OSS開発における共進化プロセスの理解のための遅延相関分析

山谷 陽亮[1,a)]　大平 雅雄[1,b)]　伊原 彰紀[2,c)]

概要：近年，Android 搭載のスマートフォンが発売されるなど，OSS を利用したソフトウェア開発が注目されている．OSS はボランティアによる開発者によって開発・保守されるため，プロジェクトが途中で崩壊したり，分裂することも珍しくない．そのような懸念を払拭するために OSS の進化に関する研究が盛んに行われている．従来の OSS の進化に関する研究では，システムあるいはコミュニティのどちらか一方のみに注目したものが多いのに対し，Ye らは OSS プロジェクトはシステムとコミュニティが互いに影響を与えながら進化していくというモデルを提案した．本稿は，遅延相関分析手法を用いることによって，Ye らが提案した OSS の共進化プロセスを定量的に表現することを目的とする．Eclipse Platform プロジェクトを対象にケーススタディを行い，OSS の進化をより正確に捉えるには，OSS のシステムとコミュニティの両方を注目する必要があることが確認できた．

1. はじめに

近年，スマートフォンに Android OS が搭載されるなど，オープンソースソフトウェア（以下 OSS とする）を利用するソフトウェア開発が注目されている．通常 OSS は，世界中に点在するボランティアの開発者によって開発されている．通常のソフトウェア開発では，開発者それぞれの役割が決められているが，OSS の開発現場では，ボランティアによる開発者同士で開発を進めていくため，保守などが義務付けられることはない．したがって，OSS 開発の方向性の違いによって，OSS プロジェクトが途中で分裂・解散することも珍しくない．最近では Netscape プロジェクトのコミュニティが崩壊したり，OpenOffice プロジェクトが分裂したことが挙げられる．

そして，このような OSS プロジェクトがいつまで存続するのかわからないことを理由に，多くの企業がソフトウェア開発に OSS を取り入れることを懸念している．独立行政法人情報処理推進機構（IPA）による，2009 年の第三回オープンソースソフトウェア活用ビジネス実態調査では，OSS をソフトウェア開発に取り入れる懸念事項として，58.8%もの企業が「利用している OSS がいつまで存続するのかわからない」ことを挙げている．

このような懸念を払拭するために，OSS の進化に関する研究が盛んに行われている [8]．OSS の進化に関する従来の研究は，OSS のシステムとコミュニティを別の系統として捉えるものばかりであった．しかし Ye らは，OSS の進化はシステムとコミュニティが互いに影響を与えながら成長をするというモデルを提案した [12]．この互いに影響を与えながら進化していく過程のことをを共進化 (co-evolution) と呼ぶ．Ye らのモデルは，アーティファクト（ソースコードなどのプロダクト），開発者（開発に携わる一個人），コミュニティのそれぞれのレイヤーが，互いに影響を与え進化していくことを前提としている．例えば，OSS の人気が上がるにつれて，コミュニティレイヤーでは，コミュニティに参加する開発者が増え，コミュニティが複雑となっていく．その一方で，アーティファクトレイヤーでは，ソースコードの行数が増えたり報告されるバグの数も増えると考えられる．また同時に，開発者レベルでは，ソースコードを変更するなどして開発者の個々のスキルが上がっていく．このようにそれぞれの進化のレイヤーが互いに影響しあいながら OSS は進化していくものだと Ye らは考えている．

Ye らは，定性的に共進化のモデルを提案しているものの，実証的な検証は行ってはいない．そこで本稿では，OSS における共進化のプロセスを定量的に表現することを目的とする．OSS における共進化を説明する知見や，プロジェクトごとの特徴を発見することができれば，進化プロセス

[1] 和歌山大学 システム工学部 情報通信システム学科
和歌山県和歌山市栄谷 930 番地
[2] 奈良先端科学技術大学院大学 情報科学研究所
奈良県生駒市高山町 8916-5
[a)] s151049@sys.wakayama-u.ac.jp
[b)] masao@sys.wakayama-u.ac.jp
[c)] akinori-i@is.naist.jp

を予測することにつながると考えられるためである．

本稿では，OSS における共進化のプロセスを定量的に示すために，遅延相関分析を用いて，各レイヤーでの進化に影響を与える要因を明らかにする．遅延相関分析を用いることによって，どれだけの期間の遅延をもって，一方のレイヤーの変化が他方のレイヤーの変化に影響を与えるか，ということを明らかにすることができる．例えば，アーティファクトレイヤーでの進化が一定期間後にコミュニティレイヤーでの進化に影響を及ぼすとする．遅延相関分析を用いない場合であれば，影響を及ぼす一定期間を導きだすことはできないが，遅延相関分析を用いると，その期間を導きだすことができる．本稿では，遅延相関係数が最も高くなる際のパラメータを自動的に求めるとともに，検定を行い相関の有意性を確かめる．

