特別研究報告

題目

ソフトウェア理解支援のための
閲覧状態復元機能付き Web ブラウザの試作

指導教員
井上 克郎 教授

報告者
今枝 誉明

平成 17 年 2 月 17 日

大阪大学 基礎工学部 情報科学科

平成 16 年度 特別研究報告

ソフトウェア理解支援のための
閲覧状態復元機能付き Web ブラウザの試作

今枝 誉明

内容梗概

ソフトウェア開発を行う際には，多くのソフトウェア開発支援システムが用いられている．その中でも，メーリングリストのアーカイブや部品検索システムといった，ソフトウェア開発の状況やプロダクトを閲覧するためのシステムは，過去のソフトウェア開発結果を理解する上で重要な役割を占めている．一般にこれらのシステムは，ユーザインターフェイスとして Web ブラウザを用いており，開発者は自由に閲覧することが可能である．しかし，それぞれのシステムは独立しているため，開発者は複数のシステムにまたがる情報閲覧を行う場合に全体のつながりを自らが管理しなければならないという手間がかかっていた．

そこで本研究では，Web ブラウザを用いるソフトウェア開発支援システムを対象として，複数のシステムを同時に利用する際に必要とされるインターフェイスを持つ Web ブラウザの試作を行った．具体的には，まず，ブラウザでの閲覧状態を逐次，履歴として保存し，またその内容をツリー状のグラフとして提示するインターフェイスを Web ブラウザに付加した．また，ある過去の時点における閲覧履歴を指定することにより，いつでもその時点で閲覧していた内容を復元するための機能を実装した．さらに，複数の利用者間で閲覧履歴を共有することにより，他の利用者が閲覧した内容に基づいて，これから閲覧することが望ましい情報の推薦機能を実装した．試作した Web ブラウザを使用して，あるソフトウェアシステムの調査を行った結果，効率的に閲覧でき，その有用性を確認した．

主な用語

ソフトウェア理解 (Software Comprehension)
Web ブラウザ (Web Browser)
協調フィルタリング (Collaborative Filtering)

目 次

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，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．

このように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，電子メールを用いたメーリングリストシステム，バグ追跡システムなど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．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．バグ追跡システムでは，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．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いる．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おり，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．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．また，これらのアーカイブの閲覧の際に Web ブラウザをユーザインターフェイスとして用いるものが一般的なってきており，開発者は自由に閲覧することが可能である．

しかし，これらのユーザインターフェイスはそれぞれ単独のシステムとして独立しているため，開発者は必要な情報を得るために複数のインターフェイスを同時に使い分ける必要がある．このとき，各システム間で関連のある情報を同時に見ることが多いが，システム間の連係が無いため，開発者自身が全体のつながりを管理しなければない．また，閲覧作業が長くなり，閲覧すべき問題が多岐に渡ってくると，こうしたシステム間の関連情報のみならず，本来解決すべき大元の問題までも見失いかねない．

そこで本研究では，Web ブラウザを用いるソフトウェア開発支援システムを対象として，複数のシステムを同時に利用する際に必要とされるインターフェイスを持つ Web ブラウザの試作を行う．具体的には，まず Web ブラウザで閲覧した内容を逐次，閲覧履歴として保存する機能，及びその内容をツリー上のグラフとして提示するインターフェイスを Web ブラウザに付加した．また，ある過去の時点における閲覧履歴を指定することにより，いつでもその時点で閲覧していた内容を復元するための機能を実装した．さらに，複数の利用者間で閲覧履歴を共有することにより，他の利用者が閲覧した内容に基づいて，これから閲覧す

ることが望ましい情報の推薦機能を実装した．

また，作成したシステムの有効性を確認する為に，実際のオープンソース開発に関連する情報を，作成したシステムを用いて調査した．その結果，調査途中で過去に一度閲覧した項目に関してさらに調査を行いたいと思った時に，保存された過去の閲覧情報を読み込むことで，過去に行った手順を繰り返すこと無く，調査を再開できることを確認した．また，それまでに蓄積された閲覧履歴を分析することで，有用であるにも関わらず，開発者が取得できなかった情報を推薦できることを確認した．このことにより，ソフトウェア開発者の負担が軽減され，ソフトウェア生産性の向上が期待できる．

以降，2 節では，オープンソースソフトウェア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．3 節では，試作したブラウザの概要と設計について述べ，4 節では実装について述べる．5 節では本システムを用いた場合に解決する問題の例を示す．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とその開発環境

本節では，オープンソースソフトウェアとその開発環境について触れ，オープンソースソフトウェア開発環境の持つ問題点を説明する．

2.1　オープンソースソフトウェア

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ソフトウェア開発と呼ばれ [19][23]，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[9] や Linux[16]，GNU[10]，Apache[1] 等は，オープンソースソフトウェアの代表である．

2.2　開発環境

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 [7] の一つである CVS(Concurrent Versions System)[2] [8][15] 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，バグ追跡システムなどについて説明する．

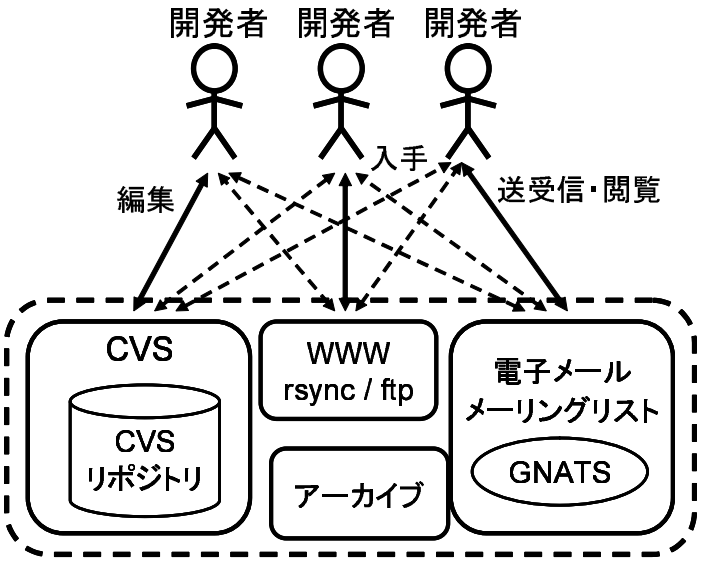

図 1: オープンソースソフトウェア開発環境の構成例

2.2.1 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．

- 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．

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[27] や CVS[2][8] といったシステムが標準で利用可能となっている．ClearCase[22] 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[18] や PVCS[17] 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 [11] も存在する．

