修士学位論文

題目

オープンソース開発の履歴情報を用いたコミュニティ検索支援システム

指導教官
井上 克郎 教授

報告者
佐々木 啓

平成 17 年 2 月 14 日

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 ソフトウェア工学講座

平成 16 年度 修士学位論文

オープンソース開発の履歴情報を用いたコミュニティ検索支援システム

佐々木 啓

内容梗概

オープンソースソフトウェアの開発では，世界中に分散した多くの開発者によって進められることが一般的である．このとき，開発を効率よく管理するため，版管理システムや電子メール，障害管理システムが広く用いられている．これらのシステムには開発時の履歴情報や開発者の情報が保存されているため，次のソフトウェア開発時に再利用可能な作業内容や，設計方針などの知識を得ることができると考えられる．しかし，蓄積された膨大な情報の中から，求める作業内容や知識を選定する作業は容易ではなく，実際の開発作業ではあまり利用されていなかった．

そこで本研究では，まず，作業の履歴情報と開発者相互の関連を開発コミュニティとして定義する．次に，開発コミュニティ中から，ある作業や知識といった着目したい内容を入力とし，関連する履歴情報や開発者からなる集合を選定するための手法について提案する．具体的には，まず既存のシステムより履歴情報と開発者を抽出し，それらの関係を分析して開発コミュニティの構築を行う．次に，ユーザの入力に応じて，開発コミュニティから履歴情報や開発者の選定を再帰的に繰り返す．これにより，求める履歴情報や開発者を含む集合全体を選定する．また，大規模なオープンソースの 1 つである FreeBSD を対象として，本手法を用いたコミュニティ検索システムの構築を行った．本システムを FreeBSD 開発者へ実際に利用してもらうことによってシステムの評価を行い，システムの有用性について確認した．

主な用語

開発履歴情報
オープンソースソフトウェア開発
コミュニティ検索

目 次

1　まえがき

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プロダクト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，開発者が分散し，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，版管理システムや障害管理システム，電子メールを用いることが多くなっている．版管理システムは，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．障害管理システムでは，プロダクトに関する機能追加や修正要求を格納し，その変更履歴を管理している．また，電子メールを介して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．

これらのシステムには，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が，個別のアーカイブとして保存されており，そこ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 [11]．また，開発に関わった開発者に着目すると，彼らはオープンソースソフトウェア開発の各分野を担当して開発を行っており，各人や彼らの属しているグループが行った開発には何らかの関連があると考えられる．このように，システムに蓄積された開発情報とそれに関わった開発者達を大きなコミュニティとして考えた場合，その中から必要な履歴情報を保持しているサブコミュニティを選定し，次のソフトウェア開発時に再利用可能な作業内容や，設計方針などの知識を得ることができると考えられる．

しかし，蓄積された膨大な情報の中から，求める作業内容や知識を選定する作業は容易ではなく，実際の開発作業ではあまり利用されていなかった．さらに，開発者が必要とするコミュニティは，万人に対して同一なものではなく，開発者と彼らが抱えている問題点に応じて動的に形成されると考えられる．そのため，単に開発情報の関連から導き出された静的なコミュニティでは，全ての要求を満たす「処理」や「知識」の選定を行うことができない．

そこで本研究では，まず，作業の履歴情報と開発者相互の関連を開発コミュニティとして定義する．次に，開発コミュニティ中から，ある作業や知識といった着目したい内容を入力とし，関連する履歴情報や開発者からなる集合を選定するための手法について提案する．具体的には，まず既存のシステムより履歴情報と開発者を抽出し，それらの関係を分析して開発コミュニティの構築を行う．次に，ユーザの入力に応じて，開発コミュニティから履歴情報や開発者の選定を再帰的に繰り返す．これにより，求める履歴情報や開発者を含む集合全体を選定する．また，大規模なオープンソースの 1 つである FreeBSD を対象として，本手法

を用いたコミュニティ検索システムの構築を行った．本システムを FreeBSD 開発者へ実際に利用してもらうことによってシステムの評価を行い，システムの有用性について確認した．

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効率良く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により，開発者の負担を軽減し効率よく開発作業を行うことが可能となり，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 節では，オープンソースソフトウェア開発とその環境について述べる．3 節では，本研究で定義するオープンソースソフトウェア開発上の語句の説明を行う．

4 節では，トピックコミュニティの検索手法についての提案を行う．5 節では，システムの実装について説明する．6 節では，システムの評価を考察する．最後に，7 節で本研究のまとめと今後の課題について述べる．

2　オープンソースソフトウェア開発

オープンソースソフトウェア開発とは，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である[29][33]．これは，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おり，そのオープンソースソフトウェア開発によって開発されたソフトウェアをオープンソースソフトウェアと言う．

オープンソースソフトウェア開発では，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[38] や Linux[23]，GNU[13]，Apache[1] 等は，オープンソースソフトウェアの代表である．

本節では，オープンソースとその開発環境について触れ，そこで用いられるシステムとその履歴情報の利用について説明する．

2.1　オープンソースソフトウェア環境

オープンソースソフトウェア開発では，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，オープンソースソフトウェア開発に参加する開発者は，オープンソースソフトウェア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．

オープンソースソフトウェア開発環境の構成例を図 1 に示す．オープンソースソフトウェア開発環境は，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 [10] の一つである CVS(Concurrent Versions System)[6][14][20][27] 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)[12] 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，オープンソースソフトウェア開発で広く用いられる版管理システムと電子メール，障害報告システムについて説明する．

2.1.1　版管理システム

版管理とは，主として以下の 3 つの役割を提供する機構である．

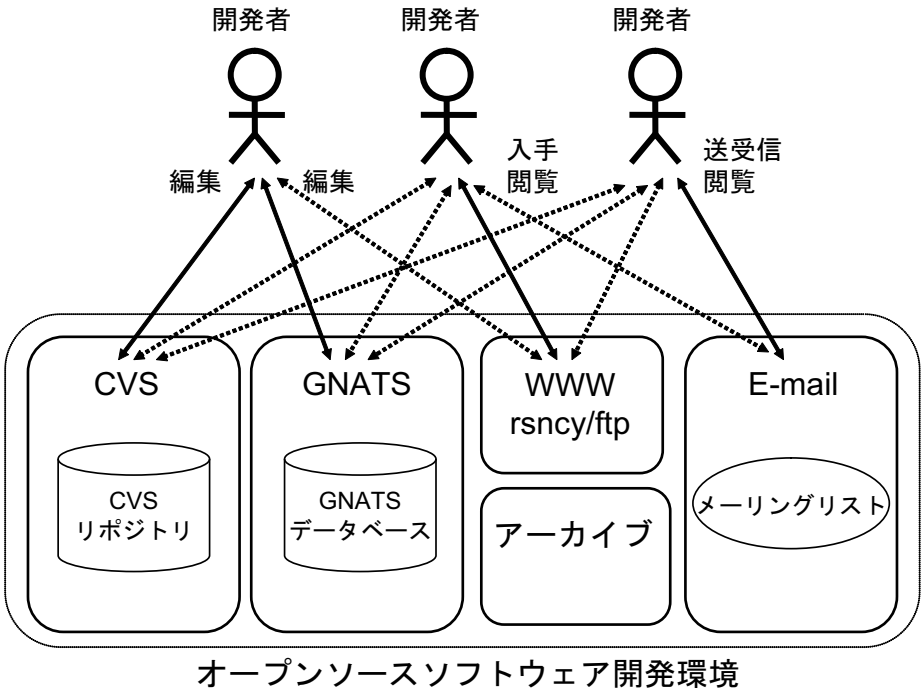

図 1: オープンソースソフトウェア環境の構成例

- プロダクトに対して施された追加・削除・変更などの作業を履歴として蓄積する.
- 蓄積した履歴を開発者に提供する.
- 蓄積したデータを編集する.

各プロダクト (ソースコード，リソースなど) の履歴データは，**リポジトリ (Repository)** と呼ばれるデータ格納庫に蓄積される．その内部では，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)** を単位として管理する．１つのリビジョンには，ソースコードやリソースなどの実データと，作成日時やメッセージログなどの属性データが格納されている.

また，リポジトリとのデータ授受をする為に，開発者はシステムに依存したオペレーション (operation) を利用する必要がある.

版管理手法を述べるにあたり，その基礎となるモデルが数多く存在する [4][7]．本節では，多くの版管理システムが採用している Checkout/Checkin モデルについて概要を述べる．なお，以降本文において，プロダクトのある時点における状態のことをリビジョンと呼ぶことにする.

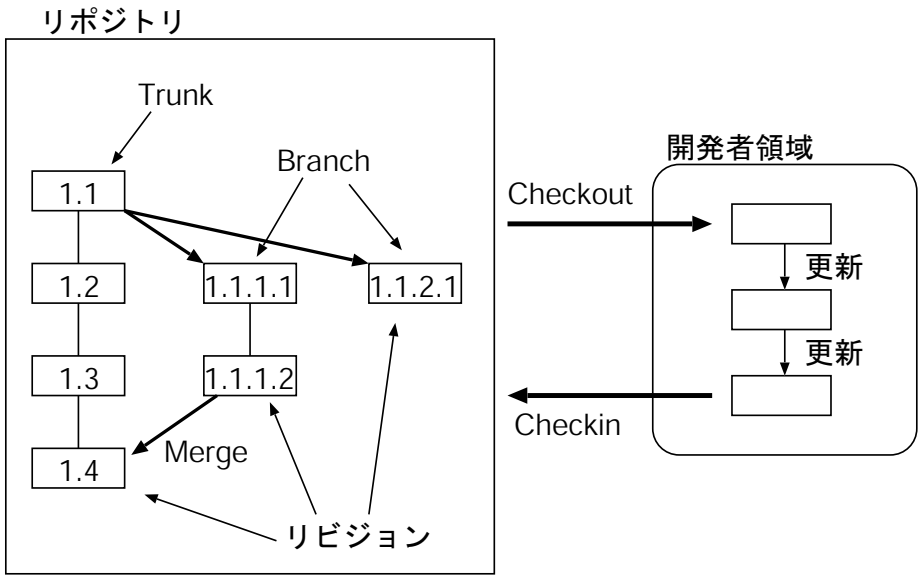

図 2: Checkout/Checkin Model

The Checkout/Checkin Model

このモデルは，ファイルを単位としたリビジョン制御に関して定義されている (図 2 参照).