本稿では Eclipse プロジェクトのデータを用い，共進化プロセス理解のための要因を導きだすために，ケーススタディを行った．ケーススタディの結果，OSS の進化をより正確に捉えるには，OSS のシステムとコミュニティの両方を注目する必要があることが確認できた．

2. 関連研究

本章ではまず，OSS のシステムとコミュニティを別々の系統と捉えた，OSS の進化に関する従来研究について述べる．次に，本研究で着目する OSS の共進化のモデルについて述べ，本研究の立場を明らかにする．

2.1 OSS の進化に関する従来研究

OSS 開発では，ソースコードのみならず，これまで報告された不具合とその修正過程，メーリングリスト上での開発者間の議論など，第三者が OSS 開発を理解するために必要となる情報がほぼすべて公開されている．そのため，OSS システムおよびコミュニティの進化について，様々な観点から分析が行われてきた．

OSS の進化に関する既存研究は，以下の 3 つの観点から大別することができる．

- アーティファクト: ソースコードの規模推移，モジュールの結合度の経時的変化など
- 開発者: コミッタへの昇格，役割の変化 (role migration) など
- コミュニティ: 参加開発者の増減，組織構造の変化など

既存研究の多くは，これらの 3 つの観点を別系統と捉え，それぞれの進化のプロセスを単体で分析・モデル化したものである．以下では，これら 3 つの系統の進化プロセスに関する主な研究を紹介する．

2.1.1 アーティファクトの進化に関する研究

アーティファクトの進化に関する研究は，主にソースコードを対象とした分析やモデル化が多い．システム開発ベンダが利用する OSS を選定するにあたって，安定して開発・保守が続けられているか，複雑になりすぎていないか，などを知るための手掛かりとなるためである．

例えば，Godfrey らは，Linux カーネルの規模推移をサブシステム毎に調査し，どのサブシステムが最も発展しているかなどを明らかにしている [3]．また，Thomas らは，ソースコードの変更履歴に対して LDA (Latent Directory Allocation) を用いることで，過去から現在に至るまでにどのような機能がコミュニティにおける開発の主流であったかを明らかにしている [10]．

その他，OSS 開発に利用されるプログラミング言語の進化について調査した Karus らの研究 [6] や，OSS のライセンス変更を追跡調査した Penta らの研究 [7] などがある．

2.1.2 開発者の進化に関する研究

OSS 開発では，コミッタと呼ばれるソースコードを直接書き換えることのできる特別な権限を持つ開発者が存在する．コミッタや管理者は，開発の品質や生産性を左右する重要な役割であるため，コミッタへの昇格プロセスや，開発者の役割変化に関する研究がこれまで盛んにおこなわれてきている．

Jensen らのコミッタ昇格プロセスに関する分析では，一般開発者がコミッタになるための方法として，現コミッタからの推薦や投票が存在することを明らかにしている [4]．また，Sinha らはこの開発者がコミッタに昇格する要因をソースコードの書き換え履歴や，報告された不具合の修正履歴などの情報を用いて分析している [9]．

コミュニティ内での開発者の役割の変化は，オニオンモデル [11] や，ピラミッドモデル [4] としてモデル化されている．これらのモデルでは，コミュニティ内には階層的に複数の役割が存在することを表している．新規参加者が時間とともに徐々にコミュニティ内で重要な役割を与えられる（役割が変わる）ことにより，開発者がコミュニティへ貢献し続けるための動機付けとなっていることを示すものであり，役割の変化はコミュニティの進化とも関連して重要な要件であるとされている．

2.1.3 コミュニティの進化に関する研究

一般的な OSS 開発におけるプロジェクト参加者の多くは，開発の義務や責任を負わないボランティア開発者である．開発者の自由意思でプロジェクトの加入・脱退を決めることができるため，プロジェクト管理者はコミュニティを維持するために工夫が必要となる．そのため，OSS コミュニティの発展または衰退する要因を明らかにしようとする研究が多数行われてきた．

例えば，Bird らは，生存分析 (survival analysis) を用いて，OSS プロジェクトに参加した開発者がプロジェクトを脱退するまでのモデルを構築している [1]．PostgreSQL, Apache, Python プロジェクトを分析した結果，開発者は平均約 1.5 年の「生存期間（プロジェクトに参加する平均期間)」が存在するため，プロジェクトにより長く貢献して

