Refereed Conference paper

OSS開発における情報交換の効率改善へ向けたタイムラグ分析手法の提案

小山 貴和子[†1] 伊原 彰紀[†1] 柗本 真佑[†1]
亀井 靖高[†1] 大平 雅雄[†1] 松本 健一[†1]

オープンソースソフトウェア (OSS) は世界中に点在する様々な生活時間帯を持つ開発者により開発が進められている．このような開発環境では，時差の違いや開発に従事する時間帯の違いにより，開発者間の情報交換にタイムラグが発生していると考えられる．ソフトウェア中の不具合やセキュリティの脆弱性など，OSS であっても早急な対応が求められることが少なくないため，OSS の信頼性を確保する上で開発者間の効率的な情報交換を支援する必要性が高まっている．本稿では，OSS プロジェクト内における情報交換のタイムラグの実態把握および情報交換の効率化を支援することを目的とした分析手法を提案する．Python プロジェクトのメーリングリスト (ML) データに提案手法を適用したケーススタディを行った結果，時差による情報交換のタイムラグの有無を確認することができた．また，情報交換のタイムラグをできるだけ解消する最適な情報交換タイミングに関する知見を得ることができた．

A Time-Lag Analysis Method Toward Improving the Efficiency of Communications among OSS Developers

Kiwako Koyama,[†1] Akinori Ihara,[†1]
Shinsuke Matsumoto,[†1] Yasutaka Kamei,[†1]
Masao Ohira[†1] and Ken-ichi Matsumoto[†1]

Open source software (OSS) is created by globally distributed developers with a variety of lifestyles. In such the development environment, the time lag of communications among developers is more likely to happen due to the time difference among locations and the difference of working hours for OSS development. A means for effective communications among OSS developers has been growing in recent years, since even an OSS product requires a prompt response to issues such as defects and security vulnerabilities. In this paper, we propose an analysis method for observing the time lag of communications among developers in an OSS project and then facilitating effective communications. We have conducted a case study in which our analysis method was applied to mailing-list data of the Python project. As the results, we have confirmed that our method could identify the existence of the time-lag in communications among OSS developers and have achieved findings on the optimum timing for communications.

1. はじめに

Linux や Apache などに代表されるオープンソースソフトウェア (OSS) は，世界中に点在する開発者が参加する分散開発の形態をとる．OSS 開発は一般的な企業で実施する分散開発とは異なり，開発者は開発に従事する時間帯をプロジェクトによって束縛されることがなく個々人の自由な意志に基づき開発が進められる[4]．

このような開発環境では，開発者のタイムゾーンや生活時間帯 (夜型・朝型など) の違いから開発者間の情報交換には少なからずタイムラグが発生していると考えられる．Robles と Gonzalez-Barahona の調査[5] によると，SourceForge.net[⋆1] における開発者数はアメリカが最も数が多く，次いで西ヨーロッパ地域，中国となっている．これら 3 つの地域間では少なくとも 5 時間以上の時差があり，開発者同士がリアルタイムに議論する時間を確保することは容易ではない．また，たとえタイムゾーンの近い地域の開発者同士であっても，開発者個人の開発に従事する時間帯に束縛がないため，迅速な情報交換が可能であるとは限らない．

多数の開発者が参加する大規模な OSS プロジェクトでは，開発者同士の情報交換を通じた意思決定や合意形成の重要性が増す一方で，多種多様なタイムゾーンや生活時間帯を持つ開発者らによる情報交換は，タイムラグが発生しやすく，迅速な開発の妨げとなる可能性がある．特に，重大な不具合や脆弱性の修正といった早急な対策が求められるケースにおいては，情報交換のタイムラグによる意思決定や合意形成の遅延は，ソフトウェアの信頼性の低下のみならずユーザからの信頼を損なう要因となり得る．

本稿では，OSS プロジェクト内における情報交換のタイムラグの実態把握および情報交換の効率化を支援することを目的とした分析手法を提案する．提案する分析手法は，OSS プロジェクト内で利用されるメーリングリスト (ML) のアーカイブを対象として，(1) 開発

