論　文

MUDABlue: ソフトウェアリポジトリ自動分類システム

川口　真司[†]　　パンカジ ガーグ[††]　　松下　誠[†]　　井上　克郎[†]

MUDABlue: Automatic Software Repository Categorization System

Shinji KAWAGUCHI[†], Pankaj K. GARG[††], Makoto MATSUSHITA[†], and Katsuro INOUE[†]

あらまし　近年，ネットワークの発達と分散ソフトウェア開発の普及に伴い，大規模なソフトウェアリポジトリが一般的なものとなってきている．ソフトウェアリポジトリとはソースコードやドキュメント，バグレポート等の各プロジェクトの成果物を蓄積するためのデータベースである．通常，ソフトウェアリポジトリは膨大な数のプロジェクトを保持しているため，例えば，開発者が現在開発中のものと似ているプロジェクトを捜したり，管理者が会社内で走っている全プロジェクトを俯瞰するといったことに活用できる．しかし，保持内容が膨大なためにプロジェクト同士の関連を判定して整理するには非常な労力を必要とする．そこで，我々はソフトウェアを自動的に分類する MUDABlue システムを作成した．MUDABlue の特長は以下の 4 つである．1) 分類にはソースコードのみを使用，2) 分類先となるカテゴリ集合も自動的に決定する，3) ソフトウェアを二つ以上のカテゴリに分類することを許す，4)Web インタフェースで分類結果を表示する．本論文では既存の分類手法との比較を通じて MUDABlue システムの有効性を議論する．

キーワード　ソフトウェアリポジトリ，ソフトウェアライブラリ，自動分類，再利用

1. まえがき

ネットワークの発達および分散ソフトウェアの一般化にともなって，ネットワーク上にソフトウェアプロダクトを保存するソフトウェアリポジトリが注目されている．特にオープンソースの世界では，SourceForge [1] を始め，さまざまなアーカイブリポジトリがある．ソフトウェアリポジトリはソースコードだけではなく各種ドキュメントやメーリングリストのログ，開発用の掲示板やバグレポート，バイナリパッケージ配布用ページ等が用意されている．2004 年 12 月現在，SourceForge を利用しているプロジェクトは 9 万を越えており，現在も急速に増加しつづけている．さらに，企業内プロジェクトを共有するために，社内向けサービスとしての導入も活発に行われている．

ソフトウェアリポジトリを利用するメリットとしては，1) 希望するソフトウェアを検索し，利用する，2) 開発者が開発中のソフトウェアと類似したソフトウェアを検索して，ソフトウェアの再利用や開発者間での情報共有を図る，3) 管理者が管理下にあるプロジェクト全体を把握するという点が挙げられる．

ソフトウェアリポジトリには膨大なプロジェクトが登録されているため，実際に希望するソフトウェアや類似ソフトウェアを検索するためには，ソフトウェアが分類，整理されていなければならない．現存するリポジトリサービスでは，プロジェクトを登録する人がそのプロジェクトの分類を行う．しかし，このような手動分類では分類先となるカテゴリ集合を定義するのに労力がかかる．また，個々のソフトウェアがどのカテゴリに分類されるのか，分類を行うソフトウェア登録者に依存するため分類の一貫性に問題がある．

このような問題を解消するために，我々は自動的にソフトウェアの分類を行う MUDABlue システムの提案を行う．MUDABlue は IR 手法(注1)の一種である潜在的意味解析手法 (Latent Semantic Analysis．以下 LSA) を用いて，カテゴリ集合の作成とソフトウェアの分類を自動的に行う．MUDABlue はソースコードのみに依存するため，管理者の入力は一切必要としない．LSA は単語間の意味的繋がりを統計的手法を用い

[†] 大阪大学大学院情報科学研究科，豊中市
Graduate School of Information Science and Technology, Osaka University, 1-3 Machikaneyama-cho, Toyonaka-shi, 560-8531 Japan
[††] Zee Source
Zee Source, 1684 Nightingale Avenue, Suite 201, Sunnyvale, CA 94087 USA

（注1）：IR: Information Retrieval

ることで抽出する手法である [2].

