’96）において入賞を果たしている。

### 電信八号の概要

電信八号のソフトウェアとしての規模は、原作者の石岡氏が開発していた時代から 5 万行程度であり、現在でも 10 万行を上回ってはいないという。

また、電信八号はエディタを自由に選ぶことができるメールソフトである。原作者の石岡氏は「メールの編集機能は当初から外部のソフトウェアに任せることにしていた」ので、それに応えて熱心にエディタを開発した人達の要望に呼応して 1995、1996 年頃に DDE[10] を作成し、公開済みである。また、その後コマンドオプションも公開したため、現在ではエディタを中心に相当数のソフトウェアが電信八号と連携して動作するようになっている。

### 電信八号の開発経緯

原作者である石岡氏が電信八号を開発しようと思った動機は、「『わかりやすいメールソフト』を自分が使いたいこと」と若干の「金儲け」であった。石岡氏がインターネットを利用し始めたころに出逢ったメールソフトは、どれも非常に使いにくく感じられ、1995 年春頃に「自分で作っちゃえ」「自分が使いやすい Windows 用メールソフトを作ろう」と考えるようになった。

開発に当たっては、UNIX 環境で広く使用されていたメールソフトである MH（Mail Handler）を意識し、平日の夜と土日を開発活動に当て、開発開始から 3 ヶ月程度経った 1995 年夏にはソフトウェアが使える状態になった。さらに開発開始から約半年後の 1995 年秋には、ソフトウェアが安定してきたと感じ、Asahi ネットのソフトウェア掲示板に電信八号を投稿した。

当時は、ユーザから寄せられる要望・期待に応えたいという気持ちがあり、公開して満足感を得ること、配布サイトでの評価、雑誌などの評価記事、ダウンロード数の多さなどを通じて、うれしさや誇らしい気持ちを感じており、それが励みになって 1 ヶ月に複数回のバージョンアップを行っていた。ところが、1998 年頃には「自分で使うには十分」と感じるようになり、石岡氏が開発に寄せる情熱は低下して、積極的にソフトウェアのバージョンを上げることはなくなってしまっていた。

この状況は、電信八号を改良するための要望を受け付けながらそれに応えられない「ソー

[10] Dynamic Data Exchange の略語で、Windows 用ソフトウェアの間でデータやコマンドをやりとりする手順のこと。

スコード死蔵」の状況であり、石岡氏はそれを苦痛と感じるようになった。そんな折、ユーザが運営する電八倶楽部のホームページを訪問した石岡氏は、その熱心さを感じ取った。その熱心さに対し、石岡氏は「要望が多数あり、熱心なユーザも多くいるのに、ソースコードを死蔵しているのは悪いことだ」と考え、条件付きでソースコード公開し電八倶楽部へ開発活動を移管した。

### ソフトウェアのバージョンアップについて

電信八号のバージョンアップの流れは、① パッチ作成、② 公式ビルダによるとりまとめ・バイナリ化（ビルド）、③ 配布およびバグ報告、①′ 新たなパッチ作成、というサイクルが基本になっている。バージョンアップに関わるさまざまな活動の流れを図示したものが図 1 である。

パッチ作成の際に主に参照され、ビルドの際にもパッチを取り込む優先度を判断するために参照される役割を果たすのが、公式ビルダの 1 人である小原良仁氏がとりまとめる WishList である。

WishList は、ユーザから電八倶楽部・電八開発倶楽部に投稿された修正や機能追加の要望をとりまとめて、日本語で記述したものである。このリストは、あくまで修正や機能追加の要望をプールしたものである。公式ビルダを含めた開発グループの誰かが、寄せられた修正や機能追加の要望に共感すれば、それをプログラムとして実現するためのパッチが作成され