†1 奈良先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

⋆1 OSS 開発のための開発環境を提供する web サイト．2009 年 2 月現在，23 万以上の OSS プロジェクトが登録されている．

者の地理的分布と地域別活動時間帯の把握，(2) 地域間毎の情報交換所要時間の分布把握，(3) 最適な送信タイミングの特定，の3つのステップからなる．本稿では，Pythonプロジェクトを対象としたケーススタディから，提案手法の有用性を確かめる．

2. 分析方法

本章では，分析に先立ち準備する必要のあるデータの収集，整形および分類方法と，分析の手順について述べる．

2.1 分析準備

2.1.1 分析データの収集と整形

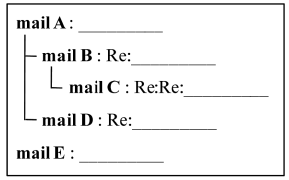
本稿で提案する分析手法は，OSS開発に携わる開発者が情報交換のために利用しているMLのアーカイブを対象とする．MLアーカイブを分析対象とした理由は，MLは多くのOSSプロジェクトで利用されており，OSSプロジェクト内の情報交換におけるタイムラグの実態を広く明らかにすることができると考えたためである．

提案手法では，まずメールの送信日時と送信場所のデータを収集する．送信日時とは送信者の現地時刻を指し，送信場所とは協定世界時 (UTC: Coordinated Universal Time) と現地時刻との時差で表す．例えば，日本の標準時はUTCより9時間進んでいるため「UTC+9」と表す．

図1はML利用時のメールの送信と返信の関係を収集する方法を示した概略図である．メールの送信者がMLにメールを送信すると，MLに登録されている開発者全員にメールが送信される．そして，送信されたメールに対して質問やコメントがある場合，返信者は送信者にメールを返信する．mail Aに対しmail Bが返信し，さらにmail Bに対しmail Cが返信する場合（図1(a)），図1(b)を基に送信と返信の関係を図1(c)のように一行（送信情報と返信情報）で表す．ただし，mail Eのように返信が行われていないメールは分析対象外とするため，図1(c)に現れない．また，収集した送信情報と返信情報を用いて，送信されてから返信までに要した時間（**返信時間**）を求める．

2.1.2 分析対象データの分類

返信時間を左右する要因としてタイムゾーンが異なる国や地域の違い，個人がOSS開発に従事する時間帯の違いなどが考えられる．例えば，タイムゾーンの異なる国の開発者間で行う情報交換では，食事や睡眠などの時間帯が異なるため，他方が睡眠中で返信できず返信時間が長引いてしまうことがある．また，OSS開発では個人の裁量で自由に活動を行うことができるため，同一地域の開発者同士であっても即座に返信が得られるとは限らない．



ID	送信日時	送信場所	協定世界時(UTC)
mail A	2009/04/29 10:33:53	UTC＋9	2009/04/29 01:33:53
mail B	2009/04/28 22:40:04	UTC－4	2009/04/29 02:40:04
mail C	2009/04/29 09:12:30	UTC＋3	2009/04/29 06:12:30
mail D	2009/04/29 02:26:59	UTC－10	2009/04/29 12:26:59
mail E	2009/04/29 08:09:27	UTC＋7	2009/04/29 15:09:27

(c) メールの送信・返信関係表

送信情報			返信情報			返信時間(時)
ID	送信日時	送信場所	ID	送信日時	送信場所	
mail A	2009/04/29 10:33:53	UTC＋9	mail B	2009/04/28 22:40:04	UTC－4	1.10
mail B	2009/04/28 22:40:04	UTC－4	mail C	2009/04/29 09:12:30	UTC＋3	3.54
mail A	2009/04/29 10:33:53	UTC＋9	mail D	2009/04/29 02:26:59	UTC－10	10.88

