特別研究報告

題目

オープンソース開発支援用メール検索システムの試作

指導教官
井上 克郎 教授

報告者
高尾 祐治

平成 14 年 2 月 19 日

大阪大学 基礎工学部 情報科学科

平成 13 年度 特別研究報告

オープンソース開発支援用メール検索システムの試作

高尾 祐治

内容梗概

インターネットに代表されるネットワーク環境の発展に伴い，多くの開発者が協調し開発を行う開発手法である,「オープンソース開発」が広く使われるようになった．オープンソース開発では，開発中のソースコード，ドキュメント等，プロダクトを広く公開し，世界中に分散する開発者が開発に参加する．オープンソース開発環境では，版管理システムを用いてプロダクトの管理を行い，電子メールを用いて開発者間の意志疎通を行うことにより，多くの開発者が並列的に開発作業を行うことができる．

しかし，現状のオープンソース開発環境では，各システムがそれぞれ固有の形式で開発履歴情報を蓄積するため,「開発を行うための情報が利用困難」という問題がある．また，開発環境を構成する各システムが互いに連係を持たず，開発履歴情報が分散して存在するため,「システム間の連係不足」という問題がある．そこで，私の所属する研究グループは，それらの問題点を解決し，大規模なソフトウェアプロダクトをオープンソース開発を用いて，開発，保守するために，オープンソース開発支援環境の構築を目指す．

本研究では，オープンソース開発で意志疎通の重要な手段として用いられるメーリングリストに注目した．メーリングリストアーカイブにはこれまでに議論された，開発に関わる情報が多数蓄積される．しかし，既存のシステムは検索機能の不足により，アーカイブに存在する有益な情報の多くを開発者に提供することができない．本稿では，アーカイブに蓄積された議論の内容を開発者に提供するために，3 種類のスレッド間の結び付きを提案する．さらに，それらの結び付きに基づく，3 種類のスレッド検索が可能なオープンソース開発支援用メール検索システムの設計と実装を行う．本システムにより，効率の良いアーカイブ検索が可能となる．

また，実際のオープンソース開発で用いられたメーリングリストアーカイブを使って，評価を行った結果，本システムを用いることで開発者は，有益な情報を入手することが可能となり，現状の問題点が解決されることを確認した．

主な用語

オープンソース開発 (Open Source Development)
メーリングリスト (Mailing List)
検索 (Search)

目 次

1　はじめに

ソフトウェアの開発規模が増大するのに伴い，その開発形態は多人数化，分散化している．ある一つの大規模なソフトウェアを開発する場合，複数の開発者が，互いにソースコードを共有しながら同時に一つの開発作業に携わることが一般的になりつつある [2]．また，インターネットやイントラネット等のネットワーク環境の充実に伴い，各開発者がそれぞれ異なる拠点で開発作業を行うことも珍しくなくなってきた．

近年では，オープンソース開発によるソフトウェアの開発が注目されている [26][27][30]．オープンソース開発は，開発中のソフトウェアプロダクトを広く公開することにより，世界中に分散した開発者がいつでも自由に参加できる．公開するソフトウェアプロダクトはオープンソース開発環境で管理されるため，多くの開発者が並列的に開発作業を行うことが可能となる．オープンソース開発環境では，版管理システムや電子メール等，多くの既存システムが用いられる．

ところが，現状のオープンソース開発環境にはいくつかの問題点が存在する．例えば，多くの既存システムがそれぞれ固有の形式で開発情報を蓄積しており，開発者がそれらの情報を利用するコストが大きい．あるいは，多くのシステムをまとめて提供されるが，各システムが連係を持たないために，開発情報が分散して存在する．それらの問題点を解決するために，私の所属する研究グループでは，オープンソース開発向けの開発支援環境の構築を目指す．この開発支援環境は，数年間のオープンソース開発で蓄積された膨大なソフトウェアプロダクトを有効に利用することにより，その生産性を高めることを目的として，開発者の作業支援を行う．

本研究では，この開発支援環境の構成要素の一つとして，オープンソース開発支援用メール検索システムの設計と実装を行った．オープンソース開発では意志疎通の重要な手段としてメーリングリストが用いられ，メーリングリストアーカイブには開発に有益な情報が他数蓄積される．しかし，既存のシステムは検索機能の不足により，アーカイブに存在する有益な情報の多くを開発者に提供することができない．この問題を解決し，アーカイブ内の情報を開発者に提供し開発を支援することを目的とする．

本システムは，既存のメール検索システムで多く用いられる全文検索を行う他に，メール本文中から取得したキーワード，ファイルパスおよび，議論者による再検索を行い，全文検索の結果として得られたメールに関連のあるメールを検索する．本システムは,「メールデータベース作成プログラム」と「検索，表示プログラム」という二つのプログラムから構成される．メールデータベース作成プログラムは，メールを効率良く検索，表示するために，検索対象のメールを解析し，そのデータベースを作るプログラムである．検索，表示プログラムは，メールを検索するためのユーザーインターフェースであり，perl による CGI を利用し

た．本システムを利用することにより，開発者はメーリングリストアーカイブから多くの情報を取得することができるようになるため，開発を支援することができる．

また，FreeBSD の開発で用いられるメーリングリストを用いて本システムの評価を行った．まずにメールデータベースを作成するのにかかる時間を測定した．その結果，十分に実用的な時間で動作し，刻々と増え続けるメーリングリストアーカイブに検索システムが追随できることを確認した．次に，既存システムとの比較を行い，情報をより多く提供することが可能であることを確認した．

本システムを組み込んだオープンソース開発向けの開発支援環境を構築することにより，限られた時間を有効に利用して開発作業を行うことが可能となり，開発者の負担を軽減し，より質の高いソフトウェアを開発するための基礎とすることができると考えられる．

以降，2. では，オープンソース開発とその開発環境について説明し，その問題点について述べる．そして，我々が研究中であるオープンソース向けの開発支援環境の提案を行う．3. では，メールおよびメーリングリストについて触れ，既存のメール検索システムの問題点について述べる．4. では，オープンソース開発支援用メール検索システムの設計と実装について述べ，評価を行う．最後に，5. で本研究のまとめと今後の課題について述べる．

2　オープンソースとその開発環境

本節では，オープンソースとその開発環境について触れ，オープンソース開発環境の持つ問題点を説明する．

2.1　オープンソース

開発中のソースコードやドキュメント等のプロダクトを広く公開して複数の開発者が並列的にソフトウェアの開発作業を行う開発手法はオープンソース開発と呼ばれ [26][27][30]，高品質で多機能なソフトウェアを開発できるとして注目を集めている．そのオープンソース開発によって開発されたソフトウエアをオープンソースソフトウエアと言う．

オープンソース開発では，世界中に分散した各開発者が，インターネットに代表される大規模ネットワークを使って開発作業を行う．そのため，開発者はいつでも自由に開発作業に参加することが可能である．FreeBSD[33] や Linux[19]，GNU[12]，Apache[1] 等は，オープンソース開発で開発されたオープンソフトソフトウェアである．

2.2　オープンソース開発環境

オープンソース開発では，各開発者がそれぞれ分散して並列的に開発作業を行うことが可能である．その一方で，開発中のソースコードやドキュメント等のプロダクトを広く公開するため，それらの管理を行う必要がある．そこで，オープンソース開発に参加する開発者は，オープンソース開発環境と呼ばれる環境の中でプロダクトの管理を行う．

