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.

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

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　開発者が抱えている開発上の問題点に対して，有用な「処理」や「知識」を保持している開発コミュニティを選定し，その開発内容をもとに問題点の解決を図ることができれば，開発コストを軽くすることができる．しかし，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，恣意的かつ相対的に変動すると考えられる．本研究ではこれを「Dynamic Community（DynC）」と定義し，開発情報と開発者の関連から DynC の選定方法を提案する．さらに，本手法を用いて実際のオープンソースソフトウェア開発の開発者と開発情報を対象とした DynC 選定のためのシステムの試作を行う．本システムを用いることで，莫大な開発情報の中から必要に応じて DynC の選定を行い，開発者の問題点を効率よく解決することを図る．

キーワード　ダイナミックコミュニティ，開発履歴情報，オープンソースソフトウェア開発

Extracting Method of Dynamic Communities from Software Development Histories

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　The development history imformation includes much information which is useful to future development. Developers can obtain a deeper understanding about former development “knowledge” by reviewing an archive in the software development, and it is expectable that they help developers. Stated another way, we think that developers have “knowledge” in their development. We call the relation group of these “developers and knowledge” to “ Dynamic Community”, because developer and knowledge are floating for each developer which needs history information. In this paper, we purpose extracting method of Dynamic Community from software development histories. And, we develop the system for extract effectively “Dynamic Community”. Then we expect the system will enable developers to reduce development costs and improve software quality.

Key words　Dynamic Community, Development history information, Open source software development

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，リビジョン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．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．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．これらの開発情報とそれに関わる開発者達を一まとまりのコミュニティとして扱う場合，そこから有

用なサブコミュニティを選定し，その「処理」や「知識」を通して開発の理解や抱えている問題の解決方法を得ることが出来れば，開発コストを軽くすることができる [2].

しかし，蓄積された膨大な情報の中から，開発者にとって有用なコミュニティを的確に選定することは容易ではない．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，この恣意的かつ相対的に変動しうるコミュニティを“ Dynamic Community(DynC) ”[11] と定義し，開発の履歴情報と開発者の関連付けからの開発コミュニティの抽出をおこない，そこから，DynC の選定手法についての提案を行う．さらに，本手法を用い，実際のオープンソースソフトウェア開発に関わった開発者と開発情報を対象とした DynC 選定のためのシステムの試作を行う．本システムを用いることで，莫大な開発情報の中から DynC の選定を行い，開発者の問題点を効率よく解決することを図る．

以降，2. 節では，Dynamic Community の概念と，そのオープンソースソフトウェア開発への適応について説明する．3. 節では，Dynamic Community の選定手法について提案を行う．4. 節では，システム実装について説明する．5. 節では，関連研究について考察する．最後に，6. 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における Dynamic Community

本節では，Dynamic Community の概念を説明し，それをオープンソースソフトウェア開発に適応することを考える．

2.1 Dynamic Community の概念

Dynamic Community(DynC) とは，複数の人と，その人が有する Knowledge から構成されるナレッジワークスペースの中に構成されるコミュニティである [11]．DynC は以下の特性を持ち，イベントの発生に応じて，関連する Knowledge とそれを有する人々を動的に選定することで構成される．

恣意性	人が必要とするイベントに応じて構成される
相対性	同一イベントであっても必要とする人によって変化する
一時性	イベントごとに発生し，イベントが終わると解消する

表 1　ダイナミックコミュニティの特性

2.2 オープンソースソフトウェア開発への適応

オープンソースソフトウェア開発に適応する場合，プロダクト開発に関わった開発者と開発者が扱う開発情報が，「人」と「Knowledge」に対応する．また，関連する開発者と開発情報から構成されるコミュニティを**開発コミュニティ**と定義するとき，これが「ナレッジワークスペース」に対応する．