図1 分析データの整形方法

提案手法では，返信時間が24時間未満の送信・返信の関係と24時間以上の送信・返信の関係に分類し，タイムゾーンの違いによる返信時間のタイムラグと，個人の活動時間帯の違いによる返信時間のタイムラグを区別する．返信時間が24時間未満のデータは，地理的要因（時差）が影響して情報交換のタイムラグが起こっているものが多く存在すると考えられる．一方，返信時間が24時間以上のデータは，地理的要因よりも開発者個々人の活動時間帯の違いといった人的要因が大きく影響してタイムラグを発生させているものと考えられる．本稿では，時差による情報交換のタイムラグを確認するために返信時間が24時間未満のものを対象とする．

2.2 分析手順

2.2.1 開発者の地理的分布と地域別活動時間帯の把握

OSSプロジェクト内における情報交換のタイムラグの実態を把握するために，OSS開発に参加する開発者の地理的分布を，返信情報に含まれる送信場所 (UTC-11～UTC+12) 別のメールの送信数（**返信数**）から求める．さらに，同一地域内でも開発者の活動時間帯は異なるため，分析対象とする地域の現地時刻別の返信数の分布を求め，地域毎の活動時間帯を把握する．本項で示した手法は，情報交換の多い地域あるいは少ない地域を特定し，地域毎

の活動時間帯を把握することが可能である.

2.2.2 地域間毎の情報交換所要時間の分布把握

時差による情報交換のタイムラグの実態を把握するために，同一タイムゾーン内において情報交換に要する時間と，異なるタイムゾーン間において情報交換に要する時間の分布を分析する．本項で示した手法は，タイムゾーンの違い（時差）による情報交換のタイムラグを確認することが可能である.

2.2.3 最適な送信タイミングの特定

送信者にとって情報交換のタイムラグをできるだけ解消する最適な情報交換タイミングを求めるために，送信者のメール送信時刻（**送信時刻**）と返信者のメール送信時刻（**返信時刻**）から各時刻の返信数を分析する（**図 2**）．図 2 の各行は送信時刻，各列は返信時刻，(a) の各セルの数字はタイムラグの大きさ，(b) の各セルの数字は返信数を表している．つまり，(a) は地域 A と地域 B の各時刻におけるタイムラグの大きさを表し，(b) は各時刻における図 1(c) のデータ（送信情報と返信情報の対）の件数を表している．例えば，地域 A の 9 時から 12 時に送信し，地域 B の 15 時から 18 時に返信があった場合，タイムラグは+3 時間であり，返信数は 80 件であることを示す．また，対角線上（左上から右下）のセルに，返信 1 時間未満である返信が対応するように地域 B の返信時刻を合わせる．なお，図 2 では 3 時間毎の返信数としている．さらに，返信数の多い時刻を俯瞰的に把握するために，全セルを返信数に合わせてグレースケールで色付けする．返信数の最小値のセルを白で色付けし，最大値のセルを黒で色付けする．本項で示した手法を用いることで，情報交換のタイムラグが発生しにくい送信時刻を特定および情報交換のタイムラグが発生しやすい送信時刻を特定することが可能である．例えば，地域 A の 21 時から 0 時（図 2(b) の一番下の行）は図 2(a) より 6 時間後に返信される割合が多く，タイムラグの少ない情報交換タイミングとしては適切ではないといえる.

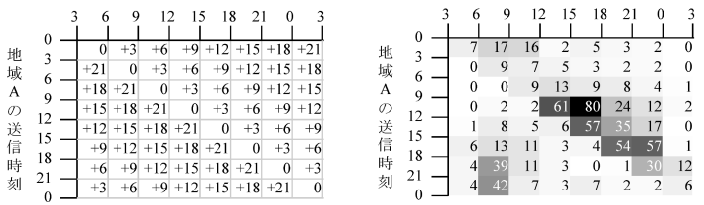


図 2 返信時刻の分布

3. ケーススタディ

本章では Python プロジェクトを対象としたケーススタディを行い，提案手法を用いることで情報交換のタイムラグの実態を確かめることができるかを確認する.