もらうためには，1.5 年以内にコミッタ権限を与えるなど，プロジェクト内でより重要な役割を担ってもらうことが必要であるとしている．

その他には，ソーシャルネットワーク分析を用いて OSS コミュニティの組織構造の特徴を明らかにすることで，成功・失敗要因を特定した研究 [2], [5] や，コミュニティを維持するうえで重要な「社会的」ポジションを担う開発者を明らかにした研究 [13] などがある．

2.2 OSS の共進化

OSS の進化の系列を別系統として捉えた前述の従来研究に対して，Ye らの研究 [12] では，OSS システムとコミュニティはともに相互作用しながら進化するものと捉えて考えられている．

図 1 は，Ye らが提案した OSS の共進化 (co-evolution) のモデル [12] を模式的に示したものである．図 1 での上段は，ソースコードの規模が増大していくなどのアーティファクトレイヤーの発展を表している．中段は，重要な役割を担わなかった開発者が，コミッタつまりプロジェクトのコアメンバーに昇格するなどの，開発者レイヤーの発展を表している．下段は，開発者同士の交流が増え，コミュニティが複雑となるなどの，コミュニティレイヤーの発展を示している．図 1 では，各レイヤーが他のレイヤーに影響を与えながら発展していくことも表現している．

Ye らはこの共進化のモデルを定性的に説明しているものの，客観的な定量的データとしてそのプロセスを示してはおらず，Ye らのモデルをベースとして OSS 開発の進化を理解するためには実証的な調査が必要となる．そこで本研究ではまず，共進化のプロセスを定量的に分析するための分析手法の構築を行う．

3. 遅延相関分析に基づく手法

本章では，OSS の共進化プロセスを理解するために用いる遅延相関分析手法について述べる．

3.1 遅延相関分析の概要

従来の相関分析手法では，相関が発生する時間的差分を考慮していないため，一定期間後に時系列データ間に相関がある場合でも，相関がないと判断してしまう．そこで本研究では，相関が発生する時間的差分を考慮する遅延相関分析に基づく手法を用いる．遅延相関分析に基づく手法を用いることで，従来の相関分析手法では明らかにできなかった知見を発見することが期待される．

遅延相関分析に基づく手法は，竹内らが提案する健康データの時系列データ解析手法を参考としている [14]．竹内らの手法は，消費エネルギーなどの生活習慣データと体脂肪率などの健康データの関係を明らかにするために，説明変数を生活習慣データ，目的変数を健康データとし，遅延相関分析を行っている．本研究では，アーティファクトレイヤー，開発者レイヤー，コミュニティレイヤーのいずれかをを説明変数と目的変数に割り当て，進化を表現するメトリクス（アーティファクトレイヤーの場合はソースコードの行数など）を用いて分析を行う．

3.2 時系列データの処理

遅延相関係数分析を行うには，加算係数，差分係数，遅延係数の 3 つのパラメータを定義する必要がある．遅延相関分析では説明変数の値が累積し，一定期間の遅延をもって目的変数の値の変化に影響を与えると想定しているため，説明変数をどれだけの期間累積したのかという指標や，目的変数のいつからの値の変化を考慮するのかという指標，説明変数の累積がどれだけの遅延をもって目的変数の変化に影響を与えるのかという 3 つの指標が必要になる．説明変数の値を累積させた期間を加算係数とし，目的変数の値の変化を考慮する期間を差分係数，説明変数の累積が目的変数の変化に影響を与える期間を遅延係数とする．

図 2 は，竹内らが提案した手法を参考に改編した遅延を考慮した相関を求めるための時系列データ処理の概念図 [15] である．図 2 では，説明変数の加算が遅延をもって目的変数の変化に影響を与えることがあることを模式的に