MUDABlue は分類した結果を Web インタフェースを通じてユーザに提供する．2. で述べるとおり MUDABlue は一つのソフトウェアに対して複数カテゴリを割りあてることを許す非排他的な分類を行う．分類結果を描画するために，我々は Cluster Map 手法 [3] を元にした Univiable Cluster Map 手法を定義する．本手法を用いることでソフトウェアリポジトリ全体を容易に閲覧することが可能となる．

以下，2. にてソフトウェア分類手法が備えるべき特性について考察し 3. にて提案する MUDABlue 分類手法について述べる．そして 4. にて MUDABlue 分類手法について実験を行い， 5. でその実験結果について考察を述べる．最後に関連文献について 6. で触れたのちに 7. で本論文のまとめを行う．

2. ソフトウェア分類

ソフトウェア自動分類手法は一般に (1) 分類先となるカテゴリ集合を定義する工程と，(2) 各ソフトウェアをカテゴリに割りあてる工程の 2 つからなる．

既存の自動的分類に関する研究 [4,5] では (2) の工程の自動化のみに着目しており，カテゴリ集合についてはリポジトリ管理者から与えられることが前提となっている．我々は (1) のカテゴリ集合を作成する工程の自動化も重要であると考える．カテゴリ集合の定義は，カテゴリ結果に重大な影響があること，そしてカテゴリ集合の作成もまた労力のかかる作業だからである．

また，これまでの分類方法では通常，ソフトウェアは用途によって分類されるが，依存ライブラリなど異なる視点での分類もまた有効である．このような分類を実現するには，非排他的な分類が必要になる．さもなければ，もっとも特徴的な側面のみしか抽出することができない [6].

さらに，既存の自動分類に関する研究ではソフトウェアに付随する各種文書に対して解析を行い，分類を実現している．しかし開発文書はソフトウェアによって質，量が大きく異なる．特に開発初期のプロジェクトでは文書が全く存在しないことも多い．そのため，どのようなプロジェクトにも存在するソースコードを用いたほうが，より適用範囲が広くなる．

これらのことを踏まえて，自動ソフトウェア分類に求められる特性として，(1) カテゴリ集合も自動的に抽出すること，(2) ソフトウェアが複数のカテゴリに分類されるのを許すこと，(3) ソースコードのみに依存すること，以上 3 点をあげることができる．

3. MUDABlue

MUDABlue システムは大きくわけてカテゴリ抽出，分類を行う分類部と，分類結果を利用してカテゴリ，ソフトの検索を行うカテゴリ描画部とで構成される．本節では，それぞれのサブシステムについて概要を述べる．

3.1 分類手法

MUDABlue は 2. で述べた 3 つの特性を備えたソフトウェア自動分類システムである．本節では，まず MUDABlue で用いている IR 手法である LSA について簡単に述べ，その後 LSA を用いたカテゴリ抽出方法の概要，および詳細ステップを述べる．

3.1.1 潜在的意味解析手法 - LSA

LSA は文書群のなかから，単語間の潜在的な関連を考慮して，単語間もしくは文書間の類似度を算出する手法である [2]．LSA では文書内に出現する単語の出現頻度を表す word-by-document 行列を作成する．そして，この行列に対して特異値分解と呼ばれる数学的操作を行うことで単語間の潜在的関連を抽出する．この操作は word-by-document 行列のみを入力として行われるため，例えば，どの語とどの語が類義語であるとか，文書 A と文書 B は同一カテゴリの文書であるといった学習用データは一切使わない． このように，学習用データを必要としないのは LSA の大きな特徴の一つである．

LSA はデータマイニングの他，人間の知識獲得過程の理解等，さまざまな分野で応用されている [7,8]. ソフトウェア工学においても，単一のソフトウェアを複数のコンポーネントに分割したり [9]，ドキュメントとソースコード間の結びつきを復元するのに活用されている [10]．LSA の詳細については文献 [2] に詳しく述べられている．

3.1.2 カテゴリ抽出手法

ソフトウェアの集合からカテゴリを抽出するために，我々は関数名や変数名，型名などの識別子に着目する．原則的には，プログラマは識別子に任意の名前をつけることが可能であるが，通常，識別子にはその動作や役割に応じた名前がつけられていることが多い．様々なコーディングスタイルについて論じた文献においても，識別子にはその実体を表す名前をつけることがソフトウェアの品質を保つ上で重要であることが指摘されている [11–13].