オープンソース開発環境の構成例を図 1 に示す．オープンソース開発環境は，一般に複数の既存システムから構成される．図 1 の構成例の場合，ソースコードやドキュメント等のプロダクトは，版管理システム [8] の一つである CVS(Concurrent Versions System)[4][11][16][28] を用いて管理される．それらのプロダクトは，rsync や ftp を利用して，各開発者に複製，配布される．また，開発者間で相互に行われる意志疎通の手段として，電子メールやメーリングリストが用いられる．その内容はアーカイブとして保存され，WWW(World Wide Web) を用いた検索エンジンによって自由に検索や閲覧が可能である．開発者からのバグ報告等フィードバックは，GNATS(GNU Problem Report Management System) を用いたバグデータベースによって管理される．

以下では，これらのシステムの中から，オープンソース開発で広く用いられる版管理システムと電子メールについて説明する．

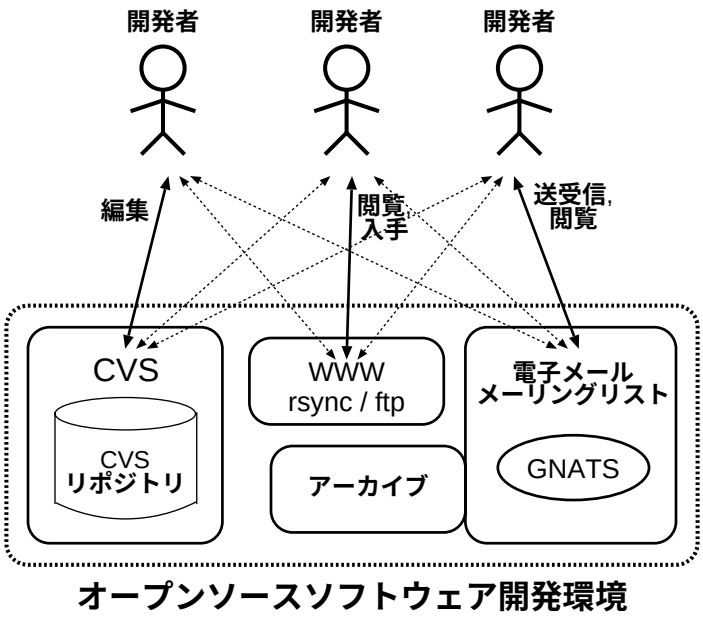

図 1: オープンソース開発環境の構成例

2.2.1 版管理システム

版管理システムとは，ソフトウェア開発の際に行われる各プロダクトに対する作業履歴を蓄積し，管理するシステムである．また，蓄積された作業履歴を開発者に提供することにより，プロダクトの修正作業を開発者間で調整することができるため，間違いのない修正を行うことが可能となる [3][9]．あるいは，ソフトウェアの再利用や開発プロセスの作成を行う際にも役立つ [10].

版管理システムは，各プロダクトに対する作業履歴を，リポジトリ (repository) にファイル単位で蓄積する．開発者がリポジトリにプロダクトを格納することをコミット (commit) と呼ぶ．リポジトリの内部では，プロダクトのある時点における状態であるリビジョン (revision) 単位で管理する．そのため，内部的にリビジョン番号，リビジョン間の差分情報，更新日時，更新者，更新時のコメント等をリビジョン情報として保持する．

版管理システムは数多く存在し，実際にさまざまな開発において利用される．また，ローカルな環境で利用するシステムもあれば，ネットワークを介してグローバルな環境で利用するシステムもある [14][15][23]．例えば，FreeBSD や Linux 等の UNIX 系オペレーティングシステム上では，RCS(Revision Control System)[35] や CVS が標準的に利用可能である．あるいは，ClearCase[29]，Visual SourceSafe[22] や PVCS[21] 等，商用の版管理システムも数多く発売されている．

また，版管理システムを用いて，ソフトウェアの開発作業や保守を支援する試みも行われ

ている．例えば，ソースコードの差分情報を利用した，ソフトウェア開発時のデバッグ支援システムがある [32].

オープンソース開発では，版管理システムとして CVS が広く用いられる．CVS は，UNIX 上で動作する標準的な版管理システム RCS を改良して設計されており，以下のような RCS が持つ問題点を解決する．

- プロジェクトがファイル単位で管理される

  RCS では，プロジェクトがファイル単位でのみ管理されるため，複数のファイルをディレクトリ構造として扱うことが困難である．

  CVS では，ディレクトリ構造を認識するため，プロジェクトをディレクトリ単位で管理することが可能である．

- 開発者がファイルをロックする方式である

  RCS では，開発者がファイルをロックして修正を行うため，他の開発者が並列的に開発作業を行うことが困難である．

  CVS では，開発者がリポジトリから作業用コピーを取得して修正を行い，同時に加えられた変更を必要に応じてマージすることができるため，ファイルをロックすることなく修正を行うことが可能である．

- ネットワーク環境に対応していない

  RCS では，各ファイルの履歴情報がそれぞれのマシンに散在する一方で，ネットワーク環境に対応していないため，開発者が分散して開発作業を行うことが困難である．

  CVS では，ネットワーク環境に対応するため，インターネット上のどこからでもアクセスすることができる．

以上のような RCS が持つ問題点を解決する CVS は，オープンソース開発の版管理システムとして非常に適する．

CVS は，GUI や Web インターフェース，エディタ等のさまざまな目的に応じて，その関連ツールが数多く開発されており，実際に利用される．例えば，CGI を利用した CVS の Web インターフェースとして cvsweb[7] がある．cvsweb を利用することにより，リポジトリ内に存在するファイル一覧や，各リビジョンのデータ，リビジョン間の差分等を既存の Web ブラウザで閲覧することが可能である．同様のシステムとして，bonsai[34] 等もある．

2.2.2 電子メール

電子メールとは，インターネットやイントラネット等のネットワークを通じて、文書や画像等のデータをやりとりするためのシステムである．今日では，ネットワーク環境の充実に伴い，容易に利用することが可能となり，単に「メール」と称されることも多い．

オープンソース開発では，開発者が世界中に分散して存在し，開発作業を行うことが多い．そのため，互いの意志疎通のための手段として，インターネットを介した電子メールが一般的に利用される．オープンソース開発では，開発者間での電子メールのやりとりを一括して管理するために，メーリングリスト (Mailing List) と呼ばれるシステムが利用されることも多い．メーリングリストでは，例えば，ある参加者がメーリングリスト宛に電子メールを送信すると，同じ電子メールを参加者全員に配信される．また，誰かがその電子メールに対して返信すると，その電子メールも参加者全員に配信される．このため，他の開発者間での議論内容を，各開発者が容易に捕捉することが可能となる．さらに，これらの電子メールが膨大な量になると，アーカイブ (archive) として管理される．また，WWW を用いて，アーカイブの中から電子メールによる議論の内容を検索するシステムも存在する．

これらのシステムを利用することにより，分散している開発者の間でさまざまな情報を共有することが可能となり，開発作業の促進につながる．

2.3 オープンソース開発の問題点

オープンソース開発では，分散した複数の開発者が自由に各々の開発作業を行う．これらの開発者は，普段から開発作業に専念するのではなく，個人的な時間を利用して作業を行うことが多い．従って，並列的に作業を行う他の開発者の進捗状況を綿密に追跡する時間が十分に確保できず，その把握が難しい．あるいは，開発者間での意志疎通が不足してしまう傾向がある．すなわち，この種の開発においては，開発者間での意志疎通の支援が不十分であることが，重要な問題点の一つである．

