グローバル環境下における OSS 開発者の情報交換に対する時差の影響

Studying the Impact of Time Difference on Global Communications among OSS Developers

亀井 靖高 (Yasutaka KAMEI)[1] 大平 雅雄 (Masao OHIRA)[2]
伊原 彰紀 (Akinori IHARA)[3] 小山 貴和子 (Kiwako KOYAMA)[4]
松本 真佑 (Shinsuke MATSUMOTO)[5] 松本 健一(Ken-ichi MATSUMOTO)[6]
鵜林 尚靖 (Naoyasu UBAYASHI)[7]

[1,7]九州大学 大学院システム情報科学研究院 [1]助教、[7]教授
[2, 3, 6]奈良先端科学技術大学院大学 情報科学研究科 [2]助教、[3]博士後期課程、[6]教授
[4]東芝 ソフトウェア技術センター
[5]神戸大学 大学院システム情報学研究科 特命助教

［Abstract］
OSS (Open Source Software) is developed by globally distributed developers who communicate between different time zones. In a large-scale project where a number of developers are involved, it is important to make a decision and build a consensus through communication among developers. The communication between different time zones, however, would cause a delay of information exchange due to time lag, resulting in impeding rapid development. Since even OSS products are required to quickly respond to issues such as defects and security vulnerabilities, better understandings of the time lag in an OSS project must be constructed in order to facilitate efficient communication among developers. The goal of this paper is to reveal the existence of the communication delay in OSS project and then to achieve a guide for improving the communication among distributed OSS developers. This paper frames two hypotheses on factors of the communication delay in an OSS project. As a result of analysis, we found that the communication delay is likely to happen when a messages is received midnight and that there are at least three hours for which developers can communicate each other without the influence of time-lag.

［キーワード］
グローバルコミュニケーション，分散ソフトウェア開発，オープンソース，情報交換のタイムラグ，時差分析

1．はじめに

近年のソフトウェア開発企業は，短期間かつ限られた開発コストの中で高品質なソフトウェアを生産することが求められている．コスト削減の一手段として多くのソフトウェア開発企業では，無償かつ商用のソフトウェアに劣らない品質と機能を備えているオープンソースソフトウェア(以降,OSS)を自社製品の一部として導入している[25]．例えば，携帯情報端末向けソフトウェアプラットフォーム「Android」は，2007 年 11 月に公開されて以降，多数の携帯電話に用いられ世界の携帯電話市場を賑わしている．OSS はソフトウェア開発企業のみならず，行政機関や教育機関においても広く導入されつつあり[32]，OSS の社会的影響は今後も拡大していくものと予想される．

OSS 開発プロジェクトの特徴の 1 つは，世界中に点在する開発者が共同開発を行うという分散開発の形態をとることである．例えば，Robles らの調査[29]によると，SourceForge.net における開発者数[1]はアメリカが最も多く，次いで西ヨーロッパ地域，中国であり，地域や時差をまたがった開発が行われていることが見て取れる．また,FreeBSD プロジェクト[2]を対象にした Diomiddis の調査[31]によると,開発者は北米やヨーロッパだけでなく，アジア，オーストラリア，南アフリカ，南米に点在していた．

[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 開発プロジェクトが登録されている．
[2] UNIX 系のオペレーティングシステムをオープンソースで開発するプロジェクト

OSS 開発プロジェクトでは地域や時差をまたがったグローバル環境下で開発が行われるという特徴から，開発者同士の情報交換は主に電子メールを代表とする非対面・非同期のコミュニケーション手段が用いられている．非対面・非同期のコミュニケーション手段によって開発者は，自身の都合の良いタイミングで，地域の離れた開発者に情報発信できる．

日常生活で電子メールにより情報交換を行う場合，時差の影響を受けて情報交換の達成に時間がかかると経験的には考えられている．その一方で， OSS 開発プロジェクトでは必ずしも当てはまるとは考えられていない．近年，ソフトウェア開発企業ではオフショア開発が積極的に実施され， その成功事例が多数報告されている．成功要因の１つは，世界中に点在するいずれかの開発者によって 24 時間体制で開発が実施され，いずれかの開発者によって常にメールが返信される体制が築かれている点である[23]．企業での成功事例から，OSS 開発プロジェクトでも高品質なプロダクトをグローバル環境下で効率的に開発できるものと考えられてきた[7]．

しかしながら，我々はグローバル環境下では各開発者の時差帯の違いから開発者間の情報交換にタイムラグが少なからず発生すると考える．例えば，先ほど例にあげた 3 つの地域間（アメリカ，西ヨーロッパ，中国）では各地域間で 5 時間以上の時差があり，リアルタイムな情報交換が困難であることが推察される．また，時差帯の違いのみならず，開発者の生活サイクルの違いも情報交換のタイムラグに影響を与えると考えられる．情報交換のタイムラグは，開発スピード，生産性，及び，ソフトウェア品質管理能力の低下など，ソフトウェア開発全体に悪影響をもたらすといわれている[1][5][6][18][23][26]．

本研究の目的は，OSS 開発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解明し，グローバル環境下におけるメーリングリストを用いた情報交換の所要時間を短縮化するための指針を得ることである．情報交換の媒体としてメーリングリストに着目する理由は，開発者全員に情報交換の過程が共有されるという利点を持ち，大半の OSS 開発プロジェクトで日常業務時の情報交換に用いられているからである． 本論文では，OSS 開発プロジェクトにおける情報交換のタイムラグに関する仮説を，大規模な複数のプロジェクトを分析することにより実験的に検証する．アプローチとして，メールのヘッダを解析することで各開発者の居住地域，及び，送返信の関係を分析した．具体的な仮説としては，仮説 1：時差のある地域間の情報交換は返信時間が長くなる，仮説 2：返信者の地域によって素早い返信を期待できる時間帯が異なる，を確かめる．

上記 2 つの仮説に関しては日常生活の電子メールのやりとりでは経験的には正しいと理解されている部分もある．しかし，OSS 開発プロジェクトでは具体的にどの程度正しいのかは明らかにされておらず，また実際のデータに対する分析の報告事例は少ない．特に本論文で用いる Apache や Eclipse のような高い能力をもった開発者が集う巨大なソフトウェア開発プロジェクトで，情報交換の所要時間を短縮化することについては調査されていない．本論文の貢献は，これら経験的に知られたグローバル環境下における情報交換に対する時差の影響に対して，OSS 開発プロジェクトデータから調査を行い，定量的な分析に基づいた客観的な知見を得たことにある．

本論文の構成は以下の通りである．続く 2 章では OSS 開発プロジェクトにおける情報交換とタイムラグについて述べる．3 章ではケーススタディとして，分析データの収集と整形手順，及び，対象のプロジェクトについて述べる．4 章では，2 つの仮説それぞれに対して分析方法を述べ，その分析結果を報告する．5 章では，分析結果に対する考察と本論文の制約について議論を行う，6 章では関連研究を述べ本研究の立場を明らかにし，最後に 7 章で本論文の結論について述べる．

２．OSS 開発プロジェクトにおける情報交換とタイムラグ

２．１ OSS 開発プロジェクトの情報交換手段

OSS は世界中に点在する，つまりグローバル環境下にある開発者によって開発されるため，開発者は情報交換手段として非対面でかつ非同期な媒体であるメーリングリスト(以降，ML)や掲示板を主に用いている．特に ML は，送信されたメッセージが参加者全員に配信され，各開発者間のやりとりをプロジェクトのメンバー全員で共有できるという特徴を持つ．そのため，ML はほとんどの OSS 開発プロジェクトで利用されており，かつ，プロジェクト内における主たる情報交換手段である場合が大半である[17]．

２．２ 情報交換におけるタイムラグ

ML はメールが参加者全員に配信されるため，グローバル環境下における OSS 開発プロジェクトでは，常に誰かが開発に従事しメッセージを読んでおり，リアルタイムな返信が期待される．しかしながら，現実の OSS 開発プロジェクトに目を向けると必ずしもリアルタイムな返信が行われているわけではない．その理由の１つは，開発者は返信に際して知識が必要とされるメールを送信している場合が多く，全開発者が当該メッセージの返信対象というわけではないからである．例えば，開発者がセキュリティ関連のソースコードの仕様（プログラムの振る