本研究では，開発コミュニティの中には過去の開発の「処理」や「知識」といった有用な**トピック**が含まれていると考える．このトピックから構成される開発コミュニティ中のサブコミュニティを**トピックコミュニティ**と定義するとき，これを選定し，必要なトピックを抽出することで，今後の開発作業に役立てることができると考える．しかし，違う人が同じトピックを求めた場合に，同じトピックコミュニティを選定するだけでは万人の要求を満たすことはできない (相対性)．また，同一人物であっても，その時々で必要なトピックは変化するため，必要なトピックコミュニティも変化する (恣意性)．このように，トピックコミュニティは静的に定まらず，必要に応じて形成されるコミュニティであると言える (一時性)．

そこで，本研究では，このトピックコミュニティを DynC と対応付け，各人に対して有効なトピックを提供するトピックコミュニティの抽出を目指す．以降で，Knowledge，開発コミュニティ，トピックコミュニティについて詳しく説明する．

2.3 Knowledge

本研究では，リビジョン管理システム CVS [3]，電子メールアーカイブ，障害管理システム GNATS に蓄積された開発情報を Knowledge として扱う．Knowledge は開発を行った理由 (**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 つに分類される (図 1 参照)．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．

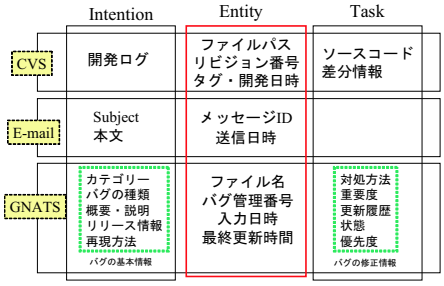

図 1　Knowledge の分類

2.4 開発コミュニティ

人と Knowledge には以下の関連が存在し，この関連から構成される集合を開発コミュニティと定義する (図 2 参照)．

（1） 人と Knowledge の関連 (H to K)
（2） 人と人の関連 (H to H)
（3） Knowledge と Knowledge の関連 (K to K)

以降，これらの関連は H to K，H to H，K to K と呼ぶ．

2.5 トピックコミュニティ

開発コミュニティ中には，複数のトピックコミュニティが存在する．トピックコミュニティは，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．

3. ダイナミックコミュニティ選定手法の提案

本節では開発コミュニティから，トピックコミュニティ(DynC) 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．本手法の手順は以下の通りである．

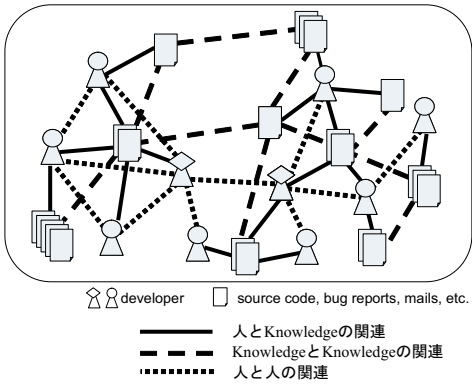

図 2　開発コミュニティ

（1） 人と Knowledge の関連抽出
（2） Knowledge による人の役割付け，重み付け
（3） トピックコミュニティの分析

最初に，システムに蓄積された開発情報から，関連を抽出する．(1)．次に，開発者の役割付け，重み付けを行い，開発コミュニティを特定する (2)．その中から着目すべき Knowledge を選定し，それをもとに複数のトピックコミュニティ群を分析し，トピックコミュニティから有用なトピックの抽出を図る (3)．

3.1 人と Knowledge の関連抽出

まず，CVS，電子メール，GNATS に蓄積された開発情報から，Knowledge の抽出を行う (図 1 参照)．図 1 の Entity は各 Knowledge の識別情報であるため，これを基準に H to K，H to H，K to K の関連抽出を行う．最初に，各システム内の関連付けを説明し，その後に，システム間の関連付けの説明を行う．

