OSS開発コミュニティの進化の理解を目的としたコミュニケーション分析：Politeness分析適用の試み

宮崎智己[1] 山谷陽亮[2] 東裕之輔[2] 大平雅雄[1]

概要：近年，OSS をソフトウェア製品の一部として再利用することが一般的になりつつある一方で，開発元の OSS 開発コミュニティが継続するのかどうかが不透明なため OSS の利用を躊躇する企業が少なくない．OSS 開発では，ボランティアの開発者が中心となり，メールや掲示板を用いた非対面環境下でのコミュニケーションが主たる情報伝達手段となる．したがって，OSS 開発コミュニティが持続的に進化する上で開発者間のコミュニケーションの質（円滑に意思疎通が図れるようにする工夫など）が極めて重要になるものと思われる．これまで，OSS 開発コミュニティの進化そのものについては数多くの研究がおこなわれてきたが，開発者間のコミュニケーションの質的側面，特に，コミュニケーションの様式（本研究では，対話者を配慮する表現形式）と OSS 開発コミュニティの持続的進化に焦点を当てた分析はほとんどない．本研究では，人間関係を維持するために対話者への配慮（本研究における Politeness）を込めた言語的表現を定量化する Politeness 分析を適用したケーススタディをおこなった．本稿では，20 年以上の歴史を有する Apache HTTPD Sever プロジェクトのトップ開発者 10 名の総計 123,538 件分のコミュニケーションデータ（開発者メーリングリスト上でのメールデータ）を対象に Politeness 分析を適用した結果について考察するとともに，今後の分析指針について議論する．

TOMOKI MIYAZAKI[1] YOSUKE YAMATANI[2] YUNOSUKE HIGASHII[2] MASAO OHIRA[1]

1. はじめに

Android をはじめとするオープンソースソフトウェア（以降，OSS と呼ぶ）は，様々な用途で幅広く普及しており，我々の日常生活にとって必要不可欠なものとなっている．また，OSS は近年，個人ユーザによるエンドユースのみならず，企業におけるソフトウェア開発にも積極的に利用されている．ソフトウェア開発企業における OSS 利用の主たる目的は，OSS をソフトウェア製品の一部として再利用することで開発の生産性を向上させることである．しかしながら，再利用する OSS がいつまで開発保守されるのか分からなかったり，不具合が発見された場合に確実に修正されるかどうかが不透明なため [1]，OSS の利用を躊躇するソフトウェア開発企業も少なくない．

ソフトウェア開発企業においての OSS 利用に関する様々な懸念を払拭するために，OSS 開発コミュニティの進化に関する研究がこれまで盛んにおこなわれてきた．例えば，Bird ら [2] は，プロジェクトに参加したボランティア開発者がプロジェクトを脱退するまでの平均活動期間は約 1.5 年であることを明らかにしており，より長くプロジェクトに貢献してもらうためには早めに（すなわち，1.5 年以内に）より重要な役割を与えることが必要であると指摘している．また，Matsumoto ら [3] は，Apache HTTPD Server プロジェクトおよび Netscape プロジェクトのメーリングリストから取得したコミュニケーションデータを対象にソーシャルネットワーク分析をおこない，コミュニティを維持するうえで重要な役割を担う開発者（コーディネータ）が存在することを明らかにしている．

特に近年では，OSS 開発コミュニティにおける礼儀や感情の表出といったコミュニケーションの質的側面に着目した分析がおこなわれている．OSS 開発コミュティの多くは，ボランティアの開発者が中心となり，主にメールや掲示板を用いた非対面環境下でのコミュニケーションを通じて開発がおこなわれるため，OSS 開発の持続的発展に

[1] 和歌山大学システム工学部
[2] 和歌山大学大学院システム工学研究科

は開発者間のコミュニケーションの質（円滑に意思疎通が図れるようにする工夫など）が重要であると考えられているためである．例えば，Tourani ら [4] は，Apache Ant プロジェクトのメーリングリストから取得したコミュニケーションデータを対象に Sentimental 分析を適用し，開発者の感情（悲しみまたは喜び）を特定するための分析をおこなっている．分析の結果，Ant プロジェクトでは，修正された不具合数と開発者の感情に相関関係があることを明らかにしている．

しかしながら，コミュニケーションの質的側面に着目した先行研究の多くは，コミュニケーションの質的側面と開発される OSS の品質との関係を明らかにすることを目的としており，OSS 開発コミュニティの形成からコミュニティの持続・発展，すなわち，OSS 開発コミュニティの持続的進化との関係については未だ十分な分析はおこなわれていない．そこで本研究は，開発者間のコミュニケーションの質的側面の中でも特に，コミュニケーションの様式に焦点を当て，OSS 開発コミュニティの持続的進化に寄与する要因を明らかにすることを目的とする．ここで，コミュニケーションの様式とは，OSS 開発コミュニティ内の人間関係を維持するうえで重要となる，対話者への配慮を表出するための言語的表現 (Politeness [5]) の方法を指す．長期に渡り OSS 開発コミュニティを維持していくためには，開発者が自己中心的な言動を控え，互いの意見を尊重し円滑に意思疎通が図れるようなコミュニケーション（言語的表現）上の工夫が存在すると考えるのが自然である．

本研究では，20 年以上の歴史を有する Apache HTTPD Server プロジェクトにおける開発者メーリングリストから取得したトップ開発者 10 名分（総計 123,538 件分）のコミュニケーションデータ対して，人間関係を維持するために表出される対話者への言語的配慮 (Politeness) を定量化するための Politeness 分析 [6] を適用するケーススタディをおこない，以下のリサーチクエスチョンに取り組む．

- RQ1: トップ開発者間で Politeness に違いはあるか？
- RQ2: トップ開発者と一般開発者の Politeness に違いはあるか？
- RQ3: 自ら発信する場合と他者へ返信する場合で Politeness に違いはあるか？
- RQ4: Politeness は時期を問わず一定しているものか？あるいは，時期により変動するものか？

以降ではまず，2 章において本研究に関する先行研究について述べ本研究の立場を明らかにする．次に，3 章では，自然言語処理の分野における Politeness の位置付けと，Politeness を定量化するための分析方法について説明する．4 章では，ケーススタディの目的と方法について述べ，5 章にてケーススタディの結果を示し考察する．最後に 6 章において本研究の今後の研究方針について議論し本論文をまとめる．

2. 関連研究

本章では，OSS 開発コミュニティの進化および，OSS 開発におけるコミュニケーション分析の観点から，本研究の関連研究をまとめ，本研究の位置づけについて述べる．

2.1 OSS 開発コミュニティの持続的進化に関する研究

OSS を利用したソフトウェア開発が増加しているものの，OSS は主にボランティア開発者によって開発および保守がおこなわれているため，OSS 開発コミュニティがいつまで存続するかわからないなどといった理由から OSS の利用を躊躇する企業も少なくない [1]．ソフトウェア開発企業における OSS 利用に対する様々な懸念を払拭するために，OSS 開発コミュニティの進化に関する研究が盛んにおこなわれている．OSS 開発コミュニティの持続的進化とは，OSS 開発コミュニティの形成から維持・発展の過程における変化を指す．OSS 開発コミュニティの持続的進化の分析では，OSS 開発コミュニティに参加するボランティア開発者数の増減や，OSS コミュニティの組織構造や活動量の変化などが調査対象となることが多い．

