リクルーティングソース
XTLA プロジェクトで実践した、開発者勧誘に基づくプロジェクト運営手法

大和　正武
レッドハット株式会社
yamato@redhat.com

概要 「ソースコード公開」に基づくオープンソース開発手法を一歩進めた、「開発者勧誘」に基づく「リクルーティングソース」開発手法を提案する。リクルーティングソース開発手法は、筆者らがXTLAプロジェクトで用いたものであり、この開発手法でもって、類似するプロジェクト乱立のためプロジェクトに開発者を集めることができないという、現在、普及を目指すオープンソースプロジェクトの多くがかかえる問題を乗り越えることができた。「リクルーティングソース」開発手法を適用できるための条件は 2 つある: プロジェクトが属するソフトウェアのカテゴリーが若いこと、プロジェクトに属する開発者が開発目標を共有できることである。

1.　オープンソース開発手法の限界

ソフトウェア開発の手法として「オープンソース」は珍しいものでなくなった。オープンソース開発手法を、Eric S. Raymondは、その著書[1]で「目玉の数さえ十分あれば、どんなバグも深刻ではない」と説明している。ソースコードへのアクセスを特定の人に「制限」するのではなく、開発の過程からソースコードを「公開」して、ソフトウェアに関心のある誰もが開発に参加できるようにすることで、ソフトウェアの品質を向上させようというのが、オープンソース開発手法の考えである。Linuxカーネルを筆頭に多くの高品質なソフトウェアがオープンソース開発手法にて開発されているのは周知の通りである。

皮肉なことに、オープンソース開発手法の普及が、オープンソース開発手法の限界を露呈しつつある。オープンソース開発手法があまりに普及してしまったため、地球上のオープンソース開発者の数に対してプロジェクトが乱立するようになった。オープンソース開発手法によるプロジェクト運営を目指して、オープンソースの定義を満たすライセンス[2]にてソースコードを公開しても、現実にはプロジェクトあたりの「目玉の数」が増えないのである。Marius Andreianaは、2003 年 3 月に発表した「Too Much Free Software[3]」にて、Freshmeattに登録されているテキストエディタに関連したプロジェクトが 385 個もあるということについて嘆いている。現在(2005 年 3 月 7 日現在)では、619 個ある[4]。SourceForge.net の Software map[5]中でテキストエディタに分類されているプロジェクトは 2097 ある(2005 年 4 月 1 日現在)。GNU Emacsのかわりに使いたくなるエディタの開発プロジェクトをどうやって捜せば良いのだろうか？

プロジェクトが一つのカテゴリーに乱立すると、「目玉」の数がいつまでたっても 2 つというプロジェクトが出てくる。多数のプロジェクトの海に溺れいつまでたっても世に認知されなければ、開発は停滞し、最後に待っているのは閉鎖である。プロジェクト創始者に問いたい。開発者も利用者も自分一人より増える可能性がほぼ 0 という状況で、ソフトウェアをオープンソースにすることに一体何の意味があるのか？フリーソフトウェアの支持者には、次のように質問を変えても良い：誰にも利用も改変も再配布もされる可能性のないソフトウェアに対して、自由を保証することにいったいどんな意味があるのか？猫に言論の自由を保証する程度の意味しかないのではないだろうか。

プロジェクト創始者としてではなく、コミュニティのメンバーとして恐ろしいのは、(開発者を引き付けることができないために)カテゴリーに属する全てのソフトウェアが品質、機能の面で中途半端になってしまうことである。状況を打破すべく同カテゴリーであなたが新しいプロジェクトを開始しても、かなりの技術、時間、仕掛がない限り、突出することはできないだろう。カテゴリーが枯れてしまっているのである。

「Emacs上で動作するGNU Arch[6]へのインターフェース」という小さなカテゴリーにおいても、5 つのプロジェクト(tla.el、tilly.el、xxx.el[1]、mst-arch.el、xtla.el)が乱立した。開発者がプロジェクト毎に分散してしまいオープンソース開発モデルが成り立たなくなる、「Emacs上で動作するGNU Archへのインターフェース」が全て中途半端になってしまう、という危機が発生しかけたのである。xtla.el開発プロジェクト(XTLAプロジェクト[7])に参加した筆者とMatthieu Moy氏は、オープンソースにおける「ソースコードの公開」という手法を一歩推し進め、「開発者の勧誘」という手法でもってこの危機を乗り越え、xtla.elをカテゴリーキラーへ導いた。現在、xtla.elの開発者は7名である。ここにはtla.el、tilly.el、mst-arch.elのプロジェクト創始者が含まれている。筆者が調べた範囲では、xtla.elと同じカテゴリーに属するプロジェクトはわずかに一つ(xxx.el)である。

---

[1] 開発者の希望により仮名とした。

xtla.el の開発が一段落したとき、XTLA プロジェクトで用いた「開発者の勧誘」という手法は、オープンソース開発手法により自身のソフトウェアの普及を目指すプロジェクト創始者やその共同開発者にとって役に立つことに気付いた。そこで、XTLA プロジェクトで筆者と Matthieu Moy 氏が実践したことを「リクルーティングソース」開発手法として本稿で紹介する。まず 2 章で「勧誘」活動を支えた 2 つのソフトウェア(QuickML と GNU Arch)を説明する。3 章にて XTLA プロジェクトと同プロジェクトで実行した「開発者の勧誘」について具体的に述べる。ここでは勧誘の成功例だけでなく、勧誘の失敗についても、勧誘された側に対するインタビューにより詳しく紹介する。4 章にて「開発者の勧誘」を一般化して「リクルーティングソース」を提案する。5 章はまとめである。