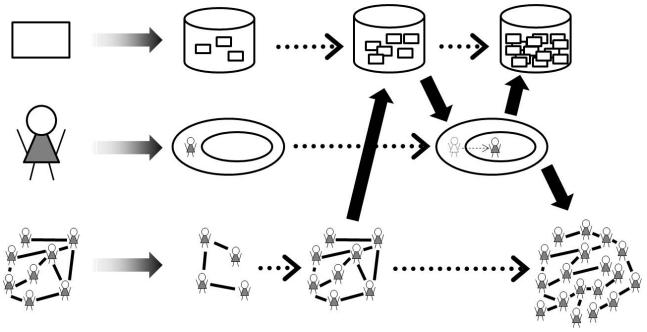

図 1 OSS 共進化の概念（[12] を参考に改編）
Fig. 1 The concept of OSS co-evolution

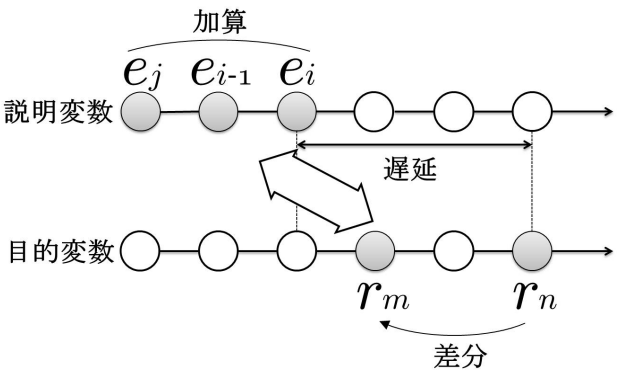

図 2 時系列データ処理の概念（[15] を参考に改編）
Fig. 2 Processing time-series data

表現している．図 2 において，加算係数は $i-j$ で表され，e_i を時刻 i における説明変数の値，e_j を時刻 j における説明変数の値とする．また，差分係数は $n-m$ で表され，r_n を時刻 n における目的変数の値，r_m を時刻 m における目的変数の値とする．さらに，遅延係数は $n-i$ で表される．

3.3 アルゴリズム

遅延相関分析に基づく手法の手順を示す．各パラメータとは，加算係数，差分係数，遅延係数を指している．

STEP1 説明変数の値の累積値を求めるための時系列データの処理

STEP2 目的変数の値の変化値を求めるための時系列データの処理

STEP3 STEP1, 2 で処理された時系列データの相関係数を算出する

STEP4 STEP1 から STEP3 を各パラメータのすべての組み合わせについて行う

STEP5 相関係数の絶対値が最大となる際の各パラメータの値を求める

STEP1 では，ある加算係数 $(i-j)$ における説明変数の値の累積値を求めるために，時系列データの処理を行う．説明変数の値の累積値を e_{ij} とすると，e_{ij} は (1) 式で定義される．

$$e_{ij} = e_i + e_{i-1} + \cdots + e_j \tag{1}$$

STEP2 では，ある差分係数 $(n-m)$ における目的変数の値の変化値を求めるために，時系列データの処理を行う．目的変数の値の変化値を r_{nm} とすると，r_{nm} は (2) 式で定義される．

$$r_{nm} = r_n - r_m \tag{2}$$

STEP3 では，STEP1 および STEP2 で処理された時系列データのある遅延係数 $(n-i)$ における相関係数を算出する．相関係数はピアソンの積率相関係数によって求める．

STEP4 では，STEP1 から STEP3 までの処理を，各パラメータのとり得る値すべての組み合わせについて行い，STEP5 では，相関係数の絶対値が最大となる際の各パラメータの値と相関係数の絶対値の値を求める．

以上の処理を行うことによって，相関係数の絶対値が最も大きくなる際の各パラメータの値を自動的に求めることができる．

4. ケーススタディ

本章では，遅延相関分析手法を用いた OSS の共進化プロセス理解のために，さまざまなメトリクスを用い，ケーススタディを行った結果を示す．

4.1 対象とするプロジェクト

本稿では，Eclipse プロジェクトの中でもツールの中核を担う Eclipse Platform プロジェクトを対象とし，分析を行う．Eclipse は OSS でありながら，大規模なプロジェクトであるため，エンドユーザが多く，長い期間盛んに開発が行われ，バグ報告やソースコードの書き換えなどが数多く行われてきた．そのため，OSS 研究によく用いられることから本稿でも Eclipse を対象に 2004 年 8 月から 2006 年 6 月のデータで分析を行う．

4.2 メトリクス

本研究では，共進化のプロセスを理解するために，進化の各レイヤーにさまざまなメトリクスを定義する．各レイヤーにおけるメトリクスを表 1 に示す．

表 1 に示すように，アーティファクトの進化のレイヤーには，12 個のメトリクスを定義する．アーティファクトレイヤーにこれらのメトリクスを定義したのは，プロダクトの進化を計測するためには，ソースコードの変化や特徴を追うことが有用だと考えたためである．アーティファクトレイヤーの各メトリクスは，ソースコード解析ツールである Understand を用い，毎月 1 日にソースコード解析を行い算出されたものである．