3.1.1 CVS 情報間の関連抽出

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 3 参照)．そのファイルの構文解析を行うことで，人と Knowledge の関連を抽出する．

```
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 3　ファイル単位のリビジョン情報

（1） H to K 関連

author とコミットしたリビジョン間の関連抽出を行う．図 3 では，開発者 sobomax が src/bin/ln/ln.c,v のリビジョン 1.19, 1.20 の開発を行っていることが分かる．そのため，sobomax とこれらのリビジョン情報に関連付けを行う．

（2） H to H 関連

（a） 同一のファイル・ディレクトリにコミットした author

同一の CVS リポジトリ内で，トランクやブランチごとに，コミットを行った author に関連付けを行う．

（b） author と開発ログに記載された協力者

例えば，図 3 のリビジョン 1.20 の開発ログに“ `Reviewed by: bde` ”と記載がある．これは，sobomax がコミットする際に，開発者 bde が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できる．そのため，author と協力者に関連付けを行う．

（3） K to K 関連

（a） 同一リポジトリに含まれるリビジョン

(2a) と同様に，CVS リポジトリの中で同じトランクやブランチに含まれるリビジョン情報に関連付けを行う．

（b） 同じ開発者が近い時間にコミットしたリビジョン

開発者と更新時間が一致するリビジョン情報に関連付けを行う．なお，本研究では開発日時の誤差を ± 3 分程度とする．

（c） 前リビジョンとの差分情報が類似しているリビジョン

図 3 の履歴情報には，リビジョン間の差分情報も付加されている．これを比較し，前リビジョンとの変更内容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 [6] を用いて行う．

（d） 開発ログ中のキーワードが類似しているリビジョン

開発ログから重要な語句をキーワードとして抽出し，そのキーワードが類似しているリビジョン情報に関連付けを行う．キーワード抽出方法として，情報検索の分野で一般的に用いられる TF-IDF 法 [7] を用いる．

3.1.2 電子メール情報間の関連抽出

電子メールアーカイブには，ファイルごとに一つの電子メールの内容が記載されている (図 4 参照)．これらのファイルを構文解析することで，人と Knowledge の関連の抽出を行う．

```
rom 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 4　電子メール情報

（1） H to K 関連 E-mail とその中に，送信元・受信先として記載されているアドレスの開発者に関連付けを行う．

（2） H to H 関連

（a） 送信者 (from)，受信者（to, cc, bcc），返信者 (reply)

同じ電子メールの受信，送信，返信，転送に関わった開発者同士に関連付けを行う．

（b） 開発者と送信，受信したメーリングリストのアドレス

開発者 と H to K 関連のある電子メールにメーリングリストの記載があれば，メーリングリストと開発者の関連付けを行う．

（3） K to K 関連

（a） 電子メールとそれに対する返信・転送メール

受信したメールに対して，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．それをもとに，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．

（b） Subject が一致し，送信日時が近い電子メール

Subject が一致し，その中でも送信日時の近い場合には，電子メール間の関連性が高いとみなし，関連付けを行う．本研究では，送信日時の誤差は± 6ヶ月程度とする．

（c） キーワードの類似性が高い電子メール

3.1.1(3d) と同様に，電子メールのタイトルおよび本文から重要と思われる語句をキーワードとして抽出し，そのキーワードが類似している電子メール間に関連付けを行う．