3.1 Python

Python はオープンソースで開発されているオブジェクト指向のスクリプト言語であり，Perl とともに欧米で広く普及している．多くのプラットホームをサポートしており，豊富なドキュメントや豊富なライブラリがあることから，Web プログラミング，GUI ベースのアプリケーション，CAD，3D モデリング，数式処理等幅広い分野で使用されている.

3.2 分析対象データ

分析対象データとしては，Python の開発に関する話題を取り扱う ML「Python-Dev」のアーカイブを用いる.「Python-Dev」は主に新規機能やメンテナンス，リリースについて議論が行われる．なお分析では，送信日時，送信場所，どのメールへ返信したか（返信情報）などが含まれていないメールを対象外とした.

分析期間としては，ML への投稿履歴が存在する期間 1999 年 4 月～2009 年 4 月を用いる．全メール数は 89,301 件であり，提案手法を行うための分析準備を行った結果，送信・返信の数が 56,707 件存在していた．また，返信時間が 24 時間未満の送信・返信の数が 51,830 件存在していた.

3.3 分 析 結 果

3.3.1 開発者の地理的分布と地域別活動時間帯の把握

各タイムゾーンの返信数の分布を**図 3** に示す．図 3 の横軸は各タイムゾーンを表し，縦軸は返信数を表す.

図 3 より，Python プロジェクトは UTC-4（米大陸東部）および UTC+2（中央ヨーロッパ地域）の開発者からの返信が多いことがわかる．Python の利用は欧米中心であり開発も欧米中心に行われているため，当該地域からのメールが多かったと考えられる．UTC-4 および UTC+2 の地域の中にはサマータイムを導入している国も多く，隣接するタイムゾーンからのメールも多いため，以降では UTC-4 周辺の地域（米大陸地域：UTC-8～UTC-4）および UTC+2 周辺の地域（欧・アフリカ地域：UTC+0～UTC+3）の 2 つの地域を分析対

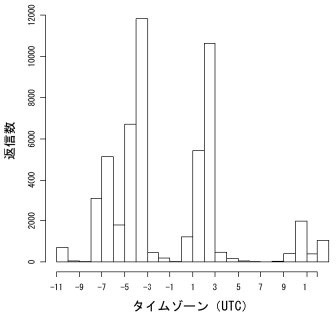


図 3　地域別返信数の分布

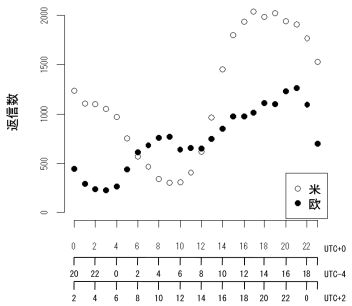


図 4　時刻別返信数の分布（米：米大陸地域，欧：欧・アフリカ地域）

表 1　分析該当地域

	タイムゾーン	該当地域
米大陸地域	UTC−8〜UTC−4	アメリカ，カナダ，ブラジル西部，チリ，ボリビア，メキシコ 等
欧・アフリカ地域	UTC+0〜UTC+3	ヨーロッパ，アフリカ，モスクワ，イラン，サウジアラビア 等

象とする．この 2 つの地域に該当する地域を**表 1** に示す．

図 4 は図 3 より決定した 2 つの地域における各時刻の返信数の推移を示す．横軸の上段は UTC+0，中段は UTC−4，下段は UTC+2 のそれぞれのタイムゾーンの時刻を表し，縦軸は各標準時の返信数を表す．

図 4 より，米大陸地域からの返信数が最大となる時間帯は現地時刻（UTC−4）の 13 時であり，最小となる時間帯は 5 時であることがわかる．このことより，米大陸地域の開発者は昼の時間帯を中心として開発に従事しているといえる．一方で，欧・アフリカ地域からの返信数が最大となる時間帯は現地時刻（UTC+2）の 23 時であることから，欧・アフリカ地域の開発者は夜の時間帯を中心として開発に従事しているといえる．このように，各地域の返信数から活動時間帯を分析することで，地域別の開発への従事時間帯の違いがわかった．