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[27] 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[2][8][15] 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．

  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 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 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，同じ電子メールが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．

2.2.3 バグ追跡システム

バグ追跡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ためのシステムである．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バグ追跡システムには，GNATS や Bugzilla といったものが挙げられる．これらのシステムでは，まず，起草者がバグ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．それら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，開発作業の促進につながっていると言える．

2.2.4 クロスリファレンサ

クロスリファレンサとは，ソースコードを解析して関数の定義場所を抽出するためのツールである．関数の定義場所を抽出しておくことで，ソースコードを閲覧する際に，呼びだされている関数がどこで定義されているかを瞬時に知ることができる．

さらに，いくつかのクロスリファレンサではソースコードを HTML として出力することが可能である．その際，関数呼び出しが書かれている場所には，定義場所へのリンクが埋めこまれる．そのため，出力された HTML を Web ブラウザで開くことで関数の処理内容を調査することが非常に容易にできる．

クロスリファレンサとしては ctags や etags，GNU GLOBAL が挙げられる．

2.3 問題点

このように，オープンソース開発ではネットワーク上に存在するオープンソース開発環境にてプロダクトの管理が行われる．そしてオープンソース開発環境を構成する各システムは一般的に WWW を通じてユーザに情報を提示する．そのため，開発者は Web ブラウザを用いて容易に現在開発しているプロダクトの情報を入手することができる．

しかし，これらの WWW を利用した情報提示システム群には「システム間の連携不足」，「複雑なリンク関係」という二つの問題点が存在する．以下これら 2 つの問題点について述べる．

2.3.1 システム間の連係不足

オープンソースソフトウェア開発では，開発管理を効率良く行うことを目的として，SourceForge[25] や SourceCast[5]，OSDL(Open Source Development Lab.)[21] 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(Computer Supported Cooperative Work) 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．

これらのシステムはそれぞれ独立して存在しているが，それぞれが保持している内容間には関連が存在する．たとえばバグトラッキングシステムに登録されたバグ情報と，メーリングリストで交された，そのバグに関する議論とは密接に関連している．そして開発者は二つのウィンドウを用いて，これら関連する情報を同時に閲覧することも頻繁に行われる．

しかし従来のブラウザでは一つのウィンドウ単位での履歴のみを管理しているため，二つのページを同時に見ていたという情報を保持することができない．そのため，これら二つのページを見ていた状態を復元したい場合にはユーザがそれらページの URL を記憶しておかねばならない．

2.3.2 複雑なリンク関係

一般にソフトウェア開発においては大量のプロダクトが存在する．そのため，プロダクト情報を提供するシステムではプロダクトおよびそれらを結びつけるリンク構造もまた大規模なものになる．そのため，リンクを辿って調査を進めているうちに以前に見た重要な情報を見失ってしまう危険性が存在する．

例えば，クロスリファレンサを用いて main 関数から始まる一連の処理の調査をしている状況を考える．このときのページ遷移状態を図示したのが図 2 である．このツリーは，ノードが 1 つの URI を示しており，ページ A からページ B へのリンクをクリックして移動した場合にノード A の子にノード B を配置する．ページ B 以下の閲覧を終えてページ A にもど

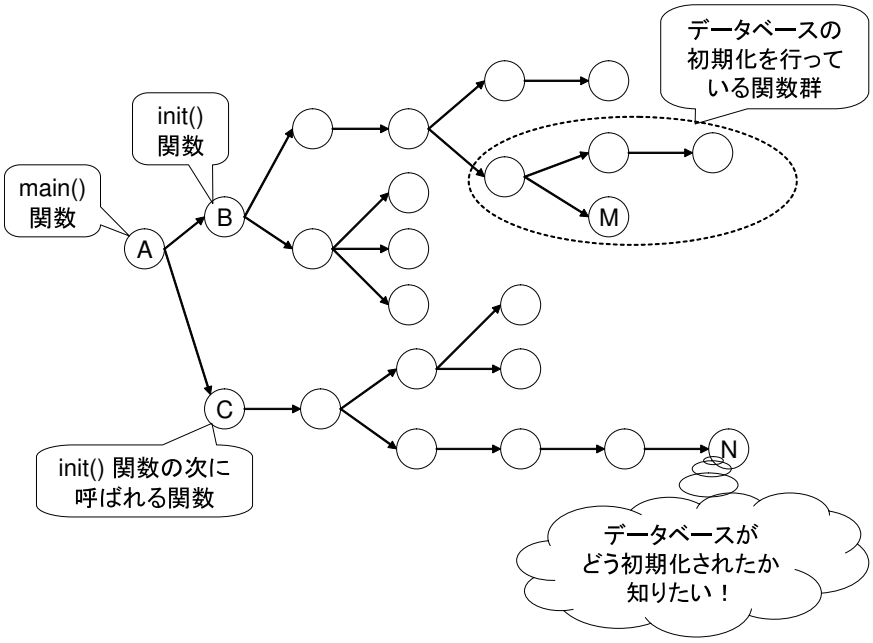

図 2: ページ遷移モデル図

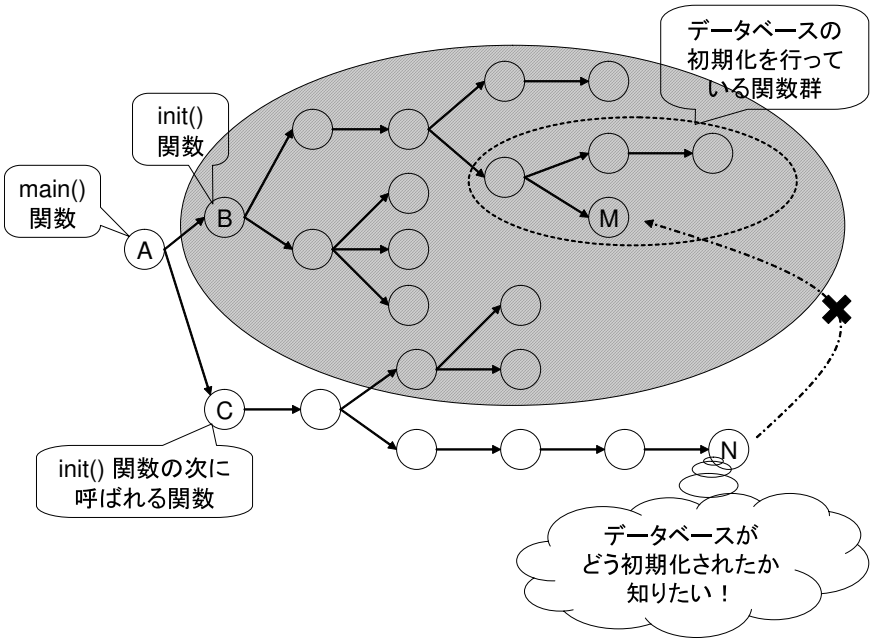

図 3: ページ N 到達時に保持されているヒストリ

り，ページ A 内の別のリンク C をクリックしたときには，ノード A の新しい子としてノード C が追加される．

図 2 は main 関数について記述されたページから呼びだされている関数を順に辿っていった様子を表している．このときページ M にはデータベースの初期値に設定された値が記述されているものとする．そして，ページ N まで遷移したときに，初期化したデータベースの内容を参照する必要が生じた．