リビジョン管理下にあるコンポーネントはシステムに依存したフォーマット形式のファイルとしてリポジトリに格納されている．開発者のそれらのファイルを直接操作するのでなく，各システムに実装されているオペレーションを介して，リポジトリとのデータ授受を行う．リポジトリより特定のリビジョンのコンポーネントを取得する操作を チェックアウト (Checkout) という．逆に，データをリビジョンに格納し，新たなリビジョンを作成する操作を チェックイン (Checkin) という．

単純にリビジョンを作成するのみでは，シーケンシャルなリビジョン列を生成することになる．しかし，過去のリビジョンに遡り，別の工程で開発を行う場合 (例えば，デバッグ等の為)，リビジョン列を分岐させるには，ブランチ (Branch) という操作により，ブランチを生成し，その上にリビジョンを作成するという手法を採る．このブランチに対して，元のリビジョン列のことを**トランク (Trunk)** という．また，ブランチ上での作業内容 (デバッグ修正部分等) を，別のリビジョンに統合する作業を マージ (Merge) という．このように，リビジョン列は木構造になることから リビジョンツリー (RevisionTree) という．

版管理システムと呼ばれるものは，多数存在する．UNIX 系 OS では，多くの場合，RCS[39] や CVS[6][14] といったシステムが標準で利用可能となっている．ClearCase[32] のように商用ものも存在する．また，UNIX 系 OS だけではなく，Windows 系 OS においても，SourceSafe[24]

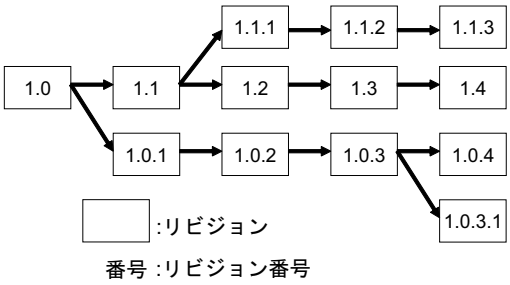

図 3: リビジョンツリーの例

や PVCS[26] をはじめ，数多く存在する．さらに，ローカルネットワーク内のみではなく，よりグローバルなネットワークを介したシステム [16] も存在する．

近年ハードウェアの進化と共に GUI 化が進んだ為，リポジトリ内部情報の視覚的な把握が可能となった．具体例として，以下のものが挙げられる．

- ネットスケープ等，ウェブブラウザを通して閲覧する．GUI の為の新たなシステム導入が不必要である．
- 版管理システムそのものが GUI 化されている．コマンドラインからの操作に比べ，間違いが少ない．

ここでは，版管理システムのうちのいくつかを紹介する．

- RCS

RCS[39] は UNIX 上で動作するツールとして作成された版管理システムであり，現在でもよく使用されているシステムである．単体で使用される他，システム内部に組み込み，版管理機構を持たせる場合などの用途もある．RCS ではプロダクトをそれぞれ UNIX 上のファイルとして扱い，１ファイルに対する記録は１つのファイルに行われる．

RCS におけるリビジョンは，管理対象となるファイルの中身がそれ自身によって定義され，リビジョン間の差分は diff コマンドの出力として定義される．各リビジョンに対する識別子は数字の組で表記され，数え上げ可能な識別子である．新規リビジョンの登録や，任意のリビジョンの取り出しは，RCS の持つツールを利用する．

- CVS

CVS[6][14][20] は RCS 同様，UNIX 上で動作するシステムとして構築された版管理システムであり，近年最も良く使われるシステムの１つである．RCS と大きく異なるのは，複

数のファイルを処理する点である．また，リポジトリを複数の開発者で利用することも考慮し，開発者間の競合にも対処可能となっている．さらに，ネットワーク環境 (ssh， rsh 等) を利用することも可能である為，オープンソースによるソフトウェア開発やグループウェアの場面で活躍の場が多い．その最たる例が，FreeBSD や OpenBSD 等のオペレーティングシステムの開発である．

2.1.2 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．

オープンソースソフトウェア開発では，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．また，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．

2.1.3 障害管理システム

障害管理システムは，開発されたプロダクトのバグ報告や機能の追加要求を管理するシステムである．これらのシステムでは，開発者が投稿した障害情報をデータベースに蓄積する．他の開発者らは，それらの問題を閲覧することで，修正が行えると思えば，それを引き受け，修正作業を行う．さらに，それ以外の開発者も，過去の解決された事例を閲覧することで，自身の問題解決に役立てることが可能である．

UNIX 系 OS の開発で用いられる障害管理システムには，GNATS[12] や Bugzilla[5] などが挙げられる．これらのシステムでは，まず，起草者がバグの報告や作成したプロダクトの機能に対して追加要求を行う．それをもとに，担当の管理人が決定され，彼らが修正担当者を割り当て，修正担当者が実際に要求の解決を図る．これらのシステムも，先に述べたリビ

ジョン管理システムや電子メールと同様にネットワークを通して障害情報の投稿や抽出を行うことが一般的に行われており，それらは開発作業の促進につながっていると言える．

2.2 オープンソースソフトウェア開発プロダクトの再利用

近年のオープンソースソフトウェア開発は，ソフトウェアプロダクトが増加するにつれて，開発基盤としての価値に加え，ソフトウェアを構成する様々な部品を発見して他のソフトウェア開発に再利用することや，参考にするといったような，ソフトウェア資産を有する大規模なライブラリとしての価値も高まっている．

このように，システムに蓄積された開発情報の中から，今後の開発に役立つプロ以下に，これらの研究について説明する．

2.3 関連研究

リビジョン管理システムなどのリポジトリ解析に関する研究としては，Mockus らが，CVS の開発ログを解析し，語句の出現頻度やキーワード分類をもとに，変更の目的や，それに伴う変更の時間や変更量の分析を行っている [22]．また，Harald Gall ら [17] は，CVS の開発情報をもとにシステム進化を解析し，システムの開発過程で，クラス・ファイル・function 間で開発者と開発時間が一致する論理的な結合の抽出を行い，それらの関連情報をユーザに提供している．

また，複数のシステムにまたがった履歴情報の分析に関する研究としては，Čubranić らが Hipikat[8] と呼ばれるシステムの開発を行っている．Hipkat では，CVS や Bugzilla・電子メールに蓄積された開発情報の関連をもとに group memory を抽出を行う．それを Eclipse の開発環境に実装することで，ユーザの行う開発に応じて，同時に変更が必要だと考えられる関連 group memory の提案を行っている．

2.4 履歴情報間の関連付けの問題点

先に挙げた研究では，あらかじめ抽出した関連もとにユーザに掲示するため，同じプロダクトに着目して検索を行った場合は，常に同じ関連を抽出する．しかし，万人が同じプロダクトに着目した場合に，彼らの要求を満たす関連が一定であることはありえない．また，同じ人でも，その時々によって必要な関連情報は変化する [42][35]．このように，静的な関連付けに基づいた検索だけでは，動的でかつ，万人によってことなる問題に対して十分に対応できない．さらに，従来の研究では，プロダクトの開発に携わった開発者の情報を考慮しているものは少ない．彼らは，実際の開発のエキスパートであるため，彼らの開発状況を考慮に入れた情報の関連は有用であると考えられる．

そのため，本研究では，開発情報間の関連に加え，そこに関わる開発者らをコミュニティとして定義し，これらの関連付けの手法を考える．さらにその関連付けをもとにして，必要に応じて動的に変化するなプロダクト間の関連情報の検索を図る．次節では，今回の研究で用いる語句の定義を行う．

3　オープンソースソフトウェア開発における開発者と開発情報について

本節では，本研究で用いる以下の語句について定義を行う．

- Knowledge
- 開発コミュニティ
- トピック
- トピックコミュニティ

さらに，前節で述べたシステムに蓄積されたオープンソースソフトウェア開発の開発情報や開発者情報間の関連について考える．

3.1　Knowledge

本研究では，リビジョン管理システム CVS[14]，電子メールアーカイブ，障害管理システム GNATS[12] に蓄積された開発情報を Knowledge として定義する．この Knowledge はさらに開発を行った理由 (**Intention**) と，行われた開発内容 (**Task**)，それぞれの Knowledge を識別する管理情報 (**Entity**) の 3 つに分類することができる．Intention は，プロダクト設計に関する議論，登録の際のコメント，バグの概要などが含まれ，Task にはソースコード，バグの対処内容，拡張機能の中身などが含まれる．各システムの Knowledge はそれぞれ図 4 のように分類される．

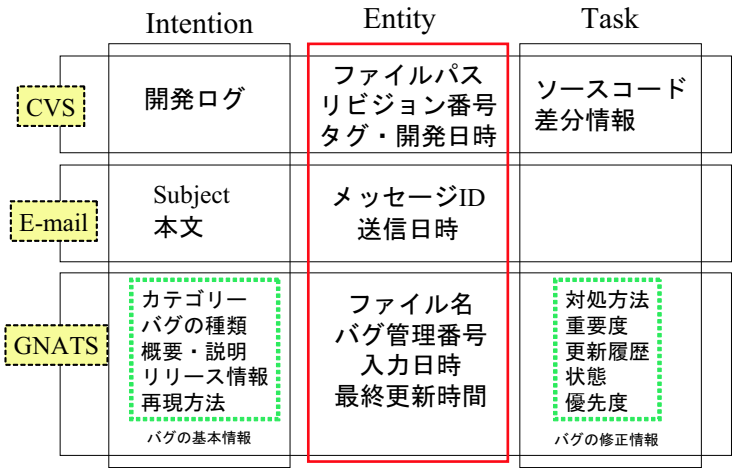

図 4: Knowledge の分類

3.2 開発コミュニティ

人と Knowledge には 3 つの関連が存在する．本研究では，これらの関連をもとに構成される集合を開発コミュニティと定義する (図 5 参照).

1. 人と Knowledge の関連 (H to K)
   開発者とその開発者が開発に関わった Knowlesge の関連

2. 人と人の関連 (H to H)
   同じプロジェクトやプロダクト開発に関わった開発者同士の関連

3. Knowledge と Knowledge の関連 (K to K)
   特定のプロダクトに関する Knowledge 間の関連

以下，これらの 3 つの関連は H to K，H to H，K to K と呼ぶ.