舞い）について質問する場合，その担当開発者がメールに返信する必要がある．そのため，常に誰かが開発に従事するグローバル環境下におけるOSS開発プロジェクトといえども，情報交換におけるタイムラグが発生する．

本論文では，上記のような状況において，送返信の関係にある開発者の間で時差が大きい場合に，情報交換のタイムラグが発生する可能性が高いと考える．

２．３ タイムラグの実態に関する予備調査

本研究を実施するにあたりPythonプロジェクトの情報交換におけるタイムラグの実態を予備的に調査した．ML上でのアメリカ在住の開発者とヨーロッパ在住の開発者との情報交換の様子を観察した結果，ヨーロッパの現地時刻1時頃（アメリカの現地時刻18時頃）にアメリカ在住の開発者がメッセージを送信しても，ヨーロッパ在住の開発者からすぐに返信されることはなく，ヨーロッパの現地時刻9時頃になってから返信されるケースが多く見られた．一例を図1に示す．

メール本文に“Good morning.”と記述されていることと，朝になってからメッセージが返信されていることから，ヨーロッパ在住の開発者はアメリカ在住の開発者がメッセージを送信した時間帯には就寝していたものと考えられる．図1の例では，返信までに要した時間（返信時間）は約9時間であり，時差の存在する地域に居住する開発者間の情報交換は，情報交換を行う時間帯が異なるためタイムラグが発生しやすい．

From: us@ait.kyushu-u.ac.jp
Date: Sun, 30 Jul 2000 **18:26:00** -0500(アメリカ)
Subject: [Python-Dev] title

messages ……..

From: eu@ait.kyushu-u.ac.jp
Date: Mon, 31 Jul 2000 **09:43:55** +0200(ヨーロッパ)
Subject: Re: [Python-Dev] title

Good morning.
messages ……..

図-1 情報交換の一例

３．ケーススタディ

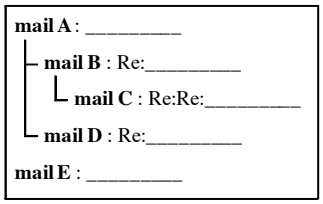
３．１ 分析データの収集と整形

本論文では，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などのアーカイブを対象とする．多くのOSSプロジェクトが情報交換に用いているMLを分析することで，OSS開発者間のタイムラグの実態を明らかにすることができると考える．

本論文では，まずメールのヘッダ（図1の"From"，"Date"の部分）を解析し，メッセージ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，図1の東部標準時からメッセージを送信した場合の標準時はUTCより5時間遅れているため“UTC-5”と表す．

図2はML利用時の送信メッセージと返信メッセージを収集する方法を示した概略図である．図2(a)はメッセージのスレッド構造を表し，図2(b)はメッセージの送信日時を表す．メッセージのスレッド構造と送信日時から，図2(c)に示すようなメッセージの送信と返信の関係を表すデータ構造を作成する．mail Aに対しmail Bが返信され，さらにmail Bに対しmail Cが返信される場合（図2(a)），図2(b)を基に送信と返信の関係を図2(c)のように一行（送信情報と返信情報の対）で表す．また，収集した送信情報と返信情報を用いて，返信時間を求める．ただし，mail Eのように返信が行われていないメッセージは分析対象外とするため，図2(c)に現れない．以降，地域Aから送信されたメッセージに対して，地域Bから返信したメッセージの対を“A → B”と示す．

(a) メッセージのスレッド構造



(b) メッセージの送信日時

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) メッセージの送信・返信関係表

送信情報			返信情報			返信時間(時)	表記方法
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	A? B
mail B	2009/04/28 22:40:04	UTC－ 4	mail C	2009/04/29 09:12:30	UTC＋3	3.54	B? C
mail A	2009/04/29 10:33:53	UTC＋9	mail D	2009/04/29 02:26:59	UTC－ 10	10.88	A? D

図-2 分析データの整形方法

３．２　分析対象プロジェクト

数多くの開発者が参加する大規模なプロジェクトを分析対象とすることにより，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明らかにできると考え，以下のプロジェクトを分析対象とする．

● Python プロジェクト[28]（以降，Python）

1990 年に Guido van Rossum によって開発され，ABC 言語の後続言語として発足したオブジェクト指向のスクリプト言語の開発プロジェクトである．Python は Perl とともに欧米に広く普及しており，多くのプラットフォームをサポートし，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．

● PostgreSQL プロジェクト[27]（以降，PostgreSQL）

BSD ライセンスによって配布されているオブジェクト関係データベース管理システム（ORDBMS）の開発プロジェクトである．1986 年にカリフォルニア大学バークレー校（University of California, Berkeley）で Michael Stonebraker によって POSTGRES プロジェクトとして発足し開発されたものが，1995 年にオープンソース版（Postgres95）として Web 上に公開された後、1996 年に名称を PostgreSQL に変更して 1997 年 1 月にバージョン 6.0 として公開され現在まで開発が続いている．

● Apache HTTP Server プロジェクト[2]（以降，Apache）

1995 年に NCSA（National Center for Supercomputing Applications）の Robert McCool らによって開発された Web サーバ（NCSA HTTPd）にパッチを提供していた開発者らによって発足されたプロジェクトである．NCSA HTTPd 1.3 をベースに障害修正や新機能の追加を繰り返し行い，1996 年に Apache 1.0 が公開され，現在はバージョン 1.3.x 系列，2.0.x 系列，2.2.x 系列の開発が並行して進められている．

● Eclipse プロジェクト[13]（以降，Eclipse）

IBM（International Business Machines Corporation）によって開発された統合開発環境（IDE: Integrated Development Environment）の開発プロジェクトである．1998 年 11 月に IBM カナダでプロジェクトが開始されたが，2001 年 11 月にオープンソース化し，現在 50 以上のサブプロジェクトの開発が並行して進められている．本論文では，Eclipse の中心プロジェクトやダウンロード数の多いプロジェクトの中から，メッセージ数が 2,000 件以上のプロジェクトを対象とした．各プロジェクトを以下に記す．

- Java development tools プロジェクト[14]（以降，JDT）
- Plug-in Development Environment プロジェクト [15]（以降，PDE）
- Business Intelligence and Reporting Tools プロジェクト [9]（以降，BIRT）
- Eclipse Modeling Framework プロジェクト [11]（以降，EMF）

- C/C++ Development Tooling プロジェクト [10] (以降, CDT)
- Eclipse Web Tools Platform Project プロジェクト [16] (以降, WTP)
- Eclipse Platform プロジェクト [12] (以降, Eclipse Platform)
  - Platform
  - SWT
  - RCP

3．3　プロジェクトの分類

OSS 開発者の多くは OSS 開発プロジェクトと関係のない企業に勤務し，勤務終了後， OSS プロジェクトに従事する． こういったボランティア型の開発者が多く在住する地域では，夕方から夜にかけて情報交換が活発に行われていると考えられる．一方，勤務する企業から派遣され，業務の一環として OSS 開発プロジェクトに従事する OSS 開発者（業務型開発者）も存在する．業務型開発者が多く在住する地域では，勤務時間帯に情報交換を行う，つまり，業務時間中に（朝から夕方にかけて）情報交換が活発に行われがちであると考えられる．朝を中心に情報交換が行われている地域や，夜を中心に情報交換が行われている地域など，業務型の開発者の多い少ないによって情報交換を行う時間帯が異なることがある．

本論文では分析を行う観点として，OSS 開発プロジェクトをボランティア型 OSS 開発プロジェクト，業務混合型 OSS 開発プロジェクト，の２つのタイプに分類する．表１に各プロジェクトと OSS プロジェクトの分類を示す．

表-1 OSS プロジェクトの分類

プロジェクトの形態	プロジェクト名
ボランティア型	Python, PostgreSQL, Apache HTTP Server (Apache)
企業混合型	Eclipse (JDT, PDE 等)

3．4　基本統計量

分析対象データとしては，主に開発に関する話題（新規機能，バグ報告，メンテナンス，リリースなど）を取り扱う ML またはニュースグループのアーカイブを用いる．各プロジェクトに対する分析対象 ML/ニュースグループの一覧および，分析期間，全メッセージ数，ML/ニュースグループ参加者数，送信・返信の対の数，返信時間が 24 時間未満のデータ数に関するそれぞれの統計量を表 2 に示す．