3.1.3 GNATS 情報間の関連抽出

GNATS では，ファイルごとに一つの修正情報が記載されている (図 5 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，人と Knowledge の関連抽出を行う．

```
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 5 GNATS 情報

（1） H to K 関連

GNATS 情報には起草者や修正の責任者の名前が記載されている．これらの開発者と GNATS 情報に関連付けを行う．

（2） H to H 関連

同じ GNATS 情報の起草や修正に関わった起草者や責任者間に関連付けを行う．

（3） K to K 関連

（a） 同一のファイルに対して修正を行った GNATS 情報

同じ CVS リポジトリに対して，変更を加えたり，機能拡張を行った GNATS 情報間に関連付けを行う．

（b） GNATS 情報とそれが参照した GNATS 情報

過去に行われた GNATS 情報に対して，参照を行った場合，参照元の管理番号が記載される．その情報を抽出し，関連付けを行う．

（c） 類似するキーワードによる関連付け

3.1.1(3d)，3.1.2(3c) と同様に GNATS 情報の再現方法・概要・対処方法からキーワードを抽出し，キーワードが類似している GNATS 情報間に関連付けを行う．

3.1.4 異なるシステム間の K to K 関連の抽出

ここまでは，各システム内の Knowledge と人の関連抽出について述べたが，次に異なるシステム間の関連抽出について説明する．本研究では，異なるシステム間には K to K 関連のみが存在するとみなす．

（1） CVS と メールアーカイブ

（a） 電子メール中に特定のファイルパスが記載されている

（b） リポジトリ情報とコミットメールに対する返信メール

CVS にコミットした際に，システムからコミットメールが送信される．それに対する返信メールは，該当リビジョンに対して議論が行われているとみなし，関連付けを行う．

（2） CVS と GNATS

（a） 開発ログと GNATS 情報から抽出したキーワードの類似性が高い

（b） GNATS 情報にファイルパスが記載がある

（c） CVS の開発ログにバグ ID が記載されている

（3） メールアーカイブ と GNATS

（a） 電子メールにバグ ID の記載がある

（b） GNATS 情報に電子メール本文の引用がある

3.2 Knowledge による人の役割付け，重み付け

次に，Knowledge をもとに関連する人の評価を行う．

3.2.1 Knowledge による人の役割付け

プロダクト開発の目的は大きく分けて,「システム開発」「機能拡張」「デバグ」「保守」に分類され，目的に応じてシステムへの Knowledge の格納状況は変化する．そのため，Knowledge の格納状況から，それに関連する開発者の役割付けを行う．

- CVS 情報

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．また，開発の目印として「タグ」を付けることもできる．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 ”と役割付る．また，タグ情報を抽出し，それを格納した開発者の役割として対応付ける．

また，3.1.1(2b) のように，開発者はコミットする際に，開発ログに協力者の名前を記載することがある．彼らは，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 2 参照)．

- 電子メール情報

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．先に 3.1.2(2b) で，受信・送信したメーリングリストと開発者の関連付けを行っているため，メーリングリストの種類をもとに開発者の役割付けを行う．

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 2　CVS 情報における開発者・協力者の役割付け