**図 1**　電信八号のバージョンアップ・プロセス

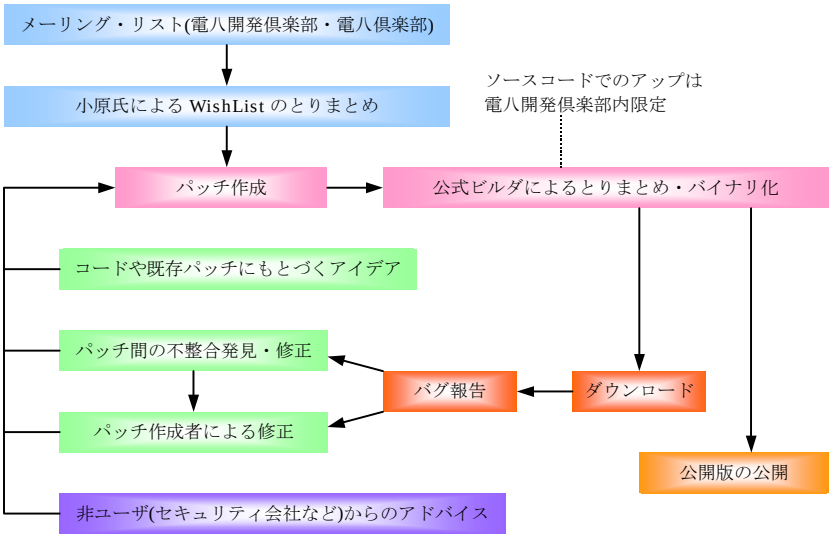


**表 2**　電信八号のバージョンアップ履歴と頻度

| | 公開日 | 一日あたり問題解決数 | 1バージョンあたり開発日数 | 1バージョンあたり問題解決数 | バージョン数 | 日数 | 問題数 |
|---|---|---|---|---|---|---|---|
| Ver.1 | 1995.7.14～ | 0.196 | 20.067 | 3.933 | 15 | 301 | 59 |
| Ver.32.1 | 1996.5.10～ | 0.152 | 43.083 | 6.533 | 60 | 2585 | 392 |
| 全体 | | 0.156 | 38.480 | 6.013 | 75 | 2886 | 451 |

注）2003 年 6 月 28 日現在。電信八号のパッケージに同梱されているリリースノートと、公式サイト「電信八号―移管後の歩み」をもとに筆者が集計。各バージョンアップで解決された問題数は、リリースノートに記載されていないものもあったので、実際にはこれよりはるかに多いはずである。

ることになるが、そうでない要望は未処理のまま WishList に残り続ける。

このようにパッチ作成のひとつの端緒は、WishList に上げられた要望を消化することである。加えて、ユーザ自身が使っている中で生じた問題を解決するためにパッチを作成したり、石岡氏のコードを参照してそれを技術的に改良するためにパッチが作成されることもある。さらには、RFC[11] が提示する規格に沿うためにパッチが作成されたり、セキュリティホールを探す企業からの報告にもとづいてパッチが作成されることもある。

こうした様々な経緯から作成されたパッチを統合し、バイナリ形式に変換してサーバにアップするのが公式ビルダである中村賢一郎氏と福井貴弘氏である。新しいパッチを統合し、バージョンアップされた電信八号は、動作が不安定になる可能性がある場合にはα版として、その危険性がなく安全だと思われる場合にはβ版としてサーバにアップされ、α版は電八開発倶楽部登録者が、β版は電八倶楽部登録者がダウンロード可能になる。

電八開発倶楽部および電八倶楽部の登録者はα版やβ版をダウンロードし、使用してみて、バグ報告やバグを修正するためのパッチを提出する。それらを受けて、再度パッチをとりまとめ、不具合の修正が行われて、一般向けの公開版がリリースされる。中村氏らは口をそろえて、電信八号の「開発スピード、バージョンアップのペースは非常に速い」という（表 2）。

バージョンアップの目安であるビルドは、毎週提出されるいくつかのパッチを受け入れる形で行われる。ビルドは中村氏と福井氏がほぼ交代で行っている。ビルドの作成自体は半日～1 日ですべて終わらせることができる機械的な作業のため、平均してほぼ 1 ヶ月に 1 回の

[11] Request For Comments の略称で、インターネットの各種プロトコルに関する標準仕様について記載した文書の名前。

ペースでビルドが続けられ、公開版が公開されている。

### バージョンアップにおけるユーザの役割

初期の電信八号のバージョンアップにおけるユーザの関与について、原作者の石岡氏は、作者が想定しなかったような使い方を、ユーザが見いだして、使用していたことが印象的だったと述べている。また、作者が考えるアピールポイントとユーザが評価するポイントが異なっていることも印象的だったという。

現在のバージョンアップに対するユーザの貢献に関しては、中村氏らが異口同音に、メーリングリスト参加者の中に優秀なデバッガ（バグを出し、報告してくれる人材）がいることが大きい」と述べていた。ユーザの中には、修正項目のすべてをテストしたり、バグの再現条件を克明に記したレポートを提出する人もいるという。ただし、電八倶楽部・電八開発倶楽部の開発者、ユーザ間の交流はメーリングリストがほぼすべてと言ってよく、オフラインでの交流は皆無に等しいという。