2. 前提となったソフトウェア

XTLA プロジェクトにて「開発者を勧誘」する上で QuickML と GNU Arch が決定的な役割を果した。2 つのソフトウェアの知識は「リクルーティングソース」を理解するのに必須なので、ここで説明する。特に XTLA プロジェクトにおいて、GNU Arch には 2 つの位置付けがあるため、本稿を読み進める上で注意を要する。xtla.el は GNU Arch に対するフロントエンドなので、XTLA プロジェクトのメンバーにとって GNU Arch は、開発対象の一部と位置付けられる。一方、xtla.el のバージョン管理にも GNU Arch を用いたので、開発プロジェクト運営ツールとも位置付けられる。xtla.el の開発が進むと XTLA プロジェクトのメンバーは GNU Arch の使いこなしが向上し、プロジェクトの運営がより円滑になった。

2.1. 分散バージョン管理システム GNU Arch

GNU Arch は、Tom Lord 氏を中心に開発されてきた、プログラムソースコードのバージョンを管理するためのソフトウェアである。管理の対象となるのは、プログラムソースコードに限らないが、プログラムの開発において特に有用な機能を多数持つ。プログラムの開発者は、開発の進捗が一段落するたびに、変更点を記したログとともにソースコードの現状を保管庫(GNU Arch ではアーカイブと呼ぶ)に記録する。開発者は一度記録したソースコードを、いつでも保管庫から取り出すことができる。あるいは現在変更中のソースコードと過去に記録したソースコードとの差分を取り出すといったこともできる。ここまでは、保管庫のことをリポジトリと呼ぶことを除いて、バージョン管理システムとして既に広く利用されている CVS と GNU Arch に大差はない。

CVS と GNU Arch の決定的な違いは、CVS が 1 つの開発プロジェクトに対して、原則一つのリポジトリを用いるのに対して、GNU Arch では、一人の開発者に対して、一つのアーカイブを用いる点である。言い換えると CVS では、開発者どうしで、保管庫を共有するが、GNU Arch では、共有しない。アーカイブを共有しないかわりに GNU Arch は 3 つの興味深い機能を持つ:

(F1) http、ftp でアクセス可能な個所に自分の所有するアーカイブのミラーを設置する。

(F2) ネットワーク上で公開している他の開発者のアーカイブから派生させて独自のアーカイブを作成する(ブランチの作成)。

(F3) ネットワーク上で公開している他の開発者のアーカイブから変更点を取得して自分のソースツリーに適用する(チェンジセットの取得)。

(F1)を用いて公開されているアーカイブに対して(F2)、(F3)を実行するためにアーカイブ管理者から許可を得る必要はない。

従来 CVS で管理されている開発プロジェクトに関わろうとすると、CVS リポジトリへのアクセス権を取得することが極めて重要であった。CVS リポジトリへのアクセス権を持たずにオリジナルのソースコードへの大幅な変更(ハック)を実行すると二つの問題が発生する: そのハック自体のバージョン管理が困難であること、CVS リポジトリ内のオリジナルのソースコードに変更があった場合、ハックとの同期を取るのが困難であること。一方アクセス権を取得するには、何がしかの開発成果を提示して、プロジェクトメンバーからの信頼を勝ち取る必要があった。プロジェクトメンバーは、その開発成果を検討して、コミット権を発行すべきか、すべきでないか判断する。技術的にあるいは人間的に信頼のおけない人にコミット権を与えて、リポジトリが破壊されてしまうのを防ぐためである。開発に参加したい人間(参加希望者)からみると、ここに CVS を用いたオープンソース開発の矛盾がある。大きな開発成果を提示してコミット権を得たいのだが、大きな開発成果を出すためにこそコミット権が必要なのである。

GNU Arch を用いた場合、(F2)を用いてブランチを作り、独自開発作業を進めつつ、(F3)で、オリジナルのソースコードの変更を取り込み続けることができる。時にはコンフリクトが発生するかもしれないが、コンフリクトへの対応も含めてバージョン管理を行なうことができる。最後に独自開発作業の成果を(F1)により公開し、プロジェクトメンバーへマージ要請を出すことができる。例えマージ要請が拒否された場合でも、作業を継続、公開し、ユーザの評価を待つことができる。

筆者が GNU Arch に強い関心を持ったのは、gs-cjk プロジェクトで苦い経験をしたためである。5 年ほど前、ghostscript における TrueType フォントの取り扱いを拡張するプロジェクト gs-cjk に筆者は参加していた。成果物(gs-cjk パッチ)をマージしてくれるよう ghostscript の作者達と交渉したのだが、拒否されてしまった。GNU Arch がなかった当時、CVS で管理されている ghostscript の更新に gs-cjk パッチを追従させることができず、結局プロジェクトは休止してしまった。当時 GNU Arch があったなら ghostscript の更新に同期した gs-cjk パッチの公開を続けることできたかもしれない、と思うと悔しい。

2.2. QuickML

QuickML[8]は、高林哲氏と増井俊之氏によって開発されたメイリングリストシステムである。同システムを使ったメイリングリストサーバがquickml.comにて運営されてい

る。QuickMLをダウンロードして、自分の管理するコンピュータにインストールすれば、quickml.comと同じ機能を持つメイリングリストを提供できるが、XTLAプロジェクトにて使ったのはquickml.comなので、以降QuickMLという言葉とquickml.comを区別しない。
QuickML の特徴は、メイリングリストの作成やメンバーの追加といったメイリングリストへの操作をメールの送信だけを用いて簡単にできることである。例 1 にメイリングリスト作成の例を示す。To:に「作成したいメイリングリストの名前@quickml.com」を指定しメールを送信すると、Cc:と From:に記載したメールアドレスを初期メンバーとするメイリングリストが作成される。例では、jet@gyve.org、tank@example.com、sub@example.net をメンバーとするメイリングリスト recruitingsource@quickml.com を作成している。以降 recruitingsource@quickml.com の送信は jet@gyve.org、tank@example.com、sub@example.net に配信される。