UTC+0 の 12 時から 23 時では，両地域共に返信数が多く，情報交換の活動が活発に行われており，時差のある地域間であっても情報交換にタイムラグは発生していない可能性がある．一方，UTC+0 の 0 時から 12 時では，欧・アフリカ地域，米大陸地域のどちらか一方の地域の返信数が少なく，情報交換の活動が活発に行われておらず，情報交換にタイムラグが発生している可能性がある．図 4 より，各地域の返信数からそれぞれの地域における活動時間の違いを見ることは可能であるが，実際に情報交換のタイムラグが発生しているのかを判断することはできない．

表 2　地域別の返信時間の統計量（米：米大陸地域，欧：欧・アフリカ地域）

送信地域 → 返信地域	返信数	最大値（時）	中央値（時）	最小値（時）
米 → 米	18,901	11.55	1.24	0.00
米 → 欧	6,942	16.34	2.07	0.00
欧 → 欧	9,426	14.69	1.59	0.00
欧 → 米	7,215	13.91	1.80	0.00

3.3.2　地域間毎の情報交換所要時間の分布把握

タイムゾーンが同一地域間における返信時間と，タイムゾーンが異なる地域間における返信時間の返信数，最大値，最小値，中央値を**表 2** に示す．以降，ある地域 X のメールに対して地域 Y から返信されたメールの対を「X → Y」と示す．

タイムゾーンが同一地域間では返信時間の中央値が米→米と欧→欧ではそれぞれ 1.24 時間と 1.59 時間であり，タイムゾーンが異なる地域間では米→欧と欧→米では 2.07 時間と 1.80 時間であった．タイムゾーンが同一地域間では 90 分程度の返答が期待できるが，異なるタイムゾーン間でも 2 時間未満の返信が多いことがわかる．しかし米大陸地域と欧・アフリカ地域では実際には 6 時間近い時差があることから，Python プロジェクトでは時差による情報交換のタイムラグが少ないことがわかる．

3.3.3　最適な送信タイミングの特定

図 5 は図 3 より決定した 2 つの地域間における各時刻の返信数の分布を示す．なお，図中には返信数の値を削除し，グレースケールで色付けした結果のみを示している．図 5 の (a)(b)(c)(d) はそれぞれ，米→米，米→欧，欧→欧，欧→米を示す．

タイムラグ 0 の返答が期待できる時刻 (対角線付近のセルの色が濃くなっている時刻) は，図 5(a) では 10 時から 17 時頃，図 5(b) では送信時刻の 9 時から 17 時頃かつ返信時刻の 15 時から 23 時頃，図 5(c) では 16 時〜23 時頃，図 5(d) では送信時刻の 16 時から 23 時かつ返信時刻の 10 時から 17 時頃であることがわかる．これらの時刻では返信時間が短いため，リアルタイム性の高い情報交換が活発に行われていることがわかる．

さらに，図 5(b) の送信時刻の 18 時から 23 時頃および図 5(d) の送信時刻の 7 時から 13 時頃では，同時刻から遅れた時間帯 (対角線付近から離れた部分) に色の濃い部分があり，返信時間が遅くなっていることがわかる．この要因として上記 2 つの送信時刻は返信地域の