- GNATS 情報

GNATS では，始めに，起草者が機能拡張やバグ修正の要求を登録する．その際に，要求の内容 (バグ修正，機能拡張など) を「class 情報」として登録する．それに対して，担当する開発者が決定され，要求解決に向けて作業を行う．そこで，起草者の役割は class 情報をもとに割り当てる．また，担当者も class 情報をもとに，バグ修正なら“ メンテナ ”．機能拡張なら“ デベロッパ ”として役割付けを行う．さらに，担当者を決定する開発者は要求の“ マネージャ ”として役割付けを行う (表 3 参照)．

class 情報	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 3　GNATS 情報における起草者・責任者・担当者の役割付け

3.2.2　開発者が行った Knowledge による重み付け

Knowledge への関わり具合から開発者の重み付けを行う．

- 開発回数

開発頻度が多い開発者は“ メイン ”，それ以外の開発者は“ サブ ”，また，開発者が開発局所的に開発を行っていれば，“ ローカル ”，広範囲を開発しているなら“ トータル ”として重み付けを行う．

- 開発時間

昔の開発よりも現在の開発に対して重み付けを行う．

- キーワード

TF-IDF 法 [7] で行ったキーワードの重み付けから，重みのある Knowledge に関連のある開発者に重み付けを行う．

3.3　トピックコミュニティの分析

先に提案した関連付けや重み付けから，開発コミュニティの抽出を行う．次に，そこから特定の Knowledge に着目し，関連するトピックコミュニティ群を導出する．これに対して分析を行うことで，必要なトピックの抽出を図る．

最初に，基点となる Knowledge の選定を行う．これは，4. 節に後述する CoxR [9] のようなシステムを利用することにより容易に行うことができる．

3.3.1　トピックコミュニティの抽出

着目した Knowledge をもとに，それに関連する人と Knowledge からトピックコミュニティの抽出を行う．ここで抽出されるトピックコミュニティは，ファイルパス，バグの管理番号，電子メールの Message-ID のように明示的に表現された Knowledge 間の関連とそこに関わる開発者から形成される．

3.3.2　トピックコミュニティの分析

次に，抽出したトピックコミュニティに対し，以下の手法で分析を行う．コミュニティを細く分けて複数のコミュニティ群

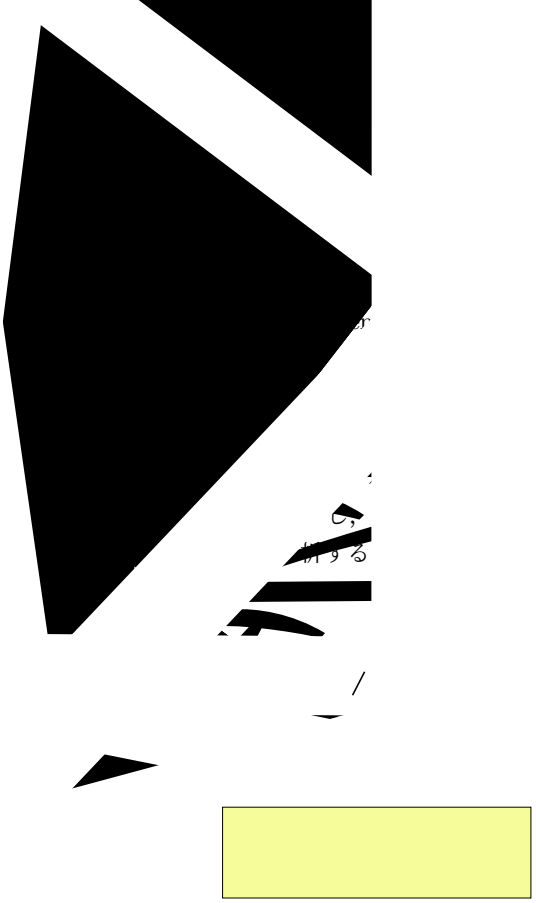

図 6　時間推移による分析

図 7　開発者グループによる分析

Ⅲ.　類似 Knowledge を有するトピックコミュニティの分析

K to K の関連を通して，選定したトピックコミュニティに類似しているトピックコミュニティの分析を行う．分析を通して，トピック間の共通 Knowledge，例外 Knowledge の抽出を図り，自身の必要とするトピックへの応用を考える (図 8 参照)．

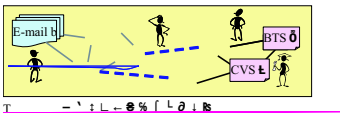

図 8　類似する Knowledge による分析

このようにして導き出せるトピックコミュニティは，必要に応じた分析から検出される (恣意性)．さらに，着目点に応じて細分化されたトピックを抽出できる (相対性)．また，あらかじめ定義された関連のみからは抽出されない (一時性)．そのため，DynC の特性を満たしていると言える．

4.　システムの実装

本研究では，現在，実際のオープンソースソフトウェア開発である FreeBSD [4] の開発情報を対象としてシステム開発を行っている．システムの構成は図 9 のようになっており，5 つの部分から構成される．

4.1　開発情報抽出部

CVS，GNATS，E-mail の開発履歴情報を抽出する．

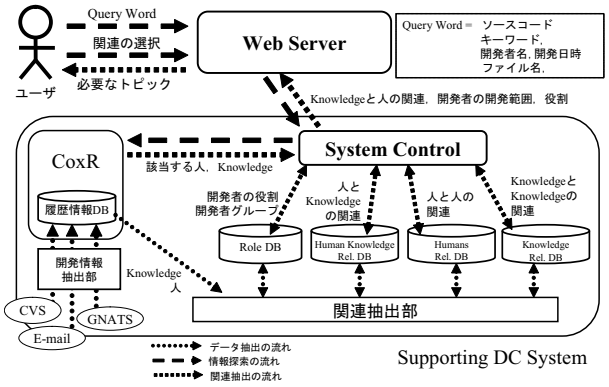

図 9　システム構成図

4.2　関連抽出部

