社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
SS2003-9 (2003-07)

リビジョン情報と電子メールを用いたオープンソース開発向き情報検索システム

佐々木　啓† 　松下　誠† 　井上　克郎†

† 大阪大学大学院情報科学研究科コンピュータサイエンス専攻　〒560-8531 大阪府豊中市待兼山町1-3
E-mail: †{k-sasaki,matusita,inoue}@ist.osaka-u.ac.jp

あらまし　オープンソースソフトウェアの開発を行う際には，リビジョン管理システムや，電子メールを用いたメーリングリストシステムが用いられる．これらのシステムの履歴情報には将来の開発に活用することのできる情報が多く含まれている．しかし，それらの情報は膨大なものであるため，開発者が必要とする情報を的確に取得することは容易ではない．そこで，本研究では，開発者が必要とする情報の抽出とそれに関連するデータの表示を行うシステムの試作を行った．さらに，実際のオープンソース開発で用いられた版管理システムの開発履歴情報とメーリングリストアーカイブを用いて，本システムの評価を行った．その結果，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が確認できた．

キーワード　オープンソースソフトウェア開発，リビジョン管理，ソースコード検索

E-mail and Source Code Revision Information Retrieval System for Open Source Software Development

Kei SASAKI†, Makoto MATSUSHITA†, and Katsuro INOUE†

† Graduate School of Information Science and Technology, Osaka University
1-3, Machikaneyama-cho, Toyonaka-shi, Osaka 560-8531, Japan
E-mail: †{k-sasaki,matusita,inoue}@ist.osaka-u.ac.jp

Abstract　In the case of open sourse software development, developers use Repository control system. and mailing list system These system store development histories of the products. Divelopers can obtain a deeper understanding about former development by reviewing an archive in the case of software development, and it is expectable that they help developers. But, it is not easy to retrieve the information from the stored information that developers require. In this research, I establish software development supporting system for searching development history information. Also, I applied this system to real data in open source development for applicable experimentation. As a result, I confirmed this system enables developers to retrieve more useful information, and solve recent problems.

Key words　Open Source Software Development, Revision Control, Source Code Searching

1. はじめに

オープンソースソフトウェアの開発規模の増大に伴い，その開発形態は多人数化，分散化している．大規模なオープンソースソフトウェア開発では，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．また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ともない，分散した多くの開発者が異なる場所で開発作業を行うことも多い．このよう複雑化しているソフトウェアシステムを効率よく管理するため，近年のオープンソースソフトウェア開発では，リビジョン管理システムや電子メールを用いたメーリングリストシステムを用いることが多くなっている．リビジョン管理システムは，プロダクトの開発履歴をリポジトリと呼ばれるデータベースに格納して管理する．メーリングリストでは，開発者相互の意志疎通や進捗状況の報告などが行われる．これらのシステムでは開発者が行ったプロダクトへの変更履歴や送信した電子メールは全て個別のアーカイブとして保存されており，その履歴の中には，将来の開発に活用することのできる情報が多く蓄積されている．ソフトウェア再利用の際にこれらのアーカイブを閲覧することによって，以前の開発についてより深い理解が得られ開発の手助けになる [1]．このように，オープンソースソフトウェア開発を行う上で，過

去に開発されたコードを参照することや再利用することは効率的な手法であると言える．

しかし，蓄積された膨大な情報の中から，開発者が必要とする情報を的確に取得することは容易ではない．必要とするプログラムは複数のファイルやリポジトリに分かれて蓄積されている可能性もある．このため，開発者は参照すべきリポジトリやファイルの履歴を見れば良いかを知ることは困難となる．

本研究ではこの問題を解決することができるソフトウェア開発支援環境の構築を行う．具体的には，先に私の所属する研究グループにおいて作成されたソースコード修正支援システム (CoDS) [7] とオープンソース開発支援のためのソースコード及びメールの履歴対応表示システム (SPxR) [5] を統合することにより，開発者が必要とする情報の抽出とそれに関連するデータの表示を行うシステムの試作を行った．本システムは特定の情報を用いることによって，該当するファイルや電子メール情報とそれに関連する情報を検索することができるシステムであり，それらの情報を提供することでオープンソースソフトウェア開発の支援を行う．