## 3. 脱パッケージソフトの開発スタイル

このような脱パッケージソフトの事例の特徴を、とくに Cusumano (1991) が描いたようなパッケージソフトとの違いを念頭において調べていくことにしよう。両者間では、パッケージング販売という流通経路を使うか使わないかという違いが最も大きいが、事例を一見してわかるのはバージョンアップ・ペースの違いである。[12] 鶴亀メールや電信八号は毎週もしくは毎月バージョンアップされているのに対して、パッケージソフトは早くても 1 年に 1 回しかバージョンアップしていない。[13] しかも、その中身に注目すると、パッケージソフトでは、Windows Update や Office Update のように部分的な脱パッケージ化によってペース自体は速まっても、提供されるのは不具合修正やセキュリティ強化などのマイナーチェンジにすぎない。ところが鶴亀メールや電信八号をはじめとする脱パッケージソフトでは、メジャー／マイナーの区別なく、毎回のバージョンアップで新機能追加や仕様変更までが実現される。

また、脱パッケージソフトの開発主体とユーザはほとんど対等な関係を結び、頻繁に直接的なやりとりをしているのに対して、パッケージソフトのユーザがソフトウェアについての

[12] 脱パッケージソフトとパッケージソフトの間におけるバージョンアップ頻度の比較は単純にはし難がたいが、本研究では便宜的に新しいリリースナンバーのソフトウェアが公開されたときにバージョンアップされたと捉えることにする。

[13] 例えば、代表的なパッケージソフトである Microsoft Windows（Outlook Express）はこの 8 年で 6 回しかバージョンアップしていない。Microsoft Office（Outlook）に関しても、バージョンアップは 8 年間で 6 回である。

要望やバグ報告をしようとしても、基本的にはサポートセンターと呼ばれるサポート部門にしかアクセスできないという違いがある。

流通経路とサポート体制の違いが、機能向上やバージョンアップのペースである開発サイクルの違いに反映されると考えられるが、そもそもこれらの相違点はどこから生じるのだろうか。実はその答えは、電信八号の開発者へのインタビューで聞かれた、User-Supported Software という「ソフトウェア像」（開発主体とユーザが共通して抱くソフトウェアに対する認識）に求められる。そしてこのソフトウェア像を起点として、開発組織や開発サイクルを含めた開発スタイル全体のあり方が決定されるのである。

### 脱パッケージ化とは—ソフトウェア像と試行コスト—

本研究が提唱する脱パッケージ化とは、単に流通経路の違いだけを意味するのではない。実はパッケージング販売をしないという意思決定は同時に、ソフトウェア像やソフトウェアの開発・流通に必要なコストの構造についても選択を行っていることになる。

そもそも日本国内のシェアウェア、フリーウェアは、User-Supported Software に源流を持つというが、これは製品であることを放棄したソフトウェア像であると理解することができる。つまり、そのソフトウェアによる収益の獲得が絶対視されず、代わりにユーザ自身がその入手、インストール、使用に責任をもって当たることが求められる。

こうしたソフトウェア像は、ユーザの役割を単なる使用者であることにとどめさせない。ソフトウェアは無料であるか、せいぜい寄付（donation）程度の値段でしかない代わりに、そのソフトウェアを使い続けたい、もっと機能を向上させて欲しいと思えば、動作テストやデバッグといった開発活動の一部を引き受け、開発者へのフィードバックを行わざるをえなくなる。つまり、脱パッケージソフトではユーザが開発に関与し、開発組織が当初の開発者に限定されない多くのユーザにまで拡大していくのである。こうした現象を「ユーザの組織化」[14] と呼ぶことにしよう。

脱パッケージ化とそれに呼応するソフトウェア像によって変化するもうひとつの要因が、試行コストである。試行コストとは、ソフトウェアを開発するコストと、それを公開・配布するコストとを指す。脱パッケージソフトでは、ソフトウェアの規模が小さかったり、次項で述べるように追加的な機能をモジュール化して外部（ユーザ）に切り出していたりするので、開発者集団も小さいしワークステーションのようなコンピュータも高価な開発ツールもあまり必要にならず、固定費用はほとんどかからない。Linux のような大規模なシステムでも、インターフェイスと仕様を固定しモジュール化しているので、1 人の開発者が担当する

