電子情報工学専攻

Advanced Course of Electronics and Information Engineering

平成 22 年度 専攻科特別研究論文

題　目

開発コンテキストの推定による支援システム間の情報統合

英文題目

Information Integration between Support Systems by Estimate of Development Context

指導教員名　上野　秀剛

論文提出者名　谷　宗一郎

独立行政法人　国立高等専門学校機構

奈良工業高等専門学校 専攻科

Institute of National Colleges of Technology, Japan

Faculty of Advanced Engineering at Nara National College of Technology

開発コンテキストの推定による 支援システム間の情報統合

Information Integration between Support Systems by Estimate of Development Context

谷　宗一郎

Souichirou Tani

独立行政法人 国立高等専門学校機構
奈良工業高等専門学校 専攻科 電子情報工学専攻
大和郡山市矢田町２２番地（〒639-1080）
Institute of National Colleges of Technology, Japan
Faculty of Advanced Engineering at Nara National College of Technology,
Yata-cho 22, Yamatokoriyama-shi, Nara 639-1080, Japan

ABSTRACT:　Many software projects use development support systems such as bug tracking system or version control system to manage development information . Such support systems collect the information based on type -not based on topics of the development- hence, a series of information which belongs to same development context spread through different systems. For example, when a developer tries to fix a bug, the developer want to check a working history on the expand feature that cause the bug. In such case, developer must search more than one system, so it makes information sharing difficult. This paper proposes a system to integrate the information which belong to same development context, but located different support systems for efficient collection. The system runs as a proxy server; developers can utilize existing systems and stored information without any conversion. Also this paper evaluates the system output by applying the system to an existing open source project. This paper evaluates the system with two ways: one is comparing hyper links made by the proposal system and links appended to items of the bug tracking system or the version control system by developers. The other one is questionnaires from developers use the proposal system.

目次

１．はじめに

ソフトウェア開発の外部委託やオープンソースソフトウェア(Open Source Software: OSS)プロジェクトの増加に伴い，地理的に離れた場所にいる開発者間で情報を共有する需要が増加している．特に，外部委託においては，海外の開発組織に委託するオフショア開発が増加していることもあり[1]，タイムゾーンの異なる遠隔地との情報交換の重要性も高まっている．

このような環境では，移動時間や移動費用のかかる開発者が直接顔を合わせて行う会議や，スケジュール調整が必要な電話会議と併用して，オンラインによる非同期な情報交換が良く用いられている[2-3]．ソフトウェア開発における，代表的なオンライン，非同期コミュニケーションのツールとしてメーリングリスト(Mailing List: ML)やバグ管理システム(Bug Tracking System: BTS)，版管理システム(Version Control System: VCS)がある．本稿では以降，これらのシステムを開発支援システムと呼ぶ．開発支援システムを用いることで，情報共有をスムーズに行うことが出来る．

これらのシステムは，ソフトウェア開発を行っている企業においては独自に開発・運営されていることが多く，複数のシステムを組み合わせた統合システムも存在する[4-5]．1つのシステムに統合することで開発中に行われた一連の作業の流れをひとまとめに管理し，効率的に検索することができる．開発における一連の作業の流れ（開発コンテキスト）を各開発システム間で共有することで，開発状況の理解が容易になり，効率的に情報を収集することができる．

一方で，小規模なOSSプロジェクトではシステムの導入と維持に必要なコストの支払いが難しいことから，無料で開発支援システムを利用可能なレンタルサービス[6-8]が多く用いられている．レンタルサービスを用いる場合，統合システムを用いる際に必要な個々のシステムの変更・改造が難しく，開発コンテキストに沿った情報を得ようとすると，それに必要な情報を各システムから手動で調査しなければならない．

複数のシステムから必要な情報を取得するためには，各開発コンテキストにおいてそれぞれの開発支援システムがどれだけ利用されたかを理解している必要があるが，そのような情報は開発支援システムには記録されていない．従って，プロジェクトに途中から参加した開発者が情報を収集しようとした場合や，古いコンテキストに関する情報を収集しようとした場合，漏れが発生する可能性がある．統合システムを無料で提供しているサービスも存在するが，既に個別の開発支援システムを利用している場合，各システムに格納されたデータの移行が難しい，プロジェクトで必要な機能が統合システムで提供されていない可能性がある，といった問題がある．

本稿では，既に利用している開発支援システムを変更することなく，複数のシステムに記録された一連のコンテキストに属する情報を合わせて表示するシステムについて提案を行う．提案システムは，開発支援システムを閲覧する際のプロクシとして存在する．開発者が閲覧しようとした開発支援システム内の情報からコンテキストを推定し，他の開発支援システムに格納された，同一のコンテキストに属すると思われる情報へのリンクを追加する．

以下，2章ではソフトウェア開発に用いられる開発支援システムの概要と問題点について説明する．3章では提案システムについて説明し，現状の実装方法とプロジェクトに適用した際の動作例を示す．4章では提案システムの動作実験を行い，その結果について考察する．5章では，提案システムを実際のプロジェクトに対して適用し，そのプロジェクト参加者にアンケートを実施し，その結果について考察する．6章では従来研究との違いを述べ，本研究の位置づけを明らかにする．最後に7章では，まとめと今後の課題を述べる．

２．開発支援システム

２．１．概要

開発支援システムとは，ソースコードの変更履歴やバグ（不具合）の検出から除去までの履歴，メールやBBSでの議論など，開発に関する様々な情報を共有するためのシステムである．開発支援システムの多くはサーバ上でCGIやサーバソフトウェアとして動作し，Webブラウザなどを経由して利用される．図1に開発支援システムの例としてバグ管理システム(Bugzilla: http://bugzilla.mozilla.gr.jp/)のスクリーンショットを示す．バグ管理システムでは，発生したバグを登録し，それに関わる担当者，作業情報などの情報を追加することで，バグの修正までの流れを管理する．

開発支援システムは共有したい情報の種類ごとに異なるシステムが開発されている．BTSは，バグを検出してからバグ位置の特定，修正方針の決定，除去までの履歴を管理する．また，これらの作業に伴って行われる議論の履歴も含まれる．VCSは，ソースコードや仕様書・マニュアルといったドキュメントの変更履歴を管理している．MLでは，調査内容の報告やスケジュール調整といったプロジェクト内の情報交換のログが蓄積される．ソフトウェアテストの結果報告なども，MLで行われる場合がある．

開発支援システムの利用者は，これらのシステムをそれぞれ参照し，開発やサポートに必要な情報を収集することができる．利便性向上のために，複数の開発支援システムを統合したシステムも存在する[4-5,9].