例 1. QuickML におけるメイリングリストの作成

```
Subject: ミーティング
To: recruitingsource@quickml.com
From: jet@gyve.org
Cc: tank@example.com, sub@cracker.net
本文...
```

例 2 はメンバー追加の例である。既にメイリングリストのメンバーであれば誰でも、Cc: に追加した人のメールアドレスを指定してメイリングリストにメールを送ることでメンバーを追加できる。例では、recruitingsource@quickml.com にsam@punk80.comを追加している。

例 2. QuickML におけるメンバー追加

```
Subject: メンバーを追加します
To: recruitingsource@quickml.com
From: jet@gyve.org
Cc: sam@punk80.com
本文…
```

3. XTLA の開発過程

XTLA プロジェクトのメンバーである筆者の視点から XTLA の開発過程を時間順に説明する。

3.1. XTLA への参加

Emacs は、さまざまな外部コマンドに対するフロントエンドを提供している。pcl-cvs もその一つで、cvs コマンドに対するものである。筆者はpcl-cvsを大変愛用している。cvs のコマンドラインを覚えていないのに、オープンソースソフトウェアの開発作業を行なうことができているのは、pcl-cvs のおかげである。そんな筆者が GNU Arch に関心を持ったとき、まず気になったことは Emacs が GNU Arch に対するフロントエンドを持っているかどうかだった。
"GNU Arch"、"Emacs"、"frontend"、"interface"などの単語を使って Google で検索したところ、なんと 3 つも開発プロジェクトが存在することがわかった。Milan Zamazal 氏の tla.el、Martin Pool 氏の tilly.el、Stefan Reichör 氏の xtla.el である。検索したおかげで、筆者が地球上で 4 つ目の開発プロジェクトを開始せずに済んだ。
3 つのソフトウェアをダウンロードして比較してみると、xtla.el が自分の必要としているものに近かった。GNU Arch の持つ機能のうち inventory(作業ディレクトリ中のファイルの一覧を列挙する機能)に対するフロントエンドが、pcl-cvs 似たユーザインターフェースで実装されていたのである。さらに Reichör 氏が Emacs 上で動作する Subversion バージョン管理システムのフロントエンド psvn.el の作者であることも心を捉えた。彼もきっと pcl-cvs を愛用しているに違いない。不満な点もあった。他人のアーカイブをブラウズする機能(abrowse)へのインターフェース(アーカイブブラウザ)が未完だった。「分散」バージョン管理システムの便利さと複雑さは、他人のアーカイブとのやりとりの過程で発生すると考えていた筆者は、アーカイブブラウザを非常に重要視していた。
自分で xtla.el を拡張してアーカイブブラウザを実装するか、Reichör 氏が実装するのを待つかで迷った。未完であるとは言え Stefan Reichör 氏が既にアーカイブブラウザ用のコードを持っていたため、しばらく待っていれば良いものに仕上がる可能性があったためである。一方で、Reichör 氏の実装を待っているうちに tla.el、tilly.el のいずれかが先に実装してしまわないかと不安になった。inventory のフロントエンドには xtla.el を、abrowse のフロントエンドには別の elisp ライブラリを使わなければいけなくなってしまう。2 つの elisp ライブラリの開発が進み、フロントエンドとしてカバーする GNU Arch の機能が重なり始めると、両方のライブラリを同時に使うことが困難になるだろう。
不安が現実となるのを避けるため、待つのをやめて自分でアーカイブブラウザを実装することに決めた。まず Reichör 氏に自分の存在を認識してもらうために、簡単な自己紹介のメールを送り(2004 年 3 月 15 日)、その後小さな不具合の修正を送ってみた(2004 年 3 月 18 日)。Reichör 氏はすぐに修正をマージした上で、新しいバージョンをウェブ上に公開してくれた。以降、2004 年 3 月中はほぼ毎日パッチを Reichör 氏に送り続けた。当時 Reichör 氏は xtla.el のバージョン管理のためにローカルに GNU Arch を使っていたが、匿名アクセスできるミラーの設置はしていなかった。筆者も xtla.el の開発作業を通して GNU Arch の使い方を学んでいるような状態だったので、ウェブ上で新しいバージョンが公開されるたびにダウンロードして、パッチを作成していた。

3.2. メイリングリスト始動

xtla.el の作業を数日続けると、自分が関わった以上 xtla.el が普及して欲しいと思うようになった。当時 xtla.el は Reichör 氏のページでそのソースコードが公開されているだけだった。メイリングリストなどは存在せず、Reichör 氏と筆者のやりとりもプライベートなメールで行なわれていた。xtla.el は既に役に立つレベルに到達していたので、例えばメイリングリストを作成し開発者間の議論を公開してアクティビティの高さを見えるようしたり、利用者からの質問を受け付けるようにしたりして、xtla.el

のユーザを増やす活動を始めても良い時期が来たと感じた。メイリングリストの作成を Reichör 氏に提案すると、彼は savannah.gnu.org(Savannah)にプロジェクトページの開設依頼を出した(2004 年 3 月 21 日)。よりオープンソース開発手法に沿ったプロジェクト運営に移行しようとしたのである。
ところが、Savannah での審査が遅れてプロジェクトページがなかなか開設されなかった。筆者は苛立ちを感じたが、かといって Savannah のかわりに sourceforge.org にプロジェクトページの開設依頼を出すよう Reichör 氏に頼むのは気が引けた。2004 年 3 月 15 日以来、彼には毎日しつこくメールを送っているので、これ以上やると嫌がられるのではないかと心配だった。かといって Reichör 氏にかわって自分で開設依頼を出すのもいやらしい。
仕方がないので応急処置として QuickML を使うことにした(2004 年 4 月 10 日)。

```
Subject: mailing list
From: Masatake YAMATO <jet@gyve.org>
To: Matthieu.Moy@imag.fr, stefan@xsteve.at
Date: Sat, 10 Apr 2004 05:11:12 +0900 (JST)

Bonjour and Guten tag!

I can set up TEMPORARY mailing list for xtla.el
quickly.  Stefan  already sent a project
setup request to Savannah.  So the TEMPORARY
mailing list is alive till the project at
savannah is set.
Are you interested in?
```