表 1 に示すように，開発者の進化のレイヤーには，Committer, Reporter, Fixer というメトリクスを定義する．一般的に OSS は，ソースコードなどが公開されているが，ソースコードを書き換えることのできる権限を持つ開発者は限られている．このソースコードを書き換えることのできる人物をコミッタと呼ぶ．一般的にコミッタではない開発者がコミッタになるには，コミッタからの推薦を受けコミッタになることが多い．コミッタから推薦を受けるためには，積極的にプロジェクトに参加していることや質の高いバグの修正を行うことが求められる．つまり，コミッタ権限をもつ開発者が多いということは，開発者自身のスキルや知識などが向上したと考えられる．また，コミッタ権限の有無に関わらず，プロジェクトに積極的に参加する開発者は報告されたバグの修正を行う．バグの報告を行った人数やバグの修正を行った人数などを分析することによって，開発者の役割の変化を追うことができると考える．本稿では，バージョン管理ツールである Git からソースコードの変更ログを取得し，一月ごとのコミッタの人数を抽出している．また，バグ追跡管理システムである Bugzilla からバグ情報を取得し，一月ごとのバグ報告およびバグ修正を行った人数を抽出している．Committer には一月ごとのコミッタ数を，Reporter には一月ごとのバグを報告した人数，Fixer には一月ごとのバグを修正した人数が時系列データとして格納される．

表 1 に示すように，コミュニティの進化のレイヤーには，NumberOfForums と NumberOfPeople というメトリクス

表 1 各レイヤーにおけるメトリクス
Table 1 Metrics in each layer

進化に関するレイヤー	メトリクス名	メトリクスの説明
アーティファクト	AddLine	追加した行数
アーティファクト	AvgCyclomatic	各メソッドの複雑度の平均
アーティファクト	CountDeclClass	クラス宣言の数
アーティファクト	CountDeclFunction	関数宣言の数
アーティファクト	CountLineBlank	空白の行数
アーティファクト	CountLineCode	総コード行数
アーティファクト	CountLineCodeDecl	宣言部のコード行数
アーティファクト	CountLineCodeExe	処理を行うコード行数
アーティファクト	CountLineComment	コメントの行数
アーティファクト	CountStmtDecl	宣言部のステートメント数
アーティファクト	CountStmtExe	処理を行うステートメント数
アーティファクト	NumberOfRevisions	ファイルを改訂した回数
開発者	Committer	ソースコードを書き換えた人数
開発者	Reporter	バグを報告した人数
開発者	Fixer	バグを修正した人数
コミュニティ	NumberOfForums	Eclipse Community Forums に投稿された回数
コミュニティ	NumberOfPeople	Eclipse Community Forums に投稿した人数

表 2 アーティファクト → 開発者での結果
Table 2 Airtifact → Developpers

説明変数	目的変数	加算係数	差分係数	遅延係数	相関係数
NumberOfRevisions	Committer	1	3	3	-0.588
AddLine	Reporter	1	3	0	0.687
NumberOfRevisions	Reporter	1	3	0	0.591
AddLine	Fixer	1	3	0	0.523
AvgCyclomatic	Fixer	3	0	3	-0.479
NumberOfRevisions	Fixer	2	3	3	-0.670

を定義する．Eclipse において，開発者間のコミュニケーションや情報交換は Eclipse Community Forums という場で行われている．Eclipse Community Forums では，コアな開発者はもちろんのこと，そうではない開発者も自由に発言することができる．Eclipse Community Forums が活発に使用されているということは，プロジェクトのコミュニティの発展を示すものだと考える．Eclipse Community Forums の中の Platform に関する月ごとの投稿数を NumberOfForums, 月ごとの投稿者数を NumberOfPeople として定義する．

4.3 分析方法

Eclipse Platform プロジェクトを対象として，遅延相関分析手法を用いケーススタディを行う方法について述べる．提案する手法を用いることによって，相関係数の絶対値が最大となるときの加算係数，差分係数，遅延係数を自動的に求めることができることから，表 1 に示すすべてのメトリクスを用いて分析を行う．遅延を考慮しない相関係数を求める場合，説明変数と目的変数の組合わせを逆にしても，相関係数の絶対値は変化しないが，遅延相関分析手法では変化する．例えば，アーティファクトレイヤーと開発者レイヤーの関係について分析を行う場合を考える．まず，表 1 に示すアーティファクトレイヤーのそれぞれのメトリクスを説明変数とし，開発者レイヤーのそれぞれのメトリクスを目的変数とした際の分析を行う．次に，開発者レイヤーのそれぞれのメトリクスを説明変数とし，アーティファクトレイヤーのそれぞれのメトリクスを目的変数とした際の分析を行う．