抽出した開発情報に対して，人と Knowledge 間の三つの関連の抽出を行う．

4.3　データベース部

H to H の関連を抽出した“ Humans relation DB ”，H to K の関連を抽出した“ Knowledge Human relation DB ”，K to K の関連を抽出した“ Knowledges relation DB ”と，開発者の役割・重み付けの情報を保持する“ Role DB ”の 4 つのデータベースから構成される．これらの DB は，ユーザが最初の Knowledge に着目した際に，それに関連するトピックコミュニティの検出やトピックコミュニティ群の分析に用いる．

4.4　CoxR　部

この部分は，先に，我々の研究グループで開発された CoxR [9] の機能を拡張し，本システムの一部として統合して用いる．具体的には，CoxR がこれまでに対応していた CVS，電子メールアーカイブの開発情報に加えて，GNATS の情報を取り込み，ユーザにその関連を提供することで，着目 Knowledge を選定するための情報検索部として用いる．

4.5　システム制御部

ユーザとシステムの情報のやり取りを管理する．まず，ユーザからの検索要求を受けて CoxR に該当 Knowledge の抽出要求を行う．さらにその Knowledge とデータベースの情報と統合し，トピックコミュニティの抽出，さらに複数のトピックコミュニティ群の分析，トピックの選定を行う．

本システムは現在，関連抽出部の開発者の役割付け・重み付けを行う部分とそれに伴うデータベース部の一部 (Role DB) の実装を行っている．さらに実装が終了後に，実際に人や Knowledge の関連や重み付け・役割付けの評価が適切であるかを判断し，その調整を行う必要がある．

5.　関 連 研 究

5.1　リポジトリ解析

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，開発ログから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サイズの分析を行っている [8]．また，Harald Gall ら [5] は，CVS の履歴情報からクラス・ファイル・function 間の論理的な結合を抽出し，その関連情報をユーザに提供している．

5.2　コミュニティ分析

複数のシステム間のコミュニティ分析の研究は，Hipikat [1] があげられる．これは，CVS や BTS・電子メールに蓄積された開発情報の関連をもとに group memory を抽出し，ユーザが行っている処理に応じた group memory の提案を行っている．

しかし，これらの手法は，あらかじめ抽出した関連を必要に応じて開発者に掲示するため，同じトピックを探索した場合は，常に同じトピックコミュニティを抽出することになる．そのため，開発者やトピックごとに柔軟に対応することは難しい．

6.　ま　と　め

本研究では，オープンソースソフトウェア開発中に蓄積された開発情報をそれに関わった開発者の関連をもとに，過去の開発情報の中から有用な処理や知識を抽出する際に，開発者や必要なトピックに応じて，動的に形成される Dynamic Community 選定のための手法の提案を行った．さらに，その手法に基いたシステム開発についても紹介した．

今後の課題としては，先に挙げたように，提案手法に基いたシステム実装の完了，そして，その評価が考えられる．本システムを利用することで莫大な開発情報の中から効率よく DynC の選定を行い，開発者の問題点を解決し，ソフトウェア開発の支援となることを図る．

文　　献

[1] 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.

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

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

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.