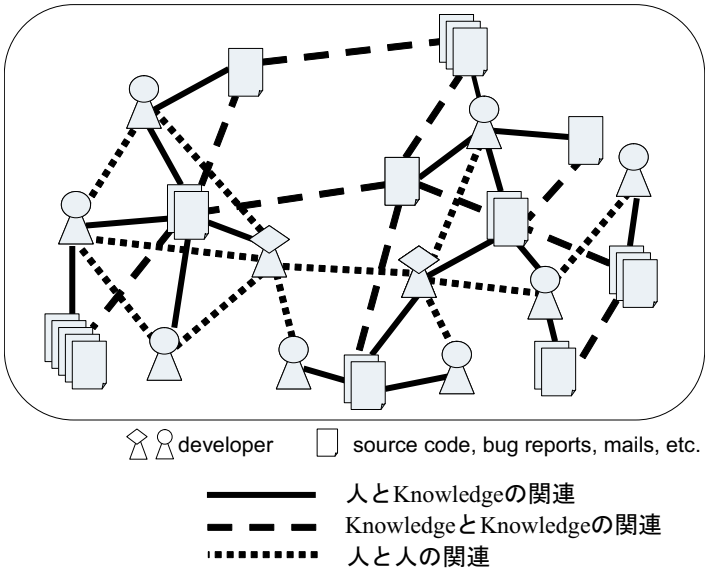

図 5: 開発コミュニティ

3.3 トピック

トピックとは，Knowledge の中に含まれているような過去の開発で行われた Intention や Task である．これらは，過去の開発を行った目的や，その実現方法が含まれている．そこで，これをもとに，行われた開発に対する理解を深め，自身の開発に再利用を行うことで，今後の開発に役立て，開発コストを軽くすることが出来る.

3.4 トピックコミュニティ

開発コミュニティ中には，複数のトピックが含まれている．これらのトピックは複数の Knowledge から構成されるが，そのトピックには，直接的，あるいは間接的に関わった開発者の関連も存在する．これらを開発コミュニティ中に存在するサブコミュニティととらえると，開発コミュニティの中には，このような複数のサブコミュニティが存在することになる．そこで，本研究では，過去の開発のトピックに関連する人と Knowledge から構成されるコミュニティをトピックコミュニティと定義する．

4　トピックコミュニティ選定支援手法

本節では開発コミュニティから，トピックコミュニティを選定し，必要なトピックを抽出するための手法の提案を行う．本手法は以下の手順で行う．

1. データベースの構築
2. ユーザは，トピックを検索対象としてシステムに入力する
3. システムは，入力をもとにデータベースを検索し，候補となる人，Knowledge を含んだ開発情報を表示する
4. トピックコミュニティを選定するまで (4b) から (4a) を繰り返す
   (a) ユーザは，開発情報の候補の人・Knowledge から，必要な人・Knowledge を選定する
   (b) システムは，ユーザが選定した人・Knowledge に対して，データベースに登録された関連を検索し，その分類を表示する
   (c) ユーザは，表示された関連の分類の中から，必要な関連を選定する
   (d) システムは，選定された関連に基づいてデータベースを検索し，候補となる人・Knowledge を含んだ開発情報を表示する

4.1　データベースの構築

まず，CVS，電子メール，GNATS に蓄積された開発情報から識別情報と人，Knowledge の抽出を行い，履歴情報 DB を作成する．さらに開発状況に応じて人に対する役割付けと重み付けを行う．それらを Role DB，Weight DB に登録する．

4.1.1　人と Knowledge の抽出，重み付け

CVS 情報

CVS では，開発の履歴情報はリポジトリごとに一つのファイルに記載されている (図 6 参照)．そこで，それらのファイルの構文解析を行うことで，リポジトリの各リビジョンにおける識別情報，人，Knowledge の情報を抽出する．

まず,「識別情報」として，**ファイルパス**，**revision**，**date** を抽出する．

「人」の情報として，各リビジョンをコミットした **author** と開発ログに記されている**協力者**を抽出する．例えば，図 6 では，開発者 sobomax が src/bin/ln/ln.c,v のリビジョン 1.19,

```
rlog ln.c,v
RCS file: src/bin/ln/ln.c,v

Working file: ln.c
```

（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 6: ファイル単位のリビジョン情報

1.20 の開発を行っていることが記載されている．さらに，リビジョン 1.20 の開発ログには“ `Reviewed by: bde` ”と記載があり，これは sobomax がコミットする際に，開発者 bde がその開発内容に対してレビューを行ったということを示している．このように開発ログに author の協力者を記載することが慣習として行われるため，その開発者を協力者として抽出する．

次に,「Knowledge」の情報として，開発者が CVS に登録する際に，登録の目的を記載する**開発ログ**と，実際に変更されたソースコード情報が記載されている**差分情報**を抽出する．開発ログは，出現語句に対して，情報検索の分野で一般的に用いられる TF-IDF 法 [21] に基づいた重み付けを行い，各語句の出現頻度 TF(Term Frequency)，とその語句が含まれる開発ログの総数の逆数 IDF(Inverse Document Frequency) の値から特徴的な語句をキーワードとして抽出する．差分情報は，各リポジトリごとに前リビジョンとのソースコードの差分情報が記載されている．そのため，変更前リビジョンの部分のソースコード片をトークン列として抽象化し，その情報を抽出する．

次に，役割付けについて説明する．2 節で述べたように，CVS に格納されるプロダクトは，リポジトリ内に木構造で格納される．一般的に，木構造の幹 (トランク) 部分はプロダクトの新規開発や拡張作業に関する格納が行われ，枝 (ブランチ) 部分ではバグ修正や保守作業に関する格納が行われる．そこで，トランクへの格納を多く行っている開発者を“ デベロッパ ”，ブランチへの格納を多く行っている開発者を“ メンテナ ”と役割付ける．また，タグ情報を抽出し，タグ名で役割付けを行う．さらに，協力者は，コミットの際にレビューを行う開発者であったり，ともに開発作業を行った開発者であるため，開発ログをもとに協力者の役割付けを行う (表 1 参照).

リビジョン	開発者	開発ログ中の記載	協力者
トランク	デベロッパ	approved by,reviewed by	マネージャ
ブランチ	メンテナ	discussed on,suggest by	開発者と同じ
タグ	タグ名	submitted by, その他	対応無し

表 1: CVS 情報における開発者・協力者の役割付け

最後に，重み付けについて説明する．本研究では，抽出した人の開発範囲や開発回数から，開発回数と開発の局所性に着目して重み付けを行う．これも先に示した TF-IDF 法を用いて，全リビジョンに対する開発者の開発回数をもとに開発者の出現頻度 (TF 値) と局所性 (IDF 値) を求める．

電子メール情報

電子メールアーカイブには，ファイルごとに一通の電子メールの内容が記載されている (図 7 参照)．これらのファイルを構文解析することで，識別情報や人と Knowledge の抽出を行う．

ここでは,「識別情報」として，各メールに含まれる **Message-ID**，**メーリングリスト名**，**送信日時**を抽出する．

「人」の情報としては，To，CC 以下に含まれる**送信先**や From いかに含まれる**受信先**の電子メールアドレスを抽出する．

「Knowledge」の情報は，電子メールの中身を示す **subject**，**本文**を抽出する．これは CVS 同様，TF-IDF 法により語句の重み付けを行い，頻出頻度の高い語句をキーワードとして抽出する．

電子メールではメーリングリストを通して受信したり，送信することが多い．メーリングリストごとに開発の目的に特化された議論がやり取りされている．そのため，メーリングリストに応じて，そこに関わっている人の開発内容が推定できるため，メーリングリスト名をもとに役割付けを行う．

重み付けは，メーリングリストごとに登場回数の多い人に重み付けを行うとともに，送信日時の近い電子メールに関わっている人や，重要なキーワードに多く関わっている人に対して重み付けを行う．

GNATS 情報

GNATS でも，ファイルごとに一つの修正情報が記載されている (図 8 参照)．そのため，これらのファイルを構文解析することにより，識別情報や人と Knowledge の抽出を行う．

まず,「識別情報」として，**number**(ファイル番号)，**Arraoval-date**，**Close-date**，**Last-modified** の情報を抽出する．

「人」の情報としては，障害情報を登録した開発者 **Originator**(起草者) や，実際に修正活動を行った開発者 **Responsible**(担当者)，さらにその修正担当者を決定したり，管理状態を変更した**管理人**を抽出する．

「Knowledge」としては，Synopsis や category，priority，Description などを**基本情報**として抽出し，その後行われた Fix や，Audit-trail の情報を**修正の履歴情報**としてを抽出する．Description や Audit-trail など開発者が自由に記載する情報は CVS や電子メール同様に TF-IDF 法を用いて語句の重み付けを行い，キーワード抽出を図る．

次に人の役割付けについて考える．起草者は障害報告の際に，報告の種類 (バグ修正，機能拡張など) を class として登録する．それに対して，担当者が決定され，要求解決に向けて

```
From owner-cvs-all  Fri Jan  1 15:07:53 1999
Return-Path: <owner-cvs-all@FreeBSD.ORG>
Received: (from majordom@localhost)
      by hub.freebsd.org (8.8.8/8.8.8) id PAA05523
      for cvs-all-outgoing; Fri, 1 Jan 1999 15:07:53 -0800 (PST)
      (envelope-from owner-cvs-all@FreeBSD.ORG)
Received: from localhost (scrappy@localhost)
      by thelab.hub.org (8.9.1/8.9.1) with ESMTP id TAA23680;
      Fri, 1 Jan 1999 19:06:29 -0400 (AST)
      (envelope-from scrappy@hub.org)
X-Authentication-Warning: thelab.hub.org: scrappy owned process doing -bs
Date: Fri, 1 Jan 1999 19:06:28 -0400 (AST)
From: The Hermit Hacker <scrappy@hub.org>
To: Harlan Stenn <Harlan.Stenn@pfcs.com>
cc: Gary Palmer <gpalmer@FreeBSD.ORG>, mark@grondar.za,
                  donegan@quick.net, dcs@newsguy.com, jmb@FreeBSD.ORG,
                  committers@FreeBSD.ORG, current@FreeBSD.ORG
Subject: Re: HEADS UP: Postfix is coming. new uid, gid required.
In-Reply-To: <18780.915229502@brown.pfcs.com>
Message-ID: <Pine.BSF.4.05.9901011903540.9916-100000@thelab.hub.org>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII
Sender: owner-cvs-all@FreeBSD.ORG
Precedence: bulk

On Fri, 1 Jan 1999, Harlan Stenn wrote:


> > I hate to ask, but is it so hard for us to d

RCS file: src/bin/ln/ln.c,v


                    (以下本文)
```

図 7: 電子メール情報

```
Subject: New port: net/ldapdiff A utility to patch ldap directories
X-Send-Pr-Version: 3.113
X-GNATS-Notify:

>Number:        34985
>Category:      ports
>Synopsis: New port: net/ldapdiff A utility to patch ldap directories
>Confidential: no
>Severity: non-critical
>Priority: medium
>Responsible: freebsd-ports
>State: closed
>Quarter:
>Keywords:
>Date-Required:
>Class: change-request
>Submitter-Id: current-users
>Arrival-Date: Fri Feb 15 17:40:01 PST 2002
>Closed-Date: Thu Jun 27 07:13:25 PDT 2002
>Last-Modified: Thu Jun 27 07:13:25 PDT 2002
>Originator: Christian Brueffer
>Release: FreeBSD 4.5-STABLE i386


（中略）


Xand compile it. Afterwards you have to adapt your config file
Xaccordingly.
X
END-of-ldapdiff/pkg-message
exit

--- ldapdiff-port ends here ---

>Release-Note:
>Audit-Trail:
State-Changed-From-To: open->closed
State-Changed-By: will
State-Changed-When: Thu Jun 27 07:13:06 PDT 2002
State-Changed-Why:
Committed, thanks.

http://www.freebsd.org/cgi/query-pr.cgi?pr=34985
>Unformatted:
```

図 8: GNATS 情報

作業を行う．そこで，起草者と担当者の役割は class をもとに決定する．さらに，担当者を決定する管理人は修正の“ マネージャ ”として役割付けを行う (表 2 参照).

class	起草者	責任者	担当者
sw-bug,doc-bug	デバッガ	マネージャ	メンテナ
change-request	ユーザ		デベロッパ
update,maintainer-upgrade	メンテナ		メンテナ

表 2: GNATS 情報における起草者・責任者・担当者の役割付け

重み付けについては，障害の修正活動に参加している回数に加え，class や priority をもとに重要な GNATS 情報に関わっている人に対して重み付けを行う．さらに，電子メールと同様に，重要なキーワードに関わっている人に対しても重み付けを行う.

