OSSシステムとコミュニティの共進化の理解を目的としたデータマイニング手法

山谷 陽亮[1]　大平 雅雄[1]　Passakorn Phannachitta[2]　伊原 彰紀[2]

概要：オープンソースソフトウェア (OSS) を活用したシステム開発が一般的になりつつある一方，「サポートが得られるかどうか分からない」などの理由から，依然としてOSSの活用に躊躇するシステム開発企業は少なくない．本研究では，OSSシステムとコミュニティの共進化のプロセスを定量的に分析するためのデータマイニング手法を提案する．本手法は，遅延相関分析の考え方に基づき，一方の進化の系列が他方の進化の系列に与える影響を一定時間後に観察できることを考慮したものである．また，遅延相関分析を容易に行うために，遅延相関係数が最も高くなる際の各種パラメータを自動的に求める点に特徴がある．提案手法の有用性を確かめることを目的として，ApacheおよびEclipseコミュニティを対象としたケーススタディを行った結果，遅延相関を考慮しない従来の分析結果と比べて，提案手法は共進化のプロセスをより正確に観察できることを確かめた．

A Data Mining Method for Understanding Co-Evolution of OSS Systems and Communities

Yosuke Yamatani[1]　Masao Ohira[1]　Passakorn Phannachitta[2]　Akinori Ihara[2]

1. はじめに

近年，システムの低価格化・短納期化の社会要請に対応するためにLinuxをはじめとするオープンソースソフトウェア（以下OSS) を活用したシステム開発が一般的になりつつある．一方で，システム開発を請け負うベンダー企業から見ると，OSSを活用したシステム開発には依然として懸念材料が数多く残されているのも事実である．独立行政法人情報処理推進機構（IPA) の「第3回オープンソースソフトウェア活用ビジネス実態調査」（有効回答数700社以上，2009年度の調査結果）によると，システム開発へのOSS導入に関する企業の懸念事項として，58.8%の企業が「利用しているOSSがいつまで存続するかわからないこと」を挙げた．

OSSは一般的に，世界中に分散しているボランティアの開発者らの共同作業によって開発が進められ [11]，開発や保守を義務付けられたり強制されることがないため，Netscapeプロジェクトのようにコミュニティが崩壊したり，OpenOfficeプロジェクトのように分裂したりすることは珍しくない [3]．したがって，「利用しているOSSがいつまで存続するかわからない」という懸念が生じるのは当然といえる．

そのような懸念を払拭するために，OSS開発およびその進化に関する分析やモデル化が盛んに行われている [12]．OSSの進化に関しての従来のモデルは，システムあるいはコミュニティを別々の系統と捉え調査したものがほとんどである．例えば，ソースコードの規模や機能追加を時系列に可視化 [4] したり，コミュニティの組織構造をの進化をソーシャルネットワークを用いて表現する [2], [8]，などである．

一方，Yeらは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，共進化 (co-evolution) のモデル [16] を提案している．共進化とは，複数の進化の系統が互いに影響を与えながら進化していく過程のことである．Ｙｅらのモデルでは，アーティファクト（ソース

[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 情報科学研究科
奈良県生駒市高山町 8916-5

コードなどのプロダクトのこと），開発者（コミュニティに参加する開発者個人のこと），コミュニティという，3 つのレイヤーが相互に関係し合い，共に進化することを前提としている．例えば，OSS の人気が高まるにつれてコミュニティの規模や複雑さが増加する一方，アーティファクトとしての OSS は多数の機能追加や不具合修正が行われる．また同時に，コミュニティに参加する開発者個々人もアーティファクトの進化とともに，知識やスキルを獲得していくといったものである．

しかしながら，Ye らの研究 [16] は共進化のモデルを定性的に示しているものの，具体的なコミュニティを対象に，かつ，定量的に進化のプロセスを明らかにしたものではない．そこで本研究は，Ye らのモデルをベースとして OSS システムとコミュニティの共進化のプロセスを定量的に示すことを目指している．共進化を説明する一般化可能な知見を導出することはもちろんのこと，個々のコミュニティに特有の知見を得ることができれば，OSS 開発に関する従来の懸念を払拭することにつながるものと期待している．

本稿ではその第一歩として，OSS システムとコミュニティの共進化を定量的に分析するためのデータマイニング手法を提案する．本手法は，遅延相関分析の考え方に基づき，一方のレイヤーが他方のレイヤーに与える影響が一定時間後に観察できることを考慮したものである．例えば，ある時点での開発者の増加がアーティファクトに与える影響は，コミュニティへ投稿されるパッチ数の増加として数カ月後に顕著に観察される，などである．また，本手法は，遅延相関分析を容易に行うために，遅延相関係数が最も高くなる際の各種パラメータを自動的に求める点に特徴がある．