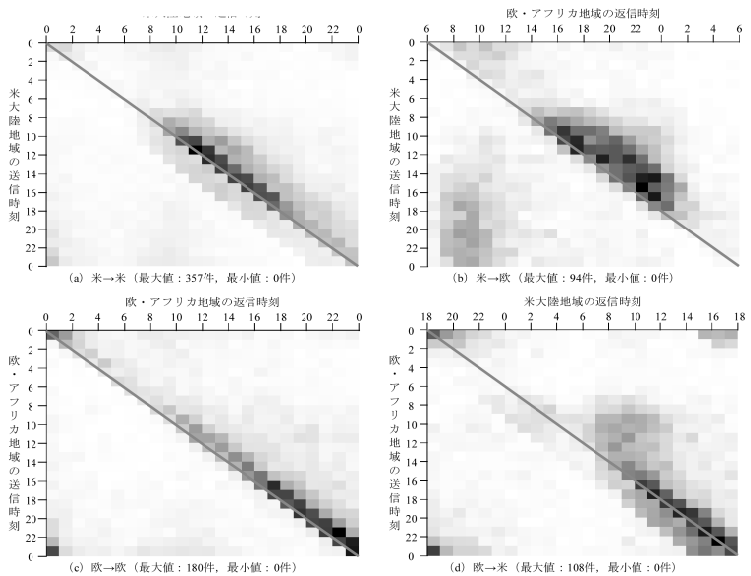


図 5 2 地域間の返信時刻の分布（米：米大陸地域，欧：欧・アフリカ地域）

深夜から早朝にかけて（0 時〜6 時）であることから，返信時間が遅くなったと考えられる．

図 5 の結果より，素早い返答を得るためには米大陸地域では 10 時から 17 時頃，欧・アフリカ地域では 16 時から 23 時頃にメールを送ることが望ましいといえる．また米大陸地域の 18 時から 23 時頃，および欧・アフリカ地域の 7 時から 13 時頃はタイムラグが発生しやすく，リアルタイムな議論を行うタイミングとしては適切ではない．このように提案手法を用いることで，情報交換のタイムラグをできるだけ解消する適切な情報交換タイミングを求めることができた．

4. 議　　論

3.3.2 項の分析結果より，Python プロジェクトでは時差の影響による情報交換のタイムラグが小さいことがわかった．この要因の一つとして，分析対象とした 2 つの地域の活動時間帯の重なりが考えられる．この 2 つの地域は 6 時間程度の時差があるものの，3.3.1 項の分析結果より，各地域の OSS 開発へ従事する時間帯が異なっていた（米大陸地域：昼，欧・アフリカ地域：夜）．そのため，図 4 に示すように米大陸地域の 10 時から 17 時（欧・アフリカ地域の 16 時から 23 時）の間に両地域の開発者の開発に従事する時間帯が偶然にも重なっていた可能性がある．もう一つの原因としては分析対象プロジェクトの規模が大きく，どの時間帯においても開発に従事する開発者が存在していたため，送信に対する素早い対応が可能であったと考えられる．

提案手法は ML における情報交換活動だけでなく，情報交換活動の媒体を変更するときにおいて利用可能であると考える．例えば，リアルタイム性の低い ML からリアルタイム性の高い IRC 媒体に移行するときは，情報交換のタイムラグをできるだけ解消するために情報交換の最適なタイミングの理解が必要であると考えられる．その際に，提案手法を用いることにより適切な送信時刻を提示できると考える．

OSS 開発はプロジェクトによって，開発者が地理的に分散していない場合がある．提案手法は時差の影響による情報交換のタイムラグの実態を確認するためのものであるが，時差のみならず開発者間の生活時間帯の違いによる情報交換のタイムラグの把握においても利用可能であると考える．OSS 開発は開発者個人の活動時間帯に束縛がなく，個人の裁量で自由に活動を行うことができるため，同一タイムゾーンに住む開発者間の情報交換においてもタイムラグが発生する可能性がある．このようなケースにおいても，提案手法を用いることにより適切な送信時刻を提示できると考える．

また，提案手法は OSS プロジェクトを対象としているが，一般企業における分散開発においても利用可能であると考える．一般企業では就業時間がある程度決まっているが，必ずしも定められた時間帯に他の開発地域との情報交換を行えるとは限らない．このようなケースにおいても，提案手法を用いることにより適切な送信時刻を提示できると考える．

本稿では，ML のアーカイブを対象として提案手法を適用した．OSS 開発の情報交換の様子がわかるコミュニケーションログは，OSS プロジェクト内における情報交換のタイムラグの実態を把握するために必要である．ML を用いた情報交換は，IRC や BBS を用いた情報交換よりもリアルタイム性は低いが，返信したいメッセージに対して返信することが可能であり，返信が遅れても使用できる．そのため，ML における情報交換はタイムラグが発生しやすい可能性がある．また，ML のアーカイブは，送信者や返信相手を明確に記録している場合が多い．一方，IRC や BBS を用いた情報交換は，ML を用いた情報交換よりも