4.1.2 人と Knowledge の関連付け

本節では，前節で抽出した人と Knowledge に対する関連付けについて説明する.

H to K 関連

同じ CVS，E-mail，GNATS から抽出された「人」と「Knowledge」には関連付けを行う．この関連は Human Knowledge Relation データベースに登録する.

H to H 関連

以下のいずれかの条件を満たす「人」には，H to H の関連付けを行う．この関連は Human Relation データベースに登録する.

1. 同一の CVS リポジトリで，同じトランクやブランチに対してコミットを行った author リポジトリの情報は，一つのファイルに記載されているため，そのリビジョン番号を参照することで，関連付けを行う.
2. 同一リビジョンで開発ログに記載された協力者と author
3. 同一の電子メールの受信，送信，返信，転送に関わった開発者
4. 開発者と送信，受信したメーリングリストのアドレス
5. 同一の GNATS 情報に記載されている originator，responsible，管理人

K to K 関連

以下のいずれかの条件を満たす Knowledge には，K to K の関連付けを行う．この関連は Knowledge Relation データベースに登録する．

1. 同一 CVS リポジトリの中で，同じトランクやブランチに含まれたリビジョン情報

   一つのファイル中に記載されているため，そのリビジョン番号をもとに関連付けを行う．

2. author と更新時間が一致するリビジョン情報

   本研究では開発日時の誤差を± 3 分とする．

3. ソースコードの変更点が類似しているリビジョン

   リビジョン間の差分情報をもとに，変更された部分が類似しているリビジョン情報に関連付けを行う．類似度の判定は，先に我々の研究グループで開発されたシステム CoDS[37] を用いて行う．

4. 電子メールとその返信・ 転送メール

   受信メールに対して，返信，転送を行った場合，参照元の Message-ID が reply-ID として付加される．その情報をもとに，電子メールと reply-ID の該当する電子メールに関連付けを行う．

5. Subject が一致し，送信日時が近い電子メール

   Message-ID の関連がない場合でも，Subject が一致し，送信日時も近い電子メールは，関連性が高いとみなす．本研究では，送信日時の誤差は± 6ヶ月とする．

6. キーワード

   キーワードは各文書中を特徴付ける語句を取り出している．そのため，これらの情報が一致するということは，直接的ではないにせよ何らかの関連があるとみなす．

7. 識別情報

   それぞれの開発情報を解析するときに抽出された識別情報をもとに，記載されている Knowledge と識別情報の Kwnoledge に関連けを行う．

これらの関連付けや重み付けを行うことで，開発コミュニティを定めることができる．次に，この中からトピックコミュニティの抽出を図る．次節からその手順について説明を行う．

4.2 トピックの入力

最初に，ユーザの求めるトピックを入力として与えることで，ユーザの求めるトピックを含んだ Knowledge，人を検索を行う．ユーザは自分が必要なトピックを言語化する必要がある．そこで，以下のようにトピックを言語化し，それをシステムへ入力として与える．これらの情報を組み合わせることで候補 Knowledge，人の検索を図る．

- トピック中に含まれているキーワード

  ユーザのトピックを表す特徴語をもとに該当する語句を含む Knowledge を検索する．これは，一般的な文書検索でも広く用いられる手法である．

- 開発者

  特定の開発者は，特定の開発に多く関わることが多い．そのため開発者をもとに，開発者と関連のある人や，Knowledge を検索を図る．

- 開発時間

  時間によってプロダクトは変化する．そのため，特定の時間の Knowlege について知りたい場合には，開発時間を入力として与えることで該当する Knowledge の検索を図る．

- 変更したいソースコード

  ユーザが抱えている問題ソースコードを入力として与えることで，該当するソースコードを持った差分情報の検索を図る．

- CVS 情報のディレクトリ名・ファイル名

  ファイル名やディレクトリ名と開発内容は何らかの関連付けが行われていることが多い．そのため，名前をもとに特定の処理を行っているであろう CVS 情報の検索を図る．

- メーリングリスト名

  前節でも述べたように，メーリングリストごとに行われる議題は異なる．そのため，知りたい議論が行われている見当がついてる場合には，メーリングリスト名をもとに，必要な Knowledge の検索を図る．

- GNATS 情報の Category，Class 情報

  GNATS 情報も，Category や Class によって分類されている．そのため，特定の分野の GNATS 情報を必要とする場合には，その情報を入力として与えることで該当する Knowledge の検索を図る．

4.3 データベースの検索

システムでは，入力されたトピックに対して，履歴情報データベースの中から該当するKnowledge を含んだ CVS，電子メール，GNATS の情報を検索する

4.3.1 検索方法

1. キーワード検索

   CVS の開発ログ，電子メールの subject，本文，GNATS の概要，対処方法，変更履歴の中から，入力されたキーワードを含んだ Knowledge を検索する．

2. 開発期間の検索

   CVS，E-mail，GNATS の登録時間をもとに，該当する期間に登録された Knowledge を検索する．

3. 開発者検索

   人の情報をもとに，該当する人が関わっている開発情報を検索する．

4. 類似ソースコード検索

   入力ソースコードをトークン化し，データベース中の差分情報と比較することで類似したソースコードを含んでいる CVS 情報を検索する．

5. 開発内容検索

   入力として与えられたディレクトリ名やファイル名，メーリングリスト名，Category，Class 情報に該当する開発情報の検索を行う．

4.3.2 検索結果の表示

システムからは，候補 Knowledge，人を含んだ開発情報が結果として出力される．まず，該当する開発情報の一覧が表示され，それらの開発情報はそれぞれ図 9 の情報が含まれる．

これらの開発情報中に，他の開発情報の識別情報が含まれていた場合，その開発情報の内容に対して明示的な関連があるとみなし，図 9 の情報に加え，識別情報の開発情報一覧を表示する．ユーザはこれらの中から，求めるトピックに適合する Knowledge や人の選定を行う．

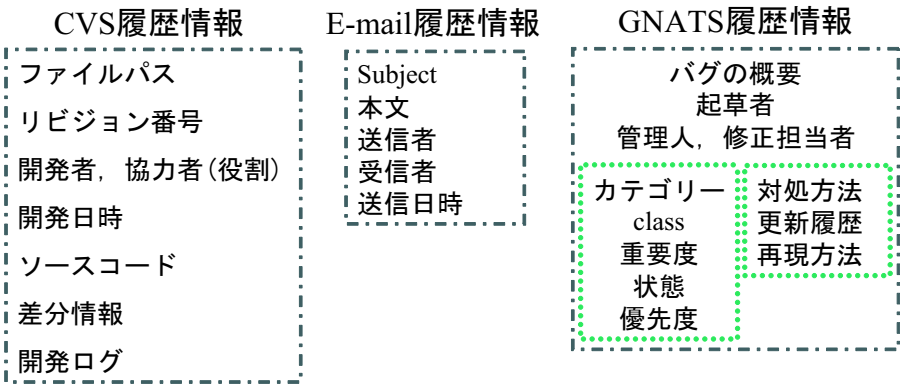

図 9: 検索結果の内容

4.4 トピックコミュニティの追跡

ユーザは，検索結果を手がかりとしてそれが含む Knowledge や人の関連を辿ることで，必要なトピックの抽出を行う．

最初に，検索結果の中から，関連追跡の基点となる人や Knowledge の特定を行う必要がある．以下にその手順について説明を行う．

4.4.1 検索結果の選定

CVS の開発ログや電子メールの Subject，GNATS 情報のバグの概要，対処方法のような，開発情報を端的に表す Knowledge を調べることで必要な情報を判断する．

さらに，開発情報に対する「人」に着目する．4.1.1 節で説明したように，人にはそれぞれ役割が割り当てられている．そのため，開発情報の中の人の役割や，その人が同時に更新した開発情報，および同じ役割で行っている他の Knowledge を考慮することで，ユーザは着目すべき人の選定や，Knowledge の選定を行う．

4.4.2 関連の分類

ユーザが着目する人や Knowledge を選定すると，システムはその選定に対して，存在する関連を分類し，ユーザに提供する．提供される関連は以下の 3 つである．

- 開発内容
- 類似性
- 開発者

開発内容と類似性による関連分類は，Knowledge rel. DB に登録された K to K の関連を検索することで抽出される．また，開発者による関連分類は，Human Knowledge rel. DB, Human rel. DB から H to K，H to H の関連を検索することで抽出される．次に，それぞれの関連分類について説明する．

開発内容による関連分類

この分類では，着目した Knowledge に対して，開発内容に基づいて関連付けられた Knowledge をデータベースから検索する．開発内容に応じて，3 種類の関連が存在する．

1. 開発変遷による関連
   この関連をもとに，着目した Knowledge の時間の経過による Knowledge の変遷を検索する．ユーザが入力として与える情報と，その結果として得られる開発情報は図 3 の通りである．

入力	検索される開発情報
ファイルパス，リビジョン番号 (CVS)	該当ファイルのリポジトリ情報
Subject（E-mail)	該当スレッド一覧
Audit-Trail(GNATS)	GNATS の修正履歴