5. 結果と考察

本章では，ケーススタディを行った結果および考察を述べる．

加算係数，差分係数，遅延係数のそれぞれの値がとり得る範囲は，加算係数 ($1 \leqq i - j \leqq 3$)，差分係数 ($0 \leqq n - m \leqq 3$)，遅延係数 ($0 \leqq n - i \leqq 3$) とする．あるメトリクスの組について，各パラメータを変化させ，分析を行うと 48 通りのパターンが存在する．この 48 通りの中から，相関係数の絶対値が最大となる各パラメータの値を算出した．表 2〜表 7 に各パラメータの値および，相関係数を示している．本稿では，相関係数の絶対値が 0.4 を超える場合を

表 3 開発者 → アーティファクトでの結果
Table 3 Developpers → Airtifact

説明変数	目的変数	加算係数	差分係数	遅延係数	相関係数
Committer	AvgCyclomatic	3	3	3	0.533
Committer	CountLineCode	2	3	0	0.491
Committer	CountLineCodeDecl	2	3	0	0.533
Committer	CountStmtDecl	2	3	0	0.577
Reporter	AddLine	3	0	2	-0.860
Reporter	AvgCyclomatic	3	0	0	0.541
Reporter	CountDeclFunction	3	3	0	-0.583
Reporter	CountLineBlank	3	3	0	-0.421
Reporter	CountLineCode	3	3	1	-0.505
Reporter	CountLineCodeDecl	3	3	2	-0.603
Reporter	CountLineCodeExe	3	3	0	-0.547
Reporter	CountStmtDecl	3	3	2	-0.620
Reporter	CountStmtExe	3	3	0	-0.532
Reporter	NumberOfRevisions	3	3	2	-0.713
Fixer	AvgCyclomatic	3	3	2	0.490
Fixer	CountLineCodeDecl	2	3	0	0.454
Fixer	CountStmtDecl	2	3	0	0.482
Fixer	NumberOfRevisions	2	0	0	0.741

表 4 開発者 → コミュニティでの結果
Table 4 Developpers → Community

説明変数	目的変数	加算係数	差分係数	遅延係数	相関係数
Reporter	NumberOfForums	3	3	3	0.482
Reporter	NumberOfPeople	3	0	2	-0.514
Fixer	NumberOfForums	2	3	0	-0.572

相関があると判断し．相関係数の絶対値が 0.4 を超える場合のみを表 2〜表 7 に示した．

5.1 アーティファクトレイヤーと開発者レイヤーの関係

表 2 より，ソースコードに追加した行 (AddLine) が増加すると，バグ報告者 (Reporter) およびバグ修正者 (Fixer) が遅延することなく増加する．したがって，新しく追加されたソースコードに新たなバグを入れてしまうことが多いと考えられる．

また表 3 より，コミッタ数 (Committer) やバグ修正者 (Fixer) が増加すると，宣言部の行数 (CountLineCodeDecl) や宣言部のステートメント数 (CountStmtDecl) が増加し，2, 3 ヶ月後に複雑度 (AvgCyclomatic) が増加することから，それぞれのコミッタやバグ修正者は独自の関数や変数を定義し，独自に宣言した関数などが 2, 3 ヶ月後に複雑となると考えられる．複雑度が増加するとバグを埋め込む可能性も上がるため，コミッタが増加した 3 ヶ月後にリファクタリングを行うことにより，複雑度を減少させ，バグを埋め込む可能性を下げることが期待できる．

5.2 コミュニティレイヤーの進化の要因

表 4 より，バグ報告者 (Reporter) が増加すると，2 ヶ月後には Eclipse News Forums に投稿する人数 (NumberOfPeople) は減少するが，3 ヶ月後には Eclipse News Forums に投稿される回数 (NumberOfForums) は増加することになる．また表 7 より，ソースコードの行数 (CountLineCode) や宣言部のステートメント数 (CountStmtDecl) , クラスを宣言している数 (CountDeclClass) などが増加すると，Eclipse News Forums に投稿する人数 (NumberOfPeople) が減少するが，投稿された回数 (NumberOfForums) との間に相関はみられなかった．Ecliupse News Forums に投稿された回数が多ければ，投稿した人数も多いことから，NumberOfPeople と NumberOfForums のどちらのメトリクスを用いた場合でも，類似する結果を得られるものと期待していた．しかし，NumberOfPeople を用いた場合と，NumberOfForums を用いた場合では，結果は全く異なるものとなった．