QuickML で作ることになったメイリングリストをこのメールでは TEMPORARY mailing list と書いている。
このメールで、本稿の主題である「勧誘」を無意識に実行していた: Reichör氏(stefan@xsteve.at)に加えてMoy氏(Matthieu.Moy@imag.fr)をTo:に指定してメイリングリスト作成を提案している。2004 年 3 月中のxtla.elへの作業中は、(メイリングリストもなかったので)Reichör氏とのやりとりのみで、Moy氏にメールを送ったことはなかった。しかし 2004 年 3 月 15 日に初めてxtla.elをダウンロードしたときには、すでにソースコード中に貢献者として彼の名前があったので彼の存在を知っていた。Reichör氏のウェブサイト[9]を見ると筆者よりも 6 日ほど前から、xtla.elの作業していることがわかる。

```
18.03.2004: New revision of xtla.el: patches
from Masatake YAMATO Use defcustom, new
command: xtla-register-archive

12.03.2004: New revision of xtla.el: Some
minor bug fixes and improvements from
Matthieu Moy
```

メイリングリストを作成するにあたって彼を勧誘したことは正しかった。開発者 1 名ではじまり、そのまま消え行くプロジェクトが多数あるなかで、メイリングリスト開設時に開発者が 3 名いるとは心強い。
TEMPORARY mailing list の開設について Reichör 氏からも Moy 氏からも賛同を得た。Moy 氏からの返事を、本人の許可を得たので引用する。

```
Subject: Re: mailing list
From: Matthieu Moy <Matthieu.Moy@imag.fr>
To: Masatake YAMATO <jet@gyve.org>
Date: Fri, 09 Apr 2004 22:20:23 +0200

Masatake YAMATO <jet@gyve.org> writes:

> I can set up TEMPORARY mailing list for
> xtla.el quickly.  Stefan
> already sent a project setup request
> to Savannah.  So the TEMPORARY
> mailing list is alive till the project
> at savannah is set.
> Are you interested in?

We're not many developers actually ;-) I think
this can wait until we have a project on
Savannah...

But *if* you set up the ML, yes, I'm
interested.

--
Matthieu
```

このやりとりの後、3 人を初期メンバーに xtla@quickml.com を開設し、QuickML の使い方をメイリングリストに流した。Moy 氏は、たった 3 人の開発者から成るプロジェクトにメイリングリストが必要かという疑問を抱いたようだが、メイリングリストの作成により 3 人の間で意見をやりとりできるようなったので、開発は勢いづいた。他の開発者ががんばっているのを見るだけで、やる気が湧いてきた。
この時期メイリングリストの開設と並行して各開発者が自身のアーカイブのミラーを設置した(2004 年 4 月 8 日から 2004 年 4 月 15 日まで)。開発者がアーカイブのミラーを設置すると、間もなくミラー機能に対するフロントエンドが xtla.el に追加された。xtla.el の分散バージョン管理を xtla.el 自体でできるようになった。そのためミラーの設置以降、メールでパッチをやりとりしなくなった。xtla.el を介して GNU Arch のチェンジセット転送機能(F3)を使うようになった。

3.3. 開発者の勧誘

3 人による共同開発を支えることができる程度に xtla.el は成長した。2004 年 3 月 19 日には xtla.el は 659 行だった。これが 2004 年 4 月 19 日には 3403 行にまで増えていた。行数の増加が直接プログラムの成長を意味するわけではないが、Reichör 氏に始めてパッチを送ったときと比べると xtla.el はかなり機能が向上していた。
xtla.el は十分高機能なので、もっと広くの人に使われても良いはずである。もっと xtla.el をアピールしないといけない。誰にアピールすれば良いだろうか考えた。そもそも誰にとって xtla.el は便利だろうか。Emacs で動作する GNU Arch のフロントエンドを欲しいと強く思っている人たちは誰か？xtla.el の価値が分かる人は誰か？それは tla.el の作者 Milan Zamazal 氏と tilly.el の作者 Martin Pool 氏であると気付いた。彼らは自分でフロントエンドが書き始めてしまうほど、GNU Arch のフロントエンドを必要としている人達である。機能が向上した xtla.el の価値をもっとも良く理解する人達に違いない。
2004 年 4 月 19 日、まず Reichör 氏と Moy 氏を Cc:に入れて Zamazal 氏に XTLA プロジェクトへの勧誘のメールを送った。

```
Subject: Merging tla.el and xtla.el
From: Masatake YAMATO <jet@gyve.org>
To: Milan Zamazal <pdm@zamazal.org>
Cc: Matthieu.Moy@imag.fr, stefan@xsteve.at
Date: Mon, 19 Apr 2004 20:10:02 +0900 (JST)

Dear, Milan Zamazal

We are interested in your tla.el. We think tla
support in GNU Emacs is extremely important
for OpenSource and Free software. As you may
know, many developers like pcl-cvs but cvs
itself. The situation is the same to tla/arch.

We are developing xtla.el, another emacs
interface for tla.  I would like to merge
yours effort for tla.el and that for xtla.el
into ONE.

Stefan Reichoer<stefan@xsteve.at>, the
original author of xtla.el already sent a
request to create a project to savannah.
However, it is not approved by savannah
administrator. So I, Masatake YAMATO
established temporally mailing list for
discussing emacs interface for tla/arch. If
you have interesting, please tell me.

Register the following location to get
xtla.el: http://xsteve.nit.at/prg/emacs/

Regards,
Masatake YAMATO
```