例えば，Bird ら [2] は，プロジェクトに参加したボランティア開発者がプロジェクトを脱退するまでの平均活動期間は約 1.5 年であることを明らかにしており，より長くプロジェクトに貢献してもらうためには早めに（すなわち，1.5 年以内に）より重要な役割を与えることが必要であると指摘している．また，Matsumoto ら [3] は，Apache HTTPD Server プロジェクトおよび Netscape プロジェクトのメーリングリストから取得したコミュニケーションデータを対象にソーシャルネットワーク分析をおこない，コミュニティを維持するうえで重要な役割を担う開発者（コーティネータ）が存在することを明らかにしている．

コミュニケーションの質的側面から OSS 開発コミュニティの持続的進化のための必須条件を明らかにしようとする点では，本研究と先行研究とではアプローチが大きく異なるが，主たる関心事である OSS 開発コミュニティの持続・発展メカニズムの解明という点では，先行研究とも目的は共通する部分が多い．

2.2 OSS 開発におけるコミュニケーション分析に関する研究

ソフトウェア開発者は，コミュニティでの活動時間の内，約 70%以上の時間を他の開発者とのコミュニケーションに費やしていることが知られている [7]．したがって，OSS 開発コミュニティが持続的に進化していくためにも，開発者間でのコミュニケーションが極めて重要であるといえる．そのため，OSS 開発における開発者間のコミュニケーションを分析する研究が盛んにおこなわれている．

特に近年では，コミュニケーションの質的側面を分析す

表 1: Politeness 分析 [6] を適用した結果の例

No.	例文	Politeness 値
1	OK, thanks for your suggestion. I will change that. Regards XXX（XXX は，ファーストネーム）.	0.997
2	Oh dear. My question is already slipping down the list into obscurity... I guess I must be the only person using eclipse 3.4 who would like to be able to do this.	0.507
3	So, what was the solution for this problem then??	0.122

る研究が盛んにおこなわれている．例えば，Tourani ら [4] は，Apache Ant プロジェクトのメーリングリストから取得したコミュニケーションデータを対象に Sentimental 分析を適用し，開発者の感情（悲しみまたは喜び）を特定するための分析をおこなっている．分析の結果，Ant プロジェクトでは，修正された不具合の数と開発者の感情に相関関係があることを明らかにしている．また，Ortu ら [8], [9] は，OSS 開発で利用されている不具合管理システムのコメントデータに対し，Sentimental 分析，Politeness 分析，および，Emotional 分析を横断的に適用し，開発者の感情と不具合の修正時間に関係があることを明らかにしている．

これら先行研究では，コミュニケーションの質的側面と開発される OSS の品質との関係を明らかにすることを目的としており，OSS 開発コミュニティの持続的進化との関係については未だ十分には分析がおこなわれていない．

そこで本研究は，開発者間のコミュニケーションの質的側面の中でも特に，コミュニケーションの様式に焦点を当て，OSS 開発コミュニティの持続的進化に寄与する要因を明らかにすることを目的とする．ここで，コミュニケーションの様式とは，OSS 開発コミュニティ内の人間関係を維持するうえで重要となる，対話者への配慮を表出するための言語的表現 (Politeness [5]) の方法を指す．長期に渡り OSS 開発コミュニティを維持していくためには，開発者が自己中心的な言動を控え，互いの意見を尊重し円滑に意思疎通が図れるようなコミュニケーション（言語的表現）上の工夫が存在すると考えるのが自然である．本研究では，OSS プロジェクトから取得したコミュニケーションデータ対して，人間関係を維持するために表出される対話者への言語的配慮 (Politeness) を定量化するための Politeness 分析 [6] を適用したケーススタディ実施する．

3. Politeness 分析

本章では，本研究のケーススタディで利用する Politeness 分析について述べる．

3.1 Politeness

他者と会話する際，会話を円滑にするために，人は，聞き手をリラックスさせたり気持ちよくさせたりしている．円滑な会話により，話し手と聞き手が不快感を感じることが少なくなり，良好な人間関係が維持される．会話を円滑にするための原理の 1 つとして Politeness[5] がある．Politeness とは，日本語の「丁寧さ」や「礼儀正しさ」とはニュアンスが異なる概念であり，円滑な人間関係を確立・維持するための相手への配慮を指す．例えば，“What is the cause of this error?” のような質問を突然聞き手に投げかけると，回答を直接的に求めることになると同時に回答しなければならない理由も理解できないため，聞き手に心理的負荷をかけてしまう．そのため，話し手は，聞き手を配慮する表現を取り入れるのが一般的である．例えば，“Could you please tell me about the cause of this error?” といった表現を取り入れた場合，聞き手は回答することでポジティブな効果（感謝されるなど）があることや，知らないのであれば質問に回答できなくても問題ないことを暗黙的に理解することができ，前述の直接的な質問形式よりも心理的負荷が低くなる．

相手を配慮するための具体的な表現を Politeness Strategy[5] という．Politeness Strategy は，Brown[5] らによって多数定義されている．以下にそのいくつかを説明する．

Please “Please” を用いた依頼表現を指す．Please は，動詞の前ではなく文中に突然出現する方が，Politeness を表現する効果が高いという調査結果がある [6].

Apologizing “Sorry” などの謝罪表現を指す．Apologizing は，相手に謝罪あるいは同情することで，相手にかかる心理的負荷を自分にそらすことができる．

Hedges “Would you by any chance...” や，“I suggest...” など，互いの面子を保つため相手が話題を否定しやすくする表現を指す．

Greeting “Hey”,“Hi” などのフランクな挨拶を指す．良好な人間関係がすでに存在する場合は，対人距離の近さや親近感を明示的に表すことができ，人間関係の維持に寄与する．

人は意識的・無意識的に，これらの Politeness Strategy を使い分けながら会話をおこなっているが，各 Politeness Strategy は基本的に，使うことのできる場面が限られているため，使う相手や場面を間違えると無礼な表現になる．そのため，文章中で Politeness を表現したい場合には，各 Politeness Strategy を効果的に使い分ける必要がある．

3.2 Stanford Politeness

本研究では，Stanford Politeness[6][*1] を用いて，Apache HTTPD Server プロジェクトの開発者メーリングリストか

[*1] https://github.com/sudhof/politeness

図 1: Stanford Politeness における Politeness の算出方法

ら取得するコミュニケーションデータに対して Politeness 分析を適用する．Stanford Politeness とは，言語表現に込められた Politeness を定量化するツールである．