また，実際のオープンソースソフトウェア開発で用いられたリビジョン管理システムとメーリングリストアーカイブを用いて，本システムの評価を行った．その結果，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が確認できた．

以上のことから本システムを用いたオープンソースソフトウェア開発支援環境を構築することで，効率よく開発作業が実現可能となり，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．

2. 節では，システムの説明を行う．3. 節で実装について述べる．4. 節では，そのシステムに対して検証を行う．最後に，5. 節で本研究のまとめと今後の課題について述べる．

2. リビジョン情報と電子メールの検索システム

本研究では，版管理システム CVS を用いたオープンソース開発を対象として，CVS の履歴情報とメーリングリストから開発者が必要とする情報を容易に取得することの出来る開発開発支援環境の構築を目指している．本システムを用いて，特定のソースコードや開発履歴情報から，それらに関連する情報を入手することが出来れば，開発にかかるコストを軽くすることが出来ると考える．本節では，リビジョン情報と電子メールの開発履歴情報検索システムについて説明する．

2.1 システムの機能と構成

本システムでは，以下の方法で開発者が必要とする開発情報とその関連情報を検索することができる．

（1） ソースコードを用いた検索

開発者が手持ちのソースコード片を入力することで，類似するソースコードを検索し，該当する CVS 情報を表示する．

（2） ファイル・ディレクトリ名検索

検索対象となるリポジトリを選択した後で，該当するファイル・ディレクトリを検索する

（3） キーワード検索

開発者が要求したキーワードを開発履歴データベースで検索し，キーを持つデータがあれば全て表示する．

（4） ファイルパス・電子メールリストを用いた検索

開発者が，視覚的に開発履歴情報の構造を見ることで検索する．

（5） 関連情報検索

キーワード検索と同様で，開発者が要求した関連情報（開発者・更新日時など）を使ってデータを検索する．

本システムは，3 つの部分から構成される．(図 1 参照)

- **類似コード検索部 (CoDS 部)**
- **開発履歴データベース管理部 (SPxR 部)**
- **データ表示部**

類似コード検索部では，(1) の機能を実現し，開発履歴データベース管理部では残りの (2)(3)(4)(5) の機能を実現する．データ表示部は，上の二つの部分と開発者を結ぶインターフェイスである．

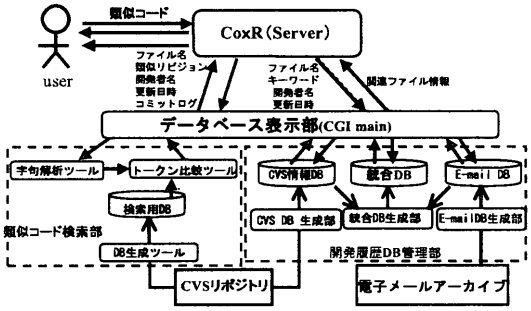

図 1 システム構成図

2.2 類似コード検索部 (CoDS 部)

この部分は，類似コード検索 DB，データベース作成ツール，字句解析ツール，トークン比較ツールの 4 部分からなる．利用者はあらかじめ版管理システムのリビジョン情報から類似コード検索 DB を作成しておく．作成後，入力したソースコード片は字句解析をすることでトークンに分解する．類似コード検索 DB 中の差分情報との類似度を計測し，一定以上の類似度を持つソースコードの情報をデータ表示部へ渡す処理を行う．

2.2.1 字句解析

字句解析では，ソースコード片をプログラミング言語の文法に従って，整数で表されたトークン列に変換する．ソースコード中の空白とコメントは無視する．さらに，ユーザ定義の変数，関数，型等は名前が異なっていても等価なトークンとみなす．言語仕様やライブラリ等で定義されている手続き・関数等の名前についてはそれぞれ別のトークンに変換する．これらのトークンと予約語を合わせて，本論文では「キートークン」と呼ぶ．出力の際には，整数で表されたトークンの番号に加え，そのトークンが出現する行番号も付加する．当該ソースコード片に出現するキートークンの行番号もこの部分に付加する．