ソフトウェア開発を行っている企業では，独自に開発支援システムを開発し，社内ネットワークで運用・管理が行われている．一方で，小規模なOSSプロジェクトでは開発のための予算を持たず，サーバの用意・運用にかかる費用を負担できないことも多い．また，OSSプロジェクトに参加する開発者はボランティアであることが一般的であり，参加者は流動的に変化するため，サーバを適切に管理できる人員を安定的に確保できないことがある．そのため，Google Code[6]やSource Forge[7], Launchpad[8]のような無償のレンタルサービスが利用されている．

図 1 開発支援システム（バグ管理システム）の例

２．２．問題点

個別の開発支援システムや統合されたシステムを用いることで，開発者間の情報共有が容易になるが，以下の問題がある．

問題1 複数の開発支援システムにまたがって検索をしなければいけない

問題2 開発コンテキストが開発支援システムに記録されない

問題3 個別のシステムから統合システムへのデータ移行が困難

問題4 外部システムと他のシステムとの結合が困難

問題5 使用できる開発支援システムに制限がある

これらの問題点について，1)複数の開発支援システムを併用した際の問題，2)統合システムを用いた際の問題の2つの観点から説明する．

２．２．１．複数の開発支援システムの併用による問題

複数の開発支援システムを用いて，1つのプロジェクトに関する情報を管理している場合，開発者はある1つの事柄を調査するために，手動で複数のシステムから情報を収集する必要がある．このとき，調査している事柄（例えば，ある不具合がどのように議論・修正されたか）を示す開発コンテキストを理解していなければならない．

図2に複数の開発支援システムを用いた開発におけるコンテキストの様子を示す．3つの長方形はVCS・BTS・MLの各開発支援システム，楕円はそれぞれのシステムに保存された情報，楕円を繋ぐ点線は同一コンテキストに属する情報のつながりを表している．それぞれのコンテキストに含まれる情報は，その種類ごとに異なるシステムに分散して保存されている．ある情

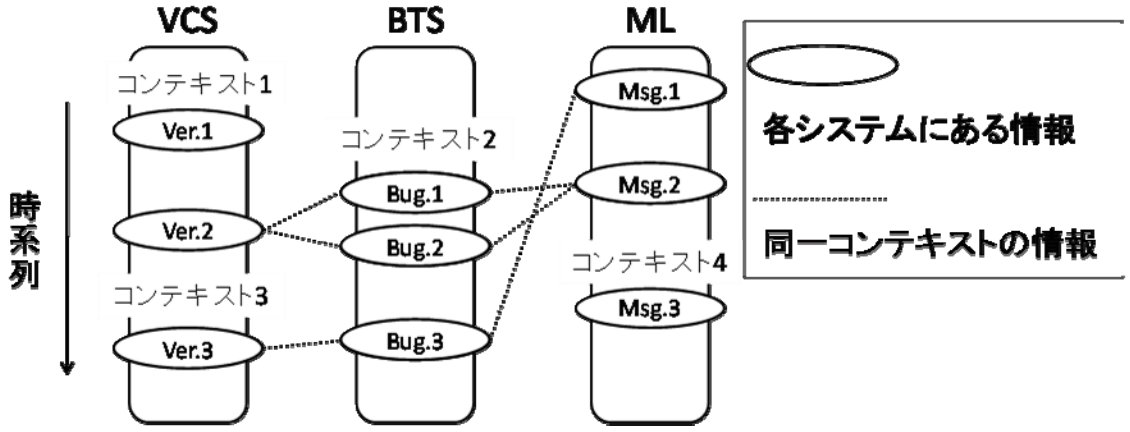

図 2　各システムが持つ情報のつながり

報に関係する情報を探したいとき，点線で示されるコンテキストを元に情報をたどることになるが，コンテキストは開発支援システムに存在せず，開発者の知識に頼っているのが現状である．

このような状況において，開発者が調査を行う際には以下のような問題点がある．

問題1　複数のシステムにまたがった検索が必要

複数の開発支援システムを併用している場合，1つの開発コンテキストに関する情報を収集するために，開発者が手動で複数のシステムを調査する必要がある．このとき，各システムが持つ，同じ開発コンテキストに関する情報は互いにリンクされておらず，必要な情報を見落としてしまう可能性がある．BTSやVCSに記録される不具合情報・履歴情報には，内容の理解や検索に用いるためのコメントや不具合IDなどが入力されるが，システム間が統合されていない場合，それらを基に手動で検索を行う必要がある．

問題2　開発コンテキストの不足

複数の開発支援システムから，必要な一連の情報を探すためには，開発時のコンテキストを理解していなければならない．例えばBTSに記録されたある不具合の情報を収集する時，不具合についてどのような議論や修正が行われてきたのかを知らなければ，その開発プロジェクトでの開発支援システムの使われ方（BTSに登録する情報は要望やソフトウェアテストの結果報告なども含めるのか，MLで取り扱う話題はどうするのか，個別のバグに関する話題も含めるのかなど）を理解できていないために，VCSやMLから効率よく情報収集することが難しい．このような開発コンテキストは開発支援システムには保存されておらず，新規にプロジェクトに参加した開発者はコンテキストの把握ができない．また，以前から参加している開発者であっても，過去のコンテキストを完全に覚えているわけではなく，記憶に頼っているのが現状である．