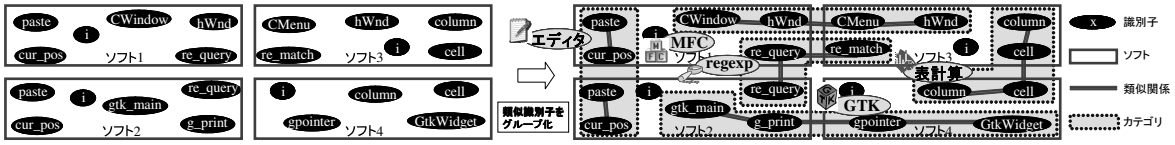


図 1　ソースコード内の識別子の関連からカテゴリを抽出
Fig. 1　Retrieve categories from source code using relationships among identifiers

そこで，LSA を利用して類似した識別子を集めて，一つの概念を構成できるのではないかと考えた．例えば “gtk_window” や “gtk_main”，“gpointer” といった識別子が存在していれば，そのソフトは GTK ライブラリを利用しているものと考えられる．我々は，このように生成された概念を**カテゴリ**と定義する (図 1)．そして，もしソフトが，そのカテゴリに含まれる識別子を一つでも持っているなら，そのカテゴリに所属するものとする．図 1 はカテゴリ抽出過程の例である．図 1 では，“cut, copy, paste” を “エディタ” カテゴリとして，“CWindow, CMenu” を “MFC” カテゴリとして抽出している．このとき，Software1 は “エディタ” カテゴリと “MFC” カテゴリに属する．

3.1.3　MUDABlue アルゴリズム

3.1.2 で述べたように，MUDABlue は類似度の高い識別子をグループ化することでカテゴリ抽出を行う．図 2 に MUDABlue 分類手法のデータフローを示す．本手法は 7 つの部分から構成される．

（1） 識別子の抽出

まず，すべてのソフトのソースコードから識別子を抽出する．抽出した単語からは予約語はとりのぞく．またコメント中の単語も除外する．コメントを除外するのは，コメントの量や質がソフトによって大きなばらつきがあることと，多くのソフトにはコピーライトやライセンス条項がコメントとして含まれており，これを含めることは分類に悪影響を及ぼすからである．

（2） identifier-by-software 行列の作成

次に LSA の入力となる識別子 - ソフト行列を作成する．これは各識別子が各ソフト中に何回現れたかを表す行列である．

（3） 識別子の選別

LSA を適用するまえに，不要な識別子の除去を行う．本手法では，単一のソフトにのみ出現する単語，および過半数以上のソフトに出現する単語を削除する．単一のソフトに現れる単語は LSA にとって完全に不要である．また過半数以上のソフトに現れる単語は，どのようなプログラムにも普遍的に出現する分類とは無関係な単語と考えられる．

（4） LSA の適用

識別子を選別した後，identifier-by-software 行列に対して LSA を適用し，行列を変換する．LSA を適用することで，類義語が多く含まれる場合にも適切な類似度測定が可能になる．

（5） 識別子間の類似度から，カテゴリを抽出

LSA の結果得られた行列から，すべての類似度のペアについて類似度を計測する．本研究では LSA における単語間類似度計測で一般的に使われる $cosine$ 尺度を用いる．そののちクラスタ分析を適用して類似度の高い識別子同士をグループ化する．クラスタ分析とはアイテム間の類似度に従って，アイテムをいくつかのクラスタにグループ化する手法である．ここで得られる識別子のグループを “識別子クラスタ” と呼ぶ．**この識別子クラスタをカテゴリとする．**