2.2.2 トークン比較

ソースコードの比較部分では，上記の字句解析によって変換されたトークンを 1 つの文字とみなし，文字列の一致問題を解

くことで，類似コードの検索を行う．

その際に，CoDS [7] 同様，類似部分を抽出する系列比較アルゴリズムとして，"**局所アラインメント** [4]" を用いている．局所アラインメントは，与えられた 2 つの系列の中のそれぞれの部分系列のうち等価で最長のものを 1 つ求める問題である．等価な部分系列は，完全に一致している必要はなく，異なる要素が存在してもよい．

2.2.3 検索効率の向上

局所アラインメントは時間計算量が 2 文字列の長さの積のオーダーと大きく，膨大な回数の比較を行う場合，実用的な時間で検索が行われない．そこで，トークンの比較を時間効率よく行うため，利用者は，入力としたソースコードの中で重要と思うキートークンを「特性キートークン」として指定する．指定がない場合は，利用者から入力されたソースコード片の中の全てのキートークンを特性キートークンとする．なお，キートークンが含まれていないソースコード片については，アラインメントを求める対象としない．

2.3 開発履歴データベース管理部 (SPxR 部)

この部分は，CVS 情報管理部，電子メール情報管理部，統合情報管理部の 3 つの部分に分けることができる．それぞれの部分はデータベース (DB) とデータベース作成ツールからなる．利用する際には，あらかじめ版管理システム CVS と，電子メールのアーカイブから情報を取得しそれぞれデータベース化する．また，それらの統合情報を抽出して統合用データベースを作成し管理する．データ表示部で検索する際にそ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．

2.4 データ表示部

この部分は開発者から検索要求を受けると履歴データベースと連結し，要求されたデータを表示する．

2.4.1 類似コード検索結果表示

検索結果を表示するのに必要な情報は，以下の通り．

（1） ファイル名とリビジョン番号の組 (F, p, q)

（2） q のコミット日時・更新者・ログメッセージ

（3） 入力ソースコード I と類似している，リビジョン p における部分トークン列

2.4.2 CVS・電子メールの開発履歴情報表示

この部分は開発者が，直接 CVS リポジトリのディレクトリ構造と電子メールのリスト一覧を見て検索することが出来る．開発者が何らかの CVS 情報・電子メール情報 I_1 をキーとして与えると，SPxR 部の CVS 情報 DB・電子メール情報 DB 内で I_1 と一致する情報を検索する．該当するものがあればその一覧を表示する．その際，統合情報 DB からそれに関連する開発情報 I_1' があれば，それも表示する．表示された情報をキー I_2 として，さらに必要な情報を絞り込むことも可能である．

2.5 データベース

本システムは 4 つのデータベースを用いて情報の検索を行う．

- **類似コード検索データベース**
- **CVS 情報データベース**
- **電子メール情報データベース**
- **統合用データベース**

2.5.1 類似コード検索データベース

類似コード検索 DB は，開発者が指定したリポジトリ内にあるすべての C 言語のソースファイルから以下の情報を収集する．

（1） **ファイル名** F

ファイルが一意に特定できるように，リポジトリ内のソースファイル名を，フルパスで取得する．

（2） **リビジョン番号の対** p,q

1 のどのリビジョン間における変更の情報であるかを特定するための情報である．p の次のリビジョンが q である．

（3） q **のコミット日時**

リビジョン q のコミット日時の情報である．

（4） q **のログメッセージ**

コミットログは，直前のリビジョンからの差分を自然語で記述したものであり，検索結果が目的としているものであるかどうかを利用者が判断することができる．

（5） **リビジョン** p **において，次のリビジョン** q **までに変更された部分（前後の数行を含む））に関する差分情報** $c(F, p, q)$

ソースコード片を検索する際，入力されたソースコード片 I とデータベースのこの部分をトークン単位で比較する．その検索の効率化のため，c を字句解析（2.2.1 節参照）し，トークン列に変換したものを格納する．