本稿では，提案手法の有用性を確認するために，Apache および Eclipse プロジェクトの共進化を，パッチ数（アーティファクト），コミッタ昇格数（開発者），開発者総数（コミュニティ）の 3 つの系統のデータを用いて行ったケーススタディについて報告する．遅延相関を考慮しない従来の分析結果と比べて，提案手法は共進化のプロセスをより正確に観察できることを確かめた．

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 [4]．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 [14]．

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 [7] や，OSS のライセンス変更を追跡調査した Penta らの研究 [9] 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 [5]．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 [13]．

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 [15] や，ピラミッドモデル [5] 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], [6] 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 [17] 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 [16] 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．

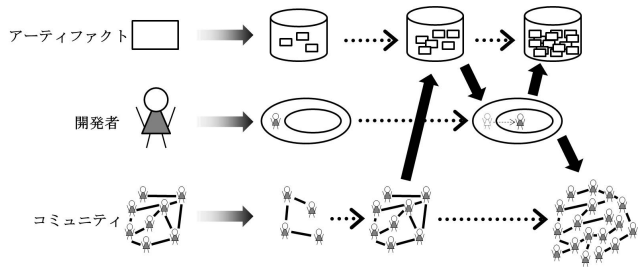

図 1 OSS 共進化の概念（[16] を参考に改編）
Fig. 1 The architecture of OSS co-evolution

図 1 は，Ye らが提案した OSS の共進化 (co-evolution) のモデル [16] を模式的に示したものである．図 1 での上段は，パッチ数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，総開発者数が増えるなどといった，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データマイニング手法の構築を行う．

3. 提案するデータマイニング手法

本章では，共進化のプロセスを定量的に分析するためのデータマイニング手法について述べる．

3.1 提案手法の概要

1 章で述べた通り，提案手法は，遅延相関分析の考え方に基づいている．つまり，一方の進化のレイヤーが他方の進化のレイヤーに与える影響が一定時間後に顕著に観察できることを前提としたものである．例えば，ある時点での開発者の増加がアーティファクトに与える影響は，コミュニティへ投稿されるパッチ数の増加として数か月後に顕著に観察される，などである．また，本手法は，遅延相関分析を容易に行うために，遅延相関係数が最も高くなる際の各種パラメータを自動的に求める点に特徴がある．

提案手法は，竹内らが提案する健康データの遅延相関分析手法を参考にしたものである．竹内の手法は，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因果関係を明らかにするために，生活習慣データ（説明変数）と一定期間後の健康データ（目的変数）との関係を分析するためのものである [18]．本研究では，説明変数と目的変数に，アーティファクト，開発者，コミュニティの 3 つの系統のうちいずれかを割当て，それぞれの系統を代表する具体的なメトリクス（例えば，アーティファクトの場合はパッチ数など）を用いる．

また，提案手法には，遅延相関分析における各種パラメータ（遅延係数，差分係数，加算係数）の最適値を遅延相関係数が最大になるよう自動的に求める処理が含まれる．収集したデータを本手法を用いて解析することで，分析者は遅延の大きさや分析窓の大きさを気にせず，最も遅延相関係数が大きくなるメトリクスの組み合わせのみを分析対象とすることができる．

3.2 遅延を考慮した相関分析

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 [19] 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表現している．

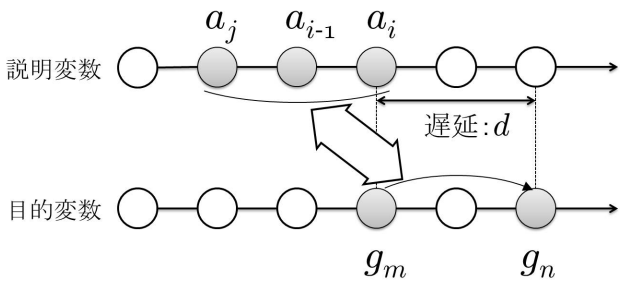

図 2 時系列データ処理の概念（[20] を参考に改編）
Fig. 2 Processing of time-series data

図 2 において，説明変数は遅れをもって目的変数に影響を及ぼすことを想定しているため，遅延係数 d は (1) 式で定義される．遅延係数とはどれだけの遅れをもつか，という値である．