（6） ソフトウェアクラスタの作成

識別子クラスタから対応するソフトウェアクラスタを作成する．ソフトウェアクラスタは，カテゴリに所属するソフトの集合を表す．ソフトウェアクラスタは，識別子クラスタに属する識別子のうち，どれか一つでも持っているソフトの集合である．

（7） カテゴリタイトルの作成

この段階で，カテゴリと各カテゴリに属するソフトの集合が得られた．最後に各カテゴリの特長を表すタイトルを作成する．

タイトルの作成のため，ソフトウェアクラスタに属するソフトのソフトウェアベクトルを抜きだし，それを合計する．そして，そのベクトル内で値の大きい識別子を 10 個取りだし，それをタイトルとする．

3.2　カテゴリ描画手法

ソフトウェアリポジトリは一般に大規模であり，得られるカテゴリ数もそれだけ多くなるため，得られたカテゴリをわかりやすくユーザに提示する必要がある．MUDABlue はソフトウェアリポジトリの閲覧，検索のために (1) キーワード検索，(2) カテゴリツリー，(3)Unifiable Cluster Map (UCM) の 3 つのビューを提供

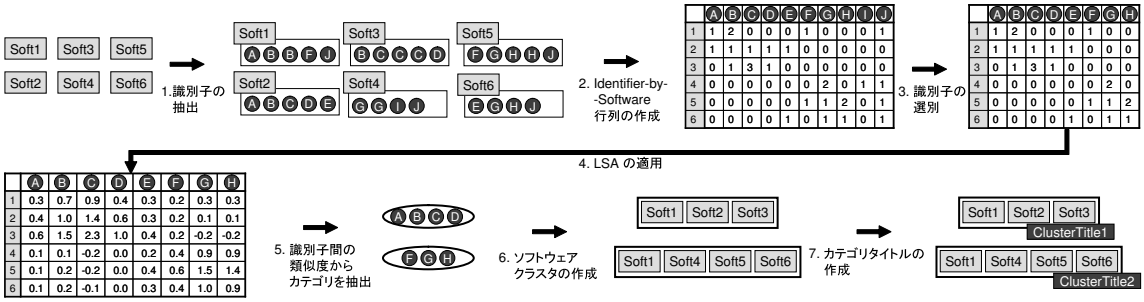


図 2　カテゴリ抽出アルゴリズム
Fig. 2　Algorithm of Retrieving Categories

する．

3.2.1　キーワード検索

キーワードを用いた検索は最も一般的な検索手法である．MUDABlue ではキーワードを用いて，カテゴリおよびソフトを検索可能である．カテゴリ検索時には，カテゴリタイトル，および対応する識別子クラスタに入力されたキーワードと部分一致するカテゴリを表示する．またソフト検索時にはソフト名，もしくはソフト内の識別子とキーワードが部分一致するソフトを表示する．

3.2.2　カテゴリツリー

カテゴリツリーでは抽出したカテゴリの一覧を階層構造にして表示する．カテゴリの階層構造はカテゴリに共通するソフトの数を元に決定する．そのため類似したカテゴリが近くに配置され，それだけユーザにとってカテゴリの閲 覧が容易になる．

本論文では，カテゴリ間の類似度として**オッズ比**を少し変更したものを用いる．母集合 S とその部分集合 $A, B \subset S$ があったとき，カテゴリ間の類似度 $or'(A,B)$ は $or'(A,B) = \begin{cases} ad/bc & \text{if } bc \neq 0 \\ ad|S|^2 & \text{otherwise} \end{cases}$ となる．ただし，$a = |\bar{A} \cap \bar{B}|, b = |A \cap \bar{B}|, c = |\bar{A} \cap B|, d = |A \cap B|$ とする．我々は $or'(A,B)$ を用いてクラスタ分析を行い，その結果をカテゴリツリーとして表示する．