以下では，本研究において，この問題点を解決する上で着目すべき要因について説明する．

2.3.1 システム固有の情報蓄積

現状のオープンソース開発は，複数のシステムを組み合わせた開発環境で作業が行われる．これらのシステムは，その目的に応じて，開発に関する情報をシステム内部に蓄積し，開発者に提供する．ところが，各システムから提供された情報は，あくまでもそのシステム固有の情報でしかない．従って，それらの情報が開発者に対して個々に提供された場合には，開発者が必要とする情報が含まれているとは限らない．そのため，開発者の立場から見ると，各システムの情報量が不足していると考えられてしまう．

例えば，多くの開発者は，版管理システムのリポジトリからリビジョン情報を取得し，参照する．ところが，版管理システムのリビジョン情報は，ファイル単位でリビジョンが管理される．このため，ファイル単位での履歴を取得することは容易に可能であるが，その他の視点から履歴を取得することは困難である．例えば，ある開発者がこれまでに行った開発作業の履歴を取得したり，特定の日時に行われた更新作業の履歴を取得することは非常に難しい．

具体的には，以下のような問題が生じる．

CVS リポジトリ内にある，`src/bin/ln/ln.c,v` というファイルについて，ファイル単位でリビジョン情報を取得するのは容易である．例えば，CVS コマンドの一つである `rlog` コマンドを利用することで，図 2 に示すように，ファイル単位でリビジョン情報を簡単に取得可能である．

しかし，リビジョン `1.19` の更新作業者 `sobomax` がこれまでに行った開発作業の履歴を取得する方法や，リビジョン `1.19` の更新日時 `2001/04/26 17:15:57` と同時に行われた更新作業の履歴を取得することは非常に難しい．

2.3.2 システム間の連係不足

オープンソース開発では，開発管理を効率良く行うことを目的として，SourceForge[36] や SourceCast[5]，OSDL(Open Source Development Lab.)[25] 等のサービスが提供されている．これらのサービスでは，電子メールや会議システム等の汎用的な CSCW ツールや，その内容を記録したアーカイブ，WWW，版管理システム等，多くのシステムをまとめて開発者に提供する．しかし，提供される各システムは互いに独立したものであり，単一の環境として何らかの連係を持っているわけではない．すなわち，電子メールで行われた意志疎通の内容はそのシステムの内部にアーカイブ化されて記録されるが，それらの情報は版管理システム CVS の情報と連係を持つことはない．そのため，CVS リポジトリを参照しながら，それに関連した電子メール上での話題を取得したい場合，あるいは，電子メールを参照しながら，それに関連した CVS リポジトリの情報を取得したい場合には，開発者が二つのシステムから個々に情報を取得して，それらを結びつける必要がある．しかし，そのように関連付けを開発者が行う場合のコストは，開発規模が増大するにつれて非常に大きくなる．

具体的には，以下に示すような問題が生じる．

あるオープンソース開発では，版管理システム CVS と，電子メールとそのアーカイブを利用するとする．ある開発者が，CVS リポジトリ内の `src/bin/ln/ln.c,v` というファイルのあるリビジョン `1.19` の情報を参照した際に，それに関連した電子メール上での話題を取得したい場合には，ファイルパスやリビジョン番号，ログメッセージからいくつかのキー

```
% rlog ln.c,v

RCS file: ln.c,v
Working file: ln.c
            .
            .
          (中略)
            .
            .
----------------------------
revision 1.20
date: 2001/04/26 17:22:48;  author: sobomax;  state: Exp;  lines: +1 -1
Previous commit should read:

style(9) Reviewed by:   bde
----------------------------
revision 1.19
date: 2001/04/26 17:15:57;  author: sobomax;  state: Exp;  lines: +28 -9
Bring in ‘-h’ compatability option and its alias ‘-n’ to match NetBSD and GNU
semantics.

style(9) Reviewed by:
Obtained from:  NetBSD
----------------------------
            .
            .
          (以下略)
            .
            .
```

図 2: ファイル単位のリビジョン情報

ワードを抽出して，電子メールの全文検索を行うなどの方法を利用する必要がある．あるいは，ある電子メールを参照しながら，それに関連した CVS リポジトリのリビジョン情報を取得したい場合には，電子メールのサブジェクトや本文からいくつかのキーワードを抽出して，それを手がかりにして CVS リポジトリを検索しなければならない．

2.4 オープンソース開発向けの開発支援環境

我々の研究グループでは，版管理システムを用いた分散ソフトウェア開発の一例であるオープンソース開発を対象として，開発支援環境の構築を目指す [17][20]．この開発支援環境は，オープンソース開発の生産性を高めることを目的として，開発者に対する作業支援を行う．

本節では，オープンソース開発向けの開発支援環境を提案する．

2.4.1 開発支援環境の設計方針

オープンソース開発が実際のソフトウェア開発の手法として取り入れられてから，既に数年が経過した．この数年間で，数多くのソフトウェアが開発されており，それらの膨大なプロダクトが蓄積された．我々の研究では，既に蓄積されたこれらのプロダクトを，今後のオープンソース開発に役立てることを主眼に置いて，研究を行う．

我々が提案する，オープンソース開発向けの開発支援環境の設計方針を以下に示す．

- 既存のシステムをそのまま利用する

  本開発支援環境では，現状のオープンソース開発で用いられる既存のシステムをそのまま利用する．具体的には，CVS や，電子メールとそのアーカイブを現状のまま変更することなく用いる．すなわち，開発者が現状でこれらのシステムによって利用可能である機能を，本開発支援環境を導入することにより失うことはない．

- 既存のシステムが蓄積した情報を有効に利用する

  本開発支援環境では，既存のシステムがこれまでに蓄積した膨大なプロダクトに関する情報を，有効に利用する．現状の CVS リポジトリや電子メールのアーカイブは，情報を蓄積することに主眼が置かれる．その一方で，蓄積された情報は，CVS コマンドや全文検索等，直接的な手段でしか参照できない．しかし，これらの情報の中に含まれる相互関係を抽出することにより，開発者はそれらの情報を結びつけて，有効に利用することができる．

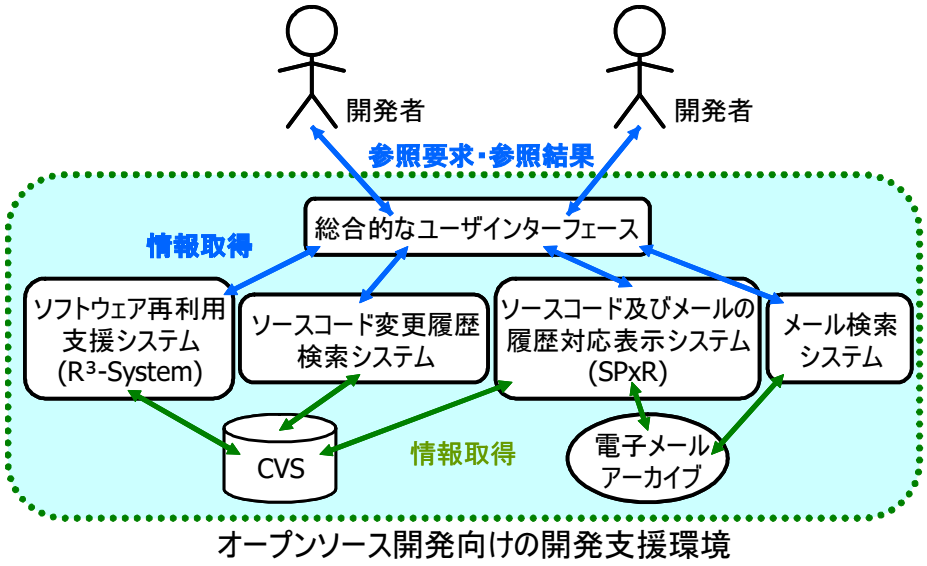

図 3: オープンソース開発向けの開発支援環境の構成

2.4.2 開発支援環境の構成

オープンソース開発向けの開発支援環境の構成を，図 3 に示す.