しかし一般的なブラウザではヒストリとして現在見ているページと直接親子関係にあるページのみしか保持されておらず，ノード B から始まる履歴も保持されていない (図 3)．そのためページ M に再び辿りつくためには，ページ A からどのようなリンクを辿っていったかを思い出しながらリンクを探索しなければならず，リンクが膨大な場合には途中で辿れなくなってしまう危険がある．

このような状況に対しては，一般的なブラウザではノード M に対してあらかじめブックマークをすることで対応している．しかし現在調査しているプログラムに詳しくない閲覧者にとっては，どのページがブックマークすべき重要なページで，どこがそうでないかを判別することは非常に難しい．このように情報を閲覧している途中に以前に見た情報を参照したいという要求に，従来のブラウザでは十分応えているとはいえない．

なお，ここでは関数呼び出し関係を描画するクロスリファレンサを取りあげたが，メーリングリストにおける議論内容を調査している状況等々，別の情報システムを利用している際にも同様の状況は発生しうる．

3　システムの提案

本節ではソフトウェア理解支援のための Web システムの提案を行う．システムの提案に基づいて前節で述べた二つの問題点を解消することを目指す．本システムは以下の機能を持つ．

- 閲覧状態の記録・復元
- 閲覧情報の検索
- 他の閲覧履歴を利用したページ推薦

3.1　閲覧状態の記録・復元

本システムでは，Web ブラウザ中のウィンドウを対象として，それぞれの閲覧状態をヒストリとして記録する．そして，ユーザはそのヒストリの中から必要なヒストリを選択することで，記録された状態の閲覧状態を復元する．

3.1.1　ウィンドウ群の閲覧状態取得

Web ブラウザには複数のウィンドウが存在する．そのため，ユーザが複数のウィンドウを開いて情報を閲覧する場合，それらのウィンドウには何らかの関連があると考えられる．しかし，従来の Web システムでは，それらは個別に独立したものとして扱われ，それらを一括して記録することは出来なかった．そこで，本システムでは，各時点での複数のウィンドウの閲覧状態をひとまとまりのセットとして扱う．そして，表示されているウィンドウに対して，移動や削除などの変化が加えられると，各ウィンドウ群の閲覧状のセットをその時点でのヒストリとして蓄積する．また，必要に応じて，閲覧状態に対してマーク付けを行うこともできる．マークにはユーザが独自にユーザが独自に名前を付けられる．

そして，蓄積されたヒストリの中から，必要に応じて該当する閲覧状態を保持したウィンドウ群を復元することで，同時に見ていて関連があると思われるウィンドウ群をユーザに対して提供する．

3.1.2　ウィンドウのヒストリ取得

従来の Web ブラウザでは，ウィンドウはシーケンシャルにしか履歴情報を蓄積出来ない．そのため，前節で述べたように複雑なリンクを辿った場合，ブラウザはそのヒストリを全て保持することが出来ない (図 3 参照)．そこで本システムでは，ウィンドウに表示されているページから関連を辿った場合，辿った先のページはもとのページのノードとして扱う．その

ようにして，ウィンドウに行われた Web ページの派生などの側面をツリー状にして逐次蓄積する．これは，ウィンドウごとにツリーを構築することで別々にヒストリの取得を行う．そして，蓄積されたヒストリの中で，必要に応じて各ウィンドウの履歴状態を復元することで，以前に閲覧した部分で再び閲覧することを実現する．

3.2 閲覧状態の検索

それぞれのウィンドウには，独自の情報が記載されている．その中でそのウィンドウの特徴を示す語をキーワードとして抽出しておく．そして，それを各ウィンドウと対応付けることで，ユーザがヒストリの中から検索する際に，キーワードをもとに該当するウィンドウの検索を行う．

3.3 他の閲覧履歴を利用したページ推薦

本システムでは，他人の閲覧履歴情報を活用することで，類似した履歴を辿っているユーザに対し，関連していると思われる情報を推薦を行うことを考える．そこで，以下で説明する協調フィルタリングを用いて，類似した履歴情報から有効な情報の推薦を行う手法を提案する．以下では，協調フィルタリングについて説明する．

3.3.1 概要

協調フィルタリング (Collaborative Filtering) とは，そのシステムにおける各ユーザの嗜好をアイテムに対する評価という形で記録し，そのユーザと似たような評価をしているユーザの嗜好情報をもとに，ユーザの嗜好を推測するシステムである．この協調フィルタリングは，リコメンデーションサービスを提供する際に使用される代表的な手法で，大量のアイテムの中から，個々のユーザの好みに合うアイテムを推薦する際の要素技術として研究が進んでいる．ここでいうアイテムとは，記事，映画，音楽，商品などの，推薦の候補となるものを指す．本研究の場合は，Web ページとなる．

協調フィルタリングと相対する概念として，コンテンツベースフィルタリング (Content-based Filtering) がある．これはアイテム自体の性質，例えば含まれる文字列や，あらかじめ入力された属性などを基にユーザにアイテムを提示する手法である．コンテンツベースフィルタリングの例として，検索システムにおけるキーワード検索や，ユーザが閲覧した文書から特徴的な単語を抽出し，それをもとに提示する手法 [14] などが挙げられる．

これに対し，協調フィルタリングでは他のユーザの評価を基にユーザにアイテムを提示する．初期の協調フィルタリングを使用したシステムとしては，E-mail や Netnews 等の推薦システムである，Tapestory[13] がある．Tapestory の場合，ユーザによって明示的に選択さ