[14] 「組織化」に関しては、Weick (1979; 邦訳 p. 4) で提示された概念を拡張して用いている。

プログラムの規模は小さくなっている。しかもできあがったソフトウェアは、現代ではほとんどがオンライン公開されるため、パッケージングや流通のコストはほとんどないことになる。[15]

### ユーザの組織化—オープンな要素—

なぜ脱パッケージソフトではユーザの組織化が見られるのだろうか。以下ではユーザの組織化の規定因について検討していくことにしよう。まず、脱パッケージソフトでは、電子メールや電子掲示板などを利用したインターネット経由のオンラインサポートが用意されていることが多い。こうしたシステムには、たとえユーザでなくても自由にアクセスすることができるようになっており、いわばオープンなサポート体制が実現されていると言える。[16]

この点は、パッケージソフトのユーザが、ソフトウェアについての要望やバグ報告をしようとしても、基本的にはサポートセンターと呼ばれる部門にしかアクセスできず、しかもインターネットなどの簡便な情報受発信手段を必ずしも使うことができないことと対照的である。パッケージソフトのユーザ・サポートは電話受付が基本であり、報告や要望の種類によっては受付窓口が異なることもままあり、初心者ユーザにとっては敷居が高いと考えられる。また、サポートセンターはサポートのみを行う部門なので開発には直接的にはタッチしていない。つまり、ユーザと開発者（開発部門）の間にサポート部門が介在し、組織的バッファとなっている。

また脱パッケージソフトの中には、ソースコードを条件つきながら公開していたり、連携ソフトウェア、プラグイン、マクロなどをユーザ自身が開発するために必要な情報を公開しているものが多い。例えば、電信八号では条件つきでソースコードが公開されているし、連携ソフトウェアの作成に必要な DDE の仕様も公開されている。このようなオープンな情報提供は、[17] オープンなサポート体制とあいまって、コンピュータおよびソフトウェアに関する知識レベルの高い（IT スキルの高い）ユーザの開発への積極的な関与を促進させる。

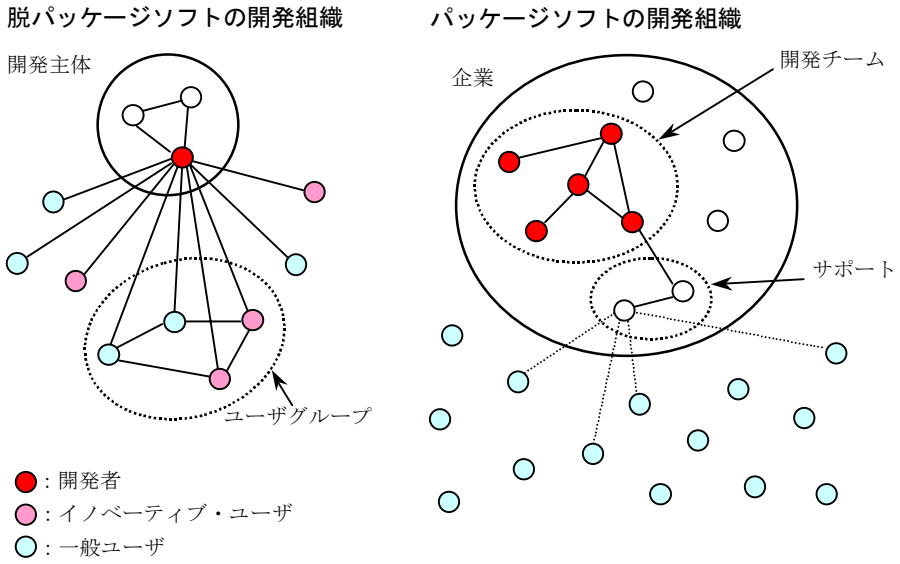
---

[15] 対照的にパッケージソフトでは、ソフトウェアの規模が大きいので、多くの人材と開発機材を抱え込むことになる。このため人件費も固定費用も膨れあがるし、製品をパッケージングして流通させるためのコスト負担もある。

[16] 鶴亀メールでも電信八号でも、開発者への受付窓口はインターネット上に設置された電子会議室やメーリングリストで、ユーザはそれらのシステムに自由に参加できる。これらのシステム上でユーザはバグ報告、修正や機能追加の要望を送信するだけでなく、他のユーザの発言も閲覧することができ、また開発者が自ら回答したりコメントを付していることも多い。実際にこれらのホームページを見てみると、膨大な数の意見や要望が提出されていることがわかる。

[17] ただし電信八号は、Linux のようないわゆるオープンソース・ソフトウェアではない。オープンソース・ソフトウェアの詳細については Raymond (1997)、高橋・高松 (2002)、Dibona et al. (1999) を参照のこと。