2.5.2 CVS 情報データベース

CVS リポジトリからリビジョン情報を取得し，CVS 情報データベースを作成する． CVS の固有情報は RCS 形式のリビジョン情報を解析することで取得する．

データベース内のそれぞれのデータ構造は図 2 の通り．

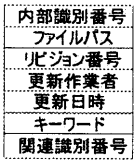

図 2 CVS 情報 DB 構成

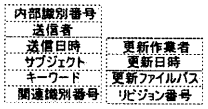

図 3 mail 情報データベース構成図

内部識別番号は，データベース作成時に各レコードごとに割り与える個別の識別番号．**ファイルパス**，**リビジョン番号**は，CVS 情報の中から特定の情報を指定するために用いる情報である．**更新作業者**，**更新日時**は，CVS のファイルのコミットした更新者とその日時である．**キーワード**は，リビジョン情報に含まれているログメッセージを解析し，出現頻度が高いファイルパスを取得する．**関連情報識別子**は，CVS 情報の中で「同一時間に更新された」「同一更新者に更新された」「キーワードが同じ」などの共通点がある場合，それらの内部識別番号を関連識別番号として登録する．

2.5.3 電子メール情報データベース

電子メールアーカイブに蓄積された電子メールを解析し，必要な情報を抽出していく．ここで対象になる電子メールは以下の 2 種類に分けられる．

- コミットメール

CVS リポジトリのリビジョンが更新された際に，CVS から送信されるメール．この中には更新されたリビジョンの更新情報

が記述されている．

- **議論メール**

コミットメール以外の全ての電子メール．開発作業に関連する話題が記述されているため，開発作業の内容理解に役立つ．

データベース内のそれぞれのデータ構造は図 3 の通り．

内部識別番号は，データベース作成時に各レコードごとに個別に割り当てられる識別番号．**送信者**，**送信日時**は，電子メールの送信者名と送信日時である．**サブジェクト**は，ファイルの概要を表すもので，容易にメールの内容をチェックできる情報．**キーワード**は，電子メールアーカイブ内のログメッセージを解析し，出現頻度が高いファイルパスや単語を取得する．**関連情報識別子**は，コミットメールに対する返信メールなど，ある電子メールと関連のある電子メールに対してそれらの内部情報識別子をお互いに関連情報識別子として取得する．**更新作業者**，**更新日時**，**更新ファイルパス・リビジョン番号**はコミットメールにのみ含まれる情報で，それぞれ CVS 情報と関連を調べる際に用いる．

2.5.4 統合情報データベース

上記の CVS 情報 DB と電子メール情報 DB の情報の情報をそれぞれのデータベースから取り出して比較し，互いに関連性があればその情報を統合し，統合用データベースに格納する．

- **内部識別番号**

電子メール (CVS) 情報 DB の内部識別番号．関連する CVS(電子メール) 情報があればその内部識別番号を登録する．

- **関連識別番号**

CVS(電子メール) 情報 DB の内部識別番号．関連する電子メール (CVS) 情報があればその内部識別番号を登録する．

電子メール情報と CVS 情報の関連性は以下の判断基準をもとに判定するようにした．

（1） 「更新日時」が一致する

（2） 「ファイルパス」と「リビジョン番号」が一致する

（3） 「キーワード」が一致する

この DB は CVS と電子メールの関連を知るためだけに用いられ，詳しい情報はそれぞれの DB から抽出する．

3. システムの実装

これまで述べた手法に基づいてシステムの実装を行った．

3.1 類似コード検索部

類似コード検索部は，C 言語で書かれたソースコードを対象とし，版管理システムとして CVS を用いていることを前提としている．また，内部で RCS のコマンドを呼び出しているため，RCS がインストールされている必要がある．

データベース作成ツール (getdb.pl)

指定されたディレクトリ以下にある C 言語のソースファイルの履歴ファイルに関してデータベース作成の処理を行う．また，Perl でデータベースを扱うルーチンは，ハッシュとファイルを結びつけることができるように GNU GDBM 1.8.0 を用いる．C 言語で実装を行った．

字句解析ツール (scanner.c)

C 言語で記述されたソースコードを入力とし，上で述べた仕様に基づいて，トークンのリストを出力する．本字句解析ツールは C 言語を用いて実装を行った．

トークン比較ツール (mathcing.c,alignment.c)

入力された 2 つのトークン列 S, T の間でアラインメントを行い，最大のスコアと，マッチしている部分の（トークン列 T の元のソースコード片における）行番号の範囲を出力する．C 言語で実装している．