5.3 コミュニティレイヤーの進化が与える影響

表 5，表 6 より，Eclipse News Forums に投稿された回数 (NuberOfForums) が増加すると，1 ヶ月後にはコミッタ数 (Committer) およびバグ修正者数 (Fixer) が増加し，2 ヶ月後にはバグ報告者数 (Reporter) が増加する．また，2, 3 ヶ月後にはソースコードの行数 (CountLineCode) や

表 5 コミュニティ → 開発者での結果
Table 5 Community → Developpers

説明変数	目的変数	加算係数	差分係数	遅延係数	相関係数
NumberOfForums	Committer	1	3	1	0.495
NumberOfForums	Reporter	3	3	2	0.419
NumberOfForums	Fixer	3	2	1	0.648
NumberOfPeople	Committer	1	3	1	0.421
NumberOfPeople	Fixer	1	0	2	0.443

表 6 コミュニティ → アーティファクトでの結果
Table 6 Community → Airtifact

説明変数	目的変数	加算係数	差分係数	遅延係数	相関係数
NumberOfForums	AddLine	3	0	2	0.592
NumberOfForums	AvgCyclomatic	3	0	2	-0.577
NumberOfForums	CountDeclFunction	3	2	2	0.507
NumberOfForums	CountLineBlank	1	1	3	0.485
NumberOfForums	CountLineCode	3	3	2	0.616
NumberOfForums	CountLineCodeDecl	3	2	2	0.694
NumberOfForums	CountLineCodeExe	3	2	2	0.514
NumberOfForums	CountLineComment	3	1	3	0.526
NumberOfForums	CountStmtDecl	3	2	2	0.689
NumberOfForums	CountStmtExe	3	2	2	0.537
NumberOfForums	NumberOfRevisions	3	2	2	0.621
NumberOfPeople	AddLine	2	0	0	0.586
NumberOfPeople	AvgCyclomatic	3	0	1	-0.767
NumberOfPeople	CountDeclClass	3	0	0	-0.687
NumberOfPeople	CountDeclFunction	3	3	2	0.741
NumberOfPeople	CountLineBlank	2	0	0	-0.643
NumberOfPeople	CountLineCode	3	3	2	0.744
NumberOfPeople	CountLineCodeDecl	3	3	2	0.766
NumberOfPeople	CountLineCodeExe	3	3	2	0.723
NumberOfPeople	CountLineComment	3	0	0	-0.681
NumberOfPeople	CountStmtDecl	3	3	2	0.746
NumberOfPeople	CountStmtExe	3	3	2	0.724

ソースコードに追加された行数 (AddLine) などが増加する．Eclipse News Forums は，バグの報告，修正，ソースコードの書き換えなどを行わないユーザが，開発者として活躍する前に利用するフィールドであると考えられるとともに，Eclipse News Forums の発展が 2, 3 ヶ月の遅延をもってアーティファクトレイヤーや開発者レイヤーに影響を与えると考えられる．したがって，Eclipse News Forums を利用しやすいものにすることで，開発者やソースコードの規模が増えることを期待することができる．

6. まとめ

本研究は，遅延相関分析手法を用いることによって，Ye らが提案した OSS の共進化プロセスを定量的に表現するために，Eclipse Platform プロジェクトを対象にケーススタディを行った．ケーススタディの結果，従来の研究のように，OSS の進化をシステムあるいはコミュニティを別々の系統として捉える場合では発見することができなかった知見を，OSS システムとコミュニティは共進化の関係にあることを考慮することによって，発見することができた．このように，OSS の進化をより正確に捉えるためには，OSS のシステム単体，あるいはコミュニティ単体に着目するのではなく，OSS のシステムおよびコミュニティの両方を着目する必要があることを確認することができた．

本研究では，Eclipse Platform プロジェクトのみを対象とし分析を行なったが，今後は Eclipse と同じ IDE ツールである NetBeans プロジェクトやモバイル OS として注目されている Firefox OS, Tizen, Android などを対象とし分析を行う．また本稿では，開発者レイヤーのメトリクスをコミッタ数，バグ報告者数，バグ修正者数と定義したが，これらのメトリクスはコミュニティレイヤーにも属すると考えられるため，今後は各レイヤーのメトリクス選定を慎重にすることはもちろんのこと，各レイヤーでのメトリクスの数を増やし分析を行う．