$$d = n - i \tag{1}$$

また，目的変数は説明変数によって変化していくものであるので，進化を定量的に表現するメトリクスの時刻 n における値と時刻 m における値の差分値を目的変数とする．$n - m$ の値を差分係数と呼ぶことにし，目的変数 g_{nm} は (2) 式で定義される．

$$g_{nm} = g_n - g_m \tag{2}$$

さらに，説明変数と考えられるメトリクスの値が累積し，目的変数に影響を及ぼすと想定しているため，説明変数 a_{ij} は説明変数と考えられるメトリクスの時刻 j の値から時刻 i の値を加算し平均をとった値とし，(3) 式で定義される．加算された期間を示す $i - j$ の値を加算係数と呼ぶ．

$$a_{ij} = \frac{a_i + a_{i-1} + \cdots + a_j}{i - j} \tag{3}$$

そして，相関係数 c_{ga} は，遅延係数 d，差分係数 $n - m$，および加算係数 $i - j$ をパラメータとして変化させ，時系列データをもとに (4) 式で計算される．

$$c_{ga} = \frac{S_{ga}}{S_g \times S_a} \tag{4}$$

ここで，S_g および S_a はそれぞれ g_{nm}, a_{ij} の標準偏差である．また，S_{ga} は g_{nm} と a_{ij} の共分散である．

最後に，相関係数の絶対値が最大となるときの遅延係数，差分係数，加算係数を求める．そのときの遅延係数が大きいということは，説明変数が遅れをもって目的変数に影響を与えているということになり，加算係数が大きいということは，長期間の説明変数と考えられるメトリクスの蓄積が目的変数に影響を与えるということになる．

4. ケーススタディ

本章では，提案手法の有用性を確かめるために行ったケーススタディについて述べる．

表 1 定義するパラメーター一覧
Table 1 Defined parameters

パラメータ名	式
遅延係数	$n - i$
差分係数	$n - m$
加算係数	$i - j$

4.1 対象プロジェクト

対象とするプロジェクトは，Eclipse Platform Project（以下 Eclipse）と Mozilla Apache HTTP Server Project（以下 Apache）である．

このプロジェクトを選んだ理由は，どちらのプロジェクトも長い間盛んに開発が行われ，OSS 研究によく用いられており，先行研究との比較が容易に行えるためである．Apache では 1998 年 1 月から 2003 年 12 月のデータを，Eclipse では 2004 年 1 月から 2007 年 12 月のデータを扱う．

4.2 対象メトリクス

2 章でも述べたように，Ye らのモデルでは，OSS はアーティファクト，開発者，コミュニティの 3 つのレイヤー（系統）の進化が互いに影響を与えながら，全体のシステムとして進化することが示されている．本研究において，共進化のモデルを定量的に示すためには，それぞれのレイヤーでの進化を測る具体的なメトリクスが必要になる．

本稿では，表 2 に示すように，アーティファクトレイヤーにパッチ数，開発者レイヤーにコミッタ昇格数，コミュニティレイヤーに総開発者数をメトリクスとして用いる．

表 2 それぞれのレイヤーに対応するメトリクス一覧
Table 2 Metris that corresponds to each layers

進化に関するレイヤー	メトリクス
アーティファクト	パッチ数
開発者	コミッタ昇格数
コミュニティ	総開発者数

各レイヤーを表すメトリクスには，上記以外にも多数のメトリクスを考えることができるが，その他のメトリクスを用いた分析は今後の課題とする．

4.3 分析パターン

進化の各レイヤーを表すメトリクスを選定した後，どのメトリクスを説明変数および目的変数とするかを決定する必要がある．

本稿では，表 3 のように決定した．表中，◯ は目的変数と説明変数の組が正当性のあるものだと判断したもので，× はその組が不自然だと判断したもの，△ はどちらともいえないものを表している．

例えば，コミュニティの開発者総数（の増加）を説明変

表 3　分析の有効性評価の一覧
Table 3　Evaluation of analysis

目的変数＼説明変数	パッチ数	コミッタ昇格数	総開発者数
パッチ数	-	○	○
コミッタ昇格数	×	-	△
総開発者数	×	△	-

数とした場合には，パッチ数（の増加）は目的変数として考えるのが自然である（開発者が増えればパッチ投稿数が増えると考えられるため）．一方，パッチ数が増えれば，開発者数が増えると考えるのは自然でないため，パッチ数を説明変数とし開発者総数を目的変数とする組み合わせは，あらかじめ除外できるものとした．