れた推薦者が評価したアイテムを掲示するという，推薦を得る相手を明示的に指定する手法をとっている．

これに対して，GroupLens[24] は推薦者や推薦する情報をシステムが自動的に決定する．GroupLens の場合，ユーザは Usenet の記事に対する評価を 5 段階で明示的に入力する．GroupLens はその情報をもとに，そのユーザと評価の傾向が似ているユーザを自動的に抽出し，ユーザに推薦を行う．現在，協調フィルタリングとして広く認識されている推薦メカニズムはこの手法を基にしており，GroupLens 手法は協調フィルタリングの基礎となっている．本研究でも，この手法を基にした手法を提案する．

さらに，アイテムへの評価をユーザに明示的に入力させず，システムが暗黙的に取得するシステムもある．例えば Phoaks[26] では，Usenet の記事中から，URL の付属した投稿を収集し，それらをその URL に対する投票と見なしている．また，大杉 [20] のシステムでは，ソフトウェア機能の使用履歴を取得して暗黙的な投票とし，ユーザに対してソフトウェア機能を自動的に推薦する．

明示的な投票を利用するシステムでは，ユーザの評価が当人の嗜好を正確に表すため，精度の高い推薦を行うことができる利点がある．しかし，ユーザに投票の操作を要求する必要があり，ユーザが非協力的な場合は，評価の絶対数を減らしてしまう可能性もある．これに対して暗黙的な投票を利用するシステムでは，ユーザに特別な操作を求めることなく評価を取得することができる．しかし，逆に正確なユーザの嗜好を把握することが難しいという欠点がある．

本研究の場合，ソフトウェア理解の支援が目的であるため，推薦機能のためにユーザに参照したページの適合性を投票させるといった余分な操作をさせることは好ましくない．このため，参照履歴をページに対する暗黙的な投票とみなし，自動的に推薦を行う手法を採用する．

3.3.2 手法

本節では，蓄積されたユーザの参照履歴に対して協調フィルタリングを適用することで，ページの推薦を行う手法を説明する．用いる推薦手法は，GroupLens で提案されたユーザ間の相関を基にした手法をもとに，Web ページの推薦に適用するための拡張を加えたものである．手法の大まかな流れは以下の通りである．

1. ユーザの評価の取得
   ユーザの部品参照履歴を取得し，ユーザの評価としてデータベースに記録する

2. 相関係数の算出
   データベースに記録された評価を基にして，ユーザ間の相関係数を求める

3. 推薦値の算出
   上記のユーザーの評価および相関係数を用いて，ページに対する推薦値を求める

4. 推薦の実行
   求めた推薦値を基にページそれぞれを推薦するかどうかを決定し，ユーザに対してページの推薦を行う

以降，これらの各段階について詳しく説明する．

1. ユーザの評価の取得
   協調フィルタリングでは，ユーザの評価を利用してアイテムの推薦を行う．そのためにユーザから各アイテムに対する評価を取得する必要があるが，ユーザに評価・投票の労力を求めることは理解効率の向上を目的とするには好ましく無いと考えられる．そのため，この手法ではユーザがページを参照した時に暗黙的に投票したとみなし，そのページに対して評価値 1 で投票したとする．

   ユーザ間の相関を基に協調フィルタリングによる推薦を行う場合，ユーザの評価傾向は変化しないことが前提となる．この前提は，GroupLens などのように評価傾向がユーザの好みを反映する場合は自然であり問題は無いと考えられるが，ソフトウェア理解の際においてはこの前提は成立しない．なぜなら，あるページについて，そのページがある目的には適合しているが，他の目的には適合していない場合が存在しうることは容易に想像でき，その時々の目的によってユーザの評価傾向は用意に変化しうるからである．

   ソフトウェア理解においては，評価傾向が変化しないのは，ユーザ単位での評価傾向ではなく，ある問題を単位とした評価傾向である．そのため，一般的な協調フィルタリングにおける「ユーザ」を「理解の際に生じる問題」とみなして，問題間の類似度から，ページの推薦に利用するのが妥当であると考えられる．これより，目的が変化した段階で別のユーザとして扱う必要があるが，システムがユーザの検索行動から暗黙的に目的の変化を取得することは一般には難しい．本研究では，ブラウザの利用終了でユーザの目的が変わったとみなし，ブラウザの使用開始から終了までを 1 セッションとし，1 セッションでの評価を 1 ユーザの評価として扱う．

2. 相関係数の算出
   協調フィルタリング手法においては，推薦を受けるユーザと似たような行動を取っているユーザを推定するために，ユーザ間の類似度を相関係数として計算する必要がある．

   GroupLens においては，2 ユーザ間の相関係数を求める際には，2 ユーザが共通して投票したアイテムに対する評価を用いている．一般にこの方法はうまく働くが，本研究

のように評価が 0，1 の 2 値である場合には意味のある値とならない．また，本研究はユーザを区切って扱うため，1 ユーザあたりの投票数が少なくなる傾向にあり，推薦の精度が落ちるという問題もある [4]．そこで，ユーザ間の相関係数を求める手法として，Breese[4] の提案する拡張を行う．これは，2 ユーザのいずれかが評価したアイテム，およびどちらも評価していないアイテムを基に相関係数を求める手法である．このとき未評価のページに対する値を設定する必要があるが，本手法では未評価のページの評価値を 0 として計算を行う．

ユーザ a とユーザ i 間の相関係数 $c(a,i)$ を求める数式は以下のようになる．式中の I_i はユーザ i が評価したページ，I はユーザ a，i のいずれかが評価したページおよび 2 ユーザのいずれも評価していない定数個のページの集合を表す．

$$v_{i,j} = \begin{cases} 1 & \text{if } j \in I_i \\ 0 & \text{if } j \notin I_i \end{cases}, \ \bar{v}_i = \frac{1}{|I|} \sum_{j \in I} v_{i,j}$$

$$c(a,i) = \frac{\sum_{j \in I} (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_{j \in I} (v_{a,j} - \bar{v}_a)^2 \sum_{j \in I} (v_{i,j} - \bar{v}_i)^2}}$$

3. 推薦値の算出

一般的な協調フィルタリングにおいては，推薦を受けるユーザと似たような行動を取っているユーザの評価をもとに，各アイテムに対して，そのアイテムの推薦値を計算する．実際には，ある未評価のページに対する推薦値を，対象ユーザがそのページに対して投票するであろう評価値の推定値として求めることができる．この推定値は，他のユーザのそのページに対する評価値を，それぞれのユーザとの相関係数を重みとして加重平均を取ることで求める．