**図 2**　ユーザの組織化




つまり、① 脱パッケージソフトではユーザが開発者と 1 対 1 の関係を結べるのに対して、パッケージソフトではサポートセンターを介して間接的にしか結びつかないのである。そしてその関係が、② 脱パッケージソフトでは緊密で強いのに対して、パッケージソフトでは稀薄で弱いと考えられる。また、ユーザが開発者と直接やりとりできる脱パッケージソフトでは、③ 開発に積極的に関与するような、いわばイノベーティブ・ユーザが存在するが、パッケージソフトではそうしたユーザの存在は期待しにくい。電信八号の事例では、メーリングリスト参加者の中に優秀なデバッガが存在するとの報告があり、鶴亀メールでは、秀丸エディタも含めて非常に多くのマクロがユーザによって提供されていた。このようなユーザが③の例としてあげられるだろう。

こうしたサポート体制・情報提供における違いが、ユーザの組織化の程度を両者で異ならせていると理解することができる。脱パッケージソフトではユーザの開発への関与度が高く、開発組織が開発者個人やグループの境界を超えて多くのコンピュータ・ユーザにまで広がっている。他方、パッケージソフトではユーザの開発関与度は低く、開発組織は開発主体である企業の境界内で完結しているか、ごく一部のユーザがたまに参加するにとどまる（図 2）。

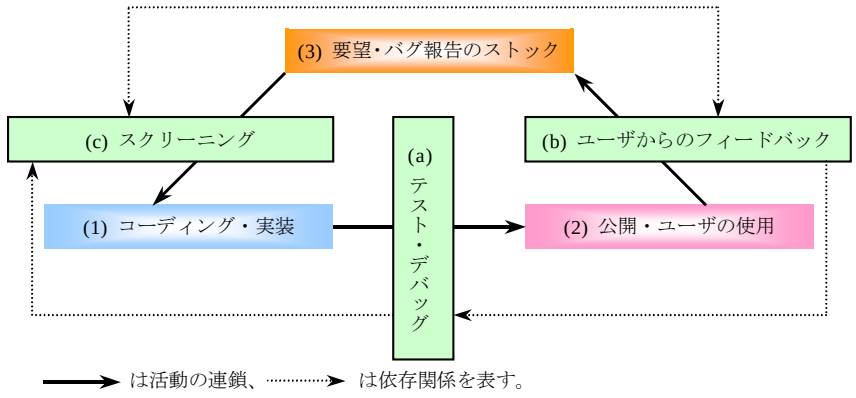
## 開発サイクルへの影響

最後に、ソフトウェアの機能向上やバージョンアップのペースである開発サイクルが、ユーザの組織化の程度や試行コストの構造によって、どのように変化しうるのかを検討しておこう。

およそソフトウェアの開発サイクルは、(1) コーディング・実装、(2) 公開とユーザの使用、(3) ユーザからの要望提出・バグ報告のストックという三つの活動から構成される。そしてそれぞれの活動の間に、(a) 動作テスト・デバッグ、(b) ユーザからのフィードバック、(c) 修正項目・追加機能のスクリーニングという活動が挟み込まれていて、(1)〜(3)の間のスムーズな移行を妨げる「関門」となっている。ソフトウェアのバージョンアップは、これら一連の活動を繰り返し循環させることによって進められる（図 3）。[18]

ところが脱パッケージソフトの開発サイクルでは、(a)〜(c)が関門にならなかったり、極端な場合はまったく存在しなかったりする。脱パッケージソフトでは、組織化されたユーザがフィードバックを寄せることで、開発者をバージョンアップに駆り立てる。つまりユーザから情報がプッシュされるため、それに応じてバージョンアップせざるをえないのである。[19] こうした情報のプッシュは、図 3 でいえば(2)と(3)の間を遮る可能性がある(b)が関門として機能せず、ユーザからの情報が迅速かつ直接的に要望やバグ報告のストックになり、その情報の重要性と膨大さがそれに続く(1)の活動を促すのである。

**図 3**　開発サイクル




[18] 既存のプロセスモデル研究で焦点が当てられていたのはこれらのうちの、(c) スクリーニング→ (1) コーディング・実装→ (a) テスト・デバッグ→ (2) 公開、という部分だけである。