以上から，本稿で対象とするメトリクスの組み合わせのうち実際の解析対象となるのは，表 4 に示した 4 つのパターンである．

表 4　解析を行うパターンの概要
Table 4　Analysis patterns

パターン	目的変数	説明変数
パターン A	パッチ数	コミッタ昇格数
パターン B	パッチ数	総開発者数
パターン C	コミッタ昇格数	総開発者数
パターン D	総開発者数	コミッタ昇格数

4.4　メトリクスの定義と算出方法

進化を定量的に表現するためのメトリクスであるパッチ数，コミッタ昇格数，総開発者数は以下のように定義する．まずパッチ数は，Phannachitta らが提案したアルゴリズム [10] によって算出された月ごとのパッチの数を表している．次にコミッタ昇格数は，コミットログから各アカウントが初めてコミットする回数を月ごとに算出したものである．そして総開発者数は，メーリングリストにメールを投稿した月ごとの人数で表現している．ただし Eclipse においては，メーリングリストはほとんど使われることがないことから，開発者が用いる News Forum に投稿した人数をカウントしている．

5.　分析結果

この章では，前節のそれぞれのパターンにおいて，Apache と Eclipse のデータを用いて分析を行った結果を示す．各パターンにおいて加算係数，差分係数，遅延係数の 3 つのパラメータを変化させ，相関係数を算出した 2028 通りの中から，相関係数の絶対値が最大となる場合，最小となる場合を抽出する．表 5 は，相関係数の絶対値が最大および最小となった場合の相関係数，加算係数，差分係数，遅延係数を示している．

また，図 3 から図 10 はパラメータを固定したときの相関係数の平均を表した図になっている．図 3 から図 10 の a は，加算係数に注目し差分係数および遅延係数を変化させたものである．図 3 から図 10 の b は，差分係数に注目し加算係数および遅延係数を変化させたものである．図 3 から図 10 の c は，遅延係数に注目し差分係数および加算係数を変化させたものである．例えば，図 3 から図 10 の a において，加算係数が 1 の場合，パラメータを変化させ算出された 2028 個の相関係数の中から，加算係数が 1 の場合の相関係数の平均値を計算している．このようにすることによって，パラメータの値ごとの相関係数を把握することができると考えている．

我々が提案する遅延を考慮したデータマイニング手法と，既存手法との比較を行うために，表 6 を示す．表 6 は Apache および Eclipse において，遅延係数および加算係数，差分係数を全く考慮しない場合での相関係数を表している．

表 6　パラメータを用いずに行った場合での相関係数の一覧
Table 6　Correlation coefficients of the conventional methods

パターン	Apache	Eclipse
パターン A	0.268	−0.199
パターン B	0.465	−0.434
パターン C	0.226	0.384
パターン D	0.226	0.384

5.1　パターン A

5.1.1　Apache

説明変数に月ごとのコミッタ昇格数の累積値，目的変数にパッチ数の変化値としたときの Apache における相関係数は，8 カ月間の説明変数の加算，5 カ月の遅延で最大の 0.618 となった．これに対し，8 カ月間の説明変数の加算，10 カ月の遅延で 3 カ月前からの変化をみることによって相関はみられなくなった．パラメータの値次第でコミッタ昇格数がパッチ数に影響を与えないことがわかる．

5.1.2　Eclipse

前節と同じように説明変数に月ごとのコミッタ昇格数の累積値，目的変数にパッチ数の変化値としたときの Eclipse における相関係数は，5 カ月間の説明変数の加算，3 カ月の遅延で 12 カ月前からの変化をみることによって最大の

表 5　相関係数の絶対値が最大となる場合の各パラメータの値の一覧
Table 5　Result of the proposed method

パターン	プロジェクト	最大相関係数	加算係数	差分係数	遅延係数	最小相関係数	加算係数	差分係数	遅延係数
パターン A	Apache	0.618	8	0	5	0.000	8	3	10
	Eclipse	0.611	5	12	3	0.000	6	0	9
パターン B	Apache	−0.639	12	12	12	0.001	1	3	1
	Eclipse	0.638	12	12	12	−0.001	3	4	2
パターン C	Apache	−0.393	12	0	8	−0.001	1	2	10
	Eclipse	0.568	11	0	10	0.000	7	1	1
パターン D	Apache	0.420	9	0	12	0.000	4	11	3
	eclipse	−0.766	9	9	6	0.000	2	5	2