本開発支援環境は，現状のオープンソース開発で用いられる既存のシステムと，これらのシステムが蓄積したプロダクトを有効に活用するためのシステムから構成される.

これらのシステムについて，以下で簡単に説明する.

- CVS

  CVS は，従来の版管理システムの機能を提供し，ソースコードやドキュメント等のソフトウェアプロダクトリポジトリとしての役割を果たす.

- 電子メール

  電子メールやメーリングリストとそのアーカイブは，開発者間に意志疎通の機能を提供し，その記録を保持する役割を果たす.

- ソフトウェア再利用支援システム (R^3-System)[13]

  ソフトウェア再利用支援システム (R^3-System) は，CVS リポジトリに蓄積された複数のソースコードファイル間の利用関係と類似度を用いて，ソースコードファイルの相対的再利用性 (Relative Reusability) を評価し，順位付け (Ranking) を行う.

  開発者が，過去に開発されたソースコードから再利用可能な部品を検索する場合，検索キーワードによる検索結果を相対的再利用性の順に表示することが可能である.

これにより，ソフトウェアの再利用を行う開発者を支援する．

- プログラム変更履歴検索システム (CoDS)[31]

  ソースコード変更履歴検索システム (CoDS) は，CVS リポジトリに蓄積されたソースコードの変更履歴をデータベース化する．それらの情報を，ソースコード片を用いて検索することが可能である．

  開発者が，手持ちのソースコード片を用いて検索を行うことで，開発者のソースコード修正作業を支援する．

- ソースコード・メールの履歴対応表示システム (SPxR)[18]

  ソースコード・メールの履歴対応表示システム (SPxR) は，CVS のリビジョン情報と電子メールから情報を取得し，管理する．また，それらの情報から統合情報を生成し，管理する．

  これらの情報を開発者に提供することで，開発者の作業を支援する．

- メール検索システム

  メール検索システムは，オープンソース開発における意志疎通の重要な手段である電子メール，メーリングリストのアーカイブから情報を取得し，管理する．それらの情報を用いて，過去にやりとりされたメールの中から，開発者が興味を持つ内容のメールを効率良く検索することができる．

  開発者が，議論に使われたキーワード，ファイルパス，共同議論者等から電子メールの検索を容易に行うことで，開発者間の意志疎通を支援する．

  メーリングリストを使う上で発生する問題の詳細は本稿の第 3 節で，このシステムの詳細については，本稿の第 4 節で説明する．

3　メーリングリストと検索システムの問題点

オープンソース開発では意志疎通の重要な手段としてメーリングリストが用いられる．開発の方向性に関する議論から，実装の詳細に関する議論まで多種多様な議論がメーリングリストでなされる．オープンソース開発に参加する開発者にとって，メーリングリストアーカイブの持つ情報は価値の高いものであり，開発を行う際非常に有用である．一方で，大規模なオープンソース開発では開発者の数も多くメーリングリストアーカイブには大量の情報が蓄積される．その中から必要な情報を取得するには困難が伴う．

本節ではメールに関する基本的な概念を説明し，既存のメーリングリスト検索システムが抱える問題点を指摘する．

3.1　メールとスレッド

メーリングリストを使って議論する際，送信されたメールに返信することで議論を行う．一通のメールに対して複数のメールが返信されるため，メールとそれに返信されるメールは一対多の関係にあると言える．議論で使われたメールに返信関係で線を引き，グラフを作成すると木構造のグラフを得ることができる．

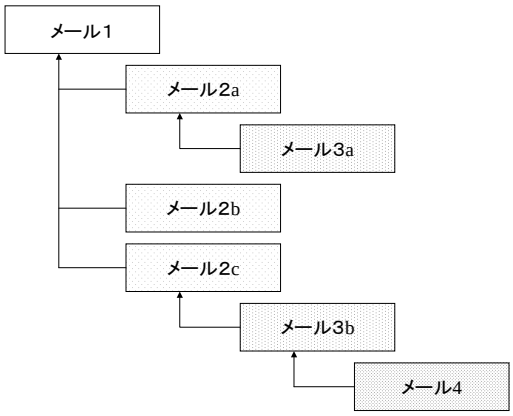

図 4: メールの返信による木構造

図 4 に返信関係を表した木構造のグラフを示す．図中の四角形は一通のメールを表し，矢印は返信先のメールを指す．メール 2a,2b,2c はそれぞれメール 1 への返信メールである．また，メール 3a はメール 2a への返信であり，同様にメール 3b はメール 2c への返信である．これらのメール全体で，木構造を成す．

このグラフに含まれるメールの集合をスレッドといい，スレッドの木構造をスレッド構造という．議論の流れを追うことは返信されたメールを読むということであり，議論の全体を

把握するためにはスレッド構造を把握しなければならない．そのため，スレッド構造を視覚的に表現することは重要である．

メールからスレッド構造を復元することを考える．メールにはヘッダと本文があり，ヘッダには様々な情報が格納される．メールのフォーマットを定める rfc822[6] によると，メールのヘッダに返信元のメールと，関連のあるメールの Message–ID を指定するフィールド（項目）がある．これらの情報を基にスレッド構造を復元する．Message–ID とは，各々のメールに付けられる識別子で，重複する Message–ID は一つも存在しない．

In–Reply–To フィールドには，返信元のメールの Message–ID を複数指定し，References フィールドには，関連のある，すなわち同じスレッドにあるメールの Message–ID を複数指定する．しかし，規格上のことであり，メールソフトによっては実装していない場合もあるし，Message–ID が重複する場合もある．

詳細は 4 章で述べるが，メール検索システムを試作するにあたり，FreeBSD の開発で用いられるメーリングリストを検索対象とした．FreeBSD のメーリングリストでは，Message–ID が重複した場合は，ごく少数であり，無視できる程度であった．返信メールであるのに，In–Reply–To，References フィールドが存在しないメールは多く存在した．この場合，どのスレッドに属するか Subject からしか判断する手段がなく，スレッド構造の正確な復元は不可能である．

In–Reply–To フィールドには返信先メール（親メールと呼ぶ）1 通の Message–ID が書かれたメールが多かった．References フィールドには親メールの Message–ID の他に，スレッドの根となるメールに返信したメールの Message–ID が全て書かれた場合が多かった．また，In–Reply–To フィールドが無く，References ヘッダには親メールの Message–ID が書かれたメールも少なからず存在した．

以上から，メールのスレッド構造を復元するには，In–Reply–To フィールドを使って親メールを求める．In–Reply–To フィールドが存在しない場合は References フィールドで親メールを求める．両フィールドともに存在しない場合は，Subject から属するスレッドを求める．ただし，Message–ID から必ず親が求まるとは限らないので注意が必要である．