本研究では，送信日時，送信場所，どのメッセージへ返信したか（返信情報）が含まれていないメッセージを対象外とした．また，表 2 における開発者数は，ML やニュースグループに一度でも登場した人物の総数とする．

ここで，各タイムゾーン（時差帯）におけるメッセージの分布を図 3 に示す．横軸は各タイムゾーンを表し，縦軸はメッセージ数を示す．図 3 に示すように，全プロジェクトに共通してメッセージ数の多い地域が 2 つ見ら

表-2 データセットの概要

分析対象プロジェクト			分析対象 ML/ ニュースグループ	分析期間	全メッセージ (件)	開発者数 (人)	送信・返信の対(件)	返信 24 時間未満のデータ数 (件)
Python			python-dev@python.org	1999/04～2009/04	89,301	2,150	56,707	51,830
PostgreSQL			pgsql-hackers@postgresql.org	1997/01～2009/09	184,492	4,731	130,239	114,514
Apache HTTP Server (Apache)			dev@httpd.apache.org	1995/03～2009/08	119,673	2,736	63,293	55,459
Eclipse	Java development tools (JDT)		eclipse.tools.jdt	2003/05～2009/09	24,691	4,717	7,894	5,951
	Plugin Development Environment (PDE)		eclipse.platform.pde	2008/05～2009/09	2,238	578	847	627
	Business Intelligence and Reporting Tools (BIRT)		eclipse.birt	2004/09～2009/09	35,108	3,886	9,058	7,798
	Eclipse Modeling Framework (EMF)		eclipse.tools.emf	2002/10～2009/09	44,216	3,337	22,654	20,259
	C/C++ Development Tooling (CDT)		eclipse.tools.cdt	2002/01～2009/09	19,088	4,290	4,727	3,296
	Eclipse Web Tools Platform Project (WTP)		eclipse.webtools	2003/07～2009/09	19,160	3,658	6,193	4,445
	Eclipse Platform	Platform	eclipse.platform	2003/05～2009/09	7,628	1,502	1,152	917
		SWT	eclipse.platform.swt	2003/05～2009/09	45,490	6,963	10,377	8,340
		RCP	eclipse.platform.rcp	2004/09～2009/09	37,583	4,796	13,614	10,243

れる．例えば，Python プロジェクトの場合，UTC-8 から-4（北アメリカ・南アメリカ地域，以降，米大陸地域）に 1 つと，UCT+0 から+3（ヨーロッパ地域・アフリカ地域，以降，欧・アフリカ地域）に 1 つ見られる．本論文では，この 2 地域を分析対象地域とし分析を進める．全プロジェクトの半数以上のメールがこれらの地域から送返信されたものであり，この 2 地域を分析することがプロジェクトの理解につながると考えたからである．分析対象地域の時差帯，メッセージ数，及び，開発者数をまとめたものを表 3 に示す．

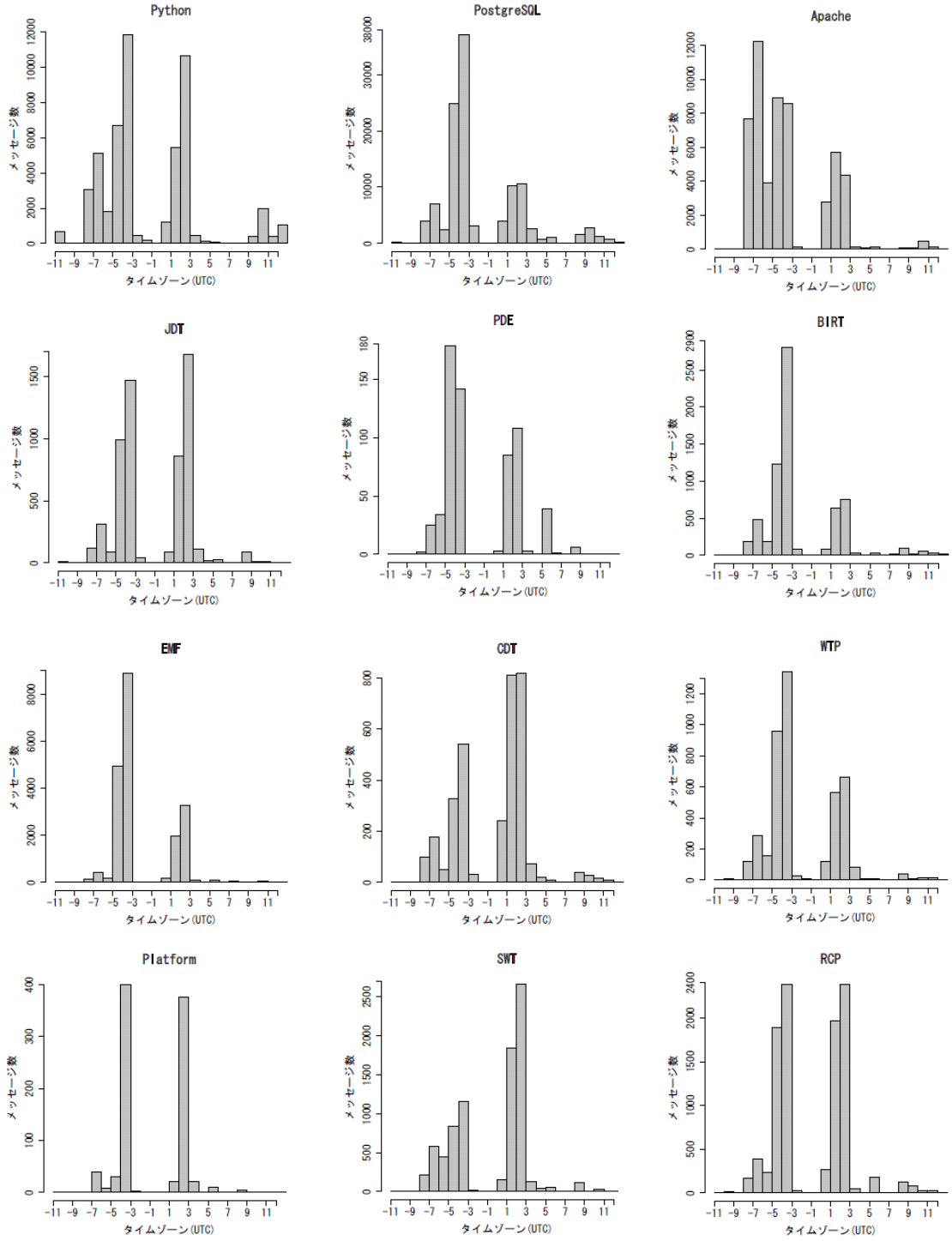


図-3 情報交換を行う時間帯

表-3 分析対象地域と統計量（米：米大陸地域，欧：欧・アフリカ地域）

分析対象プロジェクト			分析対象地域		メッセージ数（件）		開発者数（人）	
			米	欧	米	欧	米	欧
Python			UTC-8～UTC-4	UTC+0～UTC+3	28,574	17,748	679	575
PostgreSQL			UTC-5～UTC-4	UTC+0～UTC+3	61,875	27,318	537	795
Apache HTTP Server (Apache)			UTC-8～UTC-4	UTC+0～UTC+2	41,302	12,840	598	451
Eclipse	Java development tools (JDT)		UTC-5～UTC-4	UTC+1～UTC+2	2,468	2,547	226	320
	Plugin Development Environment (PDE)		UTC-5～UTC-4	UTC+1～UTC+2	321	193	28	68
	Business Intelligence and Reporting Tools (BIRT)		UTC-5～UTC-4	UTC+1～UTC+2	4,042	1,401	152	287
	Eclipse Modeling Framework (EMF)		UTC-5～UTC-4	UTC+1～UTC+2	13,807	5,219	165	510
	C/C++ Development Tooling (CDT)		UTC-5～UTC-4	UTC+1～UTC+2	872	1630	119	258
	Eclipse Web Tools Platform Project (WTP)		UTC-5～UTC-4	UTC+1～UTC+2	2,302	1,225	171	285
	Eclipse Platform	Platform	UTC-4	UTC+2	431	398	64	100
		SWT	UTC-8～UTC-4	UTC+1～UTC+2	3,248	4,517	410	613
		RCP	UTC-5～UTC-4	UTC+1～UTC+2	4,269	4,354	202	570