２．２．２．統合システムを用いる際の問題

前節で述べた問題を解消するために，複数のシステムを統合したシステムが提案・実装されている[4-5,9]．これらの統合システムではBTSやVCS，MLの他にWikiやテスト管理システムなどを統合することで，プロジェクトの管理に必要な情報を1つのシステム内で管理できる．従来の統合システムには，既存の開発支援システムを拡張したものと，統合システムとしてあ

らかじめ用意されたシステムを利用するものがある．しかし，これらの統合システムをプロジェクトに導入するためには以下の問題点がある．

問題3　既存開発支援システムのデータ利用が困難

プロジェクトが既に開発支援システムを導入している場合，これまでに開発支援システムに蓄積された情報の新規に導入する統合システムでの使用が難しい．一部の統合システムでは他の開発支援システムからのデータインポートをサポートしているが，全ての情報が適切に変換されるわけではない．

問題4　外部システムを統合できない

既に運用している開発支援システムが存在する場合，複数のシステムを接続することで蓄積された情報をそのまま使用しながら統合システムとして動作させることができる．しかし，レンタルサービスのような外部で提供された開発支援システムを用いている場合，システムの変更ができないため，統合することができない．

問題5　使用できる開発支援システムに制限がある

個別の開発支援システムを用いる場合，それぞれの情報の種類に応じて，プロジェクトで必要な機能を持ったシステムを選択する事ができる．しかし，統合システムを用いる場合，各統合システムが提供する機能しか利用できず，個別の開発支援システムを組み合わせた場合に比べて制限がある．

３．提案システム

３．１．概要

本稿では，２．２節で述べた問題点に対応するために，既に利用している外部の開発支援システムを変更することなく，開発情報を統合して表示するシステムを提案する．提案システムの構成及び動作例を図3に示す．なお，図中では説明を容易にするために，開発者がアクセスする開発支援システムとしてBTSを想定している．本システムはプロクシサーバとして動作し，開発者の開発支援システムに対するアクセスを検出すると自動的に動作する．なお，開発支援システム以外へのアクセスでは，本システムは基本的なプロクシの動作以外何も行わない．この処理はアクセス判定部で行われる（図3 (1)）．

開発支援システムへのアクセスを検知すると，本システムはその参照結果を処理にかける(2)．まず，参照結果から推定に利用する情報を抽出収集する．これはテキスト解析によって行われる．収集する情報は，BTSでは投稿ごとに振られる固有番号，投稿者名，投稿時刻，投稿されたバグに関する文章など，VCSではリビジョン番号，変更者名，コミットメッセージ，コミットファイル名などである．収集した情報は，システム内部のキャッシュデータベースに保存される．

情報収集の後，収集した情報に対して同一コンテキスト情報の推定が行われる(3)．収集した情報は，キャッシュデータベースに保存された他システムの情報一つ一つと関連性が推定される．この際の推定方法は特に提案しない．現在の実装では，単語の出現頻度で推定を行ってい

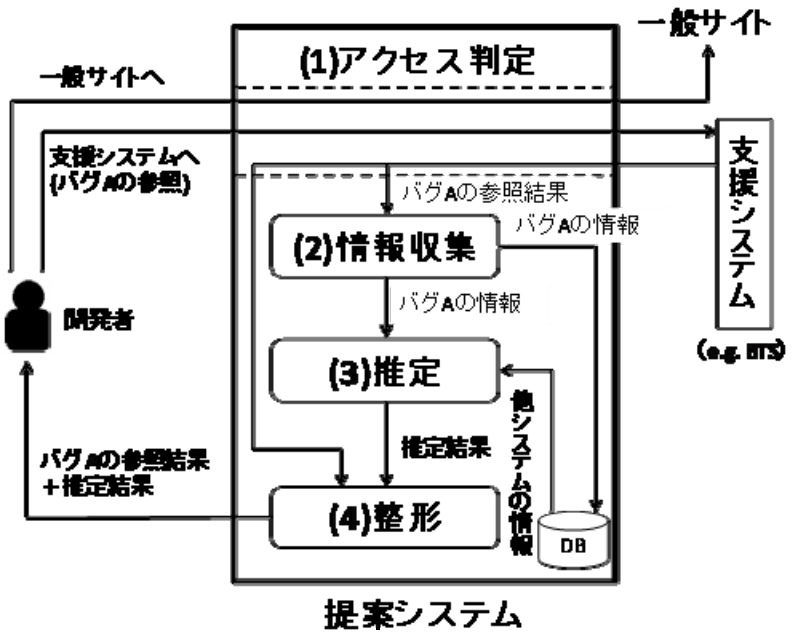

図 3　提案システムの構成と動作例

る（３．３節で詳述）．推定精度を高める方法については今後の検討課題である．

推定処理の結果はそれぞれの開発支援システムに合わせて整形され，情報収集の対象となった開発支援システムの参照結果の中に組み入れ，開発支援システムを閲覧しようとした開発者に提示される(4).

このように，本システムの一連の動作は開発支援システムへのアクセスをトリガとして開始され，開発支援システムの出力を改変し，開発者に渡して終わる．このときの開発者（開発支援システムの閲覧に利用したブラウザ）と提案システム，各開発支援システムの関係を図4に示す．本システムは，それぞれの開発支援システムのホストとは独立し，開発者の開発支援システムへのアクセスに従って情報を自動的に取得するため，開発者は本システムの存在を意識することなく，他のシステムに保存された関連情報を見ることができる．

３．２．問題点との対応

提案システムを用いて，単体で動作している開発支援システムにアクセスすると，自動的に他の開発支援システムに保存されている関連の情報を抽出し，表示する．これによって，問題1（複数のシステムにまたがった検索）が解決される．このとき，複数のシステムに保存された情報間の関連性を推定し，開発コンテキストを知らない開発者に一連の情報を示す事で，問題2（開発コンテキストの不足）に対応している．

問題3（既存開発支援システムのデータ利用が困難）及び問題4（外部システムを統合できない）については，既存のシステムがあらかじめ持っているインタフェース（例えばHTTP）を介して通信を行い，システムを変更することなく，蓄積された情報をそのまま利用することで