[19] 事例で、鶴亀メールのバージョンアップがユーザからのフィードバックを契機に行われたこと、電信八号で石岡氏が開発意欲を低下させた時に、ユーザの情報交換を目的に設立された電八倶楽部が開発を引き継いだことを想起されたい。

このような情報のプッシュが行われると、情報をユーザからいかに集めるか、引き出すかということより、ユーザから寄せられた情報にいかに対処するかということがより重要になる。これは開発サイクルの一部をなす開発プロセスの始点・終点であり、また開発主体がコントロール可能な関門である(c)と(a)をどの程度厳密に行い、自然に流れてしまう情報の流れと、開発サイクルの回転速度をいかにコントロールするかを考慮する必要あるいは余地が生じることを意味する。[20]

このコントロールの強度は、脱パッケージソフトではソフトウェア像と試行コストの構造のために、非常に弱くなる傾向がある。ソフトウェア像が User-Supported Software であるため、ソフトウェアの機能と完成度における希求水準はそれほど高くならず、(a)はほとんど行われなくなる。また、試行コストが低いため、結果的には無駄になるような試作を行っても開発パフォーマンスの低下は起こりえないか、無視できる程度にとどまるので、(c)を実施する必要はあまりない。

つまり、脱パッケージソフトでは開発サイクルに関門がほとんど存在せず、ユーザから寄せられる情報のプッシュひとつずつにすばやく反応し処理する「情報の 1 個流し」をしているため、非常に速いペースで機能向上やバージョンアップが果たされる。そこで脱パッケージソフトの開発サイクルを「関門のない相対的に速い開発サイクル」と名づけることにしよう。

その一方で、パッケージソフトの開発サイクルでは、(a)～(c)が関門として厳然と存在する。ソフトウェア像は「製品」であるので、(a)を厳密に行ってソフトウェアの機能と完成度をできるかぎり向上させなくてはならない。またユーザの組織化の程度は低いので、ユーザからの情報のプッシュは弱くなる。つまり、(b)が関門として機能する可能性が高く、ソフトウェアの改良方針の設定や機能追加・バグ修正のリストアップに時間とコストがかかってしまうことになる。さらに試行コストの高さが、結果的に無駄になると予想される試作をできるだけ行わない方向に導くし、開発主体（企業内）の開発部門とサポート部門の組織構造上の分断が存在するので、ソフトウェアの概要設計・詳細設計のために(c)を厳密に行う必要が生じる。

それゆえパッケージソフトでは、ユーザから寄せられた情報をある程度ストックしてから

[20] 従来の製品開発研究では開発組織内でいかに情報を円滑に流すかに焦点が当てられていたことと、ここで述べた情報が勝手に流れる、流れてしまう開発組織のあり方とその運営は対照的である。脱パッケージソフトの事例では、当初の開発主体にとってコントロール不可能なユーザという存在が組織化されて開発に関与することによって、開発組織内に当初の開発主体がコントロールできない情報の流れが生み出され、それが開発組織全体の活性化、開発サイクルの変化につながっていると解釈することができるだろう。

処理する「情報のバッチ処理」を行わざるをえないし、開発サイクルの中に厳然とした関門が存在するために(1)〜(3)のフェーズの移行に時間がかかってしまう。その結果、機能向上やバージョンアップのペースが遅くなるので、パッケージソフトの開発サイクルは「関門のある相対的に遅い開発サイクル」と呼ぶのが適切であろう。

## 4. 結論

本稿では、既存研究でほとんど注目されることのなかった脱パッケージソフトを対象に、その開発サイクルを明らかにし、それとユーザとの関わり方を考察してきた。これまでにも開発主体（企業）とユーザとの関係、相互作用を扱っている研究は存在した。例えば、宮垣・佐々木 (1998) は、21 人のシェアウェア作者へのインタビューを通じて、その開発の動機、ユーザとの関わり方について記述し、一般的なソフトウェア製品とは異なるシェアウェアの位置づけ、存在の合理性を明らかにしている。佐々木・北山 (2000) は、Linux の開発コミュニティと企業の関係を記述・分析している。また野島 (2002) は、オンライン・ゲーム世界の中でユーザが形成するコミュニティを、企業の収益に結びつけるためには企業側にどのような取り組みが必要かを論じている。しかし、いずれもユーザあるいはユーザ・コミュニティとの関係の構築、維持にのみ関心が寄せられており、ソフトウェアの開発組織や開発過程がどのようなものであり、その中でユーザが具体的にどのような役割を果たしているかは明らかにしてこなかった。