提案手法においては，各部品の推薦値を求める際の計算量を考慮し，あらかじめ関連のあるユーザの集合を抽出する．この集合に含まれるユーザは，対象ユーザと参照したページが 1 つ以上重複し，かつ，相関係数が負でなく，かつ，投票数が 2 以上のユーザとしている．また，相関係数をそのまま重みとして利用せず，相関の高いユーザにはより高い重みを，相関の低いユーザにはより低い重みを与えるよう，累乗して重みとして利用することで，より有効な推薦が行えることが知られている [4]．本手法でもそれを採用し，相関係数の 2 乗を重みとして利用する．

$U = \{i | I_a \cap I_i \neq \phi \wedge c(a,i) > 0 \wedge |I_i| > 1\}$ とした時，ユーザ a のページ k に対する推薦値 $p_{a,k}$ は，以下の式により求められる．

$$p_{a,k} = \frac{\sum_{i \in U} c(a,i)^2 v_{i,k}}{\sum_{i \in U} |c(a,i)^2|}$$

4. 推薦の実行

   前節で説明した式により，データベース中のページに対して推薦値を求め，ユーザに対して推薦値の高い順にページを推薦する．ただし，ユーザが既に参照したページに関しては，ユーザはそのページについては既知であると見なし，推薦対象から除外する．

3.3.3 計算例

提案手法での推薦値の計算例を，表 1 で示される履歴の中のセッション 5 のユーザに対して示す．表中の “○” は，セッション中でページが表示

まず，セッション 5 とセッション 1 の相関係数 $c(5,1)$ を求める．

$$\begin{aligned} c(5,1) &= \frac{(4+10000)(0\cdot 1+1\cdot 1+1\cdot 1+0\cdot 1)-(0+1+1+0)(1+1+1+1)}{\sqrt{(4+10000)(0^2+1^2+1^2+0^2)-(0+1+1+0)^2)(4+10000)(1^2+1^2+1^2+1^2)-(1+1+1+1)^2))}} \\ &= 0.71 \end{aligned}$$

同様にしてセッション 2，3，4 に対しても求めると，表 2 の様になる．

続いて，ページ 1 に対する推薦値を求める．セッション 5 との相関係数が 0 より大きいセッションは 1，2，4 であり，推薦値は以下の様になる．

$$\begin{aligned} p_{5,1} &= \frac{0.71^2\cdot 1+0.71^2\cdot 1+0.32^2\cdot 0}{|0.71^2+0.71^2+0.32^2|} \\ &= 0.91 \end{aligned}$$

同様にして他のページに対して求めた推薦値を表 3 に示す．

表 1: 例:表示履歴

セッション＼部品	1	2	3	4	5	6	7	8	9	10
1	○	○	○	○						
2	○	○	○		○					
3						○		○		○
4			○			○	○	○	○	
5		○	○							

表 2: 例:セッション 5 との相関係数

セッション	相関係数
1	0.71
2	0.71
3	0
4	0.32

表 3: 例:セッション 5 に対する各部品の推薦値

部品	1	2	3	4	5	6	7	8	9	10
推薦値	0.91	(表示済み)	(表示済み)	0.45	0.45	0.09	0.09	0.09	0.09	0

4　システムの実現

本節では，これまで述べたシステムの実現について説明する．本システムはクライアント部とサーバ部からなる．

クライアント部は，MDI(Multiple Document Interface) とタブインターフェースを併用した Web ブラウザであり，ユーザインタフェースを提供する．

サーバ部はクライアント部で閲覧した Web ページの情報を収集する．また，収集した情報を用いて，クライアントに Web ページを推薦する．サーバ部は， CGI プログラムとして実現した．

以下，クライアント部とサーバ部について説明する．

4.1　クライアント部

クライアント部は，MDI とタブインターフェースを併用した Web ブラウザである．クライアント部の主となるウィンドウを，図 4 に示す．

ウェブページ表示部では，複数の内部ウィンドウを用いて，複数の Web ページを同時に閲覧できる．以下，内部ウィンドウのことを，単にウィンドウと表記する．ウィンドウ操作として，新しい空のウィンドウを開く，ウィンドウを閉じるといった操作ができる．ウィンドウの最大化ボタンを押すと，ウィンドウの内容がウェブページ表示部全体に表示される．この状態で他のウィンドウの内容を表示するときは，ウェブページ表示部の上にあるタブを選択することで切り替えることができる．

Web ブラウザとしての基本的な機能として， URL を入力してページを開く，表示されている Web ページのリンク先のページを開くという操作ができる．ここでリンク先のページを開くときには，リンクの表示されているウィンドウで開くか，新しいウィンドウで開くかを選択することができる．

以上が，ブラウザとしての基本的な機能である．これに加えて，試作したシステムは，ページの親子関係を表示する機能と，ユーザの閲覧状態の履歴を記録して利用する機能がある．

まずページの親子関係を表示する機能について説明する．ページのリンクを開くときに，リンク元のページを親，リンク先のページを子とする関係を作る．この親子関係をツリーにして表示するのが，図 4 のページ親子関係表示部である．頂点がページで，ページの親子関係が頂点の親子関係に対応する． URL を入力して開いたページには親は存在しないので，木の根になる．リンク先のページが，ツリー上に既に存在する場合には，新しいツリーの頂点を作ることはしない．

閲覧状態とは，ある時点で表示されている全てのウィンドウの位置・サイズなどの情報と，全てのウィンドウで表示されているページやそのキーワードをまとめた情報である．そして，

図 4: クライアント部のインタフェース概要

閲覧状態の履歴とは，閲覧状態を時系列に並べて作った列である．閲覧状態の履歴についての詳細は後で述べる．

この閲覧状態の履歴を用いた操作について説明する．

まず，履歴の中で，一つ前の閲覧状態を復元する操作がある．この操作により，各ウィンドウで表示していたページや，ウィンドウのサイズ・位置などを，まとめて復元することができる．また，この操作は繰り返すことができ，いくつでも前の状態に戻ることができる．閲覧履歴の末尾にいないときには，一つ後の状態を復元することもできる．

閲覧履歴が短いときは，一つ前や後に戻る操作を繰り返すことで，容易に意図した閲覧状態を復元することができる．しかし閲覧履歴が長くなってくると，一つずつの操作では，なかなか意図した状態に辿りつけない．そこで，閲覧状態の一覧 (図 5) や，閲覧状態を時系列に並べたバー (図 6) を用いて，過去の閲覧状態に一度に戻ることができる．