表 1 に，OSS 開発者のメーリングリスト上の議論に対して Stanford Politeness を適用した例を示す．表 1 の Politeness 値は，0 から 1 の間で表され，その値が大きいほど相手を配慮する表現が取り入れられている．例えば，1 番目の文章の Politeness 値は 0.997 と非常に高いが，これは文中に含まれる “suggestion” が，Politeness Strategy の Hedges に該当しているためである．2 番目の文章は，単語数が多く情報量の多い文章ではあるが，Politeness Strategy には該当しておらず，0.507 と中間の値が算出されている．3 番目の文章は，Politeness Strategy に該当しないだけではなく，“So, what...” の部分が相手を不愉快にさせる可能性の高い表現であるため，0.122 と非常に低い値となっている．

3.3 Stanford Politeness のアルゴリズム

Stanford Politeness は，大きく分けて 2 つの処理からなる．図 1 に処理方法の概要を示す．

3.3.1 Politeness Strategy の検出

まず，テキストに含まれる Politeness の言語的表現そのものよりも，Politeness 表現の利用される場面，すなわち，Politeness Strategy が Politeness を数値化するうえで重要なため，分析対象（入力）となるテキストから Politeness Strategy を検出する．後述の SVM による機械学習では，入力テキストとともに検出した Politeness Strategy を学習させる．Stanford Politeness では，grep を用いて 20 個の Politeness Strategy を検出している．

3.3.2 SVM による Politeness 値予測

Stanford Politeness は，機械学習アルゴリズムの一つである Support Vector Machine（SVM）[10] を用いて Politeness 値の算出をおこなう．学習データには，Stanford Politeness Corpus[6] が用いられている．Stanford Politeness Corpus は，Wikipedia ユーザのトークページ [*2] から抽出した 4,353 の質問と，Stack Exchange[*3] から抽出した 6,604 の質問と，各質問の Politeness を人手によって評価したデータの集合である．

入力テキストに含まれる Politeness Strategy を SVM に入力することで，学習データを用いて構築した判別モデルからそのテキストの Politeness 値が返される．ただし，テキストを SVM で扱えるようにするにはテキストを数値化する必要がある．Stanford Politeness では，テキストを Bag of Words ベクトルに変換して数値化している．

4. ケーススタディ

本章では，Apache HTTPD Server プロジェクトを対象に実施したケーススタディの方法について述べる．

4.1 対象プロジェクト

ケーススタディでは，Apache HTTPD Server プロジェクトにおける開発者メーリングリストから取得したトップ開発者 10 名分（総計 123,538 件分）のコミュニケーションデータに対して Politeness 分析を適用する．本ケーススタディにおいて Apache HTTP Server を選択した主な理由は以下の通りである．

- ボランティア開発者が中心であり，コミュニケーションの様式に対して役職や階級などの社会的地位による影響が少ないこと（企業のような上司部下の関係や厳格な指揮系統が存在しないこと）[*4]
- 開発者メーリングリストを通じてやり取りされたメールデータがすべてアーカイブされており，かつ，容易に取得可能であること
- 長期にわたり存続しているプロジェクトであり開発コミュニティの持続的進化を分析する対象として適していること
- 同じプロジェクトを対象とした先行研究が数多くあり，ケーススタディにより得られた知見の妥当性を確保しやすいこと

なお，本ケーススタディにおける分析対象期間は，Apache HTTPD Server プロジェクトが設立された 1995 年 3 月から 2016 年 3 月時点の約 21 年間とする．対象開発者の選出方法については，4.2.2 項で述べる．

4.2 Politeness 分析の手順

以下では，本ケーススタディにおいて実施する Politeness 分析の手順について述べる．

4.2.1 コミュニケーションデータの取得

まず，分析対象となるコミュニケーションデータを Apache HTTPD Server プロジェクトの開発者向けのメーリングリスト [*5]（図 2）から mbox ファイル形式で取得する．mbox ファイルは，メーリングリストを用いて送信さ

[*2] http://en.wikipedia.org/wiki/Wikipedia:User_pages
[*3] http://stackexchange.com/about

[*4] Apahce HTTPD Server プロジェクトの場合，ASF (Apache Software Foundation) により一部の開発者はフルタイム雇用されているが，他のボランティア開発者へ作業を指示する権限等は存在しない．
[*5] Apache HTTP Server Development Main Discussion List: `https://httpd.apache.org/lists.html#http-dev`

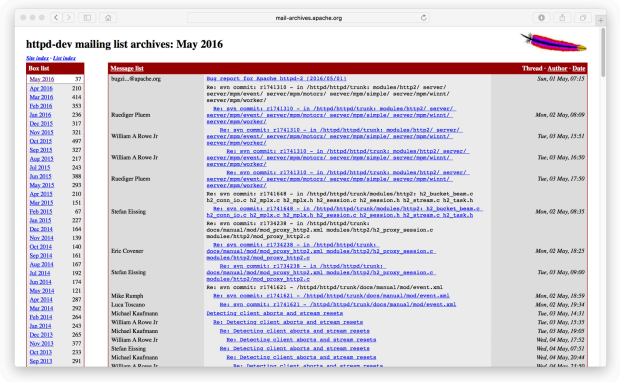

図 2: Apache HTTPD Server のメーリングリスト

れた全てのメールをアーカイブしたもので，分析に必要となるデータ（メッセージ ID，プリメッセージ ID，送信者名，送信者のメールアドレス，件名，送信日時，本文など）が 1 ヶ月毎に保存されている．

次に，取得した mbox ファイルからメールの情報を一件ずつ抽出しデータベースに格納する．ただし，メール本文には，署名や引用文などが含まれており，Politeness 値の算出に悪影響が出る可能性が高いため，あらかじめ不必要なテキストは取り除いている．なお，本ケーススタディでは時間の都合上，メール本文中に含まれるソースコード片については除去していない．また，メール返信や転送メールに含まれる「On 〜 wrote」や「Fowarded Message」などの Politeness 分析には不必要なテキストも完全には除去できていない場合がある．

4.2.2 対象開発者の選出

次に，メールデータを集計し分析対象とする開発者を選出する．本ケーススタディでは，分析対象期間中に送信したメールの多い順に上位 10 名の開発者（トップ開発者）を選出する．これらの開発者は，コミュニティ内で最も活発に議論をおこなった開発者であり，他の開発者とも積極的にコミュニケーションをおこなっていることから，コニュニティの持続的進化に重要な役割を果たしている開発者と考えることができる．なお，選出した開発者の中にはメールアドレスを複数所持している者もいるため，著者らが目視によってメールアドレスを確認しデータを統合している．

4.2.3 Politeness 分析

最後に，選出した 10 名の開発者が送信したすべてのメール（開発者が送信したメール），及び，送信したメールに対して返信された全てのメールに対して，開発者別に Politeness 分析を適用する．本ケーススタディでは，3.2 節で述べた Stanford Politeness と，構文解析ツール Stanford Parser[*6] を用いて，メール 1 件ずつに対して Politeness 分析を適用し Politeness 値を算出する．