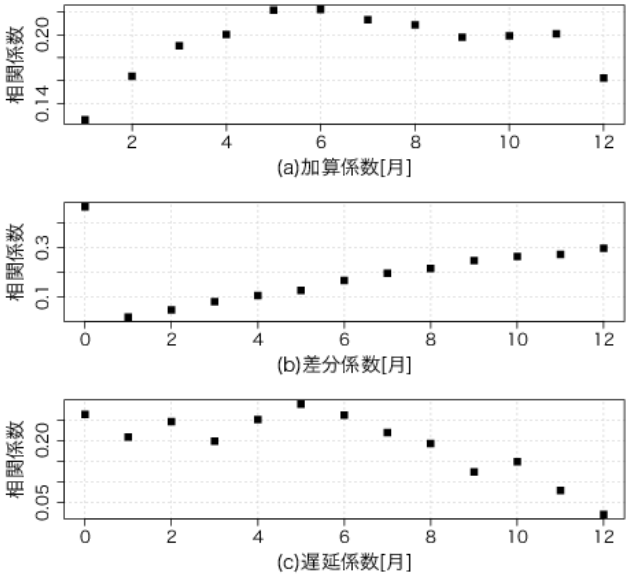

図 3　パターン A-Apache でのパラメータごとの相関係数の平均値
Fig. 3　Average value of the correlation coefficient of each parameter(Pattern A-Apache)

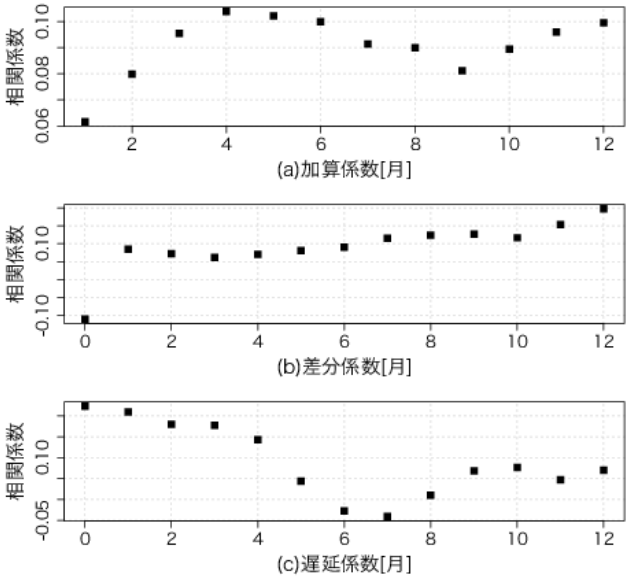

図 4　パターン A-Eclipse でのパラメータごとの相関係数の平均値
Fig. 4　Average value of the correlation coefficient of each parameter(Pattern A-Eclipse)

0.611 となった.

5.2　パターン B

5.2.1　Apache

説明変数に月ごとの総開発者数の累積値，目的変数にパッチ数の変化値としたとき，Apache における相関係数は，12 カ月間の説明変数の加算，12 カ月の遅延で 12 カ月前からの変化をみることによって最大の −0.639 となった.

5.2.2　Eclipse

前節と同じように，説明変数に月ごとの総開発者数の累積値，目的変数にパッチ数の変化値としたとき，Eclipse における相関係数は，12 カ月間の説明変数の加算，12 カ月の遅延で 12 カ月前からの変化をみることによって最大の 0.638 となった.

5.3　パターン C

5.3.1　Apache

説明変数に月ごとの総開発者数の累積値，目的変数にコミッタ昇格数の変化値としたとき，Apache における相関係数は，12 カ月間の説明変数の加算，8 カ月の遅延で最大の −0.393 となった.

5.3.2　Eclipse

前節と同じように，説明変数に月ごとの総開発者数の累積値，目的変数にコミッタ昇格数の変化値としたとき，Eclipse における相関係数は，11 カ月間の説明変数の加算，10 カ月の遅延で最大の 0.568 となった.

5.4　パターン D

5.4.1　Apache

説明変数に月ごとのコミッタ昇格数の累積値，目的変数に総開発者数の変化値としたとき，Apache における相関係数は，9 カ月間の説明変数の加算，12 カ月の遅延で最大の 0.420 となった.