表 3: 開発変遷による関連検索

2. 識別情報による関連
   4.1.2 節で述べたように，本研究では識別情報が記載されている開発情報間には，明確な関連があるとみなす．そのため，着目した Knowledge の識別情報が他の Knowledge に記載されていたり，着目した Knowledge に他の識別情報の記載があれば，該当する Knowlege を含む情報の検索を行う．

3. 開発課程による関連
   同一ディレクトリは何らかの関連があることが考えられる．また，同じ開発者によって同時にコミットされた CVS 情報は関連があるということはこれまでにも考えられている．このように，着目した Knowledge に対して，開発課程で関連が深い Knowledge の検索を行う．入力として与える情報や検索結果として得られる Knowledge は図 4 の通り．

入力	検索される開発情報
ファイルパス	上位ディレクトリの開発履歴
CVS 開発日時，電子メール送信日時	同時にコミットされた CVS 情報

表 4: 開発課程による関連検索

類似性による関連分類

この分類では，着目した Knowledge に対して，類似 Knowledge をデータベースから検索する．提供される関連は以下の 2 つである．

1. 各システムにおける開発情報の類似性
   ここでは，それぞれのシステムに蓄積された情報の類似性をもとに着目 Knowledge に類似した Knowledge の検索を行う．入力として与える情報や検索結果として得られる Knowledge は図 5 の通り．

入力	検索される開発情報
ソースコード，差分情報 (CVS)	類似ソースコードを持つ CVS 情報
Subject(E-mail)	subject が同じで送信が近い E-mail
カテゴリ・class・重要度・状態・優先度 (GNATS)	情報が合致する GNATS 情報

表 5: 類似性による関連検索

2. キーワードによる類似性
   キーワードは，4.1.1 節で述べたように，その文書を特徴つける語句である．そのため，多くのキーワードが一致するものは内容が類似している考えられる．ここでは，着目した Knowledge にキーワードがある場合には，その語句が抽出された Knoweldge の検索を行う．

開発者情報による関連分類

この分類では，着目した人に対して，関連する人と Knowledge の検索を行う．4.1.1 節で開発者の重み付けと役割付けについて説明した．ここでは，着目した開発者と関連のある開発者の検索や，役割を踏まえた Knowledge の検索を行う．入力として与える人の情報とその結果得られる人や Knowledge は図 6 の通り．

入力	検索される開発情報
開発者，協力者 (CVS)	役割，該当開発者の開発回数，開発範囲の検索
送信者，受信者（電子メール)	同じ開発者間でやりとりされたメール
起草者，管理人，修正担当者 (GNATS)	同じ役割で行われた GNATS 情報

表 6: 開発者による関連検索

4.4.3 トピックコミュニティの検索

先に述べた手順で，ユーザはシステムが検索した開発情報の中から，必要な人や Knowledge の選定を行う．システムはその選定に対して，関連の分類を提供する．その分類の中からユーザは，自身の興味や関心に基づいて関連の選択を行う，そうして，システムは選定情報の関連を辿る．そこで得られた開発情報に対して，再びユーザが検索結果の人や Knowledge を選定する．これを繰り返していくことで，ユーザは求めるトピックの抽出を図る．そして，そのトピックを得たときに形成されているサブコミュニティこそが，トピックコミュニティであるといえる．

5 システムの実装

本節では，システムの構成について説明する．

システムは5つのセクションで構成される (図 10 参照)．本システムは，我々の研究グループで開発した CoxR[34] を拡張し，その機能を統合することで実現を行う．CoxR は CVS，電子メールアーカイブの履歴情報を抽出し，それらの関連情報をユーザに提供するシステム．そこで，そこに GNATS の履歴情報を追加して再構築を行う．まず，開発情報抽出部で CVS, E-mail，GNATS の人，Knowledge，ファイルの識別情報を抽出し，履歴情報データベースの構築を行う．次に，抽出した人と Knowledge の関連付けを行い関連データベース部を構築する．システム制御部では，ユーザの入力をもとにデータベースを検索し，ユーザに検索結果を表示する．

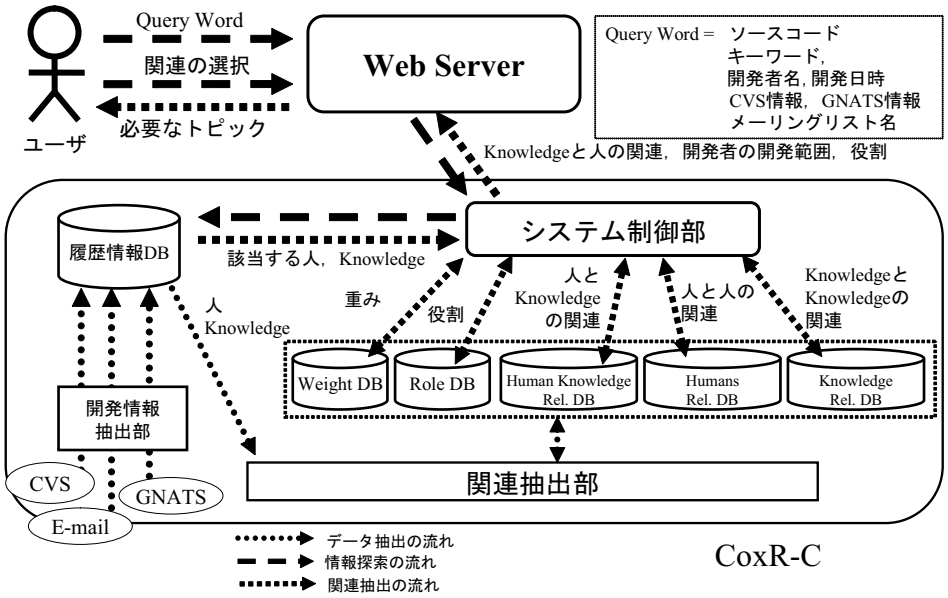

図 10: システム構成図

本研究では，実際のオープンソースソフトウェア開発である FreeBSD[38] の開発情報を対象として 1994 年から 2004 年までに蓄積されたデータをもとに，システムの実装を行った．FreeBSD 開発の実装を行った (表 7 参照).
記述言語として Perl と C を用い，データベースには PostgreSQL と BerkeleyDB を用いている．

開発環境は以下の通りである．

- CPU : Pentium4　1.5GHz

システム	対象データ	総数
CVS	ソースコードの履歴情報	ファイル数 57822, リビジョン数 618186
E-mail	ソースツリーへの変更に関するメール	213723 通 (cvs-*で始まる mailing list)
GNATS	GNATS に報告されたバグ情報	82350 件

表 7: 対象データ

- RAM : 1GB
- OS : Debian GNU/Linux 3.0
- データベース: PostgreSQL 7.4，BerkeleyDB (perl 5.8.4)
- Web サーバ : apache2.0

5.1 開発情報抽出部

この部分では，CVS，E-mail，GNATS の開発履歴情報をファイルごとに perl の正規表現を用いて，構文を解析を行う．そして，4.1.1 節で述べた人と Knowledge，各ファイルの識別情報を抽出し，それらを履歴情報データベースに格納する．

CVS 情報解析部

CVS 情報の履歴情報は図 6 のように定められている．そのため，各リビジョンごとに解析を行い，識別情報，人，Knowledge の抽出を行う．キーワード抽出に関しては，開発ログを単語ごとに切り分け，TF-IDF 法 [21] で重み付けを行い，各文書中で一定以上の重みの語句を抽出した．

ソースコードの差分情報の取得には，先に我々の研究グループで開発した CoDS[37] のデータベース作成の手法を用い，ソースコードをトークンに抽象化することで，造をデータベースに登録する．

E-mail 解析部

E-mail アーカイブに蓄積されている E-mail は図 7 のように定められている．そのため，文書ごとに解析を行い，識別情報，人，Knowledge の抽出を行う．ここでも，Subject，本文に対して TF-IDF 法を用い，文書中で一定の重みがある語句をキーワードとして抽出する．

GNATS 情報解析部

GNATS 情報は図 8 のように定められている．そのため，ファイルごとに解析を行い，識別情報，人，Knowledge の抽出を行う．また，Synopsis や Description，Audit-trail に記載されている文書を単語ごとに切り分け，TF-IDF 法を用いることで，キーワードの抽出を行う．

履歴情報データベース

CVS のリビジョン，E-mail・GNATS の各ファイルごとに一意な ID を振り当て,「人」と「Knowledge」,「識別情報」を登録する．さらに，差分情報を抽象化した情報もパターンごと蓄積する．類似コード比較のデータベースは，CoxR で用いたデータベースをそのまま利用する．

システムはここに蓄積された人や Knowledge，識別情報をユーザに表示する．このデータベースは，ユーザに提供する開発情報を含んでいる．

5.2 関連抽出部

履歴情報データベース (5.3 節参照) に登録された人と Knowledge を対象として，4.1.2 節で説明した条件をもとに，人と Knowledge の関連付けを行う．その中で H to H の関連に該当するものは“ Humans relation DB ”，H to K に該当するものは“ Knowledge Human relation DB ”，K to K の関連に該当するものは“ Knowledges relation DB ”に登録する．

また，4.1.1 節で説明したように，CVS のリビジョン番号やメーリングリスト名，GNATS の参加内容で開発者の役割付けを行い Role DB に蓄積する．

さらに，開発リビジョン数やメールの送信数，GNATS 情報への参加数・基本情報 (class, Priority) から人の重み付けを行い，これらの情報を Weight DB に蓄積する．

5.3 関連データベース部

ここでは，関連抽出部で関連付けられた人や Knowledge の情報とそれが含まれている ID を登録する．これらのデータベースは，ユーザが選定した人や Knowledge に対する関連分類の提供や検索に用いる．それぞれ関連する情報をキーとして登録するが，どちらの情報でも関連するキーを検索することができる．

Knowledge Human relation DB

このデータベースでは,「人」と関連する Knowledge の「ID」を登録する．ここでは，開発情報を表示する際に，その ID が関連している人の情報を提示したり，人が関わった開発

内容を検索する際に用いられる．

Human relation DB

「人」と関連する「人」と，その強度を登録する．人と人の情報を検索する際には，このデータベースを元に検索を行う．

Knowledge relation DB

このデータベースは，Knowledge を含む開発情報の「ID」とそれに関連する Knowledge の開発情報の「ID」が登録される．ここでは，先に 4.1.2 節に示した 7 つの条件ごとにデータベースを作成する．