3.2.3　Unifiable Cluster Map

MUDABlue で抽出したカテゴリの描画には，(1) 複数カテゴリへの所属を効率よく描画できること，(2) 十分なスケーラビリティを持っていること，以上 2 点の要求を満たす描画方法が求められる．このような要求を踏まえ，我々はカテゴリ描画手法として **Unifiable Cluster Map** を定義した．Unifiable Cluster Map は Cluster Map [3] を元にした手法である．

カテゴリ	アイテム
A	1, 2, 3, 7, 8, 9
B	1, 2, 3, 4, 5, 6, 10, 11
C	3, 4, 5, 6
D	7, 8, 9, 10, 11
E	7, 8, 9, 10, 11, 12

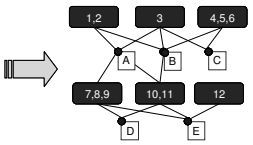


図 3　Cluster Map の例
Fig. 3　Example of Cluster Map

Cluster Map では，図 3 の左側のような関係は，右側のグラフとして表す．カテゴリは黒い丸であらわし，その丸からのびる辺によって所属しているアイテムを示す．例えばカテゴリ C にはアイテム 3, 4, 5, 6 が属しているため，それらの間に辺を引く．また，アイテム 4, 5, 6 はカテゴリ B にも属するため，カテゴリ B とアイテム 4, 5, 6 の間にも辺を作成する．スペースの節約のため，所属するカテゴリが完全に同一であるアイテム同士は一つの四角でまとめる．Cluster Map はこのような形で複数に所属するアイテムの関連を効率的に描画する．しかし，もしカテゴリが数十個を越える規模になるとノードが画面を埋めつくしてしまい，理解困難な状態に陥いる．

そこで，我々は「カテゴリの融合・展開」という操作を追加した Unifiable Cluster Map を定義，実装した [14]．UCM ではカテゴリを融合させることで，一度に 表示されるノードを減らすことができ，画面がノードで埋めつくされてしまうことを防いでいる．ユーザは融合したカテゴリを再び分離することもできる．ソフトウェアクラスタと，結合したカテゴリが線で結ばれている場合，それらのソフトは結合しているカテゴリのうちいずれかに所属していることを示している．実際には，初期状態ではさきほど定義したカテゴリ階層の上位 7 個のカテゴリのみが表示され，その他のカ

Category	Software
boardgame	Sjeng-10.0, bingo-cards, btechmux-1.4.3, cinag-1.1.4, faile_1_4_4, gbatnav-1.0.4, gchch-1.2.1, icsDrone, libgmonopd-0.3.0, netships-1.3.1, nettoe-1.1.0, nngs-1.1.14, ttt-0.10.0
compilers	clisp-2.30, csl-4.3.0, freewrapsrc53, gbdk, gprolog-1.2.3, gsoap2, jcom223, nasm-0.98.35, pfe-0.32.56, sdcc
database	centrallix, emdros-1.1.4, firebird-1.0.0.796, gtm_V43001A, leap-1.2.6, mysql-3.23.49, postgresql-7.2.1
editor	gedit-1.120.0, gmas-1.1.0, gnotepad+-1.3.3, molasses-1.1.0, peacock-0.4
videoconversion	dv2jpg-1.1, libcu30-1.0, mjpgTools, mpegsplit-1.1.1
xterm	R6.3, R6.4

表 1　実験に利用したソフトウェア
Table 1　The list of sample software systems

テゴリは融合された状態で表示される．

3.3 利　用　例

本節では，使用例を通して MUDABlue がどのように利用できるかを示す．サンプルデータとして SourceForge から 5 つのカテゴリをランダムに選択し，その中から C で書かれた 41 個のプログラムを取得した．サンプルプログラムには計 2,663,215 行，164,102 識別子，9,519 個のファイルが存在した．実際に用いたカテゴリ，およびソフトについては表 1 に示す．