4．実験結果

4．1　仮説1：時差のある地域間の情報交換は返信時間が長くなる

［概要］ 情報交換を行う開発者間に時差がある場合と時差がない場合とでは，時差がない場合の方が開発に従事する時間帯を一致させやすく，リアルタイムに情報交換を行うことが容易となり，返信時間が短くなると考えられる．例えば，アメリカとヨーロッパでは少なくとも6時間以上の時差があり，これらの地域間では開発に従事する時間帯が異なる可能性が高く，返信時間が長くなると考えられる．

［分析］ 仮説1を検証するために，同一タイムゾーン内（時差のない地域間）の返信時間と異なるタイムゾーン間（時差のある地域間）の返信時間の分布を確認し，時差の有無による返信時間の差が統計的に有意かどうかを検定する．検定にはウィルコクソンの順位和検定を用いる．時差の有無別に返信時間の分布を確認することで，情報交換のタイムラグに時差が影響するかを把握する．時差のある地域間の方が返信時間が長くなれば，仮説1が支持されたとみなす．

ただし，MLなどの非同期な媒体を用いた情報交換は，開発者の都合の良いときに返信することができるため，メッセージが送信されてから何日も経過した後に返信される場合もある．返信時間が24時間未満のケースでは，地理的要因（時差）が影響して情報交換のタイムラグを発生させ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時間以上のケースでは，地理的要因よりも開発者個々人の都合（例えば，返信するために事前の調査が必要である）といった人的要因が大きく影響して情報交換のタイムラグを発生させているものが多く存在すると考えられる．そこで本論文では，時差による情報交換のタイムラグの把握が目的であるため，返信時間が24時間未満のものを対象とする．

［結果］ 時差のない地域間と時差のある地域間における返信時間の中央値と有意水準5%のウィルコクソンの順位和検定による p 値を表 4に示す．返信時間に有意差があった p 値を太字で表す．

ボランティア型OSS開発プロジェクトに着目すると，表4より，Apacheの米→米では返信時間の中央値が0.87，米→欧では1.75であることが確認でき，これらの返信時間に有意差が見られた．これはPython，及び，PostgreSQLでも同様の結果を確認できる．しかしながら一方で，欧から送信されたメールに対する返信（欧→欧，欧→米）に着目すると，Apacheにおいては返信時間に有意差が見られたものの，Python，PostgreSQLでは有意差が見られなかった．つまり，米大陸地域の開発者が送信した場合は，情報交換のタイムラグに時差が影響し，欧・アフリカ地域の開発者が送信した場合は，情報交換のタイムラグに時差が影響しないといえる．

次に，企業混合型OSS開発プロジェクトに着目すると，9プロジェクト，計18ケース（米大陸地域からのケースと，欧大陸地域からのケース）のうち，14ケースにおいて返信時間の差に有意差が見られた．特に，米大陸地域からの開発者が送信する場合，9プロジェクト中，PDEを除く全てのプロジェクトにおいて有意差が確認できた．例えば，BIRTプロジェクトの米から送信されたメールに対する返信（米→米，米→欧）に着目すると，8時間以上の差があることがわかる．これらの結果より，仮説 1は弱くではあるが支持されたといえる．

表-4 地域別の返信時間帯（米：米大陸地域，欧：欧・アフリカ地域）

分析対象プロジェクト			米→米（時）	米→欧（時）	p値*	欧→欧（時）	欧→米（時）	p値*
Python			1.24	2.07	**0.00**	1.59	1.80	0.57
PostgreSQL			0.76	1.88	**0.00**	1.19	1.42	0.95
Apache HTTP Server (Apache)			0.87	1.75	**0.00**	1.30	1.70	**0.00**
Eclipse	Java development tools (JDT)		1.91	6.88	**0.00**	2.75	3.39	0.91
	Plugin Development Environment (PDE)		2.10	3.86	0.17	1.69	5.12	**0.00**
	Business Intelligence and Reporting Tools (BIRT)		2.33	10.41	**0.00**	1.48	4.72	**0.00**
	Eclipse Modeling Framework (EMF)		0.93	1.72	**0.00**	0.97	1.00	0.69
	C/C++ Development Tooling (CDT)		2.16	8.56	**0.00**	2.42	4.70	**0.02**
	Eclipse Web Tools Platform Project (WTP)		2.06	6.13	**0.00**	3.04	4.27	0.15
	Eclipse Platform	Platform	1.88	4.28	**0.03**	2.12	3.56	**0.00**
		SWT	2.42	3.25	**0.00**	0.96	3.42	**0.00**
		RCP	2.90	3.54	**0.01**	1.52	4.38	**0.00**

*p値：有意水準5%によるウィルコクソンの順位和検定

4．2 仮説2：返信者の地域によって素早い返信を期待できる時間帯が異なる

［概要］ 本論文では，仮説1で開発者間に時差があると返信時間を長くさせ，情報交換にタイムラグが発生していることを示した．しかしがなら，開発者間に時差がある場合においても素早い返信が期待できる時間帯が存在し，その時間帯にメッセージを投稿すれば情報交換のタイムラグを解消できると考える．最も単純な例としては，両地域の開発者の作業時間が重なる時間帯である．例えば，UCT+0における14時は米大陸地域の現地時刻の9時，欧・アフリカ地域の17時であり，両地域の開発者が業務型の場合，作業に従事している可能性が高いと考えられる．一方の開発者がボランティア型の場合，別の時間帯が素早い返信が期待できる時間帯であると考えられる．

ヨーロッパや北アメリカに在住するOSS開発者を中心にアンケートを行ったFLOSS-USの調査[8]によると，ボランティア型のOSS開発者の75%は1週間あたり7時間，つまり，1日1時間程度開発に従事している．開発者はこの開発従事時間に情報交換を行っていると考えられるが，開発にはコーディングや不具合修正などの作業も含まれるため，情報交換を行うことができる時間はさらに短い．たとえ時差のない地域間の情報交換であっても，開発者個々人の限られた開発に従事する時間を1日の中で合わせることは難しいと考えられる．本分析の結果によって，時間の限られたOSS開発者がその時間をより効率的に利用できるようになると期待できる．

［分析］ 仮説2を検証するために，各送信時刻における返信時間の分布を示し，返信の早い時間帯を確認する．具体的には，0時，1時，2時といった各時刻に送信されたメッセージに対する返信時間の分布を確認する．特に，米→米や米→欧・アフリカなど地域別に返信時間を分析し，各地域においてリアルタイムに情報交換を行うことができる時間帯を確認する．地域別に返信時間を分析することで，返信者の地域によって素早い返信が期待できる時間帯が異なるかを把握できる．なお，本節においても返信時間が24時間未満のものを対象とする．返信者の地域によって素早い返信を期待できる時間帯が異なれば，仮説2が支持されたとみなす．

［結果］ 各時刻における返信時間の分布は図4-1，図4-2に示す．図4-1，図4-2の横軸は送信地域の現地時刻を表し，縦軸は各返信時間におけるメッセージ数の割合を表す．Eclipseはメッセージ数が少なかったため，横軸の時刻を3時間毎で表す．凡例の“0 - 0.5”は返信時間が0.5時間未満を表し，“8 - 24”は返信時間が8時間以上 24時間未満を表す．

図4-1より，PostgreSQLの米大陸地域の2者間で情報交換を行う場合（米→米），10時から19時頃にメッセージを送信すると約60%の確率で送信してから1時間未満に返信される．一方，米大陸地域から欧・アフリカ地域に対してメッセージを送信する場合（米→欧），米大陸地域の5時から12時頃にメッセージを送信すると約50%の確率で送信してから1時間未満に返信される．よって，米→米で素早い返信が期待できる時間帯は米大陸地域の10時から19時頃であり，米→欧では米大陸地域の5時から12時頃であり，それぞれ返信者の地域が異なると素早い返信が期待できる時間帯も異なることがわかる．同様に，PostgreSQLの欧→欧で素早い返信が期待できる時間帯は欧・アフリカ地域の10時から18時頃，欧→米では欧・アフリカ地域の16時から0時頃であることを確