3.2 開発履歴データベース管理部

ツールはすべて Perl 言語で実装し，DB は，PostgreSQL と，BerkeleyDB を利用した．設定ファイル (.commsysconf) は読み出し開始するルートディレクトリなどを管理する．

- CVS 情報生成ツール (cvsinfogen3.pl)

ルートディレクトリから，CVS リポジトリ内のディレクトリ構造を再帰的に辿って，各 RCS ファイルの解析を行い，リビジョン情報を取得する．その情報を CVS 情報 DB に登録する．

- CVS 情報 DB

開発者から検索キーを受けると，ハッシュデータベースの中でそれと一致する内部識別番号の集合を探す．その内部識別番号をもとに PostgreSQL から各レコードが持つ情報を表示する．

- 電子メール情報生成ツール (mailinfogen3.pl)

ルートディレクトリから，電子メールアーカイブのディレクトリ構造を再帰的に辿り，解析を行い，電子メール情報を生成して，電子メール情報データベースに登録する．

- 電子メール情報 DB

CVS 情報 DB と同様．

- 統合情報生成ツール (integinfogen3.pl)

CVS，電子メール情報 DB から情報を取得して，両者のデータを比較する．相互に関連を持つ情報であれば，両者を結合して統合情報を生成し，統合情報 DB に登録する．

- 統合情報 DB

CVS 情報 DB と電子メール情報 DB の関連する一方の内部識別番号を検索を行うと，それをもとに関連する CVS 情報 DB・電子メール情報 DB の内部識別番号を返す．

3.3 データ検索・表示部

データ検索・表示部では開発者からの検索要求を受けて，類似コード検索部・開発履歴データベース管理部に検索を依頼し，開発者が必要とする情報を参照する．このシステムの実装は全て Perl 言語で行った．

4. 適用実験

本節では実際にシステムを動作し，結果について考察する．

4.1 実験対象

本システムの適用実験を行うために，実際のオープンソース開発の中からオープンソフトウェアの代表例である FreeBSD [9] の開発をモデルとして仮想的なオープン開発環境を用意する．

- CVS リポジトリ

FreeBSD の CVS リポジトリを複製した．ファイル総数 51379，リビジョン総数 511054 を対象とする．

- 電子メール

2001 年の FreeBSD のメーリングリストアーカイブを複製した．

49736 件の電子メールを対象とする.

4.2 実験の概要

本システムの評価実験として以下の 2 つの実験を行った.

- 検索実例に基づいたデータ検索

利用者が FreeBSD 開発を行っていると想定して，過去の修正事例をもとに類似コードとその関連情報を検索を行う.

- 検索速度評価

CoDS では検索に一つのキートークンしか用いなかったが，本システムは複数のキートークンを用いることで類似度を測定する範囲を絞り込むことができる．この部分について評価する.

4.3 検索実例に基づいたデータ検索

今，利用者は OpenSSH のパスワードの送信を行うソースコードの開発を行っているとする．このコードはパスワードをソケットで送信する際にパスワードの長さもデータとして送信してしまう．そのため，これを用いたパスワード攻撃が行われる可能性が指摘されている．利用者はその問題点を解消するためにパケット送信の際に長さを隠蔽子して送信している利用例を検索する. なお，入力として与えるソースコードは以下の通り.

```
if (i != 0)
    error("Permission denied, please try again.");
password = read_passphrase(prompt, 0);
packet_start(SSH_CMSG_AUTH_PASSWORD);
packet_put_string(password, strlen(password));
memset(password, 0, strlen(password));
xfree(password);    ;; ;;
packet_send();
packet_write_wait();
```

図 4 入力ソースコード

上のソースコードを用いて類似コードの特定を行い．その関連情報を参照することでバグ修正に役立てるものかどうかを検証する．類似したソースコードは複数存在した．(図 7 参照)

その中で複数の類似コードが検索された検索結果の一つとしてファイル sshconection1.c-リビジョン 1.8(更新者：green) で図 5，図 6 のような類似コードが検索された.

```
Make password attacks based on traffic
analysis harder by requiring that*non-
echoed* characters are still echoed
back in a null packet, as well as pad
passwords sent to not give hints to the
 length otherwise.
```