5.4.2　Eclipse

前節と同じように，説明変数に月ごとのコミッタ昇格数の累積値，目的変数に総開発者数の変化値としたとき，

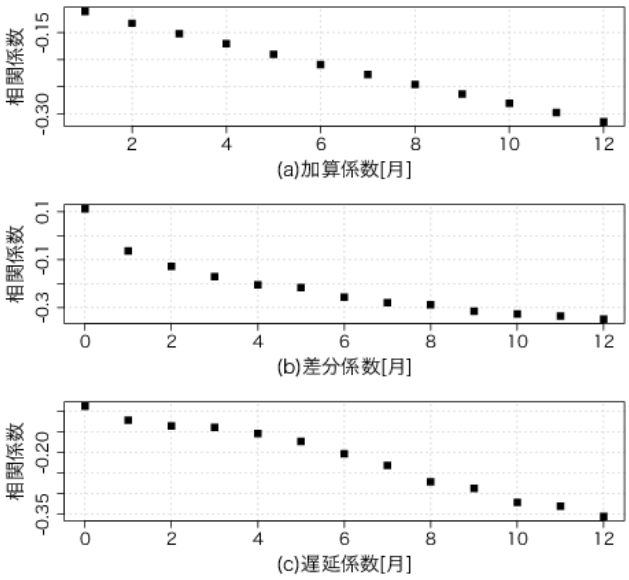

図 5 パターン B-Apache でのパラメータごとの相関係数の平均値
Fig. 5 Average value of the correlation coefficient of each parameter(Pattern B-Apache)

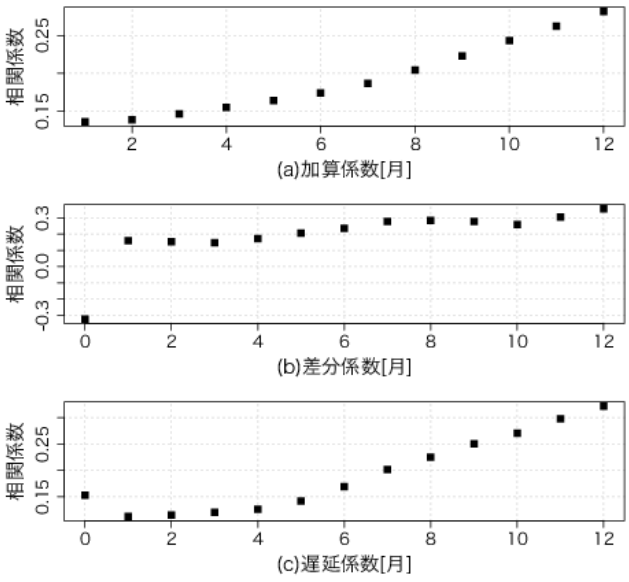

図 6 パターン B-Eclipse でのパラメータごとの相関係数の平均値
Fig. 6 Average value of the correlation coefficient of each parameter(Pattern B-Eclipse)

Eclipse における相関係数は，9 カ月間の説明変数の加算，6 カ月の遅延で 9 カ月前からの変化をみることによって最大の −0.766 となった．

6. 考察

本章では，ケーススタディから得た結果をもとに考察を行う．まず，遅延相関を考慮しない従来での相関分析による結果と，遅延相関を考慮した提案手法による相関分析の結果を比べ．考察を行う．次に，各プロジェクトにおける共進化のプロセスを定量的に示す．

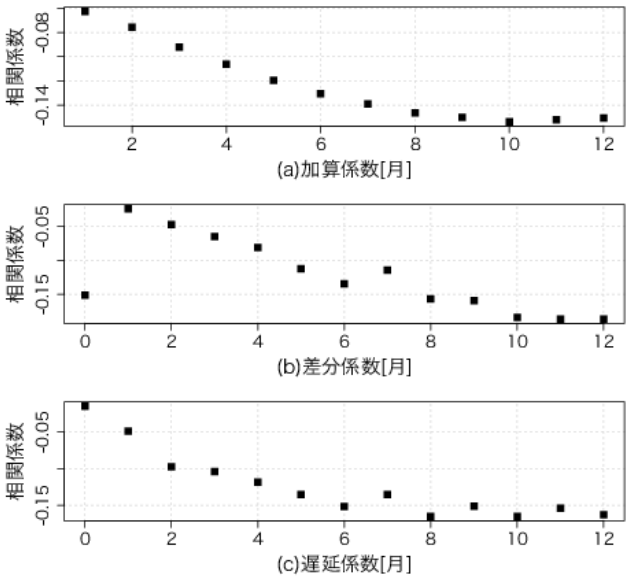

図 7 パターン C-Apache でのパラメータごとの相関係数の平均値
Fig. 7 Average value of the correlation coefficient of each parameter(Pattern C-Apache)