なお，解析対象のメールは，メール送信，メール発信，メール返信，メール受信の 4 種類に分類したうえで Politeness 値を算出する．メール送信とは，開発者がメーリングリストを通じて送信したメール（= メール発信+メール返信）を指す．メール発信とは，開発者が返信の一部としてではなくスレッドの先頭として議論を開始するために送信したメールを指す．メール返信とは，他の開発者への返信として送信したメールを指す．メール受信とは，開発者のメールに対して他者が返信したメールを指す．返信されたメールの Politeness 値を計測することで特定の開発者とその他の開発者とのコミュニケーションの質が見て取れる可能性がある．

以上の手順に従い算出された Politeness 値を開発者毎に集計し，基本統計量を求めるとともに，Politeness 値の推移を時系列に見ることができるようデータを出力する．

[*6] http://nlp.stanford.edu/software/lex-parser.shtml

5. 結果と考察

本章では，ケーススタディの結果をリサーチクエスチョン毎に示すとともに，その結果を考察する．

5.1 RQ1: トップ開発者間で Politeness に違いはあるか？

分析意図: トップ開発者はメーリングリスト上で活発に議論をしている開発者であり，コミュニティの持続的進化に重要な役割を担っている可能性が高い．そのため，トップ開発者はメールでのコミュニケーションの際に，他者への配慮を十分におこなっているものと予想でき，結果的に，総じて Politeness の高いコミュニケーションをおこなっているものと考えた．そこで，トップ開発者間の Politeness 値には大きな違いがないかどうかをまず確かめることとした．

分析方法: 開発者毎に算出した Politeness 値の基本統計量を比較する．

結果: 表 2 に，各開発者の活動期間とメール送受信数（メール送信数，メール発信数，メール返信数，メール受信数）の集計結果を示す．表 2 中の発信割合，返信割合，受信割合はそれぞれ，メール送信に対するメール発信，メール返信，メール受信の割合を意味する．また，表 3 に各開発者のメールの Politeness 値の基本統計量（平均値，中央値，最大値，最小値，標準偏差）を示す．

表 2 より，活動期間が異なるため単純比較はできないものの，活動期間の最も長い Dev#1 のメール送信およびメール受信はともに 10 名のトップ開発者の中では最多であることがわかる．一方，1 ヶ月当たり（月平均）のメール送信数は 16.0 件〜109.8 件と，開発者間で大きなばらつきが見られる．メール発信では，Dev#6 および Dev#9 が他の開発者に比べて発信数，発信割合ともに顕著に高い値を示しており，メール返信とメール受信の返信・受信割合はそれほど高い値でないことから，開発者メーリングリス

表 2: 開発者毎の活動期間とメール数

開発者	活動期間		メール送信		メール発信		メール返信		メール受信	
	開始期間 ~ 終了期間	期間（ヶ月）	送信数	月平均	発信数	発信割合	返信数	返信割合	受信数	受信割合
Dev#1	1995/10/3~2016/3/29	245	**8006**	32.7	1014	12.7	6992	87.3	**5610**	70.1
Dev#2	2000/1/31~2016/3/30	194	7391	38.1	694	9.4	6697	**90.6**	5580	**75.5**
Dev#3	1999/1/26~2003/1/13	48	5268	109.8	503	9.6	4765	**90.5**	4186	**79.5**
Dev#4	1995/10/6~2007/1/13	135	5063	37.5	790	15.6	4273	84.4	3303	65.2
Dev#5	2000/1/29~2016/3/29	194	4820	24.8	569	11.8	4251	88.2	3399	70.5
Dev#6	1996/12/23~2008/1/30	133	4064	30.6	**2472**	**60.8**	1592	39.2	2051	50.5
Dev#7	1995/9/11~2015/9/5	240	3838	16.0	443	11.5	3395	88.5	2437	63.5
Dev#8	1996/11/24~2004/12/13	97	3738	38.5	1070	28.6	2668	71.4	2513	67.2
Dev#9	1995/9/11~1999/6/17	45	3279	72.9	**1699**	**51.8**	1580	48.2	1741	53.1
Dev#10	1995/9/12~2002/7/3	82	2629	32.1	620	23.6	2009	76.4	1526	58.0

表 3: 開発者毎の Politeness 値

開発者	メール送信					メール発信					メール返信					メール受信				
	平均値	中央値	最大値	最小値	標準偏差	平均値	中央値	最大値	最小値	標準偏差	平均値	中央値	最大値	最小値	標準偏差	平均値	中央値	最大値	最小値	標準偏差
Dev#1	**0.50**	**0.48**	1.00	0.01	0.21	0.57	0.57	1.00	0.10	0.21	0.49	0.47	1.00	0.01	0.21	0.56	0.55	1.00	0.03	0.23
Dev#2	**0.62**	**0.64**	1.00	0.01	0.24	0.66	0.71	1.00	0.05	0.24	**0.62**	**0.63**	1.00	0.02	0.24	**0.59**	**0.58**	1.00	0.03	0.23
Dev#3	0.64	0.66	1.00	0.01	0.22	0.73	0.78	1.00	0.11	0.21	**0.63**	**0.65**	1.00	0.01	0.22	**0.62**	**0.63**	1.00	0.01	0.23
Dev#4	0.55	0.55	1.00	0.01	0.23	0.61	0.65	1.00	0.01	0.25	0.54	0.53	1.00	0.01	0.23	0.60	0.60	1.00	0.04	0.23
Dev#5	0.57	0.56	1.00	0.03	0.23	0.63	0.64	1.00	0.12	0.23	0.56	0.55	1.00	0.03	0.23	0.60	0.59	1.00	0.06	0.23
Dev#6	0.70	0.75	1.00	0.01	0.26	**0.79**	**0.91**	1.00	0.12	0.24	0.56	0.55	1.00	0.04	0.23	0.55	0.53	1.00	0.02	0.23
Dev#7	0.54	0.52	1.00	0.01	0.22	0.59	0.60	0.99	0.03	0.23	0.53	0.51	1.00	0.01	0.22	0.55	0.58	1.00	0.05	0.23
Dev#8	0.57	0.56	1.00	0.03	0.24	0.61	0.61	1.00	0.06	0.25	0.56	0.55	1.00	0.03	0.23	0.55	0.55	1.00	0.01	0.23
Dev#9	0.58	0.57	1.00	0.03	0.24	**0.54**	**0.50**	1.00	0.07	0.25	0.61	0.62	1.00	0.03	0.23	0.56	0.54	1.00	0.04	0.22
Dev#10	0.59	0.59	1.00	0.02	0.22	0.63	0.65	1.00	0.07	0.23	0.57	0.57	1.00	0.02	0.21	0.54	0.53	1.00	0.03	0.23