図 4 は “video” というキーワードでカテゴリを検索した画面である．この画面から，動画編集に関するソフトである “dv2jpg” や “libcu30”，“mjpgTools” が提示されていることがわかる．また結果一覧からソフトを選択した場合，UCM も選択されたソフトを中心にしたグラフになる．例えば gedit を選択した場合，エディタとして類似しているソフトや，gtk を利用しているソフトが容易に把握できる．また，UCM によってリポジトリ全体を描画してるのでリポジトリ全体を横断的に把握することができる．詳細を知りたい部分については適宜カテゴリを展開することも可能である．

4. 実　　験

本論文では，(1) MUDABlue が SourceForege で定められたカテゴリをどれだけ抽出できるのか，(2) ライブラリやアーキテクチャによる分類をどれだけ抽出できるのか，という二つの観点から MUDABlue システムの評価を行った．

4.1 実 験 方 法

評価用データセットとして，3.3 で示したものと同じ 41 個のソフトを用いた．これら評価用データセットに対して MUDABlue メソッドを適用してカテゴリ抽出を行った結果を評価する．

評価基準としては，検索アルゴリズムを評価する際によく用いられる適合率 (precision) と再現率 (recall) を用いる．本論文では各ソフトごとに，システムが関連すると提示したカテゴリが適合しているかどうかという観点で適合率と再現率の計算を行った．すなわち，まず各ソフトごとに適合率，再現率を計算し，それらの平均を全 体の適合率，再現率とした．

また，適合率と再現率は一般に相反する関係にあるため，これら二つの指標を統合する指標である F 値による評価も行った．F 値は，適率合と再現率の調和平均であり，適合率を p，再現率を r としたとき，$\frac{2pr}{p+r}$ と定義される．

4.2 分 類 結 果

表 2 は MUDABlue によって抽出されたカテゴリの抜粋である．各行がひとつのカテゴリを表しており，各列は左からカテゴリタイトル，所属しているソフト名，対応する識別子クラスタの識別子数である．この実験では合計 40 個のカテゴリが得られた．そのうち 18 個は SourceForge で決められた用途による分類に合致するものであり，11 個はライブラリやアーキテクチャに基づく新しく抽出されたカテゴリであった．残り 11 個はそのどちらでもなく，意味のないカテゴリであった．表 2 では，カテゴリ No. 1，2，4，5，6，7，10 が用途によるカテゴリであり，カテゴリ No. 3，8，9 が新しいカテゴリの例である．新しいカテゴリは No.3 (YACC カテゴリ)，No.8, 9 (GTK カテゴリ)，No.14 (SSL カテゴリ)，No.22 (regexp カテゴリ)，No.25 (JNI カテゴリ)，No.29, 33, 40 (Win32 API カテゴリ)，No.30 (getopt カテゴリ)，No.32 (Python/C カテゴリ) の 11 個である．

図 5 に適合率と再現率の結果を示す．比較のため，GURU [4] と SVM(注2) を利用した分類手法である Ugurel ら [5] の適合率・再現率もあわせて示す．

図 5 から，MUDABlue はこれらの研究に対して同程度の適合率，再現率となっていることがわかる．実際に MUDABlue の結果の F 値が 0.72 であるのに対し