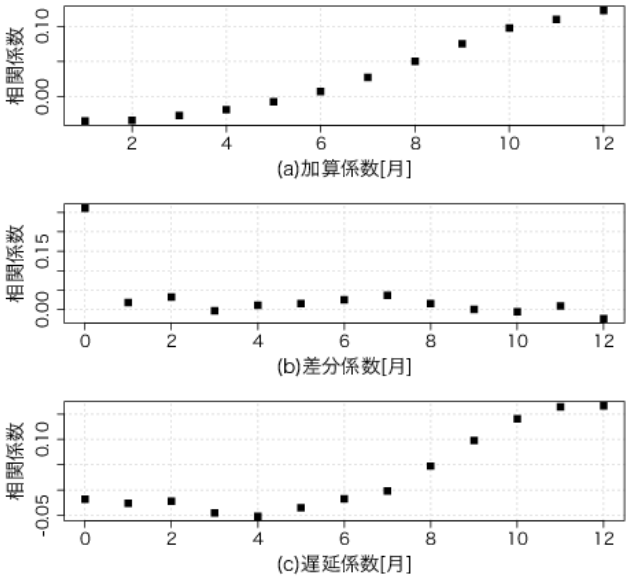

図 8 パターン C-Eclipse でのパラメータごとの相関係数の平均値
Fig. 8 Average value of the correlation coefficient of each parameter(Pattern C-Eclipse)

6.1 従来の手法での結果と提案手法での結果の比較

表 6 で示されている従来手法で算出された相関係数と，表 5 で示されている提案手法で算出された相関係数を比較する．すべてのパターンでどちらのプロジェクトであっても，従来手法で算出された相関係数の絶対値より，提案手法で算出された相関係数の絶対値の方が大きくなっていることがわかる．したがって，提案手法を用いると，従来手法では明らかにされなかった相関ルールが，遅延相関を考慮することによって，明らかにされていると考えられる．

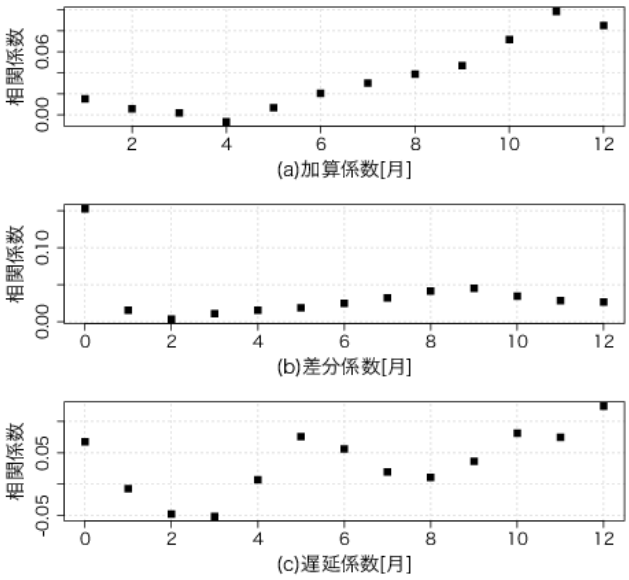

図 9 パターン D-Apache でのパラメータごとの相関係数の平均値
Fig. 9 Average value of the correlation coefficient of each parameter(Pattern D-Apache)

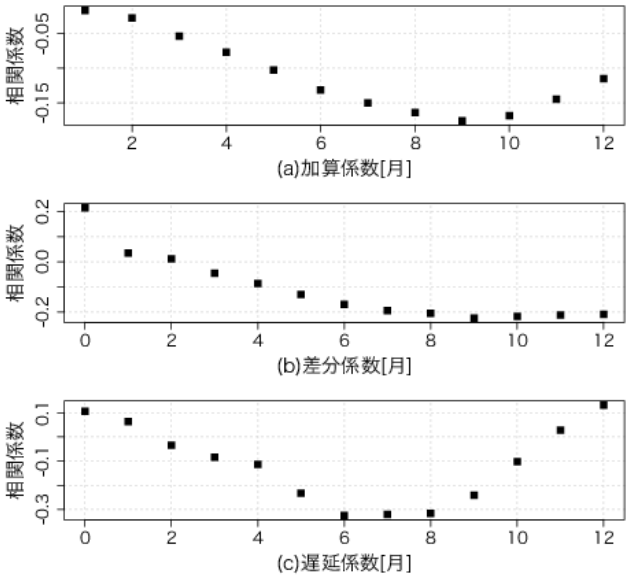

図 10 パターン D-Eclipse でのパラメータごとの相関係数の平均値
Fig. 10 Average value of the correlation coefficient of each parameter(Pattern D-Eclipse)