認でき，返信者の地域によって素早い返信を期待できる時間帯が異なることがわかる．

図 4-1，図 4-2 の各地域の素早い返信が期待できる時間帯をまとめたものを表 5 に示す．各時間帯は送信地域の現地時刻を表す． PostgreSQL プロジェクトと同様，その他のプロジェクトにおいても素早い返信が期待できる時間帯は異なることがわかった．例えば，表 5，及び，図-4-1 の Python，Apache プロジェクトの米→欧に着目すると，米大陸地域の現地時刻 6 時から 15 時（欧・アフリカ地域の 12 時から 21 時）にメッセージを送信すると早く返信がもらえる．一方，表 5，図-4-2 の Eclipse Platform，SWT，RCP プロジェクトの米→欧に着目すると，米大陸地域の現地時刻 6 時から 12 時（欧・アフリカ地域の 12 時から 21 時），つまり，現地の業務時間内にメッセージを送ることがよいことがわかる．これらの結果から，返信者の地域が異なると返信の早い時間帯も異なり，仮説 2 は支持されたといえる．

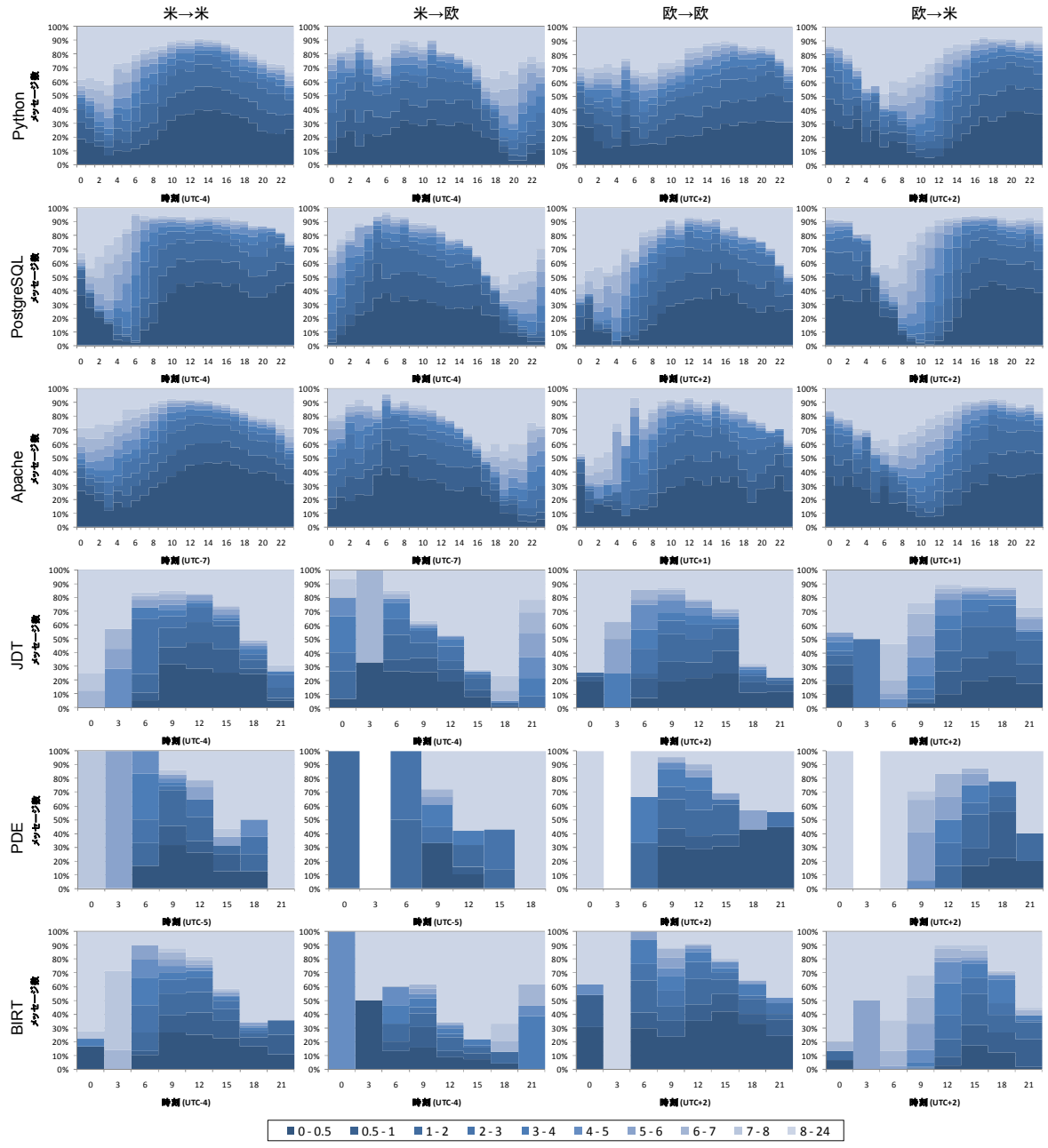


図-4-1 時刻別の返信時間の分布

5. 議論

5．1 考察

本節では2つに分類したOSSプロジェクト毎に考察を行う． 5.1.1項では ボランティア型OSS開発プロジェクトに関する考察を行い， 5.1.2項では企業混合型OSS開発プロジェクトに関する考察を行う．そして，5.1.3項で本知見に基づく開発プロセスの改善案を述べる．

5．1．1 ボランティア型OSS開発プロジェクト

仮説1より，時差のない地域間（米→米）と時差のある地域間（米→欧）の返信時間の差には有意差があったが，返信時間の差は1時間程度であった．米大陸地域と欧・アフリカ地域に6時間以上の時差があることに対して，返信時間の差は短い．この要因の一つとして，欧・アフリカ地域の開発者が米大陸地域の開発者の開発に従事する時間帯に合わせて情報交換を行っている可能性が高いと考えられる．そのため，米大陸地域の開発者と欧・