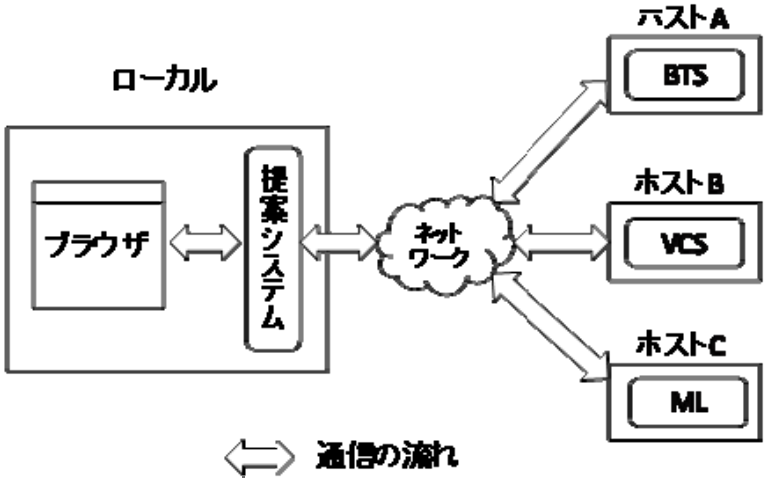

図 4　システムの実装のハードウェア構成

対応している．また，上記の点に加え，開発支援システムとの通信モジュールの新規実装を容易にするために，インタフェースクラスを用意している．プロジェクトで利用したい開発支援システムと提案システムの結合を容易にすることで，問題5（使用できる開発支援システムに制限がある）に対応している．

３．３．実装

提案システムが適切に動作するか確認するために，実装を行った．現在，manatubbs(BTS)[10]とGoogle Code[6](VCS)に対応したシステムの試作を行っている．本実装は，C#で記述されており，.NET Framework上で動作する．ソースコードは約2000行，約8個のクラスで構成される．本実装では各開発支援システムへの対応（情報の抽出，推定を行うための共通フォーマットへの整形，提示情報の整形など）を，共通処理を除き外部ライブラリとして実装しているため，他の開発支援システムへの対応も容易に行うことができる．

現在の実装では，推定は以下の手順で行っている．

1. 本システムを利用するプロジェクトで話題のキーとなる単語や特徴的な単語を設定する．３．４節に示す動作例，4章に示す動作実験，および5章に示す適用実験では，特徴的な単語としてBTSのサマリにおける頻出単語61個を利用した．
2. 現在閲覧中の情報と，本システム内にキャッシュされたすべての情報について，設定した単語の出現回数をカウントする．
3. それぞれの情報について，各単語の出現回数を，その情報での単語の総出現回数で割り，単語の出現割合とする．
4. 現在閲覧中の情報と，本システム内にキャッシュされたすべての情報それぞれの間で，単語ごとに出現割合の差の二乗を取り，各情報の組で総和を求める．この値を現在閲覧中の情報と，本システム内にキャッシュされた情報のひとつとの推定スコアとする．
5. 推定スコアが小さい組のシステム内の情報ほど，現在閲覧中の情報と同一コンテキストに属している可能性が高いと見なす．

以上の手順を数式で表すと次のようになる．

$$\boldsymbol{t} = \left((\forall j)\mathrm{count}(text, word_j) \right) \div \sum_j \mathrm{count}(text, word_j) \qquad (1)$$

$$\boldsymbol{i}_k = \left((\forall j)\mathrm{count}(cache\_text_k, word_j) \right) \div \sum_j \mathrm{count}(cache\_text_k, word_j) \quad (2)$$

$$ans = \min((\forall k)\sqrt{(\boldsymbol{t}-\boldsymbol{i}_k)(\boldsymbol{t}-\boldsymbol{i}_k)^T}) \quad (3)$$

なお，数式中，$word_j$はj番目の特徴的な単語，$text$は閲覧中の情報中のテキスト情報，$cache_text_k$はk番目のキャッシュされた情報中のテキスト情報，countはテキスト情報の指定単語の出現回数を数える関数，$\boldsymbol{t}$は閲覧中の情報の出現回数ベクトル，$\boldsymbol{i}_k$はk番目のキャッシュされた情報の出現回数ベクトル，minは系列中の最小の値を返す関数であるである．

現在の実装では，同一コンテキストに即している可能性が高い情報のうち上位5件を提示している．

図5と図6に，本システムを介してmanatubbsおよびGoogle Codeにアクセスした時の出力例を示す．図の中央付近，「システムの出力」の下（点線内）に，本システムが出力した結果が表示されている．次の節で提案システムの動作例について説明する．