ユーザが関連を選択した際に，それぞれの条件ごとに該当するデータベースを検索する．

Role DB

各役割ごとに該当する人の情報と含まれている開発情報の ID を登録する．人に関連する役割を抽出する際に，人と ID を検索することでその役割の検索を行う．

Weight DB

人と重みの値，それが含まれている開発情報の ID を登録する．人に関連する重みを抽出する際に，人と ID を検索することでその役割の検索を行う．

5.4 システム制御部

データ表示は CGI であり，Web インタフェースを利用している．ユーザから受け付けたクエリをもとに履歴情報 DB を検索する．検索結果に対して，ユーザが開発情報を選定すると，システムは次にその開発情報に対し，関連データベース部の中で存在する関連の分類を表示を行う．ユーザがそれを選択すると関連に応じて，該当する DB を検索し開発情報の提供を行う．

システムとやり取りを行うことで，ユーザはトピックの選定と必要なトピックコミュニティの検出を目指す．

5.5 実行手順

まず，ユーザは導入画面 (図 11) からトピックの検索を開始する．

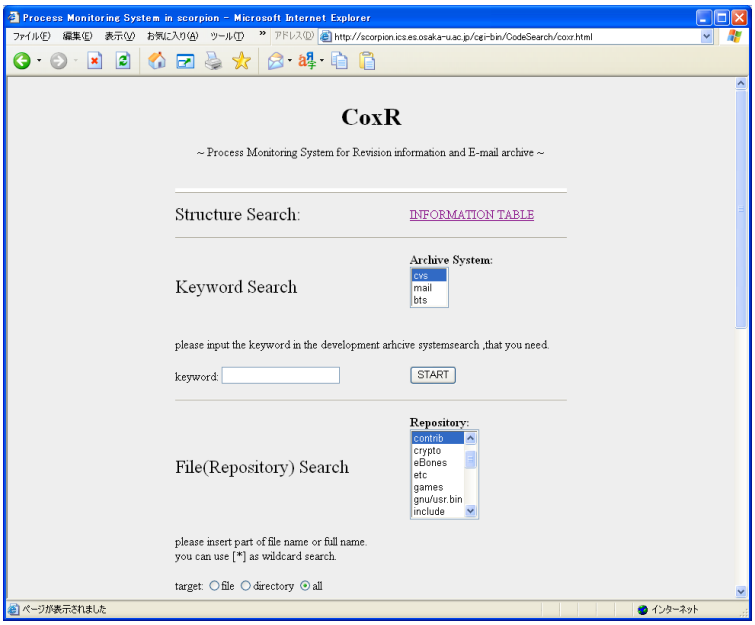

図 11: 導入画面

検索方法は，キーワードによる検索（図 12），類似コードによる検索（図 13），CVS のファイル名やディレクトリ名による検索，開発情報による検索（図 14，図 15）を行う．それらの中からユーザは必要な Knowledge の選定を行う（図 16）．選定した Knowledge は複数の関連分類が表示される (図 19，図 18)．そのため，これらの関連の中からユーザが必要な関連を選定し，それを繰り返すことで必要なトピックの抽出を図る．

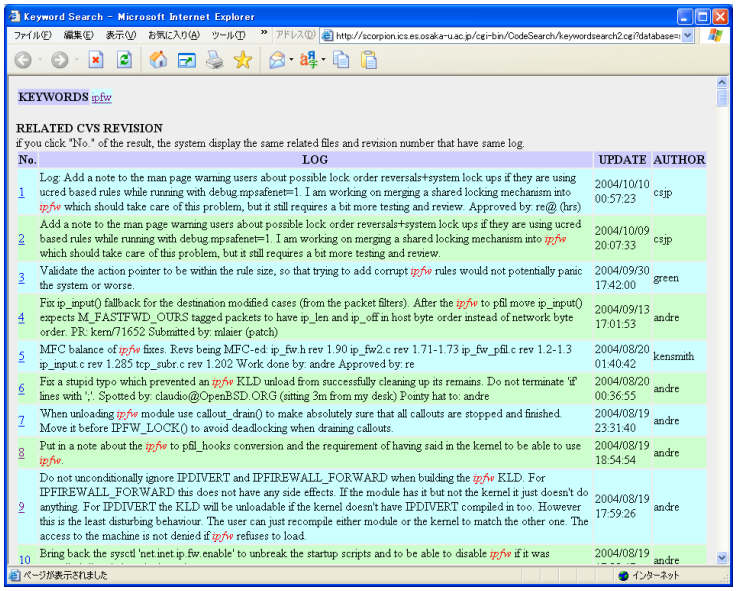

図 12: キーワード検索

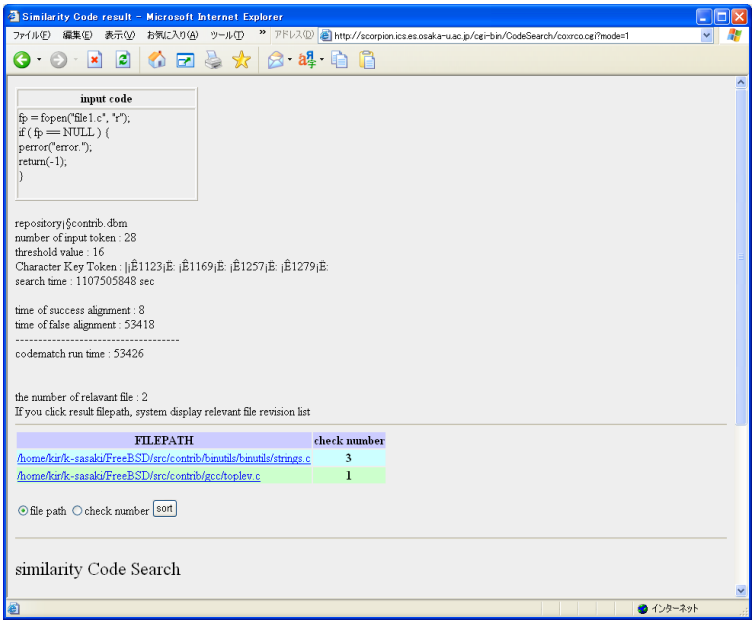

図 13: 類似コードによる検索

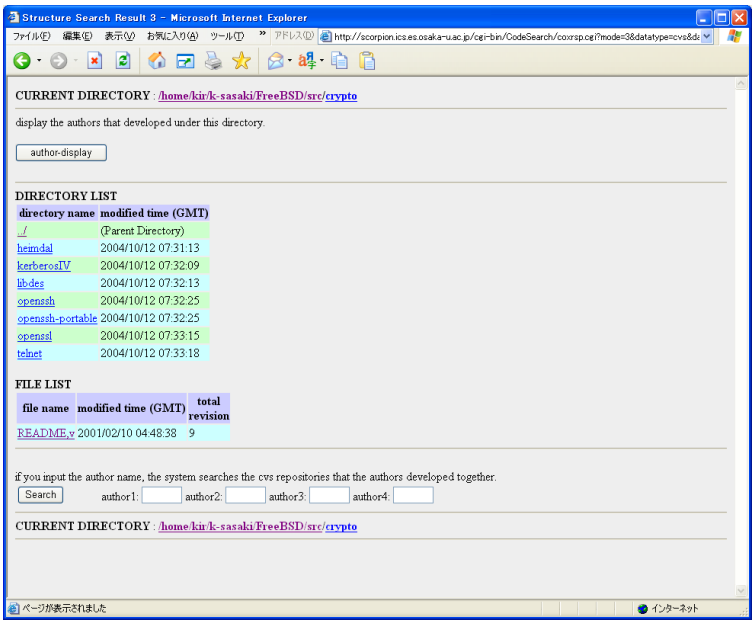

図 14: 開発情報による検索 (CVS)

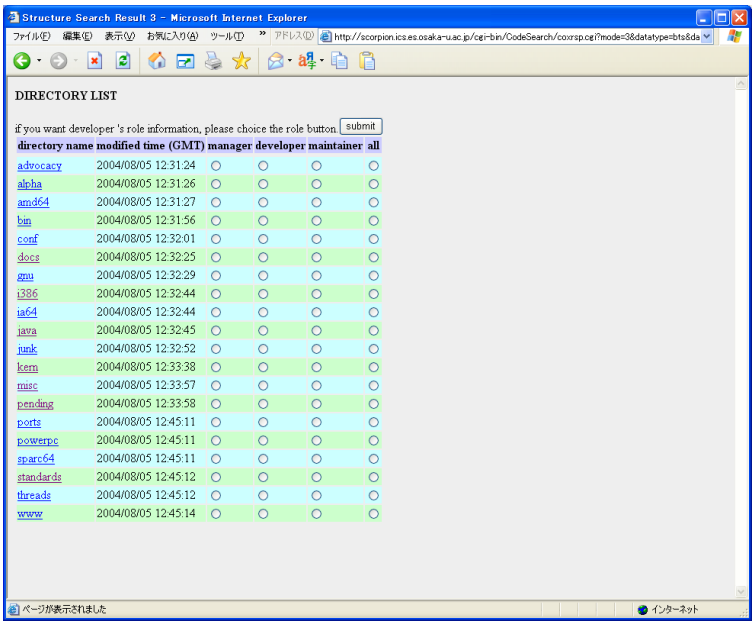

図 15: 開発情報による検索 (GNATS)