xtla.el と tla.el は同じカテゴリーに属しており、互いにユーザを奪い合う関係にあるので、もちろん Zamazal 氏からひどい怒りを買う可能性もあった。そこでいきなり QuickML を使って Zamazal 氏をメイリングリストのメンバーに追加することはしなかった。
その日のうちに Zamazal 氏から、tla.el と xtla.el の統合を快諾する返事が来た。以降 xtla.el を使い、フロントエンドの開発作業は xtla.el に対して行なうと書いてくれた。Zamazal 氏を Cc:に、To:に xtla@quickml.com を指定して返事を書いた。QuickML のおかげで、メイリングリストへ参加するための手間を Zamazal 氏に一切かけさせなかった。1 往復のメールのやりとりが、tla.el のコードと tla.el を書けるほどに elisp と GNU Arch について理解している開発者を XTLA プロジェクトにもたらしたのである。
QuickML は、勧誘される側だけでなく、勧誘する側にも都合が良かった。QuickML で運営されるメイリングリストにおいては、他の参加者の断りを待つ必要なくだれもが管理者としてメンバーを招く権限を持つ。
筆者の Zamazal 氏勧誘が 1 日で成功したことに刺激を受けたのか、今度は Moy 氏が Pool 氏に XTLA プロジェクトへの勧誘のメールを送った。文面を引用しないが、既に勧誘にのってくれた Zamazal 氏の名前を入れこんで、GNU Arch に対するフロントエンドのプロジェクト統合が進んでいることを匂わせている。Pool 氏もその日のうちに XTLA プロジェクトに合流することになった。2004 年 4 月 21 日には、Reichör 氏のアーカイブからブランチした、Pool 氏の新しいアーカイブの所在がメイリングリストに投稿された。
この頃、XTLA プロジェクトのページが gna.org(Gna!) に開設された。Savannah での審査の遅れにしびれを切らした Reichör 氏が gna.org(Gna!) にプロジェクトページ開設の申請を出していたためである。QuickML から Gna! にメイリングリストは移行したが、Google で同じカテゴリーのソフトウェアを捜して、その開発者に XTLA を紹介、統合を申し込むという「勧誘」作業を続けた。
2004 年 5 月 16 日には、mst-arch.elの作者であるMark Triggs氏が筆者の勧誘にのってくれた。 2004 年 5 月 17 日には Moy 氏が、nbbba.el の作者である Christian Neukirchen氏を勧誘している。nbbba.elは、主にアーカイブブラウザを実装していた。Neukirchen氏は統合に関心を示してくれたが、xtla.elのアーカイブブラウザを紹介したところ、xtla.elが彼の要望を満してしまっていたのか[10]、彼がXTLAプロジェクトで活躍することはなかった。xxx.elの作者であるFoobar氏(仮名)も勧誘してみたが、残念ながら断られてしまった。Foobar氏の失敗については、3.6にて詳しく述べる。

3.4. 無秩序開発

3.3にてどのようにQuickMLを活用したかを述べた。ここではGNU Archの活用について述べる。
XTLAプロジェクトでは、GNU Archの「分散」機構を乱用していた。xtla.elの作業を通じてGNU Archの使い方を学んでいたような状態の筆者はこのことにまったく気付いてなかったが、GNU Archを良く理解していたMoy氏はメイリングリストで指摘している[11] (筆者による訳、括弧の注も筆者)。

> 実際のところ、我々は `tla` の分散機構を乱用している。`Tom` が `tla` 自体(の開発)でやっているように、一人の「主」コミッターが一つの「主」アーカイブを持つのが普通のやり方だ。他の貢献者は「主」アーカイブだけからマージを実行する。
> 我々(`XTLA` プロジェクトの)分散開発のやり方は、もう少し「無秩序的(`anarchic`)」だ。みんなが互いにマージを実行し、例えば互いに関係の無い多数のパッチを一度にマージしてしまうなど、「大きい」マージを実行している。結果として我々の `ChangeLog` はあまり明瞭でない...
> 我々は、(「無秩序的」な開発を)続けるべきだと思う。興味深い実験だ。開発が安定期に入り、機能の破綻が致命的になるので、`1.0` をリリースした後に考え直す必要があるだろう。

ここで Tom とは、GNU Arch の創始者 Tom Lord 氏のことである。「主」アーカイブを持つ、Lord 氏も実践している GNU Arch の使い方を星型開発(Star-shaped development)、この時期XTLAプロジェクトで採用していた GNU Arch の使い方を無秩序開発(Anarchic development)と、XTLA のマニュアルでは呼んでいる。
無秩序開発にはトラブルが多かった。二人の開発者が同じ個所を変更してしまい不整合が発生したり、似た機能を持つ関数を二人の開発者が別々に定義してしまい「車輪の再発明」が発生したりした。互いにマージするのでトラブルがアーカイブの間を伝播した。こういったトラブルを 7 名[2]の開発者が頻発したのである。まさに無