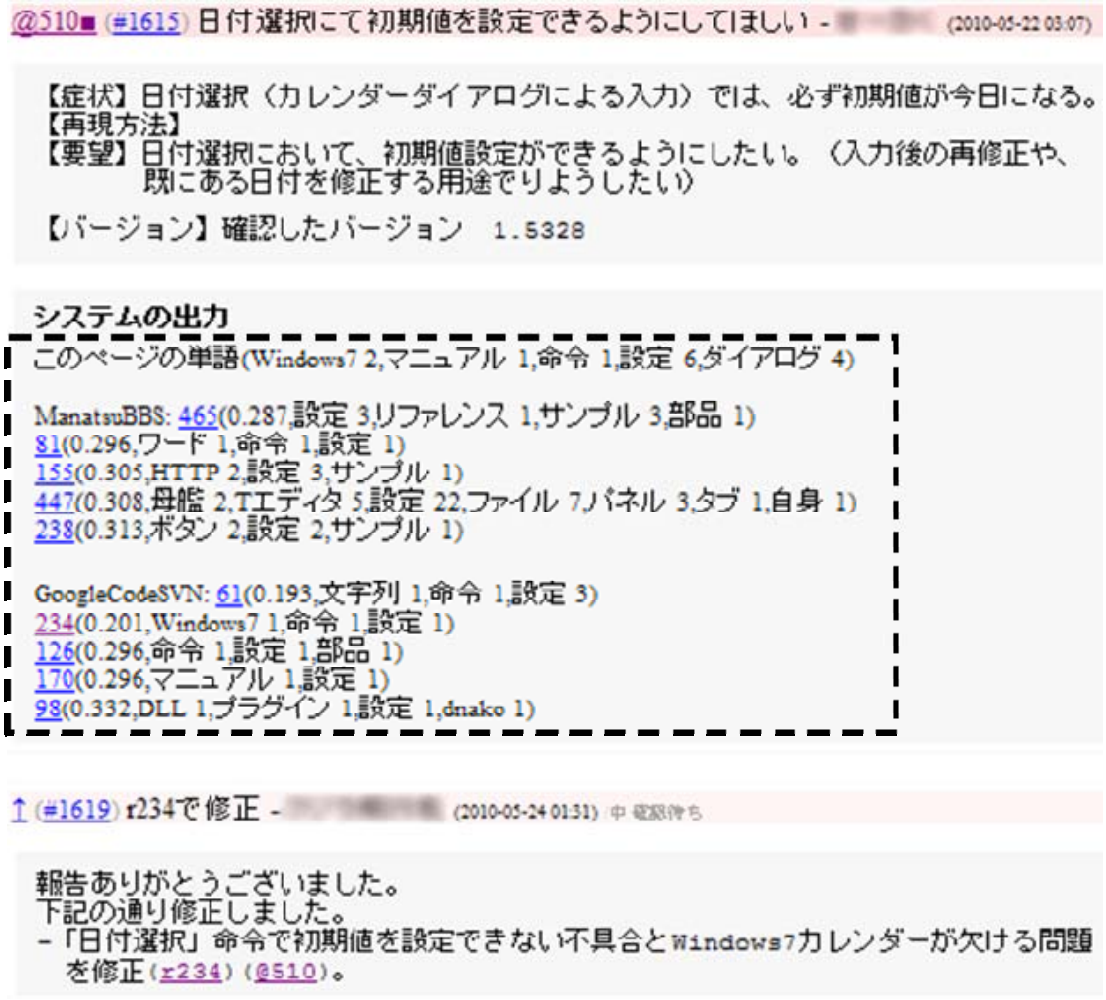

図 5　提案システムを介したバグ管理システムの出力

３．４．動作例

図5と図6は日本語プログラミング言語"なでしこ"の開発プロジェクト(http://nadesi.com/)で実際に使われているmanatubbs(BTS)とGoogle Code(VCS)に，提案システムを介してアクセスした様子を示している．出力の一行目は現在表示しているページに含まれる特徴的な単語（Windows7，マニュアル，命令，設定，ダイアログ）と，その出現回数を示している．2行目以降には，それぞれの開発支援システムについて，このページで得られた単語から推定されたページへのリンクが5つずつ提示されている．ここでの表記のフォーマットは「ページ番号(スコア,各単語の数...)」となっている．スコアは推定によって求めた推定スコアであり，この値が0に近いほど，現在のページと同一のコンテキストに含まれる可能性が高い．各単語の数の表記は，一行目と同一である．なお，この例では，話題の特徴を表す単語はなでしこプロジェクトに参加している著者の経験に基づいて主観的に決定した．

図5に示したBTSのページには，なでしこに含まれる日付選択用のダイアログボックスにつ

図 6　提案システムを介した版管理システムの出力

いての要望と対応結果(ID:@510)が記述されている．また，図6には同じ要望に対する変更履歴(ID:r234)が記録されている．

このプロジェクトでは，開発者が各システムへの入力を行う際に，関連する情報のIDを手動で入力することで，他のシステムとの関連性を管理している（図5下部，および図6上部）．この動作例の場合，開発者は変更履歴r234と@510が関連しているとして，BTS，VCSそれぞれにコメントで記録している．これらの履歴に対して，提案システムのBTSへの出力にr234へのリンクが，VCSへの出力に@510へのリンクが含まれており，適切なリンクを出力できていることを示している．

以上の結果により，提案システムの機能が適切に動作していることが確認できた．これに対して動作速度などの性能は次のようになっている．VCS・BTS双方から無作為に抽出した40件の項目を提案システム経由で閲覧したとき，データ収集にかかる時間はVCSに対しては平均0.173秒, BTSに対しては平均0.230秒, 同様に推定の場合はVCSが平均1.05秒, BTSが平均1.06秒，提示の場合は，VCSが平均0.750ミリ秒，BTSが平均1.75ミリ秒となった．総合した処理時間で見ると，VCSが平均1.23秒，BTSが平均1.29秒である．この時間は，閲覧にやや引っかかりを感じるものの，許容できる範囲内の時間であると思われる．なお，この時使用したPCは，OS・Windows XP SP3，CPU・Core2Duo E4600，メモリ2GBである．

提案システムの実行ファイルの構成を見てみると，実行ファイル・プラグインを含めたファイルサイズが約2MB（.NET Frameworkを除く）となっている．これに追加して実行後に作成されるデータベースファイルがあるが，次章で行う実験の後にその容量を確認したところ1.2MB程度であった．また実行時のメモリ使用量は25MB程度となっている．以上のように，提案システムはディスクスペース，実行時メモリともに省スペースで動作させることが出来る．

４．動作実験

４．１．実験方法

提案システムの適切な出力件数について評価するために実験を行う．前述のなでしこプロジェクトでは，手動にて関連するBTS・VCSの項目へのリンクが示されている．実験では開発者が手動で設定しているリンクを，システム関連する情報として出力すべきリンクとみなし，提案システムの出力と開発者によるリンクの比較を行う．実験手順としては，開発者によるリンクをスクリプトによって抽出する．同様に抽出した提案システムの出力に開発者によるものと同じものが含まれているかを確認，集計する．なお，システムの出力は最も推定スコアの小さいものから5件を出力し，上位1件から5件までそれぞれにおいて，開発者によるリンクと同じものが含まれているかどうか確認する．