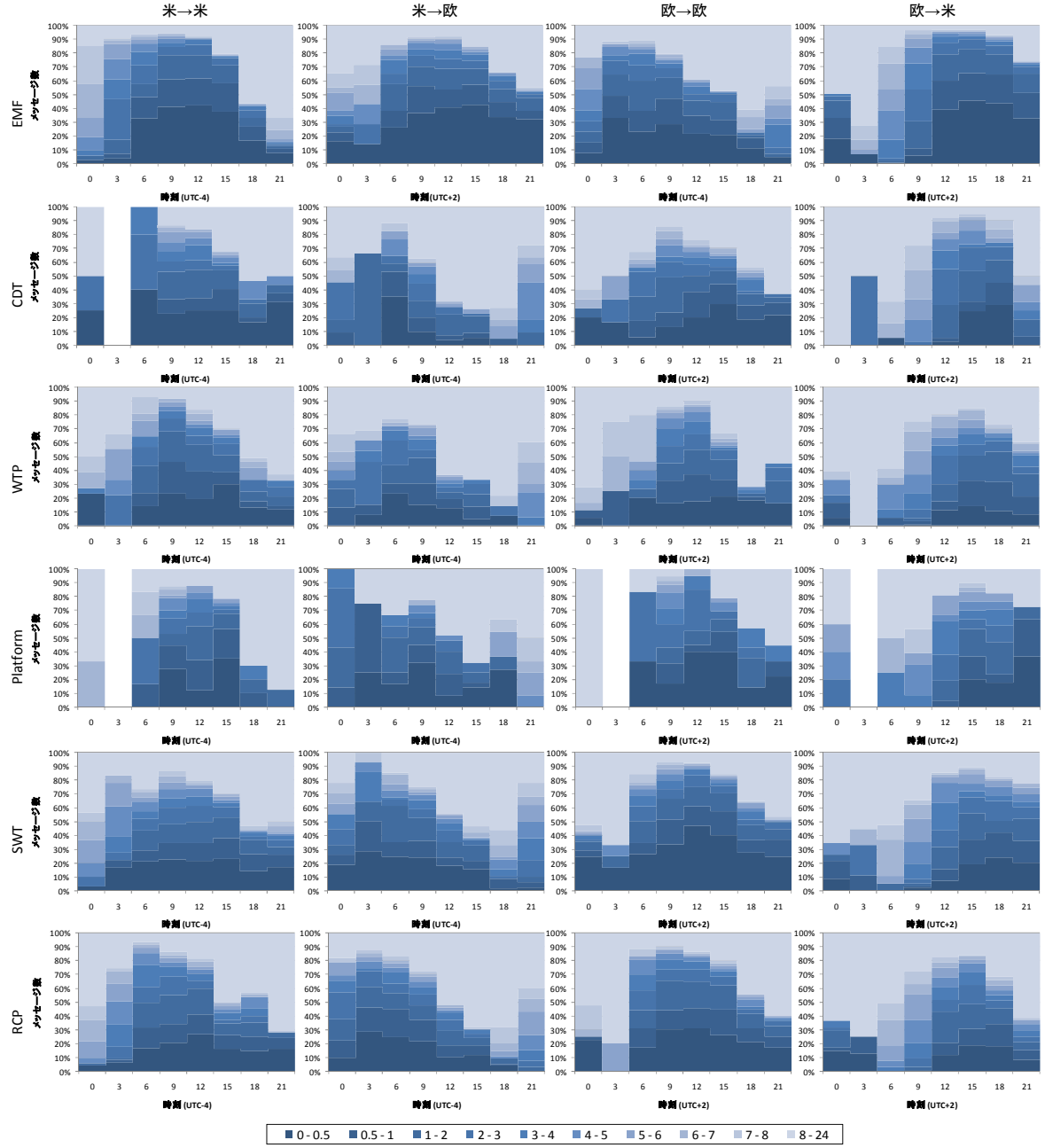


図-4-2 時刻別の返信時間の分布

表-5 素早い返信が期待できる時間帯（米：米大陸地域，欧：欧・アフリカ地域）

分析対象プロジェクト			返信の早い時間帯（現地時刻）			
			米→米	米→欧	欧→欧	欧→米
Python			9～20	7～16	14～23	16～2
PostgreSQL			10～19	5～12	10～18	16～0
Apache HTTP Server (Apache)			10～19	5～13	9～19	18～1
Eclipse	Java development tools (JDT)		9～21	3～15	9～18	15～21
	Plugin Development Environment (PDE)		6～12	6～12	9～18	15～21
	Business Intelligence and Reporting Tools (BIRT)		6～15	3～12	6～21	12～21
	Eclipse Modeling Framework (EMF)		6～18	3～12	6～18	12～0
	C/C++ Development Tooling (CDT)		6～18	6～12	9～18	15～21
	Eclipse Web Tools Platform Project (WTP)		6～18	6～12	9～18	12～21
	Eclipse Platform	Platform	9～18	3～12	6～18	15～0
		SWT	9～18	3～12	9～18	15～0
		RCP	6～15	3～12	6～18	12～21

表-6 各地域のトップドメイン（企業と混合のOSS開発プロジェクト）

分析対象プロジェクト	米大陸地域	欧･アフリカ地域
Java development tools (JDT)	企業(22.4%)	企業(23.6%)
Plugin Development Environment (PDE)	企業(28.0%)	その他(15.5%)
Business Intelligence and Reporting Tools (BIRT)	企業(67.8%)	その他(13.2%)
Eclipse Modeling Framework (EMF)	企業(73.1%)	その他(13.8%)
C/C++ Development Tooling (CDT)	企業(35.8%)	企業(15.3%)
Eclipse Web Tools Platform Project (WTP)	企業(31.2%)	その他(11.4%)

アフリカ地域の開発者の両者が情報交換を活発に行う時間帯が同じになり，タイムラグの少ない情報交換が可能になったと考えられる．これは，図5に示す米大陸地域，及び，欧・アフリカ地域の時刻別のメッセージ送信数からも見てとれる．また，表4より，PostgreSQLとApacheがPythonより返信時間の中央値が短いことがわかった．その要因として，PostgreSQLとApacheはデータベース管理システムとWebサーバといった信頼性の高いシステムが求められ，重大な不具合や脆弱性などの問題が起こると早急な対応が求められるといった，ソフトウェア自体の性質も情報交換のタイムラグに影響していると考えられる．

仮説2より，OSS開発プロジェクトでは返信地域の深夜から早朝にかけた時間帯にメッセージを送信すると返信が遅くなるため，この時間帯を避けてメッセージを送信すれば，情報交換のタイムラグが解消される．また，米大陸地域の開発者は10時から12時頃，欧・アフリカ地域の開発者は18時頃にメッセージを送信すると，どの地域からも素早い返信が期待できるため，情報交換の所要時間を短縮化することが可能と考える．

５．１．２　企業混合型OSS開発プロジェクト

図5から米大陸地域では情報交換が活発な時間帯がボランティア型OSS開発プロジェクトに比べて短く，UTC+0の23時から13時頃（米大陸地域の19時から9時頃）にかけてメッセージ数が極めて少ないことがわかった．その要因の一つとして，企業の開発者が多く存在していることが考えられる．Eclipseは IBMで開発が開始されたプロジェクトであることから，米大陸地域の開発者には企業の開発者が多く存在する可能性がある．

本研究では，企業の開発者がどのくらい存在しているかを調査するために，メールアドレスの “@”以降のドメイン情報[3]から企業の開発者であるかを調査した．その結果を表6に示す．表6は各地域において最も多かったドメインが企業であるかどうかをまとめたものであり，括弧の中の数字は各地域のメッセージ数のうち，トップドメインが占める割合を表す．なお，企業以外のドメインは “その他 ”とする．表6より，米大陸地域には企業の開発者が多く存在することを確認できた．一方，欧・アフリカ地域では米大陸地域のような特徴はみられ