表 7 アーティファクト → コミュニティでの結果
Table 7 Airtifact → Community

説明変数	目的変数	加算係数	差分係数	遅延係数	相関係数
AvgCyclomatic	NumberOfForums	1	0	1	-0.422
NumberOfRevisions	NumberOfForums	3	0	0	-0.560
AddLine	NumberOfPeople	2	0	0	0.615
AvgCyclomatic	NumberOfPeople	1	0	1	-0.583
CountDeclClass	NumberOfPeople	1	0	0	-0.627
CountDeclFunction	NumberOfPeople	1	0	0	-0.649
CountLineBlank	NumberOfPeople	1	0	1	-0.613
CountLineCode	NumberOfPeople	1	0	0	-0.633
CountLineCodeDecl	NumberOfPeople	1	0	0	-0.628
CountLineCodeExe	NumberOfPeople	1	0	0	-0.637
CountLineComment	NumberOfPeople	1	0	1	-0.634
CountStmtDecl	NumberOfPeople	1	0	0	-0.628
CountStmtExe	NumberOfPeople	1	0	0	-0.634
NumberOfRevisions	NumberOfPeople	1	3	0	-0.410

7. 謝辞

本研究の一部は，文部科学省科学研究補助金 (基盤 (B):23300009)，(基盤 (C):24500041) および (若手 (B):25730045) による助成を受けた．

参考文献

[1] Bird, C., Gourley, A., Devanbu, P., Swaminathan, A. and Hsu, G.: Open Borders? Immigration in Open Source Projects, *Proceedings of the 4th International Workshop on Mining Software Repositories (MSR'07)*, p. No.6 (2007).

[2] Bird, C., Pattison, D., D'Souza, R., Filkov, V. and Devanbu, P.: Latent Social Structure in Open Source Projects, *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE'08)*, pp. 24–35 (2008).

[3] Godfrey, M. W. and Tu, Q.: Evolution in Open Source Software: A Case Study, *Proceedings of the International Conference on Software Maintenance (ICSM'00)*, pp. 131–142 (2000).

[4] Jensen, C. and Scacchi, W.: Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study, *Proceedings of the 29th International Conference on Software Engineering (ICSE'07)*, pp. 364–374 (2007).

[5] Kamei, Y., Matsumoto, S., Maeshima, H., Onishi, Y., Ohira, M. and ichi Matsumoto, K.: Analysis of Coordination between Developers and Users in the Apache Community, *The Fourth International Conference on Open Source Systems (OSS2008)*, pp. 81–92 (2008).

[6] Karus, S. and Gall, H.: A Study of Language Usage Evolution in Open Source Software, *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR'11)*, pp. 13–22 (2011).

[7] Penta, M. D., German, D. M., Gueheneuc, Y.-G. and Antoniol, G.: An Exploratory Study of the Evolution of Software Licensing, *Proceedings of the 32nd International Conference on Software Engineering (ICSE'10)* (2010).

[8] Scacchi, W.: *Understanding Open Source Software Evolution*, chapter 9, pp. 181–205, John Wiley & Sons, Ltd. (2006).

[9] Sinha, V. S., mani, S. and Sinha, S.: Entering the Circle of Trust: Developer Initiation as Committers in Open-Source Project, *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR'11)*, pp. 133–142 (2011).

[10] Thomas, S. W., Adams, B., Hassan, A. E. and Blostein, D.: Modeling the Evolution of Topics in Source Code Histories, *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR'11)*, pp. 173–182 (2011).

[11] Ye, Y. and Kishida, K.: Toward an understanding of the motivation Open Source Software developers, *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, pp. 419–429 (2003).

[12] Ye, Y., Nakakoji, K., Yamamoto, Y. and Kishida, K.: *The Co-Evolution of Systems and Communities in Free and Open Source Software Development*, chapter 3, pp. 59–82, Idea Group Publishing (2004).

[13] Zhou, M. and Mockus, A.: Developer fluency: achieving true mastery in software projects, *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering (FSE'10)*, pp. 137–146 (2010).

[14] 竹内裕之，児玉直樹：生活習慣と健康状態に関する時系列データ解析手法の開発，*Proceedings of the 3th Forum on Data Engineering and Information Management(DEIM'08)* (2008).

[15] 黛　勇気, 竹内裕之, 児玉直樹：生活習慣と健康状態の時系列データ解析における重み付けの検討 (I) -日毎の任意係数による重みづけ-，*Proceedings of the 3th Forum on Data Engineering and Information Management(DEIM'11)* (2011).