４．２．対象プロジェクト

なでしこプロジェクトのBTSには2010/07/15時点で529件のバグが登録されている． 開発者によるリンクは，修正が行われたバグに対して示されるが，その数はVCSへのリンクが106件（全数の20%），BTSへのリンクが38件（同7%）である．同様に，VCSには2010/07/15時点で238項目が存在し，開発者によるリンクは，BTSに対しては90件（全数の37%），VCSに対しては15件（同6%）存在する．

４．３．実験結果

実験の結果を表1に示す．開発者リンク数とは，BTS・VCSそれぞれにおいて，その各項目のうちプロジェクト参加者によるBTS・VCSへのリンクが示されている項目数である．システムの出力数とは，キーワードの不一致などによってシステムが推定を行えなかった項目を除いた項目数である．上位1件から上位5件での一致とは，システムによる出力上位1件から上位5件それぞれの中に，開発者によるリンクと同じものが1つでも含まれている項目数である．なお，括弧内は開発者リンク数に対する割合である．

表 1　開発者によるリンクとシステムによる出力の項目数

	BTS(全 529 項目)		VCS(全 238 項目)	
リンク先	BTS	VCS	BTS	VCS
開発者リンク数	38	106	90	15
システムの出力数	476	476	184	184
上位 1 件での一致	15 (39%)	12 (11%)	34 (38%)	3 (20%)
上位 2 件での一致	17 (45%)	16 (15%)	47 (53%)	7 (47%)
上位 3 件での一致	18 (47%)	16 (15%)	54 (60%)	7 (47%)
上位 4 件での一致	21 (55%)	17 (16%)	57 (63%)	8 (53%)
上位 5 件での一致	22 (58%)	18 (17%)	62 (69%)	8 (53%)

４．４．考察

まず, システム出力の上位5件とBTS･VCSにおける開発者によるリンクとの比較を考える．BTSからBTS，VCSからBTS，VCSからVCSへのリンクは，開発者によるリンクの半数以上(53%)をシステムの出力によってカバーできていることが確認できる．しかし，BTSからVCSへのリンクは18件(17%)しかカバーできていない．この原因として考えられるのは，BTSでは多くの単語を使ってその項目の説明が書かれていることが多く，逆にVCSでは少ない単語で簡潔に書かれていることがあげられる．例えば，「なでしこが持つ暗号化の機能を用いて作成された暗号が解読される恐れがある」というトピックについて，BTSはその方法について多くの言葉を尽くして議論を行うが, VCSでは対策を行ったとしか書かれていない．この違いのため，関連する項目であってもBTSに多く含まれる単語がVCSにないために，推定がうまく出来ていないのではないかと考える．しかし，単語数の違いとすると，なぜVCSからBTSへのリンクは半数以上という結果となっているかという疑問がある．この理由としては，VCSでは単語数が少ない，つまり注目する単語が特定されているため，単語数が多くても同じ単語の出現数が多いBTSの項目を推定できているのではないかと考える．

次に，システム出力の上位1件から上位5件の違いについて考える．表1のどの列の結果を見ても，出力する件数を減らすごとに一致項目数も減少している．表2は表1における一致項目数を，上位1件と上位2件の差，上位2件と上位3件の差と，順に計算してまとめたものである．VCSへのリンクの場合は出力する件数を，上位2件から上位1件へと減らしたときに一致する項

表 2　一致項目数の差

	BTS(全529項目)		VCS(全238項目)	
リンク先	BTS	VCS	BTS	VCS
上位1件と上位2件の差	2 (6%)	4 (4%)	7 (15%)	4 (27%)
上位2件と上位3件の差	1 (2%)	0 (0%)	7　(7%)	0　(0%)
上位3件と上位4件の差	3 (8%)	1 (1%)	3　(3%)	1　(6%)
上位4件と上位5件の差	1 (3%)	1 (1%)	5　(6%)	0　(0%)

目の割合がBTSでは15%から11%，VCSでは47%から20%と，その他の上位5件から上位4件，上位4件から上位3件などの減少幅（それぞれBTSでは17%から16%と16%から15%，VCSでは53%から53%と53%から47%）より大きい．また，上位1件と上位5件の一致割合を比較したとき，BTSからVCSへのリンクの場合を除いて，約1.5倍から2.5倍の一致率の向上が見られる．以上より, 複数の推定結果を出力する方が, 有用なリンクを提示できる可能性が高い. しかし，出力する件数が多いと，その出力の中に不要な情報も多くなるため，目的の情報を探す手間が増える．この考察より，上位1件のみの出力では有用な出力が出来る可能性が低いが，あまり多い出力件数も好ましくないため，上位5件の程度の出力がよいと考える．

以上の2つの考察を行ったが，開発者によるリンクは開発者が任意で入力しているため6%～37%しか入力されておらず，システムの出力が有用にもかかわらず適切であると判断できなかったものがある．例えば，BTSに登録されている「配列のメモリリーク」に関する項目から，これに対応したVCSの「配列のメモリリーク対策」の項目へのリンクは開発者により設定されていないが，提案システムの出力にはこれらに対して相互にリンクが示されている．よって，この実験結果は実際の適切なリンクを示している数よりは少なめに示されていると考えられる．

５．適用実験

５．１．実験方法

提案システムの出力について，開発者にとって有用な情報を出力しているか確認するために，実際のプロジェクトに提案システムを適用し，実験を行った．実験結果は開発者に対するアンケートによって得る．実験に用いたシステムでは，３．３節における実装の下，キーワードは4章の実験と同様のものを用い，実験開始時点でのBTSとVCSの情報を予めキャッシュデータベースに格納した状態で実験を行う．

アンケートに含まれる質問は次の通りである．

問1 実験システムは開発プロジェクトにとって有用であると感じましたか？

問2 実験システムが出力する5件の関連する項目の中に，実際に関連する項目，出力が妥当だと思われる項目が含まれていましたか？

問3 実験システムが出力する項目の数は適当な数でしたか？

問4 実験システムが出力する項目の数はどの程度が良いと思いますか？

問5 実験システムを経由してBTSなどを閲覧するとき，経由しないで閲覧するときと比べてストレスを感じましたか？

なお，回答は問4以外“1:あてはまらない”・“2:どちらかといえばあてはまらない”・“3:どちらかといえばあてはまる”・“4:あてはまる”の4択，問4は“1:もっと少なく”・“2:少し少なく”・“3:少し多く”・“4:もっと多く”の4択である．