[2] 2004 年 5 月に XEmacs への移植に関心を持つ Robert

秩序だった。
しかし誰も無秩序開発をやめようとは言わなかった。すでに述べたように XTLA は展開の速いプロジェクトだった。開発者が全て Emacs ユーザだったため pcl-cvs という既存のソフトウェアを共通の目標に定めることができたこと、Reichör 氏、Moy 氏、筆者がコードを書きまくっているところへ、さらに tla.el、tilly.el というフロントエンドの開発経験者が参加したことが原因である。無秩序開発は「主」アーカイブというボトルネックの設置を避けることで、ソフトウェアの品質を犠牲にするかわりにこの速い展開を維持していたのである。星型開発に乗り換えて「主」アーカイブを設ければ、そのアーカイブの管理者である「主」開発者の作業が、プロジェクトのスピード感を殺してしまったに違いない。「主」開発者も作業時間の多くをマージ作業に取られてしまい、自分の欲しい機能の実装のために時間を減らす必要が出てきただろう。我々は手元で頻繁に発生する小さなトラブルを我慢して、プロジェクトの勢いを優先した。
Moy 氏が主に開発を担当した xtla-bookmark が「無秩序開発」を一層促進した。XTLA プロジェクトのメンバーが公開するアーカイブを巡回し、自分のアーカイブにない他の開発者が追加した変更(チェンジセット)を列挙する機能を xtla-bookmark は提供する。さらに利用者が望めば、キー操作一つで列挙されたチェンジセット全てを転送して自分の作業ディレクトリに取り込むことができる。xtla-bookmark のおかげでトラブルが微少化した。頻繁にマージして他の開発者の作業を知ることができるようになったので、トラブルが大きくなる前に、マージした側の開発者がトラブルの小さくなる方向に作業をしたためである。

3.5. 星型開発へ

各開発者がGNU Archのフロントエンドに欲しいと考えていた機能を一通り実装してしまうと、xtla-0.9 をリリースしようという話題がメイリングリストに投稿されるようになった。そして機能追加から不具合の修正に注力することになった。「主」アーカイブの管理作業をMoy氏が買って出たため、遂にXTLAプロジェクトは、星型開発へ移行した。2004 年 11 月 7 日のことである。現在、1.0 に向けた作業がゆっくりと進行している[3]。

3.6. 勧誘失敗に関するインタビュー

この論文の作成にあたり、うまく勧誘できなかった Foobar 氏にメールにてインタビューを試みた。勧誘活動の裏舞台としてインタビューの内容をここにまとめる。勧誘してからインタビューまで時間を経ているので、インタビューする側、される側の記憶が正確でない可能性がある代わりに、冷静な応答がなされた。

Q *勧誘されてどのように感じたか？*
— 「これは率直に申し上げて、その時はうれしかったです。」、「あとは、自分の elisp に自信がなかったものですから失礼ですが、『はたしてコードを見た上で誘ってくださったのかな？』とも感じました。」

実は筆者は、勧誘以前にコードを読んでおりコード中に示されていた Emacs とフロントエンドの融合の方針に魅力を感じていた。しかし、勧誘にあたり、技術的な詳細に触れすぎると、ストーカーっぽく気持ち悪がられるリスクがあるので勧誘の文面には記載しなかった。

Q *勧誘をお断りになった理由*
— 「勧誘頂いた当時、本当に時間がありませんでした。」、「アクティビティの高い（XTLA の）開発を追えない」、「elisp の技術に自信がなく、elisp の学習のために xxx.el の開発をしている。」、「FSF への著作権譲渡の手続きが負担に感じる。」、「xtla.el と xxx.el のデザインに差異がある。」

勧誘により開発を募ることで強烈な勢いを得ていた XTLA プロジェクトは、Foobar 氏の目から見ると勢いがありすぎて参加しづらかったようである。しかし勧誘の文面に「気ままに参加できます」と嘘を書くわけにはいかなかった。
XTLA プロジェクトでは xtal.el を GNU Emacs の一部に含めようと考えていたので、著作物の権利を FSF へ譲渡することに同意した貢献者からのみコードを受け取ることにしていた。勧誘時に著作権譲渡のことを文面に含めていた。
xtla.el と xxx.el のデザイン上の差異というのは次のようなものである：xtla.el が多人数の開発者をかかえるプロジェクトにて GNU Arch を使うことを重視してデザインされていたのに対し、xxx.el は、一人（Foobar 氏本人）が複数のアーカイブを使い分け、例えばネットワークから切り離された環境でもバージョン管理システムの恩恵にあずかれることを重視してデザインされた。2つのデザインは矛盾するものではないが、xtla.el のデザインは Foobar 氏のニーズを超えていた。

Q *Foobar 氏の勧誘を成功させる方法はあったでしょうか？*
— 「『なかった』が答えです。『無秩序開発』に協力する技量がないことは自分が一番よくわかっていた、というのがその理由です。」

Foobar 氏が「なかった」という以上、勧誘を工夫しても無駄だったわけであるが、それでも勧誘される側には、プロジェクトへ参加することにはそれなりのプレッシャーがあることが分かった。
具体的な質問からではないが、Foobar 氏とのメールのやりとりからわかったのは、氏が自身のソフトウェアの普及にはあまりこだわってはいないことである。「もし役立つならば、ご自由に使うなり取り込むなりしてくださいという軽い気持ちでした。」と述べている。著作権譲渡について考慮の必要がなければ、勧誘が失敗してもライセンスの範囲でソースコードを流用できた。XTLAプロジ

Widhopf-Fenk 氏が(自主的に)プロジェクトに参加した。勧誘によるものではない。
[3] 本稿のカメラレディを作成中の 5 月 2 日に 1.0rc1 がリリースされた。

ェクトには他のプロジェクトからソースコードを流用するだけのリソースがあった。

4. リクルーティングソース

XTLA プロジェクトで実行した「勧誘」を一般化して、自身のソフトウェアの普及を目指す開発者にリクルーティングソース開発手法を提案する。

「オープンソース開発手法」が特殊化したものが「リクルーティングソース開発手法」である。「オープンソース開発手法」では、ソースコードを「公開」して、開発者、利用者がプロジェクトに関心を持ってくれるのを「待つ」。「リクルーティングソース開発手法」では、プロジェクトに関心を持ちそうな開発者を「捜して」、参加を「勧誘」する。