それに対して本稿では、脱パッケージ化という意思決定は、単なる流通経路の選択にとどまらず、ソフトウェア認識と開発サイクルの選択であり、同時に流通経路の違いは、ユーザの組織化の程度、試行コストの違いを必然的に生じさせ、それがソフトウェアのバージョンアップ・ペースに結びつくことを明らかにした。また、ユーザの組織化の程度と試行コストは、ソフトウェアの流通経路を脱パッケージ／パッケージのいずれにするかを選択した時点で決まるので、もともとパッケージング販売されていたソフトウェアが脱パッケージ化してオンライン販売されたところで、その開発スタイルはいっさい変化しないことも理解される。

インターネットとブロードバンドが普及した現在では、インターネットを通じてユーザを組織化し、脱パッケージソフトのような「関門のない相対的に速い開発サイクル」によってソフトウェアを開発していくことが可能である。具体的には、ソフトウェア企業は脱パッケージ／パッケージの開発スタイルを組み合せることで、より効果的にソフトウェア開発・ビジネスを展開していくことができるだろう。例えば、ソフトウェアをコア・コンポーネント（本体プログラム）と、追加機能・不具合修正のためのサブ・コンポーネント（パッチ）とに大別し、各々が脱パッケージ／パッケージで提供される可能性を考えてみると表 3 のよう

**表 3**　ソフトウェアの開発・提供方法の組み合わせ

| コア | サブ | 例 | |
|---|---|---|---|
| パッケージ | パッケージ | ① 業務用基幹システム | Cusumano、Iansiti らの一連の研究 |
| | 脱パッケージ | ② Microsoft Windows、Office、一太郎 | |
| 脱パッケージ | パッケージ | ③ Linux ディストリビューション | 佐々木・北山 (2000) |
| | 脱パッケージ | ④ シェアウェア、フリーウェア | 本研究 |

になる。

従来ソフトウェア企業は、表3の①と②を中心にソフトウェア・ビジネスを展開しており、近年になってようやく③に取り組み始めた。しかし、④のような開発・提供方法もありうるべき選択肢であることが今回の事例からは示唆される。実はユーザの組織化による情報収集が、ソフトウェアの迅速な開発に非常に有効であることは、実務界でも認められ始めている。

しかし、残された課題が多いことも否定できない。今回の調査では、膨大な数の脱パッケージソフトの中から、たった二つの事例しか取り上げていない。したがって、この研究で発見された開発サイクルやその規定因について、より多くの事例収集や実態調査を通じて、妥当性・一般性を確認していかなくてはならない。

また、この論文で提示した二つの開発スタイルは、現時点で観察された事例から抽出された典型的類型であるということにも注意しておく必要があるだろう。現実にはパッケージソフトと脱パッケージソフトの開発サイクルの対応関係は、これまで議論してきたような明確な対応関係を持っていない可能性もあるし、将来その対応関係が変化するかもしれないからである。したがって今後の研究においては、ソフトウェア像や試行コストといった、開発サイクルの規定因に遡って、個々のソフトウェアおよびその開発サイクルを適切に位置づけることが必要であると考えられる。

## 謝　辞

本稿の作成にあたり、東京大学経済学研究科・現代企業ワークショップで発表を行った際に、貴重なコメントを頂いた。同ワークショップでコメントを下さった諸先生方に、深く謝意を表したい。

## 参考文献


青島矢一 (1997)「新製品開発研究の視点」『ビジネスレビュー』*45*(1), 161-179.

青島矢一, 延岡健太郎 (1997)「プロジェクト知識のマネジメント」『組織科学』*31*(1), 20-36.

Brooks, F. P., Jr. (1995). *The mythical man-month: Essays on software engineering* (Anniversary eds.). Reading, MA: Addison-Wesley. 邦訳, F･P･ブルックス (1996)『人月の神話』滝沢徹, 牧野祐子 他訳. アジソン・ウェスレイ・パブリッシャーズ・ジャパン (星雲社).

Cusumano, M. A. (1991). *Japan's software factories: A challenge to U.S. management*. New York: Oxford University Press. 邦訳, M･A･クスマノ (1993)『日本のソフトウェア戦略 : アメリカ式経営への挑戦』富沢宏之, 藤井留美 訳. 三田出版会.

Cusumano, M. A., & Selby, R. W. (1995). *Microsoft secrets*. New York: Free Press. 邦訳, M･A･クスマノ, R･W･セルビー (1996)『マイクロソフトシークレット―勝ち続ける驚異の経営』山岡洋一 訳. 日本経済新聞社.