５．２．対象プロジェクト

4章の実験と同じく，なでしこプロジェクトのBTS/VCSを対象に実験を行った．実験対象者はなでしこプロジェクトにコミッタとして参加する開発者3名である．

５．３．実験結果

前述のようになでしこプロジェクトの開発者3名に対してアンケートを行い，回答を得た．その結果を表3に示す．表中，各行は各質問，各列は各回答に対応する．

表 3　アンケート結果

	あてはまらない ／もっと少なく	どちらかといえば あてはまらない ／少し少なく	どちらかといえば あてはまる ／少し多く	あてはまる ／もっと多く
問 1	0	1	1	1
問 2	0	3	0	0
問 3	0	0	1	2
問 4	0	1	1	1
問 5	1	1	1	0

５．４．考察

各質問の結果について検討する．

問1は“1:あてはまらない”以外の回答が1名ずつという結果となっており，システムを有用と感じたという回答が優勢である．否定的な回答者の自由記述を見ると，「BTS/VCSの項目番号のリンクだけを示すのではなく，そのタイトルも表示して欲しい」とあった．これは，項目番号だけではその内容がどの程度関連しているかどうかが直感的にわからないため，いちいちその項目を確認しなければならないということであると考える．この問題については，自由記述による指摘のようにタイトルも含めての提示を検討する必要がある．

問2は全員の回答が“2:どちらかといえばあてはまらない”と否定的な結果となっている．この理由として考えられるのは，関連項目の推定方法が簡易的であるために，うまく推定できていない項目があるということである．アンケートの自由記述にも，「キーワードに入っていない単語の関連項目が出ない」という意見が見られた．このことから，キーワードの動的決定などの推定方法の改善が必要であると考える．

問3と問4は，どちらも出力する項目数についての質問であり，関連が強い項目なので併せて考える．まず問3は，“3:どちらかといえばあてはまる”が1名，“4:あてはまる”が2名と，肯定的な結果が得られた．一方問4は，“1:もっと少なく”以外が1名ずつとなっている．同一の回答者による問3と問4の回答を合わせて書くと，“回答3—回答2”・“回答4—回答4”・“回答4—回答3”という組み合わせになる．これらの結果のうち，まず問3の回答が“3:どちらかといえばあてはまる”・“4:あてはまる”のみであることから，上位5件という出力がおおむね妥当であることが確認できる．問題3と問題4の対応を見ると，問3が回答3であったものについては，単純に5件よりやや少ない結果の出力を望んでいるものと思われる．対して，問3が回答4であったものにつ

いては，自由記述に「より深く関連を追いたいときは，さらに多くの結果が欲しい」とあることから，通常時は5件の出力で十分にあるものの，広範にわたる問題に関する項目などについてはより多くの出力が必要なのではないかと考える．また，単純に出力数をいくらかの数に増やすという方法以外に，推定のスコアによって，ある一定スコアを閾値にすることで，出力数を変動させることも考えられる．

問5は“4:あてはまる”以外の回答が1名ずつと，閲覧によるストレスをあまり感じない回答者の方が多い結果となっている．回答がばらついた原因として，推定処理を各個人のPCで行っているということが考えられる．提案システムの構成上，推定処理にかかる時間が，BTS/VCSの各項目にアクセスしてから，閲覧できる状態になるまでの時間に加わる．そのため，PCの性能や常駐ソフトの影響などによって，閲覧までのタイムラグが変動し，閲覧よるストレスの受け方が変わってきていると考えられる．この点に関する解決策として，閲覧よるストレス軽減のために，推定処理におけるそもそもの処理量を減らし，環境による差異を少なくすることがあげられるが，これは推定結果のキャッシュ処理など，計算手順上の工夫である程度軽減できるものと考えられる．

以上の考察をまとめると，システムの出力のうち，有用なものは少なく，現状の推定方法を改良し，推定精度および速度の向上が必要であると考えられる．しかし，出力件数という点においては4章の実験の考察で導かれた「5件程度がよい」という結果をおおむね満足させる結果となった．ただし，この実験は3名のみという少ない人数で行ったため，今後より多くの人数に対する調査により詳細な検討が必要である．

６．関連研究

関連研究として先に挙げた二つのシステムについて述べる．

Trac[4]は版管理システムと連動するバグ管理システムである．Tracの利用者はバグ管理システムに保存された不具合に割り当てられたIDを，版管理システムへの投稿に含めることで，関連づけを行う．しかし，このシステムでは関連づけを開発者の手動で行っており，開発者がVCSの情報とBTSの関連性を把握していなければならない．また，既存プロジェクトで他のBTSを利用している場合，既存のシステムに登録された情報をTracに移す必要があるが，手動で再投稿を行う場合，多くの労力が必要な上，開発コンテキストの理解に重要な情報である投稿日時が失われてしまう．既存システムのデータベースをTracで用いられているデータベースに変換する事もできるが，両者のデータベース構造に関する知識が必要となり，容易に情報を移行できるものではない．

石川らはMLとVCSを組み合わせた検索システムについて提案している[9]．このシステムはそれぞれの開発支援システムから情報を取得し，投稿者や投稿時間，メッセージから抽出したキーワードなどの要素から情報を関連づけている．既存のシステムをそのまま取り込む方式のため，移行の問題は解決されるが，ホストとなるマシン上にシステムを構築するため，マシンが用意できない場合や，レンタルサービスを利用している場合など，適用できないケースが存在する．

大平らは複数の開発支援システムの情報を収集し，プロジェクトのリアルタイムな管理を行うシステムEPM(Empirical Project Monitor)を提案している[5]．EPMはCVSやMailman, GNATSなど6つの開発支援システムに対応しており，未対応のシステムに対してもRubyのスクリプトを作成することで比較的容易に対応することができる．プロジェクトで既に利用している開発支援システムをそのまま利用でき，提案システムと類似している．