リアルタイム性は高いが，返信相手を記録していない場合が多く，返信が遅れると議論が既に終了している可能性もあり，いつでも情報交換をすることができるとは限らない．そのため，IRC や BBS における情報交換はタイムラグが発生しにくい可能性がある．また，IRC や BBS のアーカイブは，時系列順に並んでいるのみで，どのメッセージに対して返信したのか判断することが難しい．これらの特徴から本稿では，タイムラグが発生しやすく，どのメッセージに対する返信であるかが明確となっている ML のアーカイブを対象として提案手法を適用した．

5. 関連研究

分散開発における情報交換のタイムラグの実態把握を目的とした研究が今までに行われている[1)2)3)]．例えば，Harbsleb ら[1)] は企業のオフショア開発を対象とし，開発者が各拠点に点在している開発プロジェクトは（していないプロジェクトと比べて）開発スピードに遅延が生じているかを調査している．調査の結果，開発者が点在している分散開発は，“face-to-face” コミュニケーションを欠落させ，開発スピードに遅延をもたらすことが明らかとなった．一方で，Nguyen ら[3)] は OSS 開発を対象とし，従来報告されているように分散開発が地域差により情報交換のタイムラグをもたらしているかを調査している．Eclipse Jazz プロジェクトを対象とした調査の結果，開発者間の地域差は情報交換のタイムラグに影響を与えてはいるが，それほど大きなものではないという結論を出している．これらの研究では開発者間の地域差に着目してはいるものの，開発者が開発に従事する時刻までは考慮されていない．本稿では，メールの送信・返信時刻を細分化し，開発者間の時差と送信時刻がそれぞれ情報交換のタイムラグにどういった影響を与えているかを分析した点，および，最適な送信タイミングを特定した点が異なる．

6. おわりに

本稿では，OSS プロジェクト内における情報交換のタイムラグの実態把握および情報交換の効率化を支援することを目的とした分析手法を提案した．Python プロジェクトの ML データに提案手法を適用したケーススタディを行った結果，以下の知見を得ることができた．

- 分析対象とした各地域の開発に従事する時間帯の違いを確認することができた．
- 分析対象プロジェクトでは，時差による情報交換のタイムラグが少ないことが確認できた．
- 情報交換のタイムラグをできるだけ解消する最適な情報交換タイミングを確認することができた．

しかしながら，本稿では時差のある地域間や複数の開発者を対象として提案手法を適用しており，時差のない地域間や個々の開発者については考慮していない．今後は時差のない地域間や個々の開発者についても考慮して提案手法の有用性を確認していきたい．

謝辞 本研究を遂行するにあたり，有益なコメントを下さった奈良先端科学技術大学院大学の藤田将司氏に感謝する．

本研究の一部は，文部科学省「次世代 IT 基盤構築のための研究開発」の委託に基づいて行われた．また，本研究の一部は，文部科学省科学研究補助費（若手 B：課題番号 20700028, 21・8995, 20・9220）による助成を受けた．

参考文献

1) Herbsleb, J.D., Mockus, A., Finholt, T.A. and Grinter, R.E.: An empirical study of global software development: distance and speed, *Proceedings of the International Conference on Software Engineering*. pp.81–90 (2001).
2) Milewski, A.E., Tremaine, M., Egan, R., Zhang, S., Kobler, F. and O’Sullivan, P.: Guidelines for effective bridging in global software engineering, *Proceedings of the International Conference on Global Software Engineering*, pp.23–32 (2008).
3) Nguyen, T., Wolf, T. and Damian, D.: Global software development and delay: does distance still matter?, *Proceedings of the International Conference on Global Software Engineering*, pp.45–54 (2008).
4) Raymond, E.S.: *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, O’Reilly and Associates (1999).
5) Robles, G. and Gonzalez-Barahona, J.M.: Geographic location of developers at SourceForge, *Proceedings of the International Workshop on Mining Software Repositories*, pp.144–150 (2006).