3.2 既存のメール検索システムが抱える問題点

現在，メーリングリストのアーカイブを検索できる web サイトは数多く存在する．FreeBSD のメーリングリストを検索できるサイトだけでも，

- The FreeBSD Project, Search (`http://www.freebsd.org/search/`)
- Geocrawler (`http://www.geocrawler.com/lists/3/FreeBSD/`)

などがある．

これらの検索サイトで主に使われている検索手法は「全文検索」と呼ばれるものである．全文検索とは文書内の全ての文字列を，直接検索対象とする検索手法である．全文検索は強力な検索手法であり，検索結果の精度は高い．しかし，メーリングリストの検索に全文検索のみを用いた時，問題が発生する．

前述したとおり，メーリングリストの議論はスレッド単位で展開される．スレッドを把握しないと，議論を把握し必要な情報を得るまでに手間がかかる．その点，全文検索の検索単位はメール一通単位なので，検索結果をそのまま利用する既存の検索システムには問題がある．

既存の検索システムの抱える具体的な問題点を挙げる．

- 検索結果がメール単位で表示されるため，同じスレッドに属するメールが複数表示され，議論の流れを追うことが難しい
- キーワード検索を行う際，キーワードが含まれる文書しか検索できない．そのため，有益な情報が含まれる文書に検索に使ったキーワードが含まれない場合，その文書を取得することができない．

以上の問題点を解決することを目的とした，メール検索システムを設計，試作した．詳細は次節で述べる．

4　オープンソース開発支援用メール検索システム

本研究ではメーリングリストアーカイブから必要な情報を容易に取得するためのメール検索システムを設計，試作した．本節では，試作したメール検索システムについて説明する．

4.1　概要

本メール検索システムでは，全文検索の結果として得られたメールと関連のあるメールを検索することで必要な情報を取得することができる．以下に示す 3 種類の方法で関連のあるメールを検索する．

- メール中に含まれるキーワードによる検索
  メールから抽出したキーワードを用いて，アーカイブ全体を検索する．この手法により対象とするメールと関連のある議論が為されたメールを検索することができる．
- メール中に含まれるファイルパスによる検索
  メール中に書かれてあるファイルパスで，アーカイブ全体を検索する．対象とするメールで議論されたファイルについて検索を行うため，そのファイルで過去，どのような議論がなされたのかを検索することができる．
- 議論者による検索
  メールスレッド中で議論した複数の開発者のメールアドレスでアーカイブ全体を検索する．開発者は特定の分野で主に開発を行うことが多い．活発に議論する複数の開発者は過去，関連のある話題で議論した可能性が高いため，関連のあるメールを検索することができる．

本研究では，オープンソース開発の例として FreeBSD[33] を選び，FreeBSD の開発で用いられるメーリングリストを用いてシステムを開発した．FreeBSD で用いられるメーリングリストには開発に関するものから，初心者向けのものまでおよそ 60 種類存在する．それらのアーカイブは，WWW を利用して，`http://docs.freebsd.org/mail/` で公開されており，その内容を閲覧したり，検索することが可能である．

4.2　システムの設計

本システムは 2 個のプログラムから構成される．

- メールデータベース作成プログラム
  圧縮保存されたメーリングリストのアーカイブを検索可能な形式に展開し, 検索，表示を容易に行うためにメール情報のデータベースを作成する．

- 検索，表示プログラム
  メールを検索するためのインターフェイスである． CGI として実装した．そのため様々なプラットフォームから WEB ブラウザを使ってメールを検索することが可能である．

CGI として実装した，検索，表示プログラムには，検索ページ，結果表示ページ，メール閲覧ページの 3 種類の HTML で記述されたページがある．

検索ページは，キーワードを入力し，検索を開始するページである．結果表示ページは，検索の結果を表示するページである．この時表示される結果は，キーワードが含まれるメール単位ではなく，そのメールが含まれるスレッドを単位として表示される．メール閲覧ページは，メールを閲覧するため，および関連のあるメールスレッドを表示するためのページである．閲覧するために選んでいるメールのほか，メール閲覧ページにはスレッドツリー，閲覧中のスレッドから求めたキーワードでの検索結果，ファイルパスでの検索結果，議論者での検索結果が表示される．

想定する検索の流れを示す．

1. 検索ページを開いて，検索するキーワードを入力し，検索を開始する．
2. 結果表示ページから閲覧するメールスレッドを選ぶ．
3. メール閲覧ページでは，スレッドで一番最初に送信されたメールの内容が表示される．スレッドツリーを使ってスレッド内の他のメールを見ることもでき，キーワードにより検索された他のメールスレッドを見ることもできる．また，キーワードを変えて再検索を行うこともできる．

なお，本システムの全文検索エンジンには Namazu[24] を用いた．Namazu とは，全文検索システムであり，大量の文書を高速に検索することが可能である．Namazu にはコマンドライン版と CGI 版があるが，CGI 版は検索結果を解析してさらなる検索を行うことが困難なため，本研究の用途には適さない．そのため，検索結果のファイルパスを標準出力に出力することができるコマンドライン版を利用した．

4.3 システムの実装

本研究では，オープンソース開発支援用メール検索システムの実装を行った．

開発に用いた言語は Perl で，コードサイズは，全体で約 1200 行となった．内訳としては，メールデータベース作成プログラムが約 300 行，検索，表示プログラムが約 900 行となってた．

開発環境は，以下の通りである．

- CPU : PentiumII 450MHz
- RAM : 256MB
- OS : FreeBSD 4.4-Release
- WEB サーバ : Apache 1.3.23

4.4 メールデータベース作成プログラム

4.4.1 メールの分割

FreeBSD の開発で使われたメーリングリストのアーカイブは様々な手段で入手できるが，本システムでは CVSup というプログラムでアーカイブを入手した．CVSup とはネットワーク経由でファイルの同期を行うプログラムである．

CVSup で入手したメーリングリストのアーカイブは，メーリングリスト毎にディレクトリが作られ，メールが格納される．送信されたメールは 1 週間を単位にして 1 つのファイルにまとめられる．しかし，メーリングリストのアーカイブを検索，表示する際メールは 1 通につき 1 ファイルにという形式で保存した方が効率が良い．そのため，まずメールを分割する必要がある．適当なディレクトリを作成し，その中に分割したメールを保存する．

4.4.2 データベースの作成

検索結果のメールを効率良く解析するために，事前にデータベースを作っておく．具体的には 1 通のメールが与えられた時，そのメールが含まれるスレッドを求めて，スレッドに含まれるメールを一覧表示するためのデータベースを作成する．

本システムではメーリングリストアーカイブ中に現れるメールのスレッドに通し番号をつけて管理する．通し番号を利用して，メールのスレッドを見つけ出すために以下の 2 個のデータベースを作成する．

- メールのパス名からスレッドの通し番号を求める
- スレッドの通し番号からそのスレッドに含まれる全てのメールのファイルパスを求める

また，メールの解析，管理を簡便に行うため，以下のデータベースを作成する．

- メールの Message–ID からメールのファイルパスを求める

各データベースは Perl のハッシュを用いて実装する．tie 関数によりハッシュを NDBM_File に結び付けハッシュのデータをハードディスク上に保存する．ハッシュは Perl のデータ構造

で，大量のデータをハッシュに格納してもわずかな時間でデータを取り出せるという特徴を持つ．そのため，簡単なデータベースを構築することができる．

4.4.3 プログラム

データベースを作成するプログラム create_mail_database.pl について説明する．このプログラムは，上で述べた 2 種類の作業を行う．起動書式は以下の通りである．

```
% perl create_mail_database.pl archive_directory \
                                target_directory
```