ト上で開発者全体へ向けてバグフィックスやリリース計画などについてのアナウンスをおこなうことが多い開発者であると推察される．メール返信とメール受信については，Dev#2 および Dev#3 が他の開発者と比べて高い返信・受信割合を示しており，他の開発者からの質問に答えたり相談に乗る機会が多い開発者であると推察できる．

表 3 より，極端に大きな違いではないものの，各種メールに対して算出した Politeness 値の平均値，中央値ともに開発者毎に違いがあることが確認できる．最多のメール送信数を示した Dev#1 のメール送信についての Politeness 値は，平均値 0.50，中央値 0.48 と，10 名の開発者の中で最低の値を示している．さらに，Dev#1 に次いでメール送信数の多い Dev#2 のメール送信についての Politeness 値は，平均値 0.62，中央値 0.64 と，必ずしも低い値ではないことから，メールの送信数と Politeness 値には相関は見られない．Dev#1 のメール送信についての Politeness 値が低いのは，気使いを必要としないコア開発者との議論がコミュニケーションの中心である可能性があり，今後詳しく調査する必要がある．

メール発信の多かった Dev#6 および Dev#9 のメール発信についての Politeness 値については，Dev#6 が最も高く（平均値 0.79，中央値 0.91），Dev#9 が最も低い（平均値 0.54，中央値 0.50）結果となった．開発者全体へのアナウンスが中心であれば Dev#6 の Politeness 値は妥当な結果といえるが，Dev#9 のメール送信についての Politeness 値は理解するのが難しい．今後の分析により，Dev#6 および Dev#9 が発信したメールの内容を精査する必要がある．

メール返信およびメール受信の多かった Dev#2 および Dev#3 の Politeness 値については，メール送信についての Politeness 値では両者が他の開発者よりも高い値を示している．また，メール受信についての Politeness 値では Dev#3 が最も高く（平均値 0.62，中央値 0.63），Dev#2 についても Dev#3 以外の開発者と同等以上の値を示している．これらのことから，Dev#2 および Dev#3 は，開発者メーリングリストに投稿された他の開発者からの疑問や質問に対して丁寧に回答していることが伺える．結果として，他の開発者からの送信されたメール（受信メール）の Politeness 値も概ね高く，Dev#2 および Dev#3 は，コミュニティ内の多くの開発者と良好な関係性を構築するために重要な役割を担っているものと思われる．

5.2 RQ2: トップ開発者と一般開発者の Politeness に違いはあるか？

分析意図: RQ1 では，トップ開発者の Politeness 値そのものに注目して分析をおこない，開発者毎に Politeness 値に違いが見られること，特に，Dev#1 のメール送信についての Politeness 値が他の開発者に比べ低いことが分かった．ただし，トップ開発者以外の一般開発者の Politeness 値との比較をおこなっていないため，Dev#1 を含むトップ開発者の Politeness 値が高いか低いかを客観的に知ることができない．一般開発者の Politeness 値を計測し，トップ開発者と比較することで RQ1 で得られた知見の妥当性を検証する必要がある．

分析方法: ランダムに選出した 100 名の一般開発者（トップ開発者以外の開発者）のメールデータから Politeness 値の基本統計量を算出し，トップ開発者の Politeness 値と比較する．

結果: 表 4 に，100 名の一般開発者のメール送受信数（メール送信数，メール発信数，メール返信数，メール受信数）の集計結果を示す．また，表 5 に各メールの Politeness 値

表 4: ランダムに選出した 100 名の一般開発者のメール数

メール送信	メール発信		メール返信		メール受信	
送信数	発信数	発信割合	返信数	返信割合	受信数	受信割合
5385	1509	28.0	3876	72.0	4006	74.4

表 5: ランダムに選出した 100 名の一般開発者のメールの Politeness 値

メール送信					メール発信					メール返信					メール受信				
平均値	中央値	最大値	最小値	標準偏差	平均値	中央値	最大値	最小値	標準偏差	平均値	中央値	最大値	最小値	標準偏差	平均値	中央値	最大値	最小値	標準偏差
0.63	0.65	1	0.01	0.24	0.65	0.68	1	0.05	0.24	0.62	0.64	1	0.01	0.24	0.59	0.60	1	0.01	0.23

の基本統計量（平均値，中央値，最大値，最小値，標準偏差）を示す．

表 4 より，ランダムに選出した 100 名の開発者の発信割合は 28.0%であることが見て取れる．これは，トップ開発者の大半の発信割合より高い値であり，ランダムに選出した 100 名の開発者は，メーリングリストを通じて質問や依頼を発信する側になることが多いためと考えられる．一方，受信割合はトップ開発者の Dev#2 および Dev#3 と同程度の 74.4%を示しており，発信した質問や依頼に対して比較的多くの開発者から返信が返ってきていることがうかがえる．

表 5 より，ランダムに選出した 100 名の開発者のメール送信，メール発信，メール返信，メール受信についての Politeness 値の平均はそれぞれ，0.63, 0.65, 0.62, 0.59 とほぼ同等の値となっている．これらはトップ開発者の各 Politeness 値の平均と比べてやや高めの値であるものが多い．ランダムに選出した 100 名の開発者は基本的には頻繁に投稿する開発者ではないため，不慣れな開発者メーリングリストでの議論において Politeness が高めになるのは不思議ではない．これらのことから，RQ1 において得られた Dev#1 のメール送信についての Politeness 値は，コミュニティ全体としても低めの値であるといえる．最もメール送信数の多い Dev#1 の Politeness が比較的低いという状況は，開発コミュニティの維持・発展においては好ましいものではないと思われるが，最も頻繁にメール送信をおこなうが故の特殊理由が存在する可能性もある．今後，Dev#1 のメールの内容を精査し算出された Politeness 値の妥当性を確認する必要がある．

5.3 RQ3: 自ら発信する場合と他者へ返信する場合で Politeness に違いはあるか？

分析意図: 開発者メーリングリストにおける発言や議論は購読しているすべての開発者に配信されるため，開発コミュニティの維持・発展に対する影響を踏まえると，極端に Politeness が低くなるような発言は控えることが望ましい．特に，自らメーリングリスト上に情報を発信する場合は，他者の発言に対して返信する場合とは異なり，特定の相手を想定した発言ではなく購読者全員に対しての発言をおこなうことが多いと考えられることから，メール発信の方がメール返信の場合に比べて Politeness 値は高くなるものと予想される．ただし，前述の理由により，返信する相手を特定できるメール返信の場合であっても，Politeness 値が低すぎるような言語的表現は差し控えるべきであるため，メール発信とメール返信についての Politeness 値には極端な差は見られない可能性があり，検証する必要がある．

分析方法: メール発信とメール返信についての Politeness 値を開発者毎に比較する．

結果: 表 2 より，メール返信に比べメール発信についての Politeness 値は，Dev#9 を除いたすべての開発者において，平均値，中央値ともにより大きな値を示していることが分かる．また，Dev#6 を除いたすべての開発者において，メール発信とメール返信についての Politeness 値の差は，ほぼ 0.1 以内に収まっており大きなものではない．