また，単語を入力して閲覧履歴を検索することもできる (図 7)．検索語を入力して Search ボタンを押すと，下部に検索語に部分一致するキーワードを持つ閲覧状態が，一覧として表示される．この一覧から一つを選択することで，その閲覧状態を復元することができる．

さらにクライアント部は，推薦するページを提案する機能を持つ．推薦ページの計算はサーバ部で実現される．クライアント部は，ユーザがリンクをクリックしたり URL を入力したりすることによって，新しいページを表示したときに，そのページの情報をサーバ部に送る．この情報は，サーバ部が推薦ページを計算するのに必要なものである．

サーバに情報を送ると，サーバからは 0 個以上の推薦ページの情報が送られてくる．この推薦ページの情報を，図 4 の推薦ページ情報表示部に，一覧として提示する．この一覧から一つを選択すると，推薦されたページを表示することができる．

推薦ページは先人が解決した同様の問題の情報を用いているので，問題を短時間で解決するのに役立つ．また，ユーザの操作環境とは別の部分に，非同期に表示されるので，ユーザの操作を妨害しないという利点がある．

4.1.1 閲覧状態の履歴

閲覧状態とは，ある時点での各ウィンドウがどこにあり，何の情報を表示しているかを記録したものである．

具体的には，閲覧状態は以下のような情報から成る．

- 時刻
- 直前の閲覧状態から変化した原因
  - ウィンドウの開閉

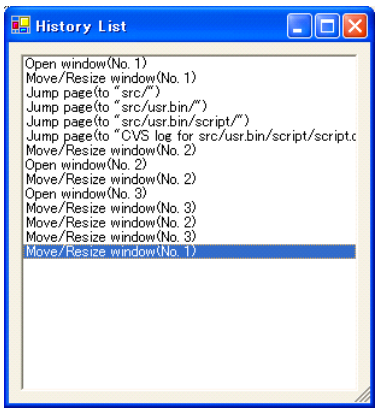

図 5: 閲覧状態の一覧ダイアログ

図 6: 閲覧状態の時系列バーダイアログ

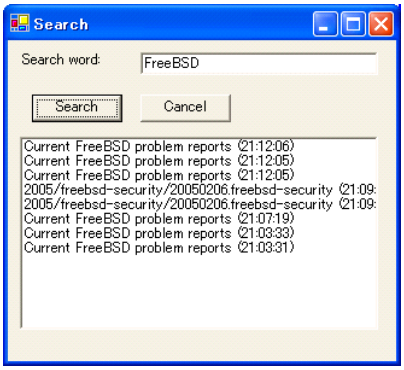

図 7: 閲覧状態の検索ダイアログ

  - ウィンドウのサイズ変更・移動
  - ページ移動
- 変化が起こったウィンドウ
- 全てのウィンドウに関する以下の情報
  - 位置・サイズ・スクロール位置
  - 表示しているページの URL とタイトル
  - キーワード

キーワードとは，閲覧状態を特徴付ける単語であり，ウィンドウが表示しているページや，直前に表示していたページから抽出する．具体的には，ウィンドウが現在あるページを表示しているとき，そのページを記述している HTML 文書から以下の単語を抽出してキーワードとする．

- タイトル
- name 属性が keywords か description いずれかである meta 要素の content 属性の値
- a 要素の name 属性の値

また，ウィンドウが現在表示しているページが，リンクを辿って表示したページである場合には，リンク元のページのアンカーのテキストをキーワードとする．

そして，閲覧状態の履歴は，時系列に並べた閲覧状態の列である．

4.1.2 クライアント部の実装

クライアント部は， C# on .NET Framework v1.1 を用いて記述した．規模は約 2000 行である．Web ページの表示および解析には， Microsoft 社の WebBrowser コントロールを使用している．

Microsoft Windows XP Home Edition 上で動作を確認した．

4.2 サーバ部

サーバ部はクライアントに Web ページを推薦する役割を持つ．

そのために，クライアント部で閲覧した一連の Web ページの情報を収集する．この，クライアントでの一連の閲覧動作をセッションと呼ぶことにする．セッションには一意な ID を付けて区別する．

サーバは過去に閲覧されたページの情報を，セッション毎に記録したデータベースを持っている．

クライアントから，新しいページを表示する度に，そのページの URL とタイトルの情報が送られてくる．送られてきた情報は，セッション ID と閲覧ページの情報を組にして，データベースに記録する．

データベースを用いて，推薦ページを計算する．推薦ページの計算には，前節で説明した協調フィルタリングを利用する．そして，計算結果の推薦ページをクライアントに送信する．

4.2.1 サーバ部の実装

サーバ部は Ruby 1.8.2 を用いて，Web サーバ Apache 2.0.53 上で動作する CGI プログラムとして記述した．データベースライブラリには GNU GDBM 1.8.3 を使用した．動作の確認は， FreeBSD 4.11-STABLE 上で行った．

5　利用例

本節では実際の使用例として，開発経験の浅い人がソフトウェア理解を試みた事例をとりあげる．ここでは私の研究グループで開発している SPARS のソースコード解析部を題材とする．SPARS について予備知識を持たない開発者が，処理の内容を理解するために， main 関数を起点として呼びだされた関数を逐次クロスリファレンスツールの Web インタフェースを用いて閲覧する．このとき，Web ブラウザとして提案ツールを使うことでどのような利点が存在するかを示す.