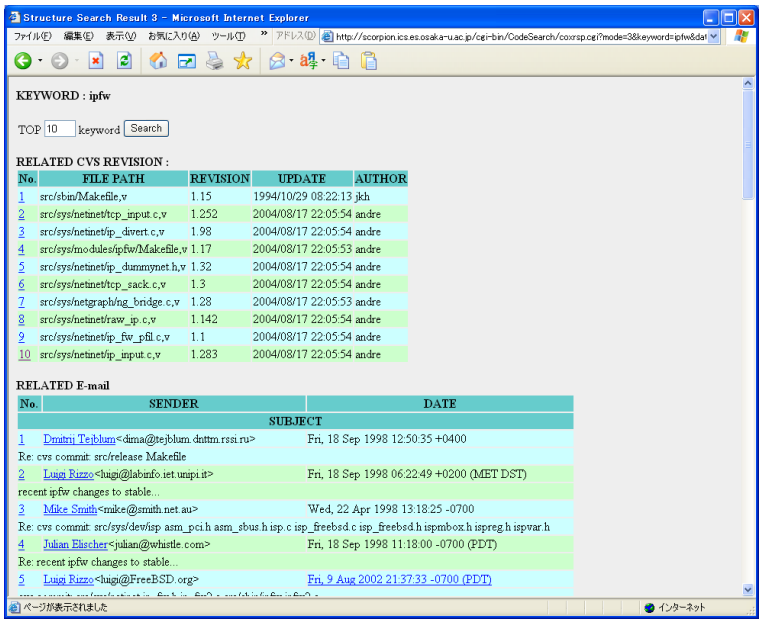

図 16: キーワード ipfw による関連

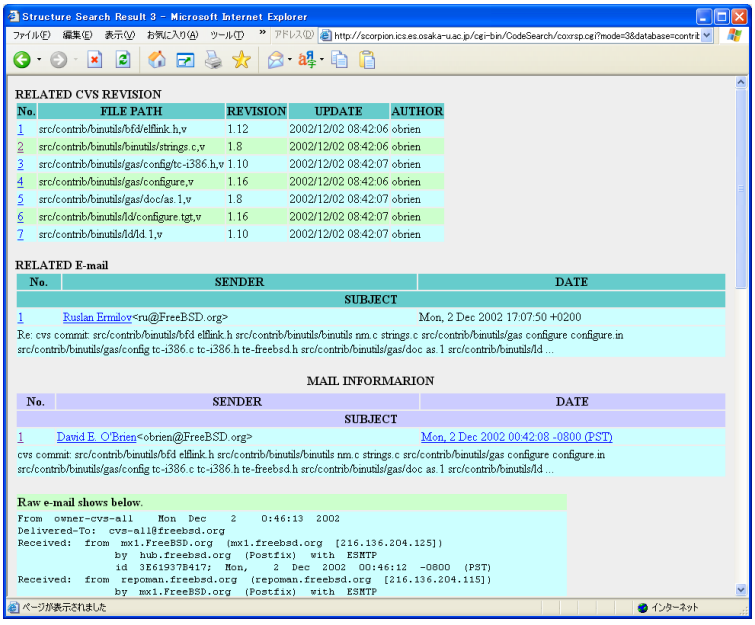

図 17: E-mail 情報と開発の関連

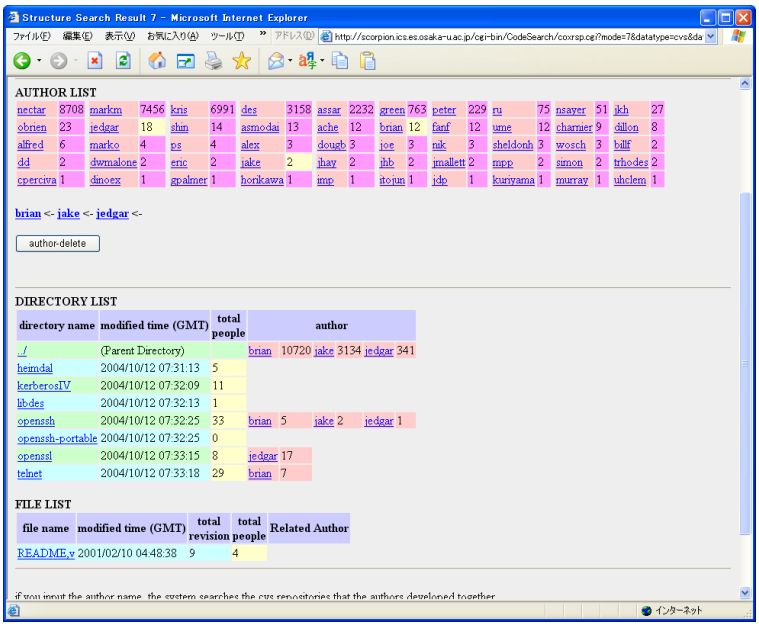

図 18: CVS 情報と開発者の関連

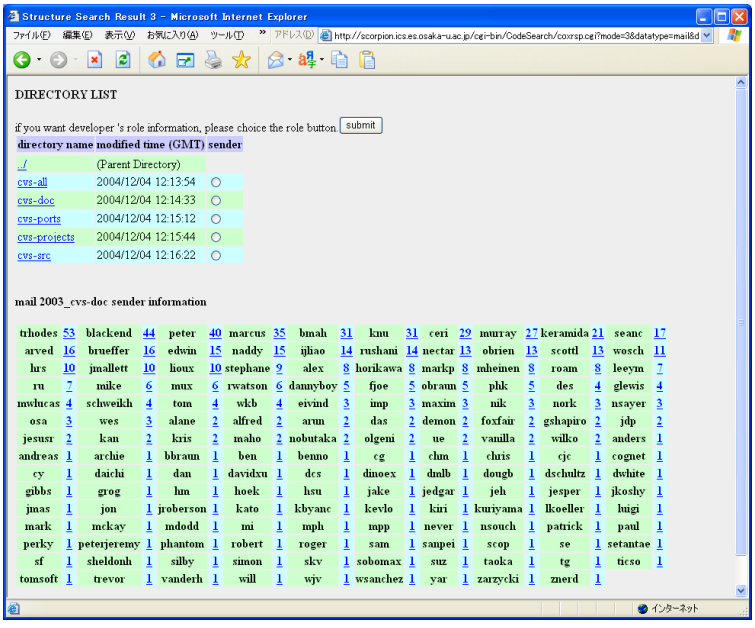

図 19: E-mail 情報と開発の関連

6 システムの評価

6.1 システムの運用

本システムを apache2 のウェブサーバ上 (http://scorpion.ics.es.osaka-u.ac.jp/cgi-bin/CodeSearch/coxr.html) で稼動した．そして，FreeBSD 開発の多くの開発者が参加しているメーリングリスト“ FreeBDS-hackers ”と“ FreeBSD-current ”に対して告知を行い，本システムの自由な使用に利用を呼びかけた．告知文には，本システムの趣旨と簡単な使用説明の記載と，本システムの動作サーバへのアドレスを記載している．各ユーザはサーバに対して自由にアクセスを行うことができ，サーバ側ではアクセスしてきたクライアントに対して cookie を発行で，その動作履歴を取得した．告知は，1 月 31 日と 2 月 7 日の二度行い，期間として，1 月 31 日から 2 月 10 日までの 11 日間のシステムの利用状況についてログを取得した．なお，システムは現在も稼動している．

以降では，ユーザの利用結果から，システムのどの機能がどれくらい使われているかを確認し，ユーザのシステムの検索手段や入力内容について追跡調査を行った．それをもとに，あらかじめ想定される利用パターンとの比較を行った．

6.2 想定される利用パターン

本節では，システムの利用用途を想定する．以下では，想定されるの想定利用パターンについて説明する．

1. プロダクト情報の再利用
   自身が着目したい機能を有した CVS 情報に着目し，それに対して，関連するソースコードやドキュメントファイルの検索を行うことで，機能実装の再利用することを図る．

2. 過去の議論の抽出
   調べたいキーワードが含まれている電子メールを検出し，それに関連する電子メールや，関連するキーワード，行われた CVS 情報や，修正された GNATS 情報について抽出を図る．

3. バグ修正履歴の利用
   抱えているバグに類似した部分を修正したソースコードや GNATS 情報を検出し，それに関する CVS や E-mail を辿ることで，該当するバグの解決方法を抽出する．

4. プロジェクト理解
   着目しているプロダクトに対して，該当期間に参加していた開発者や更新された CVS

情報，やり取りされた E-mail を検索することで，着目プロジェクトについての理解を深める．

6.3 利用履歴の内容

延べ 79 名のユーザが，検索および閲覧作業を行った．この中で，実際に意味のある検索および閲覧を行ったユーザの情報のみを結果として抽出することを考える．本研究では，システムにアクセスしたユーザの中で，検索を 3 回以上行ったユーザを利用ユーザとして，調査の対象と考える．その結果，今回対象となるユーザは 31 名であった．彼らは，それぞれ独立して検索，閲覧が行い，協調して作業を行っているといったような傾向は見られなかった．

31 名のユーザが行った検索方法は，図 8 の通りである．

検索方法	総数	検索全体に対する割合
キーワード検索	48	0.303797
開発時間検索	23	0.14557
開発者情報検索	24	0.1518987
類似コード検索	16	0.101266
開発内容検索	47	0.297468
合計	158	1

表 8: トピックの検索方法

今回の分析では，追跡ログからユーザの利用状況を再現するため，ユーザが，どの時点で人や Knoweldge を選択したのかという判定を厳密に行ことが難しい．そこで，開発情報を閲覧した後で，その開発情報に含まれる関連の分類の中から次の関連情報の検索を行ったり，そこで検索を終了した場合に，その開発情報に含まれる人か Knowledge の選定を行ったとみなした．ここでは，人と Knowledge のどちらの選定を行ったかを判断することは難しいため，以降では,「開発情報の選定」と述べる．

実際に，30 名中 18 名のユーザが，開発情報の選定を行った．さらに 12 名の開発者は，その選定した開発情報に対して，関連の分類を選択していた．選択された分類を調べると，分類を選択が合計で 20 回行われ，開発内容に関する分類を辿った回数が 10 回，類似性による関連を辿った回数が 2 回，開発者による関連調査が 8 回行われた (図 9 参照).

	開発内容に関する分類			類似性に関する分類		開発者に関する分類		
選定情報\	開発遷移	識別情報	開発課程	システム	キーワード	CVS	E-mail	GNATS
CVS	3		5			6		
E-mail		2			1		2	
GNATS					2			

表 9: 関連分類の選定回数

6.4 利用履歴の分析

本節では，先に示した利用履歴の分析を行う．最初に本システムの検索状況による分析を行い，次に実際に 6.2 節に示した想定する利用パターンと比較を行うことで，実際の利用状況についての分析を行う．

6.4.1 トピックの検索方法についての分析

表 8 に示されるように，今回の利用履歴では，キーワードによる検索と，開発内容による検索が多く用いられている．検索の過程を追跡すると，ユーザはキーワード検索で情報を絞り込んだ後に，開発内容のよる検索を行うことで，開発情報を選定するケースが多く見られた．そのため，これらの検索方法は，ユーザにとって有用であると言える．また，開発者や開発時刻に基づく関連も，キーワード検索による絞り込みの後に開発内容による検索とともに併用して用いられていため有効に活用されていると言える．

一方，類似コード検索について考察する．本システムでは，CVS の差分情報と，入力コードを比較することで，類似ソースコードの検索を行う．そのため，比較的短いソースコードを対象として，設計が行われている．そのため，ユーザが長いソースコードを入力として与えた場合には，一部が類似していても大部分が一致しないという結果になり，類似度の閾値が低くなってしまう為に，判定がうまくいかないケースが見られた．以上のことから，実際の検索に用いるにあたり，長いソースコードを対象として検索を行う必要性が課題として挙げられる．さらに，ユーザインタフェイスの問題が挙げられる．本システムは，入力場所で，それぞれ別の検索を行うように設計している．そのため，キーワード検索の欄にファイル名や開発者名を入れて検索を行うというユーザが存在した．本システムの設計では，入力スペースを分割することで，ユーザの目的に応じた検索を行うことが目的であったが，この問題も今後，考慮すべき点と言える．

6.4.2 関連分類の選定についての分析

本節では，想定された利用パターンと蓄積された利用履歴の比較を行う (図 9 参照).