[3] kamei@ait.kyushu-u.ac.jp の下線部分

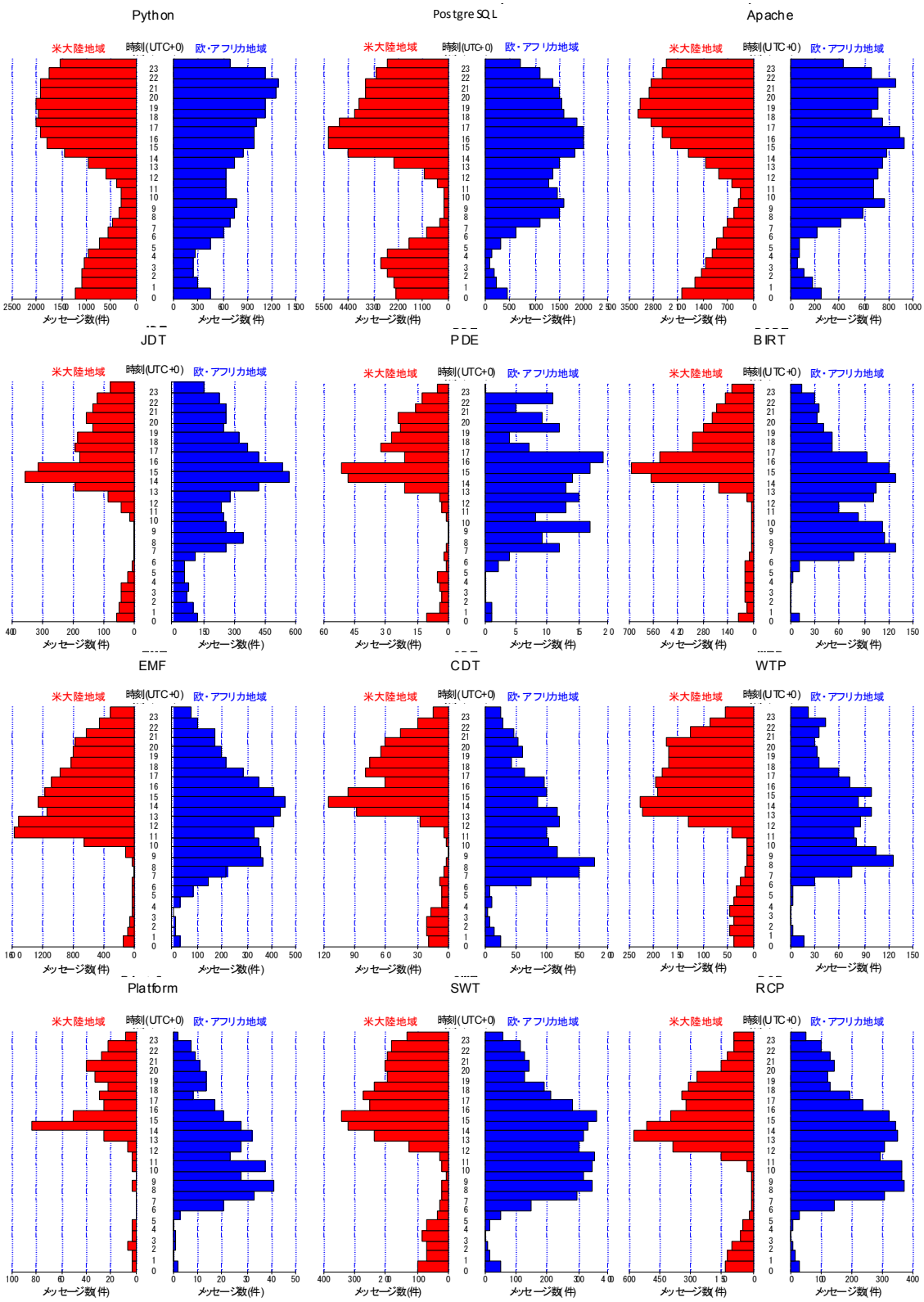


図-5 各地域におけるメッセージ送信時刻の分布

なかったため，企業の開発者は少ないと考えられる．表 6 からも，欧・アフリカ地域には企業の開発者が少ないことを確認できた．

仮説 1 の分析を通して，EMF を除いた返信時間の中央値が OSS 開発プロジェクトと比べて大きいことがわかっ

た（表4）．その要因としては，図5より米大陸地域の開発者が情報交換を活発に行っている時間帯が短く，欧・アフリカ地域の開発者との情報交換を行っている時間帯がずれているため，情報交換にタイムラグが発生する可能性が高くなったと考えられる． EMF では米大陸地域の開発者と欧・アフリカ地域の開発者の両者が互いに情報交換を行う時間帯を合わせている可能性が高く，タイムラグの少ない情報交換が可能となったと考えられる．

仮説2より，企業混合型OSS開発プロジェクトでは返信地域の夕方から早朝にかけた時間帯にメッセージを送信すると返信が遅くなるため，この時間帯を避けてメッセージを送信すれば，情報交換のタイムラグが解消される．また，米大陸地域の開発者は9時から12時頃，欧・アフリカ地域の開発者は15時から18時頃にメッセージを送信すると，どの地域からも素早い返信が期待できるため，情報交換の所要時間を短縮化することが可能と考える．

５．１．３　知見に基づく開発プロセスの改善

本論文では分析を通して，時差の影響を受けない時間帯は，(A) 米大陸地域のボランティア型OSS開発プロジェクトでは10時から12時頃であり，企業混合型OSS開発プロジェクトでは9時から12時頃，また，(B) 欧・アフリカ地域のボランティア型OSS開発プロジェクトでは18時頃であり，企業混合型OSS開発プロジェクトでは15時から18時頃である，という知見が得られた．この知見から，(A) 米大陸地域ではボランティア型・企業混合型ともに12時頃が時差の影響を受けない時間帯であるため，業務開発者は昼食時間帯を1時間程度遅らせ，ボランティア開発者は昼食時間を活用する，(B) 欧・アフリカ地域では18時頃が時差の影響を受けない時間帯であるため，業務開発者は業務終了時間を1時間程度遅らせ，ボランティア開発者は業務終了後，ソースコードの実装などではなく情報交換を行う，といった改善が考えられる．例えば，Bird らはMicrosoft の Windows Vista の開発を対象に分析を行った結果，各国の業務時間が重なるように変更するといった開発プロセスの改善を行うことで，分散開発の開発効率が高くなったと報告している[3]．Bird らの報告は企業を対象に実施された分析であるが，OSS開発プロジェクトの情報交換に対しても適用可能であると考える．

ただし，本改善案の制約として，すべての参加者に対して適用可能ではない点が挙げられる（参加者によっては業務時間をずらすことが不可能な場合もあるため)．そういった参加者が多いOSS開発者を多く有するOSS開発プロジェクトでは，作業割り当ての改善が考えられる．一般的に１つのソースコードには複数の開発者が作業担当者として割り当てられる．割り当てられた開発者は，機能追加やバグ修正などの議題をML上で議論する．もし担当者間で時差があると，本研究で示したように情報交換にタイムラグが発生する．そのため，時差が近い開発者を同一のソースコードに割り当てることで，情報交換の効率化を図ることが可能であると考える．

５．２　本論文の制約

OSS開発プロジェクトでは開発に従事する時間帯に束縛がないため，開発者個々人の都合の良い時間帯に開発を行うことが可能である．そのため，時差のない地域間の情報交換であっても，開発者個々人の開発に従事する時間帯が異なることで，開発者間の情報交換にはタイムラグが発生する可能性が高くなる．本論文では，開発者間の時差に着目して情報交換のタイムラグの実態を解明したが，開発者個々人の開発に従事する時間帯の違いまでは考慮していない．開発者個々人に着目して情報交換のタイムラグの実態を解明することで，情報交換所要時間の更なる短縮化が可能となり得る．

本論文では，情報交換のタイムラグが解消される方法として，素早い返信が期待できる時間帯にメッセージを送信することを述べた．しかしながら，送信するメッセージの内容によってすぐに返信できない場合も考えられる．このような場合，情報交換のタイムラグを解消できる時間帯にメッセージを送ったとしても，素早い返信が行われずタイムラグが発生し，情報交換のタイムラグを解消できない可能性がある．どういったメッセージの場合において時差を考慮しても情報交換のタイムラグが発生するか否かを機械的に判別することは今後の大きな課題である．

６．関連研究

OSS開発におけるプロジェクト遅延に関する問題として,OSSプロジェクト内のバグトラッキングシステムを用いた不具合修正プロセスに関する研究が今までに行われている [20][22][32]．例えば，Wang ら[32]は OSSの進化を計測するための適切な指標の必要性を挙げ，OSSの進化の特徴を考慮した指標を提案している．提案指標の一つとして，ソフトウェア中に存在している不具合の数や修正された不具合の数などが挙げられている．Wang ら[32]の研究では，提案指標のケーススタディとして Linux ディストリビューションである Ubuntu を対象とした分析が行われている．その結果，報告されている不具合のうち，修正されている不具合は20%程度に過ぎず，修

正されていない不具合の中には開発者の割り当てが行われていないものが多く存在していることが明らかにされている．これらの調査結果はプロジェクトにおける仕事量の多さの観点からプロジェクト遅延について分析されているが，本研究のようなコミュニケーションの観点からの分析はなされていない．

Mockus ら[24]と Herraiz ら[19]は，OSS 開発プロジェクトにおいて不具合を解決する平均時間を報告した．Mockus ら[24]は OSS 開発が成功する理由を調査するために，2 つの有名な大規模なプロジェクトである Apache プロジェクトと Mozilla プロジェクトの分析を行った．分析の結果，カーネルやプロトコルといった基盤系のモジュールや広い範囲で利用される機能を持つモジュールに存在する障害は不具合が解決されるまでの時間が短いことがわかった．また，優先度別の不具合が解決されるまでの時間は，P1 と P3 の 50%の不具合が 30 日以内，P2 の 50%の不具合が 80 日以内，P4 と P5 の 50%の不具合が 100 日以内に解決されることがわかった（P1 が最も優先度が大きく，P5 が最も小さい）．これらの研究は，OSS 開発において不具合修正プロセスの正確な知識に焦点が当てられているが，不具合修正プロセスと開発者間の情報交換のタイムラグが明らかにされていない．

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 [3][4][21][23][30]．例えば，Herbsleb ら[18]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，各拠点に点在していない開発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，対面の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にされている．一方で， Nguyen ら[26]は OSS 開発プロジェクトを対象とし，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は与えているが，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．

本論文では，メッセージの送信時刻と返信時間を細分化し，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，情報交換のタイムラグを解消する時間帯を示した点が異なる．

7．おわりに

本論文では OSS 開発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解明し，グローバル環境下におけるメーリングリストを用いた情報交換の所要時間を短縮化するための指針を得ることを目的として，OSS 開発プロジェクトにおける情報交換のタイムラグに関する仮説を実験的に検証した．12 件の大規模な OSS 開発プロジェクトを分析対象とし，それらをボランティア型 OSS 開発プロジェクト，企業混合型 OSS 開発プロジェクトの 2 つに分類して分析を行った結果，得られた主な知見は以下の通りである．