図 5 コミットログ

```
error("Permission denied, please try again.");
password = read_passphrase(prompt, 0);
packet_start(SSH_CMSG_AUTH_PASSWORD);
 ssh_put_password(response);
* memset(password, 0, strlen(password));
 xfree(password);    ;; ;;
 packet_send();
 packet_write_wait();
```

図 6 類似コード

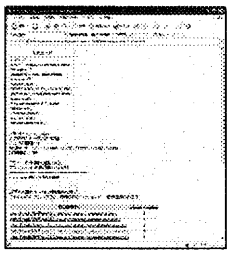

図 7 検索結果の例

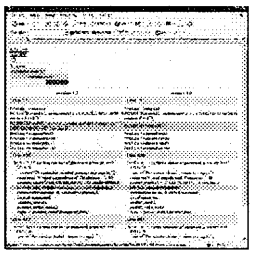

図 8 差分の表示

コミットログから「パスワード送信の際に長さが分からないように送る」と記述してあることから，利用者が必要とする情報だといえる. しかし，中身を見ると関数の変更

`packet_put_string(response_stren(response)`

→ `ssh_put_password(response)` は記載してあるが，関数 `ssh_put_password` についてはこれ以上記載されていない.

そこで，関連情報の検索を行う．ここで考えられる検索方法としては類似コード検索により出力された結果を用いる. 検索方法として以下の 4 つを考えられる.

（1） 開発者 green による検索

（2） 更新日時が 2001/03/20 02:06:40 前後のファイル検索

（3） それに伴うコミットメール情報の検索

（4） キーワード「openssh」による検索

その結果，(2) の同時間帯にコミットされたファイル sshconnection.c の中に関数 ssh_put_password の内容が記載されている部分を検索できた.(図 10，図 11 参照)

```
void
  ssh_put_password(char *password)
  {
    int size;
    char *padded;

    size = roundup(strlen(password) + 1, 32);
    padded = xmalloc(size);
    memset(padded, 0, size);
    strlcpy(padded, password, size);
    packet_put_string(padded, size);
    memset(padded, 0, size);
    xfree(padded);
  }
```

図 9 関数 ssh_put_password

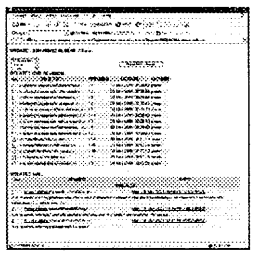

図 10 更新日時間帯による検索

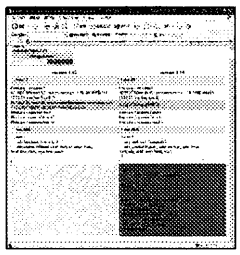

図 11 関数 ssh_put_password

本システムを用いて実例に基づいた適用実験を行った結果，類似コードによる検索で修正部分の特定が可能となることが分かる．しかし，その情報だけでは完全に利用者の要求を満たすことはできなかった．そのため関連情報の中から必要となる情報を取得することができた.

4.4 検索速度評価

ここでは 3 つの類似検索データベースに対し，3 つのソースコードを与えて検索時間の計測を行った．表 1 のデータを評価に用いた．これは FreeBSD の CVS リポジトリ内のソースコードについての情報を取得したものである.

	データ習得ファイル数	差分数	総ファイル数
src/crypto/openssh	67	6382	251
src/usr.bin	763	18403	827
src/contrib	318	31542	5395

表 1 データベースに用いたリポジトリに関するデータ

また，入力として与えたソースコード片 X・Y・Z に関するデータは表 4 に示す．特性キートークンを変えて計測する.

	行数	トークン数	特性キートークン
X	5	28	(if),(if,return),(if,return,fopen)
Y	25	134	(if),(if,return),(if,return,fopen)
Z	38	206	(if),(if,return),(if,return,fopen)

表 2 与えた入力に関するデータ

	X	Y	Z
openssh	32	32	32
usr.bin	55	74	80
contrib	111	113	191

表 3 検索の所要時間 (if)

	X	Y	Z
openssh	10	18	20
usr.bin	20	27	30
contrib	47	59	67

表 4 検索の所要時間 (if,return)

	X	Y	Z
openssh	2	4	5
usr.bin	7	9	10
contrib	9	14	15

表 5 検索の所要時間 (if,return,fopen)