次のように飲食店なりの運営に例えることができる: 「オープンソース開発手法」は入店しやすくなるよう店の入口を小綺麗にしてお客さんを待つ。「リクルーティングソース開発手法」では、街頭なりに呼び込みを配置して、お客さんなってくれそうな人に声をかけて店に誘導する。もちろん呼び込みを配置しつつ、店の入口が小綺麗にしていても何の不具合もない。

4.1. 勧誘時の注意

「勧誘」にあたって重要なことは、まず人選である。プロジェクトにとって真に力となってくれそうな開発者を選ぶべきである。単に開発力があるというのではダメである。プロジェクト特有の知識、技術を有しているかどうか、開発目標を共有できるかどうかで判断すべきである。そういった人であれば、「勧誘」にのってくれる可能性が高い。Zamazal 氏や Pool 氏がまさにそういう人だった。彼らは自身で GNU Arch のフロントエンドの開発経験があり、Emacs の利用者であるため pcl-cvs について知っていたと考えられる。単に開発力があるというだけで、ランダムに声をかけても無駄である。

プロジェクトの適正度を測るのは簡単でない。筆者があえて声をかけなかった人が二人いる。Stefan Monnier氏とMiles Bader氏である。二人ともGNU Emacsの開発者である。Monnier氏はpcl-cvsの開発者で、現在もメンテナンスを続けている。Bader氏はGNU EmacsのCVSリポジトリをミラーしたGNU Archアーカイブを運営している[12]。Bader氏は、GNU ArchベースでGNU Emacsの開発作業をしているようである。実績を見ればXTLAプロジェクトにうってつけである。しかし直接には、「勧誘」しなかった。まず二人共GNU Emacsの作業で忙しいと考えられ、展開の速いXTLAプロジェクトに参加してくれるようなことがあれば、GNU Emacsの作業がおろそかになるだろうと思った。GNU Emacsの品質が下ってしまっては、元も子もない。筆者がGNU Emacsのメイリングリストに投稿したパッチを、Monnier氏はみっちりレビューしてくれる。無秩序開発の実体を知ってしまったら二度とパッチをレビューしてもらえなくなりそうだった。Bader氏については、何もしなくてもXTLAプロジェクトに参加してくれると考えていた。氏がGNU Arch、GNU Emacs両方に強い関心があることがわかっていたからである。しかし結局 8 本のパッチがMoy氏経由でxtla.elにマージされたこと以外、特にプロジェクトに強い影響を及ぼさなかった。

次に「勧誘」にあたって重要なことは 勧誘の言葉である。「似たようなもの作っているのでマージしよう」といっても勧誘にはのってもらえないだろう。一般にハッカーは技術的な利害を冷静に判断する能力を有しているが、それでも自分の作っているものには愛着が湧いてしまうものだ。マージを提案する以上、マージの利を相手にわかるよう説明する必要がある。Foobar 氏のインタビューにあるとおり、自分の成果物に関心を持つ人からのコンタクトは喜ばれる可能性は十分にある。その関心の強さは積極的に示した方が良いようだ。

一方でプレッシャーをかけ過ぎてもいけない。勧誘対象者がオープンソースプロジェクトでの経験を多く持つ場合、適度なプレッシャーは参加意欲を高める効果があろうが、経験が少ない場合、逆に「引いて」しまうことになろう。

巨大なコードベースを背景にマージを提案するのは間違っている。単なる新規参入者に対する妨害である。巨大なコードベースを持つ以上、既にそのプロジェクトは世に名が知れている可能性が高い。それにもかかわらず勧誘相手が独自実装を進めているのは理由があるに違いない。その理由を良く考慮して勧誘すべきである。また巨大なコードベースが既にあるなら、「オープンソース開発手法」で十分かもしれない。xtla.el もすでに巨大なコードベースを持つに至ったので、「勧誘」しなくなった。

4.2. 勧誘した後の注意

「勧誘」にのってくれた後に注意すべき点は、新しい開発者の勢いを殺さないことである。複数の共同開発者からの「勧誘」というイベントを前に、新しい開発者はプロジェクトへの関心と活躍してやろうという気持ちが高まっているはずである。その勢いを存分にコーディングにぶつけてもらわないといけない。勧誘した側は勢いを削ぐ要因を積極的に排除しておくべきである。

XTLA プロジェクトでは、QuickML と GNU Arch が新しい開発者の勢いを後押しした。 QuickML のおかげで、新しい開発者の手間を煩わせることなく瞬時に新しい開発者を議論へ参加させることができた。勧誘者側が説明しなければ、新しい開発者はいつ自分がメイリングリストへ参加することになったのが気付かなかったと思う。

GNU Arch を用いて無秩序開発を行なったおかげで、新しい開発者も既存の開発者も好きなペースで作業でき、互いにボトルネックとならなかった。星型開発では、パッチが「主」アーカイブに統合されるのを待っているうちに勢いが失われる恐れがあった。

もちろん勧誘する側にとっても勧誘の手間は少ないほうが良い。勧誘の手間が大きければ勧誘する側の勢いを削いでしまう。こ QuickML により既存の開発者は誰でもメールを一通送るという手間で、いつでも勧誘対象者をメイリングリストへ招くことができた。QuickML がなければ、メイリングリスト管理者、勧誘する開発者、勧誘された開発者の間で合意を取るためのメールを書く手間で勧誘する気が失せたであろう。