Cusumano, M. A., & Yoffie, D. B. (1998). *Competing on internet time*. New York: Free Press. 邦訳, M･A･クスマノ, D･B･ヨッフィー (1999)『食うか食われるか―ネットスケープ vs.マイクロソフト』松浦秀明 訳. 毎日新聞社.

Dibona, C., Stone, M., & Ockman, S. (1999). *Open sources: Voices from the open source revolution*. Cambridge, MA: O'Reilly. 邦訳, C･ディボーナ, M･ストーン, S･オックマン (1999)『オープンソースソフトウェア』倉骨 彰 訳. オライリー・ジャパン.

Iansiti, M. (1998). *Technology integration: Making critical choices in a dynamic world*. Boston, MA: Harvard Business School Press. 邦訳, M･イアンシティ (2000)『技術統合―理論・経営・問題解決』NTT コミュニケーションウェア 訳. NTT 出版.

Iansiti, M., & MacCormack, A. (1997). Developing products on internet time. *Harvard Business Review*, *75*(5), 108-17. 邦訳, M・イアンシティ, A・マコーマック (2001)「インターネット時代の製品開発」D･タプスコット編, Diamond ハーバード・ビジネス・レビュー編集部 訳『ネットワーク戦略論』(3 章). ダイヤモンド社.

宮垣元, 佐々木裕一 (1998)『シェアウェア』(金子郁容 監修). NTT 出版.

延岡健太郎 (1996)『マルチプロジェクト戦略―ポストリーンの製品開発マネジメント』有斐閣.

野島美保 (2002)「コミュニティと企業戦略の適合モデル―オンライン・ゲームの産業の事例」『赤門マネジメント・レビュー』*1*(7), 1-33. 2005 年 1 月 22 日検索, http://www.gbrc.jp/GBRC.files/journal/amr/AMR1-7.html

Raymond, E. (1997) *The cathedral and the bazaar*. Retrieved January 22, 2005, from http://www.catb.org/~esr/writings/cathedral-bazaar/ 邦訳, E･レイモンド (1997)『伽藍とバザール』山形

浩生 訳. 2005 年 1 月 22 日検索, http://cruel.org/freeware/cathedral.html

佐々木裕一, 北山聡 (2000)『Linux はいかにしてビジネスになったのか―コミュニティ・アライアンス戦略』(國領二郎 監修). NTT 出版.

高橋伸夫 (2003)『経営の再生―戦略の時代・組織の時代―』(新版). 有斐閣.

高橋伸夫, 高松朋史 (2002)「オープンソース戦略の誤解―Linux はなぜ成功したのか」『赤門マネジメント・レビュー』*1*(4), 1-26. 2005 年 1 月 22 日検索, http://www.gbrc.jp/GBRC.files/journal/amr/AMR1-4.html

立本博文 (2002)「ソフト開発プロセスモデルと製品属性」『赤門マネジメント・レビュー』*1*(4), 309-336. 2005 年 1 月 22 日検索, http://www.gbrc.jp/GBRC.files/journal/amr/AMR1-4.html

von Hippel, E. A. (1988). *The sources of innovation*. New York: Oxford University Press. 邦訳, E・A・フォンヒッペル (1991)『イノベーションの源泉』榊原清則 訳. ダイヤモンド社.

von Krogh, G., Spaeth, S., & Lakhani, R. (2003). Community, joining, and specialization in open source software innovation: A case study. *Research Policy*, *32*(7), 1217-1241.

von Krogh, G., & von Hippel, E. A. (2003). Special issue on open source software development. *Research Policy*, *32*(7), 1149-1157.

Weick, K. E. (1979). *The social psychology of organizing* (2nd ed.). Reading, MA: Addison-Wesley. 邦訳, K・E・ワイク (1997)『組織化の社会心理学 (第 2 版)』(第 2 版の訳). 遠田雄志 訳. 文眞堂.

West, J. (2003). How open is open enough?: Melding proprietary and open source platform strategies. *Research Policy*, *32*(7), 1259-1285.

〔2005 年 1 月 25 日受稿; 2005 年 2 月 5 日受理〕

**赤門マネジメント・レビュー　4 巻 2 号** 2005 年 2 月 25 日発行

編集　東京大学大学院経済学研究科 ABAS/AMR 編集委員会

発行　特定非営利活動法人グローバルビジネスリサーチセンター

理事長　片平 秀貴

東京都千代田区丸の内

http://www.gbrc.jp