RQ3 については概ね予想通りの結果であったが，Dev#6 と Dev#9 については RQ1 においても解釈が困難な結果を示しており，RQ3 の結果をより正確に理解するためにもメールの内容を精査する必要があることがわかった．

5.4 RQ4: Politeness は時期を問わず一定か？時期により変動するか？

分析意図: RQ1〜RQ3 までは，開発者が送受信したすべてのメールの Politeness 値を算出し，基本統計量に基づいて議論した．しかしながら，OSS 開発コミュニティは，新機能の追加などにより開発が急速に進められる時期や，主にバグフィックスなどの保守が主流となる時期が存在するなど，常に一定して進化するわけではない [11]．一方，開発コミュニティの進化とコミュニケーション様式の関係性について調査した研究が存在しないため，Politeness 値が時期を問わず一定して推移するものなのか，あるいは，時期により変動するものなのかについては検証する必要がある．また，開発コミュニティの状態（活発な開発時期なのか，開発が停滞している時期なのかなど）とは別に，個々の開発者が開発コミュニティに参加してから脱退するまでの Politeness 値の推移についても不明なため調査する必要がある．

分析方法: 各種メールについての Politeness 値を時系列にプロットし，開発者毎の Politeness 値の推移を分析する．

結果: 図 3〜図 6 にそれぞれ，開発者毎の各種メールの Politeness 値の時系列の推移を月毎にプロットしたグラフ（青色の折れ線）を示す．各図中には，参考までに各種メー

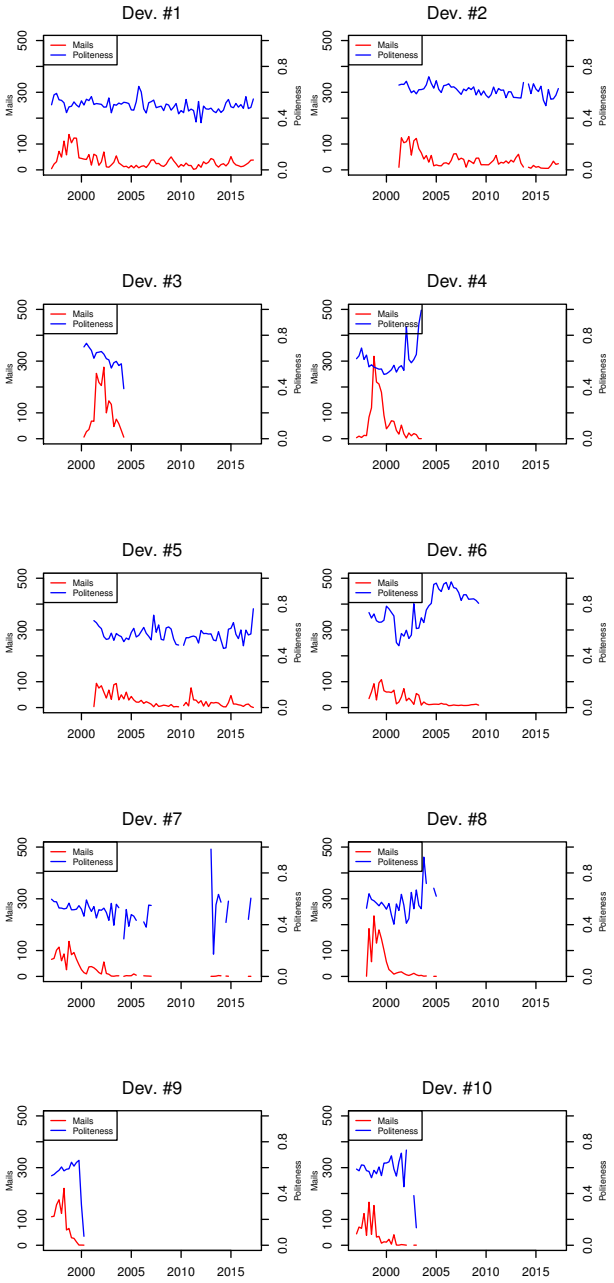

図 3: 開発者毎のメール送信数と Politeness 値の 3 ヶ月平均の時系列変化

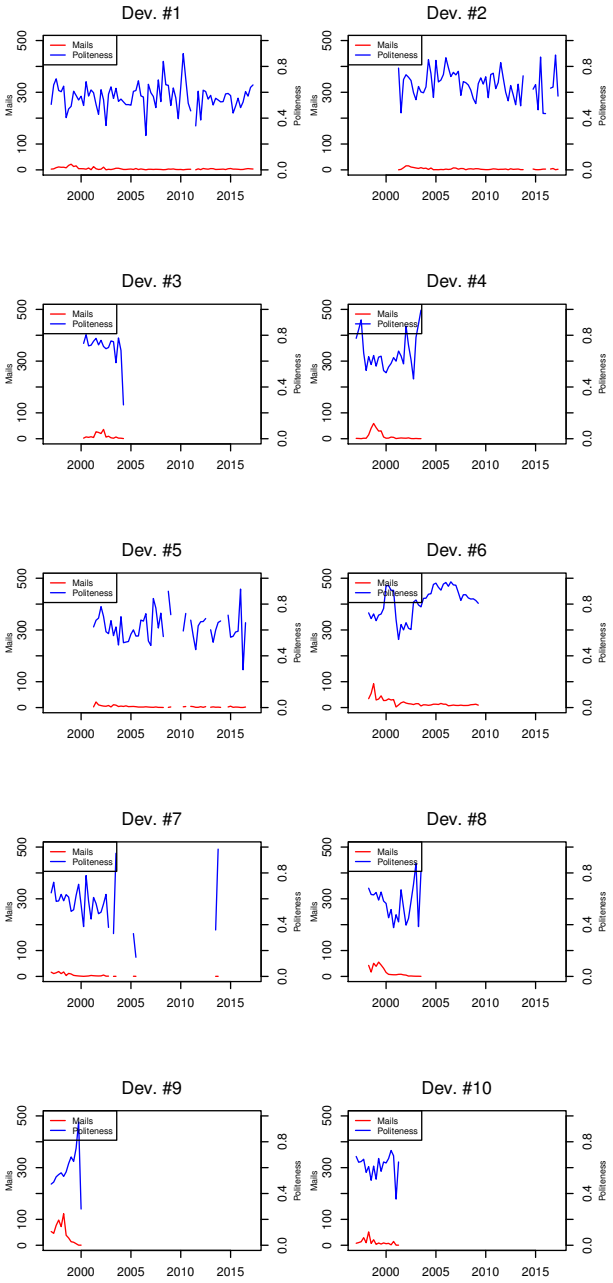

図 4: 開発者毎のメール発信数と Politeness 値の 3 ヶ月平均の時系列変化

ルの月毎の件数も同様にプロット（赤色の折れ線）している．なお，単純に月毎にデータを集計してプロットするとメール件数の少ない期間の外れ値の影響を受けてグラフが読み取りづらくなるため，Politeness 値，メール件数ともに，3 か月間の移動平均をプロットしたグラフであることに注意されたい．