1. プロダクト情報の再利用
   今回の利用履歴では，最初に選定する情報が CVS 情報であることが多かった．その中でも，同一の開発者による分類や，開発課程に関する分類を選択するケースが多く，プロダクトの機能についての情報抽出を図る利用履歴が多かった.

2. 議論の抽出
   選定された電子メールに関する分類では，メールに含まれるキーワードに関する分類の検索の履歴が抽出された．しかし，予想していたスレッドを辿るような開発変遷に関する分類は一度も行われなかった．その理由として，今回対象としている電子メールは，主に CVS の登録に関するコメント等を中心としたメーリングリストであり，ユーザが同じスレッドの情報を必要としなかったことが挙げられる.

3. バグ修正履歴の利用
   この利用パターンに対しては，類似コード検索で選定された CVS 情報に対して，その差分情報の調査が行われている．GNATS 情報は全体的にあまり検索が行われなかった．その理由としては，キーワードをもとにした関連では，入力として与えられたキーワードが一般的なものであったため，限定した情報を提供できなかったことが考えられる．Class 情報や Category による情報検索を行ったユーザは一人であったが，そのユーザは情報を選定し，図 9 に示すように分類を選択した.

4. プロジェクト理解
   今回の利用履歴に該当する利用パターンは含まれなかった．今回の利用履歴で選定された開発情報は，同じシステム内の関連のみを含んだ開発情報が多く選定され．システムをまたがった情報抽出がなされなかった．大規模な人数で行うことにより，これがシステムの問題点かどうか分かると思われる.

以上のように，今回の利用履歴を分析すると，CVS 情報に関する検索や選定が多かった．理由としては，電子メールや GNATS の情報を検索する際にキーワード検索が多く用いられた．しかし，これらの情報は文書量が多いため単語の数も多くなる．そのため，一般的なキーワードは多くの文書に含まれ，キーワードの質に検索結果が大きく左右される．また検索結果として表示できた場合でも，上位に来なければユーザは確認しないことが多い．そのため，現在の静的な重み付けに加え，多くのユーザが使った情報を上位に持ってくるなどの動的な重み付けを行うことは有用だと考える.

次に，利用履歴から，選定後の分類選択について考察を行う．ユーザは，選定した開発情報から関連を選択する時，検索結果の中から，再び関連を選定し，それを繰り返して関連の木構造を辿るケースはあまり見られず，最初に選定した開発情報を軸として，それに関連する情報を次々に閲覧していくケースが多くみられた．このことから，関連する情報は一つの群として，関連付けが実現されていることが考えられる．よって本システムの関連付けは有用であると言える．

6.5 個別の利用例に対する分析

次に実際に本システムを通して行われた個別の検索事例をもとに分析を行う．

6.5.1 利用例 1

このユーザは，FreeBSD 上で TV 表示やビデオキャプチャの開発に用いられるようなキーワード「bktr」や「conextant」「LeadTek」をもとに該当する Knowledge の選定を行った．その際に，キーワード Ledtek で検出された CVS 情報は，開発者 fjoe と roger の行った 4 つの開発ログ情報が検出されたが，fjoe の過去の開発内容を調査し，彼の開発内容に興味を持ち，それに関する開発情報を抽出した．

考察

この例では，キーワード探索をもとに，開発情報を辿り，開発者情報を検索している最も多く見られた利用例の一つである．この例では，キーワード検索から，必要な情報を選定するまでに，その分野に関連したキーワードを複数用いなければならなかった．この例からも質の良いキーワードが入力されない場合，検索が遠回りになる可能性も含まれていることが分かった．そのため，現在行っている静的な重み付けに加え，語句間の類似性について考慮することが今後の課題として挙げられる．

6.5.2 利用例 2

ユーザは，パケットフィルタに関する検索を行うため，pf というディレクトリ名の検索を行ったその中で特定のパスを発見し，そのディレクトリ構成を辿ることで src/sys/contrib/pf/net/pf.c,v に着目した．そのリポジトリ情報は，mlaier と dhartmei,kan によって開発が行われていたが，開発者の開発回数をもとに開発者 kan に興味を持ち，彼が登録したリビジョン 1.17 を調査した．

考察

ユーザが調べたいリポジトリの大まかな部分と内容に見当がついていれば比較的容易にこのような情報を取り出すことが出来るといえる．今回はユーザが開発回数の少なかった開発者 kan に着目したため，容易に彼の開発情報の取得が行われたが，そうでない場合には，同じ開発者で多くの開発を行っているような場合には情報を抽出することが困難になることが考えられる．現在の UI はユーザが選択しなければ役割や重みといった情報を表示していない．そのためそれらの情報をユーザの邪魔にならないような形で表示することを検討することは有用であると言える．

7 まとめ

本研究では，過去の履歴情報の中に蓄積された有用なトピックを効率よく抽出することを目的として，トピックに関連のある開発情報と開発者の構成されるトピックコミュニティの検索を行うための手法の提案を行った．さらに実際に既存の CoxR を元にシステムの実装を行い，その評価について考察を行い，本システムを利用することでユーザは膨大な開発履歴情報の中から効率よく自身が必要なトピック抽出を図ることが可能であることを確かめた．

今後の課題としては，以下の事柄が挙げられる．

- 役割の細分化

  現在の役割分類は，基本的な項目に沿ったものであるが，様々なユーザの開発の目的を考えたとき，それらの役割によって着目したい開発者の情報は異なると考えられる．そこで，一意的な開発者の役割付けにとどまらず，プロジェクトに応じたの役割の細分化を考える．

- 抽出したトピックコミュニティの再利用

  ユーザの求めるトピックが類似している場合，先に他人が着目したトピックコミュニティを推薦することは，トピックコミュニティ検索の手助けになると考えられる．そこで強調フィルタリング [30]，[31] などの手法を用いることにより，検索されたトピックコミュニティの再利用を図る．

- 重み付けの改善

  現在は開発情報に応じて重み付けを行っているが，これはデータベースを作成した際に一意に定まってしまう．そこで，利用状況やユーザの好みに応じて重みの表示を変更することを考える．

- ユーザインターフェイスの改良

  現在，Knowledge の情報表示は開発情報単位で行っている．しかし，この情報表示は，一覧として閲覧することには向いているが，Knowledge 一つ一つに特化して着目することを考える場合には不向きであるとも言える．そこで，必要に応じて特定の Knowldege のみに着目することができるユーザインターフェイスの作成を考える．

謝辞

本研究の全過程を通して，常に適切な御指導および御助言を賜りました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 井上 克郎 教授に心より深く感謝いたします.

本論文を作成するにあたり，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情報科学研究科コンピュータサイエンス専攻 楠本 真二 助教授に心から感謝いたします.

本論文を作成するにあたり，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情報科学研究科コンピュータサイエンス専攻 松下 誠 助手に心から感謝いたします.

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情報科学研究科コンピュータサイエンス専攻井上研究室の皆様に深く感謝いたします.

参考文献

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

[2] H. Agrawal, R. A. DeMillo and E. H. Spafford: “An execution backtracking approach to program debugging”, IEEE Software, pp.21-26(1991).

[3] H. Agrawal, R. A. DeMillo and E. H. Spafford: “An Execution Backtracking Approach to Program Debugging”, Techmical Report SERC-TR-22-P, Software Engineering Research Center, Dept. of Computer Sciences, Purdue University, IN, 1989.

[4] Ulf Asklund, Lars Bendix, Henrik B Christensen, and Boris Magnusson, “The Unified Extensional Versioning Model”, 9th International Symposium, SCM-9, LNCS1675, pp.100–122, 1999.

[5] Bugzilla,
http://www.bugzilla.org/.

[6] 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.

[7] Reidar Conradi and Berbard Westfechtel, “Version models for software configuration management”, ACM Computing Surveys, Vol. 30, No.2, pp.232–280, June 1998.

[8] D. Cubranic and G. C. Murphy. “Hipikat: Recommending pertinent software development artifacts”, In Proceedings of the 25th International Conference on Software Engineering (ICSE 2003), pp.408-419, Oregon, USA, May.2003.

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

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

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

[12] GNATS,
http://www.gnu.org/software/gnats/.

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

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

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

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

[17] H. Gall, M. Jazayeri, and J. Krajewski: “cvs release history data for detecting logical couplings”, in International Workshop on Principles of Software Evolution (IWPSE 2003), pp.13-23, Helsinki, Finland, Sep.2003.

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

[19] 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, Sep.2001.

[20] 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

[21] 北, 津田, 獅々堀, “ 情報検索アルゴリズム ”, 共立出版, 東京, 2002.

[22] A. Mockus, and L. G. Votta. “Identifying reasons for software changes using historic databases”, In Proceedings of International Conference on Software Maintenance (ICSM 2000), pp.120-130. California, USA, Oct. 2000. IEEE.

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

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

[25] 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.

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

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

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

[29] 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.

[30] P. Resnick and H. R. Varian, “ Recommender Systems ”, Communications of the ACM, Vol.40, No.3, pp.56-58, 1997.

[31] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “ GroupLens: An Open Architecture for Collaborative Filtering of Netnews ”, Proceedings of ACM 1994 Converence on Computer Supported Cooperative Work(CWSW ’94), pp.175-186, 1994

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

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

[34] 佐々木啓, 松下誠, 井上克郎, “ 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 ”, 電子情報通信学会技術研究報告 Vol.103, No.189, SS2003-9, pp.19-24 Jul.2003.

[35] 佐々木啓, 松下誠, 井上克郎, “ 開発履歴情報に基づいたダイナミックコミュニティ選定支援手法 ”, 電子情報通信学会技術研究報告 Vol.104, No.571, SS2004-50, pp.1-6 Jul.2003.

[36] Temple Smith and Michael Waterman, “Identification of Common Molecular Subsequences”, J.Molecular Biology, 147, pp.195-197, 1981.

[37] 田原靖太, 松下誠, 井上克郎, “ 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, Mar.2002.

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

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

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

[41] ViewCVS, http://viewcvs.sourceforge.net/

[42] 葉雲文, 山本恭裕, 岸田孝一, “ 知識共創のためのダイナミックコミュニティ:理論・アーキテクチャ・応用 ”, DynC Symposium 2004, 東京, Nov.2004.