5.1　利用例 1: クロスリファレンサを用いた場合

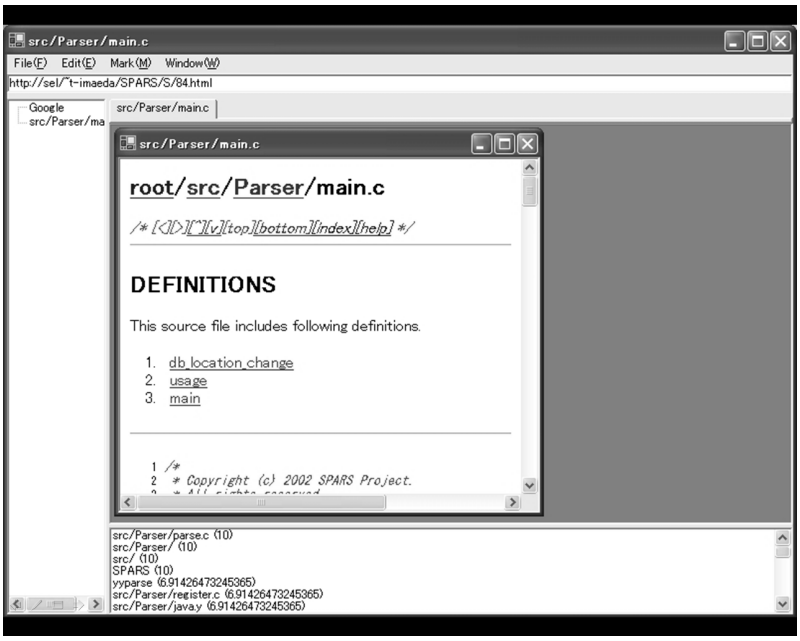

図 8: 初期状態: main 関数の内容が表示されている

まずユーザは本ツールを通じてクロスリファレンサにアクセス，main 関数のページを表示する (図 8)．そして main 関数を起点に呼びだされている関数を順次読みすすめていったところ，閲覧者は構文解析ライブラリ yacc を利用していることがわかった (図 9)．開発者はひととおり yacc についての解析を行いひとまず構文解析部でどのような処理が行われているか，その概要を理解したものとする．

そして一度 main 関数にもどり，さらに読みすすめていくと解析結果をデータベースに入力している部分を発見した (図 10)．閲覧者は利用しているデータベースのマニュアルも同時に開きながら解析を進めていく．そして，閲覧者はパース部分からのデータを加工してい

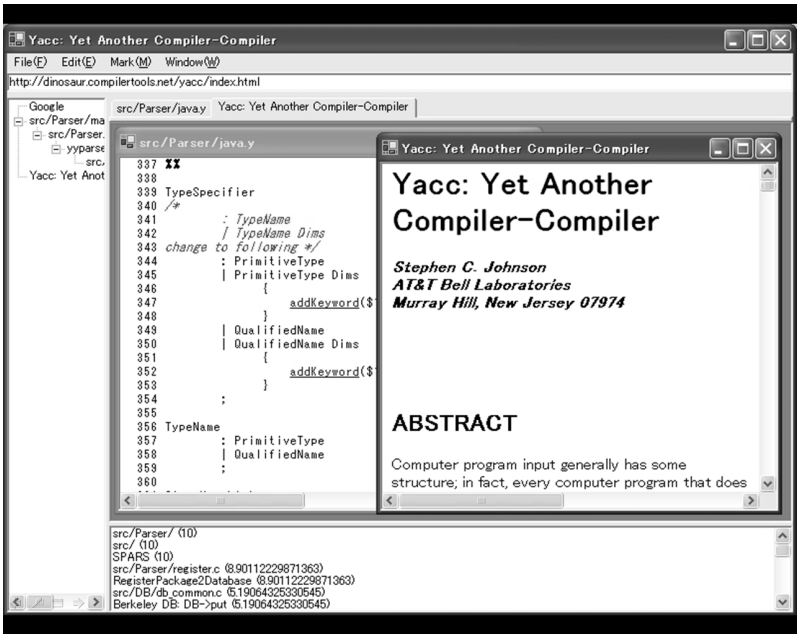

図 9: yacc ライブラリについて調査している状態

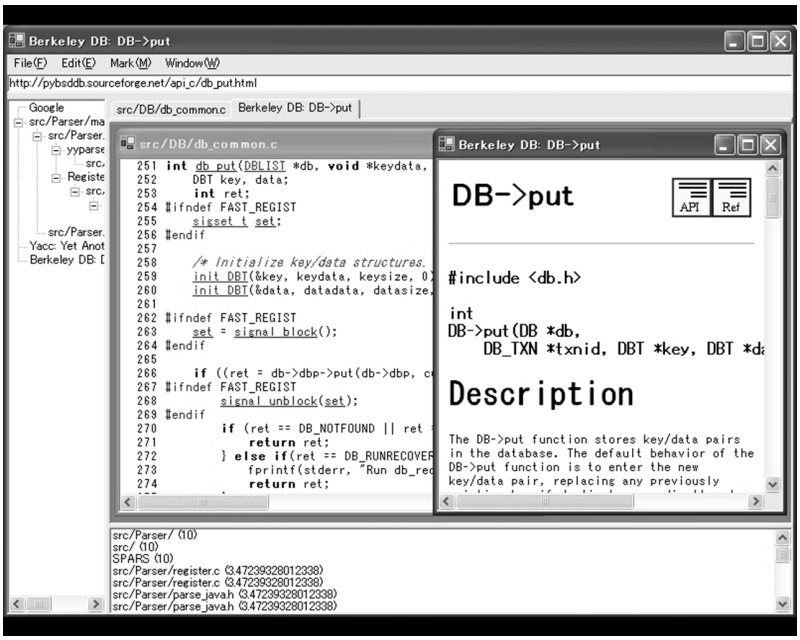

図 10: db について調査している状態

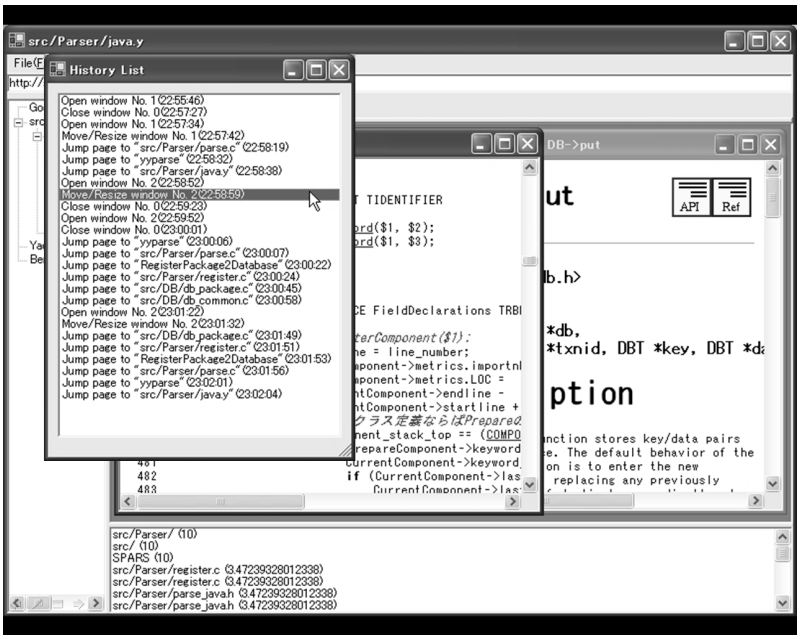

図 11: 閲覧履歴一覧リスト

るソースを発見したため，さきほど閲覧した yacc のソースにもどって，どのようなデータがはいるのかを確認をする必要に迫られた．

このとき，通常の閲覧ツールであれば各ウィンドウごとに現状をブックマークで保存して，ウィンドウの戻るボタンで main 関数まで戻り，そこから再び yacc 解析部までのリンクをたどらなくてはならない．しかし，このとき正確に yacc 解析部へのリンクをたどらなければ yacc 解析部を閲覧することができない．yacc 解析部は main 関数からは何段もの関数呼び出しが間に存在するため，最初に yacc 解析部にたどりついたときと同じ行程をたどるのは閲覧者にとって負担となる．