引数に，メーリングリストのアーカイブが存在するディレクトリ archive_directory と，分割したメールを保存するディレクトリ target_directory を指定する．target_directory にまだ分割していないメールが archive_directory に存在するとそれらのメールだけを分割する．そのため，データベースを新規作成するのかデータベースを更新するのかに関わらず同じ書式でプログラムを起動する．

また，Namazu は検索を行う際，インデックスファイルというものを使い，検索を高速化する．メールを分割した際インデックスファイルの作成も行う必要がある．インデックスを作成する作業には時間がかかるため，必要なファイルだけ処理しないと効率が悪い．そのためには更新 (update) オプションをつけてインデックス作成コマンドを起動する．その書式は以下のようになる．

```
% mknmz -h archive_directories --update=index_path
```

archive_directories は，分割したメールが保存してあるディレクトリ群であり，index_path はインデックスファイルを保存するディレクトリである．

4.5 検索，表示プログラム

本メール検索システムは，多く使われる検索手法である「キーワードによる全文検索」に加え，検索結果のメールに対して以下の三種類の方法で関連のあるメールを検索する．

- メールから抽出したキーワードによる再検索
- ファイルパスによる検索
- 議論者による検索

以下の章で各々の方法について説明する．

4.5.1 キーワード，ファイルパスの取得法

メール本文中に現れたキーワードで検索するために，メールからキーワードを抽出する作業が必要となる．その作業の手順を以下に示す.

1. メール中の文章を単語単位に分割する
2. 単語の出現回数と，出現した箇所を考慮して重みを計算する
3. 前置詞，助詞などの出現頻度の高い単語は除外する
4. 単語のうち重みの大きいものをキーワードとする

メールを単語単位で分割するのは，キーワードを抽出する際，必須の作業といえる．本システムでは英語の文章が対象のため，スペース，改行などの空白文字および，コンマやコロンなどの記号を区切り文字として単語への分割を行った．日本語等，単語がスペースによって区切られないタイプの言語を解析する際は形態素解析を行う．形態素解析とは，文章を単語に分解し品詞を求める処理である．

キーワードを判定するために,「重み」という量を考える．単語の出現回数を数え，出現した箇所を考慮して重みを計算する．Subject はメールを代表する文章であると考えられるため，Subject から取得したキーワードは大事である．すなわち，本文中に出現した場合より重みは大きいと評価する．

単語の重みを集計する際，単語の大文字と小文字は区別しない．しかし，大文字だけの単語は重みをつける．コンピュータ用語は単語の頭文字をとった言葉が多い．そのため，大文字だけの単語はキーワードの候補と考えることができるためである．

重み (W) は，Subject での出現回数 (A_s)，本文中の出現回数 (A_c) を用いて以下のように書ける．

$$W = \beta(\alpha A_s + A_c)$$

α は Subject から取得した単語に重みをつけるための定数である．本システムでは $\alpha = 2$ とした．β は単語が大文字のみの時，重みを多めに評価するための変数で，単語が大文字のみの時，2，それ以外の時 1 とした．

出現回数を数える方式だと，“a” や，“the” などの前置詞もキーワードとして評価される．前置詞や助動詞はキーワードとして必要ないので除外する．以上のようにして求めた単語をその重みの順に並べてキーワードとする．

また，ファイルパスを求める方法は，単語中に「.」と「/」のファイルパスの構成要素が含まれるとキーワードではなくファイルパスとする．

4.5.2 検索手法

上記の方法でキーワードを求め，そのキーワードを使って，メール閲覧ページで再検索を行う．検索を行う際は，キーワードの上位 3 個で再び検索をする．3 個という数字は経験的に適切な結果が得られた数字で理論的な根拠はない．また，キーワード上位 8 個をページ上に表示して，自由にキーワードを選んで検索できるようにする．

議論者での検索の仕方を説明する．議論者での検索の場合もメール閲覧ページで再検索を行う．閲覧中のメールスレッドに含まれる開発者で，発言回数が多い開発者 2 名が含まれるスレッドを探す．

4.5.3 ユーザーインターフェイス

作成した検索，表示プログラムのユーザーインターフェイスを以下に示す．前述のとおり，本プログラムは検索ページ，結果表示ページ，メール閲覧ページの 3 種類からなる．

検索ページを図 5 に示す．キーワードを入力し，検索を開始するページである．また，検索を実行するメーリングリストと，結果として表示する最大の数を指定する．

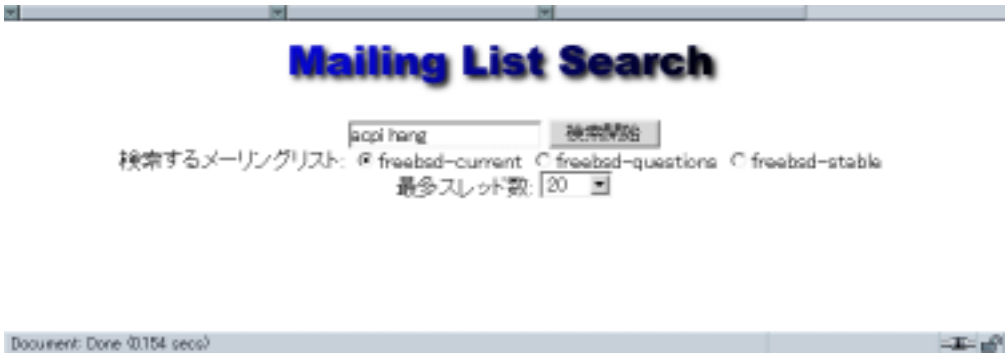

図 5: 検索ページ

結果表示ページを図 6 に示す．例として,「large size MFS」という 3 個の単語を検索に使用し，freebsd–current メーリングリストを検索した．それぞれのスレッド毎に Subject，最初の送信日時およびスレッドに含まれるメールの数を表示する．結果表示ページでは検索結果をメール単位ではなく，スレッド単位で表示し，1 ページにより多くの情報を記載することで効率の良いメール検索，閲覧を可能にする．

メール閲覧ページを図 7，図 8，図 9，および図 10 に示す．各図は一つの HTML ページの一部分であるが紙面の都合上分割してある．

図 7 はメール表示部である．メール表示部でメールを閲覧する．図 8 はスレッド表示部である．スレッドに含まれるメールを木構造で表示する．閲覧中のメールに返信したメール

図 6: 結果表示ページ

や，同じスレッド中のメールを読むにはスレッド表示部のリンクをからたどって読む．図 9 は， キーワードによる検索部である．スレッド構成するメールから抽出したキーワードと，重みの大きいキーワード 3 個で再検索した結果を表示する．この部分にリストアップされたスレッドは，閲覧中のスレッドと関連のあると判定されたスレッドである．また，検索に用いるキーワードを変更して再検索を行うこともできる．図 10 はファイルパスおよび議論者による検索部である．スレッドを構成するメールから抽出したファイルパスの一覧と，出現回数の多いファイルパス 3 個で検索した結果を表示する．また，スレッドでの発言回数が多い 2 名のメールアドレスで検索した結果を表示する．他のメールアドレスを加えた検索を実行することもできる．