- ボランティア型 OSS 開発プロジェクトでは，返信地域が深夜から早朝となる時間帯にメッセージを送信すると，日中に比べて開発者間の情報交換にタイムラグが発生する可能性が高いことがわかった．また，時差の影響を受けずに情報交換を行うことができる時間帯は，米大陸地域では 10 時から 12 時頃，欧・アフリカ地域では 18 時頃であることがわかった．
- 企業混合型 OSS 開発プロジェクトでは，OSS 開発を仕事とする企業の開発者が多く存在するため，返信地域が夕方から早朝となる時間帯にメッセージを送信すると，日中に比べて開発者間の情報交換にタイムラグが発生する可能性が高いことがわかった．また，時差の影響を受けずに情報交換を行うことができる時間帯は，米大陸地域では 9 時から 12 時頃，欧・アフリカ地域では 15 時から 18 時頃であることがわかった．

OSS 開発プロジェクトにおいて，本論文で示唆したタイムラグが解消できる時間帯に情報交換を行うと，情報交換の所要時間を短縮化することが可能であると考える．本論文の貢献は，これら経験的に知られたグローバル環境下における情報交換に対する時差の影響に対して，OSS 開発プロジェクトデータから調査を行い，定量的な分析に基づいた客観的な知見を得たことにある．本分析の結果によって，1 日 1 時間程度しか開発に従事できない開発者がメールの送信時刻を工夫し，この限られた時間をより効率的に利用できるようになると期待できる．

今後の課題としては，得られた知見の妥当性向上を目的として，上記の知見によりプログラムの実装やバグの修正がどの程度スムーズに行うことができるかを実験的に検証することがあげられる．また，メーリングリスト以外の情報交換手段（IRC 等のインスタントメッセージ）について分析することも今後の課題の１つである．

謝辞

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（基盤 B：課題番号 23300009，若手 B：課題番号 22700033）による助成

を受けた。

［参考文献］

[1] Ban Al-Ani and H. Keith Edwards. A comparative empirical study of communication in distributed and collocated development teams. In Proceedings of the 3rd International Conferenceon Global Software Engineering (ICGSE'08), pp.35-44, 2008.
[2] Apache Software Foundation. Apache HTTP Server Project. 'http://httpd.apache.org', accessed 2011-05-03.
[3] Christian Bird, Nachiappan Nagappan, Premkumar Devanbu, Harald Gall, and rendan Murphy. Does distributed development affect software quality? An empirical case study of windows vista. In Proceedings of the 31st International Conference on Software Engineering (ICSE'09), pp.518-528, 2009.
[4] Erran Carmel. Global Software Teams: Colloborating Across Borders and Time Zones. Prentice Hall, Upper Saddle River, USA, Jan. 1999.
[5] Marcelo Cataldo and Sangeeth Nambiar. On the relationship between process maturity and geographic distribution: an empirical analysis of their impact on software quality. In Proceedings of the the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'09), pp.101-110, 2009.
[6] Jorge A. Colazo. Following the sun: Exploring productivity in temporally dispersed. In Proceedings of the 14th Americas Conference on Information Systems (AMCIS'08), 2008.
[7] Kevin Crowston and Barbara Scozzi. Open source software projects as virtual organizations: Competency rallying for software development, IEE Proceedings Software, Vol.149, No.1. pp.3-17, 2002.
[8] Paul A. David, Andrew Waterman, and Seema Arora. FLOSS-US: The Free/Libre/Open Source Software Survey for 2003. available from・http://www.stanford.edu/group/floss-us/', accessed 2011-05-03.
[9] Eclipse Foundation. Business Intelligence and Reporting Tools (BIRT). ・http://www.eclipse.org/birt/phoenix/', accessed 2011-05-03.
[10] Eclipse Foundation. Eclipse C/C++ Development Tooling (CDT).・http://www.eclipse.org/cdt/', accessed 2011-05-03.
[11] Eclipse Foundation. Eclipse Modeling Framework (EMF). 'http://www.eclipse.org/modeling/emf/・, accessed 2011-05-03.
[12] Eclipse Foundation. Eclipse Platform.・http://www.eclipse.org/platform/', accessed 2011-05-03.
[13] Eclipse Foundation. Eclipse Project. Eclipse.org home ・http://www.eclipse.org/', accessed 2011-05-03.
[14] Eclipse Foundation. Java development tools (JDT).・http://www.eclipse.org/jdt/', accessed 2011-05-03.
[15] Eclipse Foundation. Plugin Development Environment (PDE). 'http://www.eclipse.org/pde/', accessed 2011-05-03.
[16] Eclipse Foundation. Web Tools Platform (WTP) Project.・http://www.eclipse.org/webtools/', accessed 2011-05-03.
[17] Karl Fogel. Producing Open Source Software: How To Run a Successful Free Software Project. O'Reilly Media, California, USA , 2005.
[18] James D. Herbsleb, Audris Mockus, Thomas A. Finholt, and Rebecca E. Grinter. An empirical study of global software development: Distance and speed. In Proceedings of the 23rd International Conference on Software Engineering (ICSE'01), pp.81-90, 2001.
[19] Israel Herraiz, Daniel M. German, Jesus M. Gonzalez-Barahona, and Gregorio Robles. Towards a simplification of the bug report form in eclipse. In Proceedings of the 5th International Working Conference on Mining Software Repositories (MSR'08), pp.145-148, 2008.
[20] Akinori Ihara, Masao Ohira, and Ken ichi Matsumoto. An analysis method for improving a bug modification process in open source software development. In Proceedings of the joint International Workshop on Principles of Software Evolution (IWPSE) and the ERCIM Workshop on Software Evolution (EVOL) (IWPSE-EVOL'09), pp.135-144, 2009.
[21] Dale Walter Karolak. Global Software Development: Managing Virtual Teams and Environments. Wiley-IEEE Computer Society Press, Los Alamitos, USA, 1999.

[22] Sunghun Kim and Jr. E. James Whitehead. How long did it take to fix bugs? In Proceedings of the 5th International Working Conference on Mining Software Repositories (MSR'06), pp.173-174, 2006.
[23] Allen E. Milewski, Marilyn Tremaine, Richard Egan, Suling Zhang, Felix Kobler, and Patrick O'Sullivan. Guidelines for effective bridging in global software engineering. In Proceedings of the 3rd International Conferenceon Global Software Engineering (ICGSE'08), pp.23-32, 2008.
[24] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and mozilla. ACM Transactions on Software Engineering and Methodology (TOSEM), Vol.11, No.3, pp.309-346, 2002.
[25] Sandro Morasca, Davide Taibi, and Davide Tosi. Towards certifying the testing process of open-source software: New challenges or old methodologies? In Proceedings of the 31st ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS'09), pp.25-30, 2009.
[26] Thanh Nguyen, Timo Wolf, and Daniela Damian. Global software development and delay: Does distance still matter? In Proceedings of the 3rd International Conferenceon Global Software Engineering (ICGSE'08), pp.45-54, 2008.
[27] PostgreSQL Global Development Group. PostgreSQL. 'http://www.postgresql.org/', accessed 2011-05-03.
[28] Python Software Foundation. Python Programming Language.Official Website 'http://www.python.org/', accessed 2011-05-03.
[29] Gregorio Robles and Jesus M. Gonzalez-Barahona. Geographic location of developers at sourceforge. In Proceedings of the 5th International Working Conference on Mining Software Repositories (MSR'06), pp.144-150, 2006.
[30] Raghvinder Sangwan, Matthew Bass, Neel Mullick, Daniel J. Paulish J., and Juergen Kazmeier. Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series). Auerbach Publications, Boston, USA, Sep. 2006.
[31] Diomidis Spinellis. Global software development in the FreeBSD project. In Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner (GSD'06). pp.73-79, 2006.
[32] Yi Wang, Defeng Guo, and Huihui Shi. Measuring the evolution of open source software systems with their communities. SIGSOFT Software Engineering Notes, Vol.32, No.6, p.7, 2007.
[33] 独立行政法人情報処理推進機構. オープンソース情報データベース OSSiPedia. 入手先・http://ossipedia.ipa.go.jp/', 参照 2011-05-03.

(2011年10月13日受理)