メール発信数と Politeness 値の時系列変化: 図 3 より，プロジェクト参加直後のメール送信についての Politeness 値は，ほとんどの開発者が 0.6 前後の値を示している．Dev#1 や Dev#2, Dev#5 といった長期間（15 年以上の）活動している開発者の Politeness 値には急激な変化が見られず，± 0.2 を超える変化は見られなかった．

一方, Politeness 値が大きく変動する開発者も確認できた．具体的には，Dev#3, Dev#4, Dev#8, Dev#9, Dev#10 といった活動期間の短い開発者である．例えば，Dev#3 は 1999 年 1 月〜2003 年 1 月の 4 年間活動をおこないってい

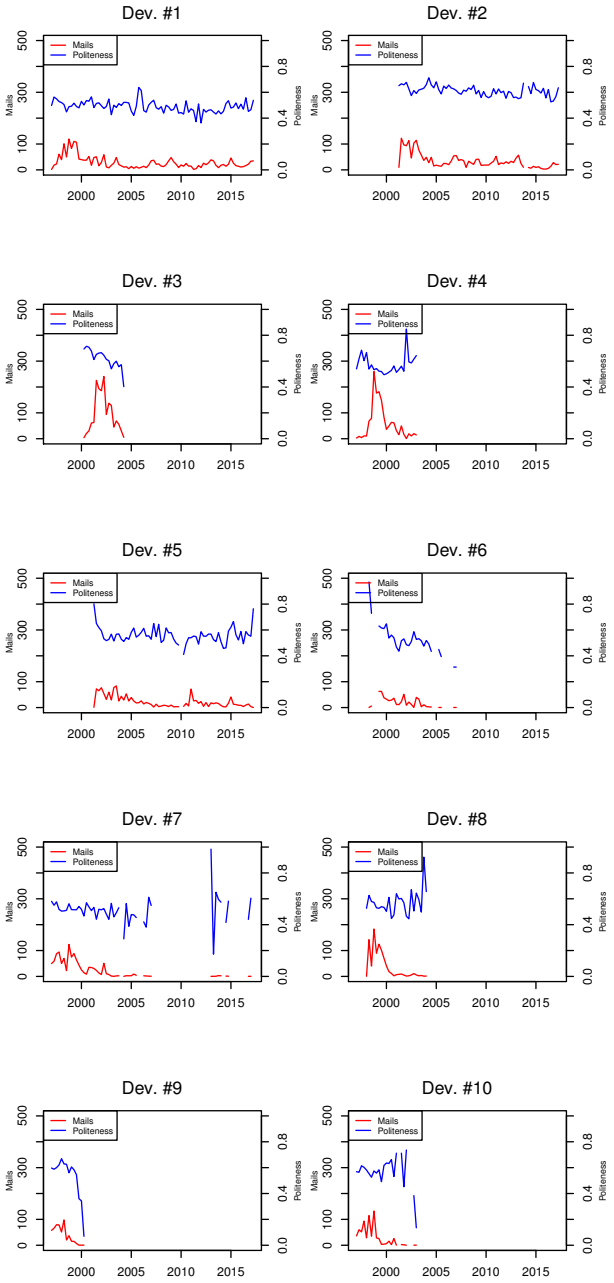

図 5: 開発者毎のメール返信数と Politeness 値の 3 ヶ月平均の時系列変化

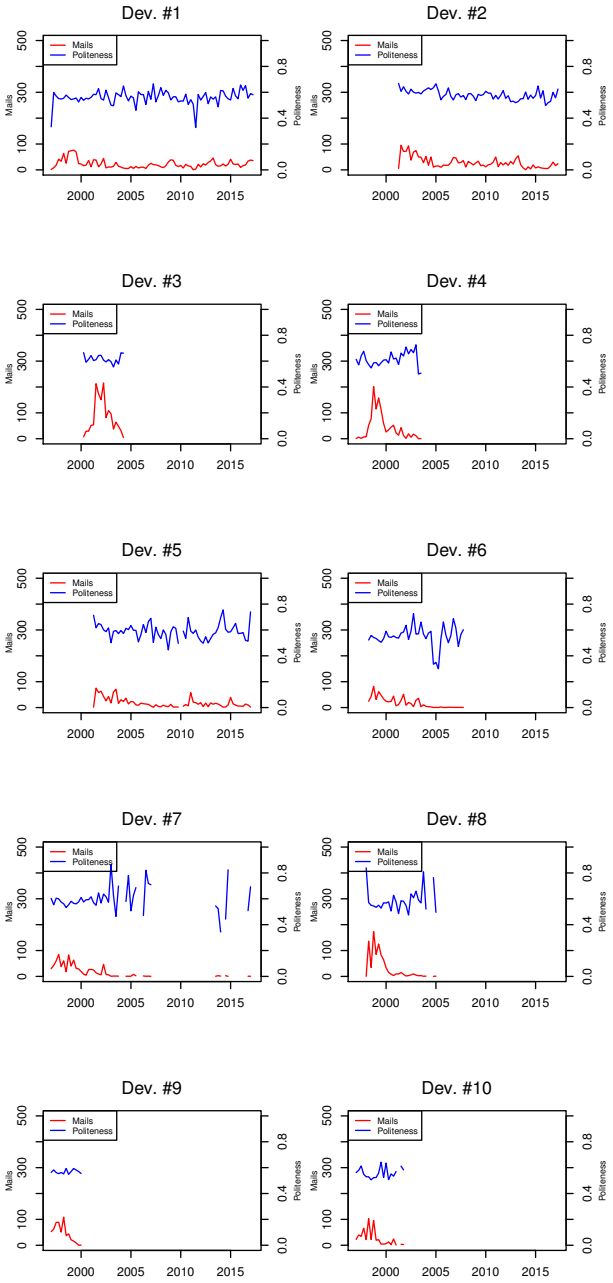

図 6: 開発者毎のメール受信数と Politeness 値の 3 ヶ月平均の時系列変化

るが，時間経過とともに Politeness 値が減少している．さらに，コミュニティから離脱する直前の Politeness 値は最も急激に減少している．一方，Dev#4 のように，時間経過とともに Politeness 値が増加している開発者も存在している．

また，Dev#10 は，平均値の増加と減少を繰り返し，離脱する際に大きく減少している．他には，Dev#4 や#8 のように月日が経つにつれ，徐々に平均値が増加するという開発者も見られた．これらの開発者に関しては，プロジェクトから離脱する際に急激に平均値が増加するという結果が見られた．