本ツールを用いている場合，閲覧者は閲覧履歴リストを呼びだして，得られた履歴一覧から yacc を見ていたときのものを選択することで yacc を調べていたときの状態に復元できる (図 11)．またそのときに参照していたマニュアルも同時に復元される．このように本ツールを用いることで容易に任意の時点での閲覧履歴を参照し，その状態を復元することができる．

また，yacc の内容が確認できたので再び元のデータベース登録部に戻りたいと思ったときには，再び閲覧履歴リストを呼びだして，適切な履歴を選択すればよい．このときも，データベース登録部を閲覧していたときに参照していたデータベースのマニュアルも同時に復元される．

5.2 利用例 2: 同様に SPARS を調査する開発者がいた場合

またページ推薦機能を利用することで，後に同様の調査をする人にとって有用なページを自動的に推薦することができる．上の行動が行われた後に，同様の調査が行われた場合，図 9 の状態にたどりついた時点で，本システムは yacc のマニュアルページを自動的に推薦する．このように先人の履歴から有用な情報を得ることができる．

6 まとめ

本研究では，ソフトウェア開発を研究対象として，オープンソースソフトウェア開発における開発環境の問題点を二つ指摘した．一つ目は，各システムが連携を持たず，開発情報が分散して存在する「システム間の連係不足」の問題である．そして二つ目は，各開発環境において大量のリンクが存在するために，ページ遷移を繰り返すうちに，以前見たページに辿り着けなくなる「複雑なリンク関係」の問題である．これらの問題のため，各システムが共通のユーザインターフェイスとして Web ブラウザを用いているにも関わらず，開発者は複数のシステムにまたがる情報閲覧を行う場合に，情報閲覧全体のつながりを自ら管理するという手間があった．

そこで，これらの問題点を解決するために，複数のソフトウェア開発支援システムを同時に利用する際に必要となるインターフェイスを持った Web ブラウザを試作した．具体的には，まず， 閲覧状態の履歴をツリー上のグラフとして提示するインターフェイスを Web ブラウザに付加した．そして，閲覧状態の履歴を用いて，任意の過去の時点における閲覧状態を復元できる機能を実装した．さらに，閲覧状態の履歴を複数の利用者間で共有することにより，他の利用者が閲覧した内容に基づいた推薦を行う機能を実装した．

また，CVS を起点にした場合とメーリングリストを起点にした場合の二つの利用事例を挙げ，作成したブラウザの有用性を確認した．

今後の課題としてまず，定量的な評価を行うことが挙げられる．試作したシステムを用いるユーザ群と，既存の Web ブラウザを用いるユーザ郡に対して，共通の課題を与え，その課題に対する解答時間を比較することで，システムの評価をすることが出来ると考えている．次に，表示していたページの内容の類似度や，ページを同時に閲覧していた時間を考慮して，ページ間の隠れた関係を抽出できるのではないかと考えている．また，現在のシステムでは，閲覧状態のリスト表示をするときに，閲覧状態が変化した理由と，変化が起こった時刻を組み合わせて，閲覧状態の名前としている．この表示は，詳細な情報が記述されているものの，視認性は高くない．そこで，閲覧状態のリスト表示を行うときに，それぞれの閲覧状態の画像を縮小して表示しておくことで，詳細を見なくても状態の概要が分かるようにすることを考えている．これらの改善点を踏まえて，ソフトウェア開発時の情報閲覧の手間を軽減し，ソフトウェア開発環境の改善に貢献することを目指す．

謝辞

本論文を作成するにあたり，常に適切な御指導および御助言を賜わ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致します．

本論文の作成にあたり，逐次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致します．

参考文献

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

[2] B. Berliner, “CVS II:Parallelizing Software Development”, In USENIX Association, editor, Proceedings of the Winter 1990 USENIX Conference, pp.341-352, Berkeley, CA, USA, 1990.

[3] Bonsai,
http://www.mozilla.org/bonsai.html.

[4] J. S. Breese, D. Heckerman and C. Kadie, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering”, Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp.43-52, Madison, Wisconsin, USA, 1998.

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

[6] CVSWeb,
http://www.freebsd.org/projects/cvsweb.html/.

[7] J. Estublier, “Software Configuration Management: A Roadmap”. The Future of Software Engineering in 22nd ICSE, pp.281-289, Limerick, Ireland, 2000.

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

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

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

[11] P. Fröhlich and W. Nejdl, “WebRC Configuration Management for a Cooperation Tool”, SCM-7, LNCS 1235, pp.175-185, 1997.

[12] GNU GLOBAL,
http://www.gnu.org/software/global/.

[13] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using Collaborative Filtering to Weave an Information tapestry.”, Communications of the ACM, Vol.35, No.12, pp.61-70, 1992.

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

[15] 鯉江, 西本, 馬場, “バージョン管理システム (CVS) の導入と活用”, SOFT BANK, 2000.

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

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

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

[19] 落水, “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.

[20] 大杉, 門田, 松本, 森崎, “ソフトウェア機能の推薦の為の協調フィルタリング”, ソフトウェアシンポジウム 2002 論文集, pp.83-89, 2002.

[21] Open Source Development Labs, Inc.,
http://www.osdlab.org/.

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

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

[24] 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, pp.175-186, Chapel Hill, North Carolina, USA, 1994.

[25] SourceForge,
http://sourceforge.net/.

[26] L. Terveen, W. Hill, D. McDonald, and J. Creter, “PHOAKS: A System for Sharing Recommendations”, Communications of the ACM, Vol.40, No.3, pp.59-62, 1997.

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