4.3. 適用可能範囲

リクルーティングソースという開発手法が適応可能なの

はどういう場合か。まず、そのプロジェクトの属するカテゴリーが新しい必要がある。あたりまえのことであるが、カテゴリーに属する全てのプロジェクトが若いことが保証できるからである。勧誘対象の開発者が運営するプロジェクトが若ければ、マージが困難になってしまうほど開発は進んでおらず、しかも開発者が抱くコードへの愛着も低いと期待できる。また成熟したプロジェクトが若いプロジェクトを勧誘すれば、それは単なる脅しであり、若いプロジェクトは勧誘を無視するであろう。逆に若いプロジェクトが成熟したプロジェクトを勧誘するのは滑稽であり、やはり勧誘を無視するだろう。プロジェクトの年齢は勧誘する方、される方ともに若い方が良い。

プロジェクトが若いことは、勧誘に成功した後でも都合が良い。プロジェクトが成熟してくると、品質が低下する恐れのある機能の追加は歓迎されない傾向にあるためである。何をしでかすかわからない新しい開発者の勢いを活かせるのは、失うものが少ない若いプロジェクトに限られるだろう。逆にリクルーティングソース開発手法により開発力を獲得したプロジェクトはいずれ成熟を迎えオープンソース開発手法へと移行することになる。

プロジェクトの開発目標が開発者間で共有しやすいことも要件である。開発目標が不明確であれば、勧誘対象、勧誘の言葉が定まらない。例え勧誘に成功しても開発が無秩序になりすぎて、まともに機能するソフトウェアを作成することはできないだろう。猿にタイプライターを叩かせているようなものである。XTLA プロジェクトの場合、pcl-cvs という明確な目標があった。無秩序開発を実行しても動作するソフトウェアを作ることができたのは、この目標を共有できたからである。

4.4. 著作権の譲渡

成果物を GNU プロジェクトのソフトウェアに統合したい場合など、著作権譲渡の手続きが必要なら、勧誘対象にそのことを早期に伝えるべきである。トラブルが発生すれば、開発者から開発意欲と時間を奪ってしまう。

4.5. 勧誘を断られた場合の心得

勧誘対象者にネガティブな印象を持つのは間違っている。Foobar 氏へのインタビューからわかる通り、勧誘対象者にも当然事情がある。オープンソースに関わる以上、世に多様な価値観があることを受け入れるべきである。

また、開発者の勧誘に失敗しても、勧誘対象者の成果物であるソースコードの流用について検討することを忘れてはいけない。扱っている問題が近いため、流用できないにしても、分析すれば得るところが多いと期待できるためである。

5. まとめ

リクルーティングソース開発手法を説明した。自身が開発中のオープンソースソフトウェアの普及を目指す読者が、新しいプロジェクトを開始するときの参考になれば幸いである。目的を共にする開発者との作業は興奮させられる。この興奮は、目的が明確で開発者が厳選されるほど強まる。リクルーティングソース開発手法をうまく適用できれば最高の興奮を得られるだろう。

プロジェクトの乱立というオープンソース界の大きな問題のうち、リクルーティングソース開発手法によって解決できたのはほんの一部にすぎない。プロジェクトの成熟に差がある場合や、カテゴリーが成熟している場合、目標の共有が困難なプロジェクトでは役に立たない。こういった条件のもとで良いソフトウェアを育てるにはどうすれば良いか、これは開発者として次の課題である。

一方、オープンソースソフトウェアの利用者の立場としても課題がある。リクルーティングソース開発手法には、多様な実装の出現とプロジェクト間の競争を抑制する可能性があり、望ましくない側面を持つ。しかし多様な実装が望ましいのは、その実装の利用者、および利用候補者が、その特徴を知りえた場合である。プロジェクトの多すぎる現状では、それを知るのは困難である。またプロジェクトが互いに切磋琢磨しているかも疑わしい。互いに相手の機能的特徴を認知した場合に、はじめて競争が成立する。現状では競争者の数は多いが競争が成立していない場合が多い。成立していたとしても、それは非常に漠然としたものであろう。単一のプロジェクトを運営するという立場から離れてより広い視点に立つと、適切な競争が発生する「仕掛け」を考えて行く必要があろう。

謝辞 貴重な体験を与えてくれた XTLA プロジェクトのメンバーに感謝する。XTLA プロジェクトの体験を公表するよう進めてくれた青笹茂氏に感謝する。快くインタビューに応じて下さった Foobar 氏に感謝する。

参考文献

[1] 伽藍とバザール. Raymond Eric 、 山形 浩生. 2000 年 5 月 2 日. http://cruel.org/freeware/cathedral.html
[2] Open Source Initiative OSI - Licensing. http://www.opensource.org/licenses/
[3] Too much free software. Marius Andreiana. 2003 年 3 月 29 日. http://freshmeat.net/articles/view/774/
[4] freshmeat.net: Browse project tree - Topic. http://freshmeat.net/browse/
[5] SourceForge.net: Software Map. http://sourceforge.net/softwaremap/trove_list.php
[6] GNU arch - GNU Project - Free Software Foundation (FSF). http://www.gnu.org/software/gnu-arch/
[7] xtla.el (project page at gna). https://gna.org/projects/xtla-el
[8] QuickML.COM: ようこそ. http://quickml.com
[9] XSteve's programming resources. http://www.xsteve.at/prg
[10] chris blogs: Lisp, Arch & Emacs. http://kronavita.de/chris/blog2/archive/2004/05/lisp-arch-emacs.html
[11] "anarchic"開発の指摘. https://mail.gna.org/public/xtla-el-dev/2004-06/

msg00191.html
[12] New arch archive for GNU Emacs.
http://lists.gnu.org/archive/html/gnu-arch-users
/2005-01/msg00002.html