メール発信数と Politeness 値の時系列変化: 図 4 より，メール発信についての Politeness 値についても，メール送信数に比べ件数が少ないため変動が大きくなっているが，Politeness 値の推移については基本的に図 3 と同様の傾向が見られる．

メール返信数と Politeness 値の時系列変化: 図 5 より，メール返信についての Politeness 値についても，図 3 および図 4 と同様の傾向が見られるが，Dev#6 は唯一他の開発者とは逆の傾向が見られる．図 3 および図 4 における Dev#6 の Politeness 値は時間経過とともに増加していくのに対して，図 5 では Politeness 値が減少していく傾向がある．

メール受信数と Politeness 値の時系列変化: 図 6 より，メール受信についての Politeness 値についても，図 3 および図 4 と同様の傾向が見られるが，Dev#3 および Dev#4 は開発者とは逆の傾向が見られる．

Politeness 値の大幅な変動とコミュニティからの離脱について: RQ4 の調査により，すべてのトップ開発者の Politeness 値は時間経過とともにある程度あるいは急激に変動するものであることが分かった．特に，図 3 のメール送信についての Politeness 値の推移から見て取れるように，数名の開発者（Dev#3 や Dev#9, Dev#10）は急激な Politeness 値の減少の後にプロジェクトを離脱していることが分かる．

そこで，Politeness 値が低いメールの本文を目視し，議論の内容と Politeness 値の変動にどのような関係があるかを確かめた．その結果，Politeness 値の低いメールでは，「この修正は必要ではない」，「この修正は結果としてバグを増やしているために間違いだ」といったように，メールを送信している開発者が他の開発者とは反対の意見を提示したり，極端な場合は口論に発展しているものがあった．特に Dev.#3 については，2003 年 1 月に開発コミュニティを離脱しているが，離脱直前に手法の違いから他の開発者と口論になっている．その期間の Politeness 値は 0.2 近く減少しており，表 3 の平均値と比較しても大きく減少しているのが分かる．

これらのことから，今回のケーススタディでは，Apache HTTPD Sever プロジェクトの持続的進化についてコミュニケーションの様式から直接的な関係を明らかにすることはできなかったが，コミュニケーションデータから算出した Politeness 値と実際のコミュニケーションの質は直感的に理解出来る程度に一致しており，今後の分析においても Politeness 分析は有効であるといえる．

6. 今後の分析方針と本研究の制約・本論文のまとめ

6.1 今後の分析方針

今回のケーススタディでは，Apache HTTPD Server を対象に OSS 開発コミュニティにおける開発者のコミュニケーションの質的側面を見るために，Politeness 分析を適用した．しかしながら，5 章で述べた様に，Politeness 値の変動理由が明確でないため，メールの内容を目視で確認することで，Politeness 値の変動理由を明らかにする必要がある．また．コミュニケーションにおいて，コア開発者同士のコミュニケーションなのか，コア開発者と多数の一般開発者同士のコミュニケーションなのか区別がついておらず，開発者間の人間関係が考慮されていない．よって，今後はソーシャルネットワーク分析を用いて人間関係を考慮した Politeness 分析をおこなう必要がある．また，コミュニティの時期や活動を考慮していないため，OSS 開発コミュニティの持続的進化と Politeness 値の関係が不明である．そこで，パッチ投稿数・バグ報告数等から図るプロジェクト活動の変化と Politeness 値を関連付けた分析が必要である．

6.2 本研究の制約

本研究では，開発者のコミュニケーションツールであるメールのデータを得るために，Apache HTTPD Server のメーリングリストのアーカイブから mBox ファイルを取得した．しかし，返信メールの約 5 %に関して，返信先のメッセージ ID の記載がされていなかった．そのため，実際は返信メールではあるが発信メールとして処理してしまっている．また，4 章で述べた通り，引用文やソースコード等の開発者本人に対しての Politeness 分析に不必要なテキストの除去ができていない．不必要なテキストの除去は，今後の分析において第一に対処しなければならない問題である．

6.3 本論文のまとめ

本研究では，Apache HTTPD Server の送信メール数トップ 10 人を対象に，OSS 開発コミュニティの進化の理解を目的とした Politeness 分析をおこなった．結果として，今回のケーススタディでは OSS 開発コミュニティの進化の理解を満たすことができなかったが，開発者毎に Politeness 値の個人差があること，時間的変化による Politeness 値の変化があることが分かった．よって，コミュニケーションの質的側面を図る上で Politeness 分析には有用性があることがいえる．今後は開発者間の人間関係やプロジェクトの時期を考慮した Politeness 分析をおこない，OSS 開発コミュニティの進化の理解を目指す．

謝辞

本研究の一部は，文部科学省科学研究補助金（基盤 (C): 15K00101）による助成を受けた．

参考文献

[1] IPA(独立行政法人情報処理推進機構)：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 (2009 年度調査) (2009).

[2] 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. 6 (2007).

[3] Matsumoto, S., Kamei, Y., Ohira, M. and Matsumoto, K.: A Comparison Study on the Coordination between Developers and Users in Foss Communities, *Socio-Technical Congruence (STC'08)*, pp. CD–ROM–NO.8–1–9 (2008).

[4] Tourani, P., Jiang, Y. and Adams, B.: Monitoring Sentiment in Open Source Mailing Lists: Exploratory Study on the Apache Ecosystem, *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering (CASCON'14)*, pp. 34–44 (2014).

[5] Brown, P. and Levinson, S. C.: *Politeness: some universals in language usage*, Cambridge University Press (1987).

[6] Danescu-Niculescu-Mizil, C., Sudhof, M., Jurafsky, D., Leskovec, J. and Potts, C.: A computational approach to politeness with application to social factors, *Proceedings of 51st Annual Meeting of the ACL (Association for Computational Linguistics) (ACL '13)* (2013).

[7] DeMarco, T. and Lister, T.: *Peopleware: Productive Projects and Teams*, New York: Dorset House Publishing (1987).

[8] Ortu, M., Adams, B., Destefanis, G., Tourani, P., Marchesi, M. and Tonelli, R.: Are Bullies More Productive?: Empirical Study of Affectiveness vs. Issue Fixing Time, *Proceedings of the 12th Working Conference on Mining Software Repositories (MSR '15)*, pp. 303–313 (2015).

[9] Ortu, M., Destefanis, G., Kassab, M., Counsell, S., Marchesi, M. and Tonelli, R.: Would you mind fixing this issue?, *Agile Processes, in Software Engineering, and Extreme Programming*, Springer, pp. 129–140 (2015).

[10] Gunn, S.: Support Vector Machines for Classification and Regression, Technical Report, School of Electronics and Computer Science, University of Southampton (1998).

[11] Nakakoji, K., Nakakoji, Y. Y., Nishinaka, Y., Kishida, K. and Ye, Y.: Evolution Patterns of Open-Source Software Systems and Communities, *5th International Workshop on Principles of Software Evolution (IWPSE2002)*, pp. 76–85 (2002).