図 7 から，図 10 に示したメール閲覧ページが本研究の一番の成果である．閲覧中のスレッドと関連のあるスレッドを検索，閲覧できる．

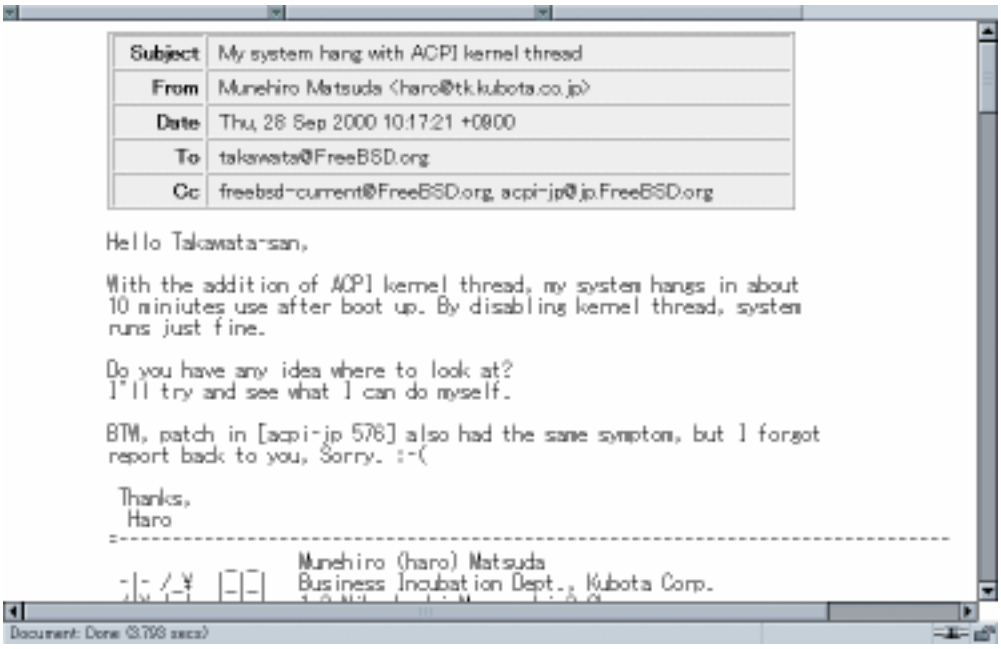

図 7: メール閲覧ページ – メール表示部

図 8: メール閲覧ページ – スレッド表示部

図 9: メール閲覧ページ – キーワードによる検索部

図 10: メール閲覧ページ – ファイルパスおよび議論者による検索部

4.6 評価

4.6.1 メールデータベース作成にかかる時間

メーリングリストのアーカイブを展開し，データベースを作成する時間が実用的なものかどうかを評価した．対象としたのは freebsd–current，freebsd–stable，freebsd–questions の 3 個のメーリングリストである．

これまでに蓄積されたアーカイブを全て処理するのにかかる時間，また，1 年分および，1 カ月分のメールをデータベースに追加するのにかかる処理時間を調べた．

表 1: freebsd–current

期間	メール数	スレッド数	総ファイルサイズ	時間
全て	88,109	18,455	259MB	9m56s
1 年	11,054	2,340	38MB	1m31s
1 カ月	1,403	278	4.9MB	11s

表 2: freebsd–stable

期間	メール数	スレッド数	総ファイルサイズ	時間
全て	56,914	13,604	174MB	6m01s
1 年	19,842	4,457	66MB	2m27s
1 カ月	1,708	379	6.0MB	13s

表 3: freebsd–questions

期間	メール数	スレッド数	総ファイルサイズ	時間
全て	278,642	107,881	800MB	43m45s
1 年	60,381	20,965	205MB	10m51s
1 カ月	5,626	1,954	19MB	1m03s

表 1 に freebsd–current，表 2 に freebsd–stable，表 3 に freebsd–questions の処理にかかった時間を示す．1 年とは 2001 年のことであり，1 カ月とは 2002 年 1 月 20 日から 2002 年 2 月 16 日までの期間である．また，各データは 2002 年 2 月 19 日時点でのものである．

メールを処理する時間を見ると，全てのアーカイブを処理するにはメールの数にも依存するが，数十分かかる．1 年分のメールを処理するには数分，もっとも頻繁に行われるであろ

う，1 カ月分のメールを追加処理するのにかかる時間は数秒であった．実際には 1 週間単位でメールの追加を行うことができるが，その時間はさらに短いものと予想される．

次に，Namazu のインデックス作成にかかる時間を調べた．freebsd–current メーリングリストを対象とした．

表 4: Namazu によるインデックスの作成時間

	全て	1 年	1 カ月
かかった時間	4h28m	1h4m	9m37s

表 4 に Namazu による検索用インデックス作成時間を示す．期間は前述した期間と同様である．

検索を行うための前準備としてメールの展開，およびデータベース作成かかる時間と Namazu のための検索用インデックス作成時間がかかるが，以上の結果から十分実用的であると言える．

4.7 再現率，適合率及び，f 値の測定

既存の全文検索システムと試作したシステムで再現率，適合率を求め，f 値を計算し比較した．

まず，再現率，および適合率について説明する．

- 再現率 – 必要な情報のうち，実際に検索された情報の割合
- 適合率 – 実際に検索された情報のうち，必要な情報の割合

再現率，適合率は上記のように定義され，両方ともに高いほど優れた検索システムであると判断できる．

また，f 値とは，

$$f\ 値 = \frac{2 \times 再現率 \times 適合率}{再現率 + 適合率}$$

と定義され，情報検索の精度を表す指標として用いられる．f 値が大きいほど，情報検索の精度は高いとする．

行なった実験の概要について説明する．

freebsd–stable メーリングリストで 2002 年 1 月にやりとりされたメールの中から,「FreeBSD 4.5–RELEASE に関する問題」を探すという課題を設定し再現率，適合率を求めた．freebsd–stable メーリングリストで 2002 年 1 月にやりとりされたメールの総数は 1263 通であり，302 個のスレッドが存在した．そのうち,「FreeBSD 4.5–RELEASE に関する問題」について論じ

られたスレッドは 60 個存在した．既存の全文検索システムと，試作したメール検索システムの両方について検索を行ない，検索結果のスレッドが「FreeBSD 4.5–RELEASE に関する問題」について論じられたか否かに分類し，再現率，適合率を計算した．

実験では，異なるキーワードを用いて 3 回検索を行ない，再現率，適合率 3 回分の平均値を求めた．既存の全文検索システムには，Namazu のみからなる検索システムを利用した．

実験結果を表 5 に示す．

表 5: 既存システムとの比較実験

	再現率	適合率	f 値
Namazu	15%	50%	0.23
試作したシステム	26%	40%	0.32

実験の結果，f 値が大きくなり，検索の精度が上がったことが示された．また，再現率は 11%高くなり，適合率は 10%低くなった．再現率が高くなるのは，検索に使われたキーワードが含まれないメールも検索できるためであり，本検索システムの有効性が示された．また，適合率が 10%低くなることに関しては，メールからキーワードを抽出することに失敗し，関連のないメールも検索したためと考えられる．

以上から本検索システムの有効性を示せたものの，キーワード抽出の精度の向上という課題が残った．

5　まとめ

オープンソース開発を研究対象として，まず，現状のオープンソース開発環境における二つの問題点を指摘した．各システムがそれぞれ固有の形式で開発履歴情報を蓄積する「システム固有の情報蓄積」の問題と，各システムが互いに連係を持たず，開発情報が分散して存在する「システム間の連係不足」の問題により，開発者が大きな負担を強いられる．そこでこれらの問題点を解決し，数年間のオープンソース開発で蓄積された膨大なソフトウェアプロダクトを開発者が有効に利用することを目的とした，オープンソース開発向けの開発支援環境を提案した．