しかし，EPMはプロジェクトの定量的な分析を目的に設計されている．そのため，ソースコードの規模推移や，不具合数の推移などを容易に把握できるが，個々の不具合に対する議論や修正作業のようなコンテキストの把握には適していない．また，石川らのシステムと同様にホストとなるマシンを用意しなければならず，外部システムの結合も対応していない．

これらのシステムと，本稿で提案したシステムの比較を表4に示す．Tracは種類の異なる情報をひとつのシステムで扱うということができるが，その中で関連する情報を自動で結合させることはできず，既存システムの情報も活用できない．石川らや大平らは関連情報の結合や既存システムの活用が可能であるが，その運用にホストとなるマシンを必要とする．また，石川

表 4　既存の統合システムと提案システムの比較

	Trac[4]	石川ら[9]	大平ら[5]	提案システム
複数種類の情報の格納	○	○	○	○
既存システムの活用	×	△	○	○
ホストマシン	必要	必要	必要	不要
関連情報の結合	手動	自動	プロジェクト単位で集計	自動
任意のシステムの追加	×	×	△	○

らは任意のシステムを追加することができず，大平らは任意のシステムを追加することはできるが，外部システムの追加ができないという制限がある．本稿の提案システムはローカル環境でプロクシサーバとして動作するためホストとなるマシンを必要とせず，任意のシステムについて，既に蓄積された情報を活用することができる．さらに，情報間の関連性の推定を行うことで，関連情報の結合にも対応している．

7．おわりに

本稿では，複数の開発支援システム間にまたがった検索における，同一コンテキストの情報にスムーズにアクセスできないという問題を解決するために，開発支援システムを変更することなく統合するシステムについて提案を行った．提案システムはプロクシして動作し，その通信の過程で得た情報により開発コンテキストを推定することで，対象とする開発支援システムを選ばず，スムーズに同一コンテキストの情報へのアクセスができる．また，提案システムが適切に動作するかを確認するために，実際のOSSプロジェクトで用いられている2つの開発支援システムに提案システムを適用し，動作の確認を行った．さらに，実際のプロジェクトに対して提案システムを適用し，アンケートによりその出力の評価を行った．その結果，提案システムにより開発者が選択した関連情報と同じものが出力可能であることを確認した．以上より，提案システムによるシステム間の情報統合は一定の有効性があることを確認した．

今後の課題としては，まず提示方法の改善があげられる．これは，適用実験の考察にあるように，項目タイトルの提示の他，その項目の投稿日時や提示する位置についても検討する必要があると考える．また，情報間の関連について推定する方法についても，精度を高めるための方法を検討する．現状の推定方法は簡易なものであるが，例えばTF-IDF法を利用して動的に単語を決定できれば精度向上につながるのではないかと考える．

また，開発のコンテキストを示す重要な要素である，日時についても推定に用いることで精度が高まると考えられる．1つのコンテキストに属する一連の情報であっても，時間の経過と共に主題が変化していくことがある．例えばあるバグに関する議論で，最初はバグの再現性や挙動についての議論をしていても，議論が進むにつれて，不具合位置の特定や修正方法についての議論に話題が遷移する事がある．このとき，話題の主題は同じバグであっても，再現性と修正方法という異なる内容が含まれるため，特徴付けるキーワードが変化すると考えられる．このような，時系列の変化をとらえることは開発のコンテキストを理解するためには重要である．今後の研究では，一連の話題から短期間のイベントを抽出して話題推移の把握を支援する手法[11]をシステムに導入することで，より精度の高い推定が行えると考えられる．

謝辞

在学中，研究に関することの他，様々な事柄についてご指導いただいた上野秀剛先生に深い感謝の意を表する．セミナー発表において，貴重なご意見いただいた世古忠先生，土井滋貴先生，山口賢一先生，芦原佑樹先生に深い感謝の意を表する．実験への協力を快諾していただいたなでしこプロジェクト代表酒徳峰章氏，ならびに開発参加者各位に感謝の意を表する．研究に関する議論にお付き合いいただいた上野研究室各位に感謝の意を表する．

参考文献

[1]情報処理推進機構: IT人材白書2010, http://www.ipa.go.jp/about/press/20100407.html (2010).

[2]Bikram Sengupta, Satish Chandra, Vibha Sinha: A Research Agenda for Distributed Software Development, In Proc. The 28th International Conference on Software Engineering (ICSE), pp.731-740 (2006).

[3]Carl Gutwin, Reagan Penner, Kevin Schneider: Group Awareness in Distributed Software Development, In Proc. The 2004 ACM Conference on Computer Supported Cooperative Work, pp.72-81 (2004).

[4]Edgewall Software: The Trac Project!, http://trac.edgewall.org/ (2010).

[5]大平雅雄, 横森励士, 阪井　誠, 岩村　聡, 小野英治, 新海　平, 横川智教: ソフトウェア開発プロジェクトのリアルタイム管理を目的とした支援システム, 電子情報通信学会論文誌D-I, Vol.J88-D-I, No.2, pp.228-239 (2005).

[6]Google Project Hosting: Project Hosting on Google Code, http://code.google.com/hosting/ (2010).

[7]Geeknet, Inc.: SourceForge.net: Find and Develop Open Source Software, http://sourceforge.net/ (2010).

[8]Canonical Ltd.: Launchpad, https://launchpad.net/ (2010).

[9]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).

[10]酒徳峰章: manatubbs - くじらはんどラボ, http://d.aoikujira.com/labo/index.php?manatubbs (2010).

[11]菊池匡晃, 岡本昌之, 山崎智弘: 階層型クラスタリングを用いた時系列テキスト集合からの話題推移抽出, 日本データベース学会論文誌, Vol.7, No.1, pp.85-90 (2008).

業績

1. コンテキストの推定による開発支援システム間の情報統合, 谷宗一郎, 上野秀剛, 第169回ソフトウェア工学研究発表会, SE-169-9, 2010年7月23日