6.2 共進化プロセスの定量的理解

表 5 に示されているケーススタディの結果をもとに，各プロジェクトの共進化のプロセスを確認する．

6.2.1 Apache

Apache では以下のような共進化プロセスが確認できた．8 カ月間のコミッタ昇格数の増加が，5 カ月の遅延をもって，パッチ数の増加に影響を与える．また，9 カ月間のコミッタ昇格数の増加が，12 カ月の遅延をもって，総開発者数の増加に影響を与える．さらに，12 カ月間の総開発者数の増加が，12 カ月の遅延をもって，12 カ月前からのパッチ数の変化の減少に影響を与える．

このように，開発者の変化がコミュニティおよびアーティファクトに影響を与え，コミュニティの変化がアーティファクトに影響を与えることを確認することができた．

6.2.2 Eclipse

Eclipse においても，以下のような共進化プロセスが確認できた．5 カ月間のコミッタ昇格数の増加が，3 カ月の遅延をもって，12 カ月前からのパッチ数の変化の増加に影響を与える．また，12 カ月間の総開発者数の増加が，12 カ月の遅延をもって，12 カ月前からのパッチ数の変化の増加に影響を与える．さらに，11 カ月間の総開発者数の増加が，10 カ月の遅延をもって，コミッタ昇格数の増加に影響を与える．

このように，開発者の変化がアーティファクトに影響を与え，コミュニティの変化がアーティファクトおよび開発者に影響を与えることを確認することができた．

7. まとめと今後の課題

本研究では，OSS システムとコミュニティの共進化のプロセスを定量的に分析するためのデータマイニング手法を提案した．本手法は，遅延相関分析の考え方に基づき，一方の進化の系列が他方の進化の系列に与える影響を一定時間後に観察できることを考慮したものであった．また，遅延相関分析を容易に行うために，遅延相関係数が最も高くなる際の各種パラメータを自動的に求める点に特徴があった．提案手法の有用性を確かめることを目的として，Apache および Eclipse コミュニティを対象としたケーススタディを行った結果，遅延相関を考慮しない従来の分析結果と比べて，提案手法は共進化のプロセスをより正確に観察できることを確かめた．

本研究では，対象とするメトリクスをアーティファクトレイヤーにパッチ数，開発者レイヤーにコミッタ昇格数，コミュニティレイヤーに総開発者数とした．しかし，上記以外にも各レイヤーを表すメトリクスは多数考えられる．今後の課題としては，本研究で扱うことのなかったメトリクスを用い，共進化のプロセスをより正確に捉えていきたい．

8. 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 (online), DOI: 10.1109/MSR.2007.23 (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] DiBona, C., Stone, M. and Cooper, D.: *Open Sources 2.0–The Continuing Evolution*, O'Reilly Media, Sebastopol, CA (2005).

[4] 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 (online), available from ⟨http://dl.acm.org/citation.cfm?id=850948.853411⟩ (2000).

[5] 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 (online), DOI: 10.1109/ICSE.2007.74 (2007).

[6] 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).

[7] 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).

[8] Kumar, A. and Gupta, A.: Evolution of Developer Social Network and Its Impact on Bug Fixing Process, *Proceedings of the 6th India Software Engineering Conference (ISEC'13)*, pp. 63–72 (2013).

[9] 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).

[10] Phannachitta, P., Ihara, A., Jirapiwong, P., Ohira, M. and ichi Matsumoto, K.: An Algorithm for Gradual Patch Acceptance Detection in Open Source Software Repository Mining, *IEICE Transactions on Information and Systems*, Vol. E95-A, No. 9, pp. 1478–1489 (2012).

[11] Raymond, E. S.: *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly Media, Sebastopol, CA (1999).

[12] Scacchi, W.: *Understanding Open Source Software Evolution*, chapter 9, pp. 181–205 (online), DOI: 10.1002/0470871822.ch9, John Wiley & Sons, Ltd. (2006).

[13] 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).

[14] 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).

[15] 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 (online), available from ⟨http://dl.acm.org/citation.cfm?id=776816.776867⟩ (2003).

[16] 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).

[17] 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 (online), DOI: 10.1145/1882291.1882313 (2010).

[18] 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).

[19] 竹内裕之，児玉直樹，橋口猛志，林　同文：個人健康管理システムのための自動相関ルール抽出アルゴリズム，日本データベース学会 Letters，Vol. 5, No. 1, pp. 29–32 (2006).

[20] 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).