本研究では，オープンソース開発で，意志疎通の重要な手段として用いられるメーリングリストに注目した．既存のメーリングリストアーカイブの検索システムには「メール単位での検索手法のみ」,「全文検索のみを行うため，取得できる有益なメールの数に限りがある」という問題点が存在することを指摘した．そこで，オープンソース開発向けの開発支援環境構成要素の一つとして，これらの問題点を解決し，開発者に有益な情報を多く提供することを目的とした，オープンソース開発支援用メール検索システムを設計し，その実装を行った．

また，実際のオープンソース開発で用いられたデータを用いて，本システムの適用実験を行った．その結果，本システムは大規模なオープンソース開発に適用した場合でも実時間で動作することを確認した．さらに，本システムを用いることにより，開発者が必要とする情報を提供することが可能となることを確認した．

現時点では，スレッドから抽出したキーワードの精度が低く，不必要な情報も取得するという問題点がある．そこで，キーワード抽出の精度を高め，スレッド間の結び付きを精度良く検索することを目指す．

謝辞

本論文を作成するにあたり，常に適切な御指導を賜わりました大阪大学大学院基礎工学研究科情報数理系専攻 井上 克郎 教授に心より深く感謝致します.

本論文の作成にあたり，適切な御指導および御助言を頂きました大阪大学大学院基礎工学研究科情報数理系専攻 楠本 真二 助教授に深く感謝致します.

本論文の作成にあたり，適切な御指導および御助言を頂きました大阪大学大学院基礎工学研究科情報数理系専攻 松下 誠 助手に深く感謝致します.

本論文の作成において，適切な御助言を頂きました大阪大学大学院基礎工学研究科情報数理系専攻 石川 武志 氏に深く感謝致します.

最後に，その他様々な御指導，御助言等を頂いた大阪大学大学院基礎工学研究科情報数理系専攻井上研究室の皆様に深く感謝致します.

参考文献

[1] The Apache Software Foundation, Apache Projects, http://www.apache.org/.

[2] Ulf Asklund, Boris Magnusson, and Annita Persson, “Experiences: Distributed Development and Software Configuration Management”, In Proceedings of SCM-9, Ninth International Symposium on System Configuration Management, J. Estublier (Ed.), LNCS1675, pp.17–33, Toulouse, France, 1999.

[3] Wayne A. Babich, “Software Configuration Management”, Addison-Wesley, Reading, Massachusetts, 1986.

[4] Brian Berliner, “CVS II:Parallelizing Software Development”, In USENIX Association, editor, Proceedings of the Winter 1990 USENIX Conference, pages 341–352, Berkeley, CA, USA, 1990.

[5] Collab. Net, Inc., SourceCast, http://www.collab.net/products/sourcecast/.

[6] David H. Crocker, “Standard for the Format of Arpa Internet Text Messages.” RFC 822. Internet Engineering Task Force, Aug. 1982.

[7] CVSWeb, http://stud.fh-heilbronn.de/˜zeller/cgi/cvsweb.cgi/.

[8] Jacky Estublier, “Software Configuration Management: A Roadmap”. The Future of Software Engineering in 22nd ICSE, pp.281–289, 2000.

[9] Jacky Estublier and Rubby Casallas, “The Adele Configuration Manager”, In Walter Tichy, editor, Configuration Management, pp.99–133, John Wiley and Sons, Ltd., Baffins Lane, Chichester, West Sussex PO19 1UD, England, 1994.

[10] Peter H. Feiler, “Configuration Management Models in Commercial Environments”, CMU/SEI-91-TR-7 ESD-9-TR-7, March, 1991.

[11] Karl Fogel, “Open Source Development with CVS”, The Coriolis Group, 2000.

[12] Free Software Foundation, Inc., The GNU Project, http://www.gnu.org/.

[13] 藤原晃, “類似度を用いたプログラムの再利用性尺度の提案と実現”, 大阪大学大学院基礎工学研究科修士学位論文, 2002.

[14] Peter Fröhlich and Woflgang Nejdl, “WebRC Configuration Management for a Cooperation Tool”, SCM-7, LNCS 1235, pp.175–185, 1997.

[15] James J. Hunt, Frank Lamers, Jürgen Reuter, and Walter F. Tichy, “Distributed Configuration Management via Java and the World Wide Web”, ICSE’97, SCM-7, LNCS1235, pp.161–174, 1997.

[16] 鯉江英隆, 西本卓也, 馬場肇, “バージョン管理システム (CVS) の導入と活用”, SOFT BANK, December, 2000

[17] 石川武志, 山本哲男, 松下誠, 井上克郎, “ソフトウェア開発時における版管理システムを利用したコミュニケーション支援システムの提案”, 情報処理学会研究報告, 2001-SE-133, Vol.2001, No.92, pp.23–30, 2001.

[18] 石川武志, “オープンソース開発支援のためのソースコードおよびメールの履歴対応表示システム”, 大阪大学基礎工学部情報科学科修士学位論文, 2002.

[19] Linux Online Inc., The Linux Home Page, http://www.linux.org/.

[20] 松下誠, 井上克郎, “自由な開発形態を支援するソフトウェア開発環境”, ソフトウェアシンポジウム 2000 論文集, pp.236–242, 2000.

[21] Merant, Inc., PVCS Home Page, http://www.merant.com/pvcs/.

[22] Microsoft Corporation, Microsoft Visual SourceSafe, http://msdn.microsoft.com/ssafe/.

[23] Bartosz Milewski, “Distributed Source Control System”, SCM-7, LNCS 1235, pp98–107, 1997.

[24] Namazu Project, 全文検索システム Namazu, http://www.namazu.org

[25] Open Source Development Lab, Inc., Open Source Development Lab, http://www.osdlab.org/.

[26] 落水浩一郎, “分散共同ソフトウェア開発に対するソフトウェアプロセスモデルに関する基礎考察”, 電子情報通信学会技術研究報告, SS2000-48(2001-01), pp.49–56, 2001.

[27] Koichiro Ochimizu, Hiroyuki Murakoshi, Kazuhiro Fujieda, and Mitsunori Fujita, “Sharing Instability of a Distributed Cooperative Work”, ISPSE2000, pp.33–42, 2000.

[28] 大月美佳, “入門 CVS Concurrent Versions System”, SHUWA SYSTEM Co., Ltd, 2001

[29] Rational Software Corporation, Software configuration management and effective team development with Rational ClearCase, http://www.rational.com/products/clearcase/.

[30] Eric S. Raymond, “The Cathedral & the Bazaar”, O’REILLY, 1999.

[31] 田原靖太, “既存ソフトウェアの変更履歴を利用したソースコード修正支援システム”, 大阪大学大学院基礎工学研究科修士学位論文, 2002.

[32] 寺口正義, 松下誠, 井上克郎, “バージョン間の差分を利用したデバグ手法の提案”, 電子情報通信学会技術研究報告, SS99-52, pp.17–24, 2000.

[33] The FreeBSD Project, The FreeBSD Project, http://www.freebsd.org/.

[34] The Mozilla Organization, bonsai, http://www.mozilla.org/bonsai.html.

[35] Walter F. Tichy, “RCS - A System for Version Control”, SOFTWARE - PRACTICE AND EXPERIENCE, VOL.15(7), pp.637–654, 1985.

[36] VA Linux Systems, Inc., SourceForge, http://sourceforge.net/.