以上のデータを用いて類似コード検索の速度の検証を行った．単位時間は全て秒である．

CoDS では，入力として与える特性キートークンに出現頻度が少ないものを用いると速くなることが分かっている [7]．本システムでは，特性キートークンを複数個用いて，比較対象の範囲を絞り込んだ結果，さらに検索速度が上昇することが分かる．

4.5 考　　察

今回の適用実験から，利用者は手持ちのコードを用いることで類似した部分の情報を検索することができることが分かる．また，情報の関連情報を検索することにより，そのプロジェクトの全体に対する理解を深めることができることが分かる．

情報検索を容易にするには，基本情報の特定が必要である．そのために，利用者自身が情報を絞り込む作業が必要となる．しかし，適用実験や開発途中の利用例からソースコードやキーワードを用いて情報を絞り込めば，そこから関連情報を検索することが出来る．また，コミットメールや同一更新者・同一時間帯に保存された開発履歴は関連性が強いことも確認できた．

一方で解決に至っていない部分も存在する．たとえば，キーワード検索はデータ作成の際にコミットログやメールの内容に大きく影響を受けるので，必ずしも有用な検索ができるとは限らない．また，表示された情報が全て利用者の求める情報であるとは限らないため，多くの情報に捕らわれて検索を繰り返していくうちに本来必要な情報を見落としてしまう恐れもある．

キーワードの問題はデータ取得の方法を改善したり，あるキーワードを持つファイルに対して他の情報で関連性を持つと思われるファイルも同時に表示できるようにすることで改善できると考える．また，キーとして与える情報を複数にすることにより一度に表示する情報の精度を上げることにより情報の見落としを防ぐことができるようになる．今後，これらの点を改善できればさらなる開発支援が期待できる．

検索速度においては，CoDS に対してさらに情報を絞り込むことにより，本システムでは検索速度が上がることを確認できた．しかし，この方法は始めに検索対象となる範囲を絞り込んでいるため，入力として与えるキートークンが間違っていれば，利用者が本当に必要としている情報を切り落としてしまう可能性がある．今後，文字列一致の計算方法を改良することにより，情報全体の検索時間を速くすることによりさらなる改善が見込まれる．

5. ま と め

本研究では，オープンソースソフトウェア開発で蓄積された膨大なソフトウェアプロダクトの履歴を開発者が有効に利用できることを目的とし，先に私の所属する研究グループでこれらの問題点を解決するために開発されたソースコード修正支援システム（CoDS）と，オープンソースソフトウェア開発支援のためのソースコード及びメールの履歴対応表示システム（SPxR）の統合を図り，新システムの実装を行った．

また，実際のオープンソースソフトウェア開発で用いられるデータを用いて本システムの実験を行った．その結果，本システムは大規模なオープンソースソフトウェア開発に適用した場合でも実時間で動作し，実用的であることを確認した．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になることを確認した．

本システムの改善点としては実行速度の向上や関連情報の充実などが挙げられる．実行速度の向上については，検索手法のさらなる改善やデータベースを効率的に構築することが挙げられる．関連情報の充実については，関連性の有無だけではなく，情報のスコア付けを行う手法を用いることや検索方法を増やすことなどが考えられる．これらの改善点を踏まえてオープンソースソフトウェア開発支援用の検索システムとしての充実を目指し，オープンソースソフトウェア開発環境支援に貢献することを目指す．

文　　献

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

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

[3] 藤原晃, “ソースコード間の関係を用いた再利用性評価手法の提案”, 情報処理学会研究報告, 2002-SE-136, Vol.2002, No.23, pp.155-162, 2002.

[4] Dan Gusfield, “Algorithms on Strings, Trees, and Sequences”, Cambridge University Press, 1997.

[5] 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.

[6] 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

[7] 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手法の提案”, 情報処理学会研究報告, 2002-SE-136, Vol.2002, No.23, pp.57-64, 2002.

[8] 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.

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

[10] 山本 哲男, 松下 誠, 井上 克郎, “バージョン管理ファイルシステムを用いた保守支援ツールの提案”, 電子情報通信学会技術研究報告 Vol.99, No.164, SS98-23, pp. 65–72, 1999.7.9.