(注2)：SVM: Support Vector Machine

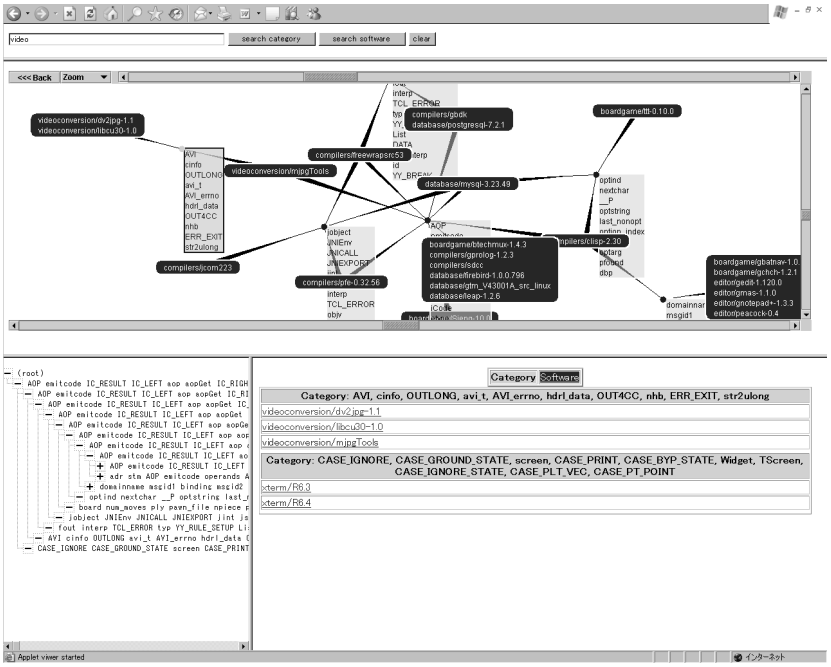


図 4　“video” キーワードで検索
Fig. 4　Searching with keyword “video”

	Title of cluster	Software	# of tokens
1	AOP, emitcode, IC_RESULT, IC_LEFT, aop, aopGet, IC_RIGHT, pic14_emitcode, iCode, etype	compilers/gbdk, compilers/sdcc	8597
2	CASE_IGNORE, CASE_GROUND_STATE, screen, CASE_PRINT, CASE_BYP_STATE, Widget, TScreen, CASE_IGNORE_STATE, CASE_PLT_VEC, CASE_PT_POINT	xterm/R6.3, xterm/R6.4	2160
3	YY_BREAK, yyvsp, yyval, DATA, yy_current_buffer, tuple, yy_current_state, yy_c_buf_p, yy_cp, uint32	compilers/gbdk, database/mysql-3.23.49, database/postgresql-7.2.1	223
4	AVI, cinfo, OUTLONG, avi_t, AVI_errno, hdrl_data, OUT4CC, nhb, ERR_EXIT, str2ulong	videoconversion/dv2jpg-1.1, videoconversion/libcu30-1.0, videoconversion/mjpgTools	177
5	domainname, msgid1, binding, msgid2, domainbinding, pexp, __builtin_expect, transmem_list, codeset, codesetp	boardgame/gbatnav-1.0.4, boardgame/gchch-1.2.1	165
6	board, num_moves, ply, pawn_file, npiece, pawns, moves, white_to_move, move_s, promoted	boardgame/Sjeng-10.0, boardgame/cinag-1.1.4, boardgame/faile_1_4_4	154
7	xdrs, blob, DB, UCHAR, XDR, mutex, key_length, logp, page_no, bdb	database/firebird-1.0.0.796, database/mysql-3.23.49	118
8	domainname, N_, binding, gchar, GtkWidget, PARAMS, codeset, gpointer, loaded_l10nfile, argz	boardgame/gbatnav-1.0.4, boardgame/gchch-1.2.1, editor/gnotepad+-1.3.3, editor/peacock-0.4	118
9	GtkWidget, gchar, gpointer, gint, widget, gtk_widget_show, N_, g_free, dialog, g_return_if_fail	boardgame/gbatnav-1.0.4, editor/gnotepad+-1.3.3, editor/gmas-1.1.0, editor/gedit-1.120.0, editor/peacock-0.4	104

表 2　MUDABlue 分類結果 (抜粋)
Table 2　MUDABlue Result (excerpt)

て，GURU の結果のうち F 値の最大値は 0.6591 であり (適合率 0.9，再現率 0.52 のとき)，Ugurel らの結果の F 値の最大値は 0.6531 である (適合率 0.66，再現率 0.65 のとき).

GURU や Ugurel らの結果はそれぞれの論文から直接引用したものであり，これらの結果は異なるデータセットに対して適用した得られた結果である．そのためこの数値だけで優劣を決定することはできない．しかしこの結果から，本研究はこれら過去の研究とは大きく異なったアプローチで分類を行っているが，同程度の信頼性を保っていることがわかる．

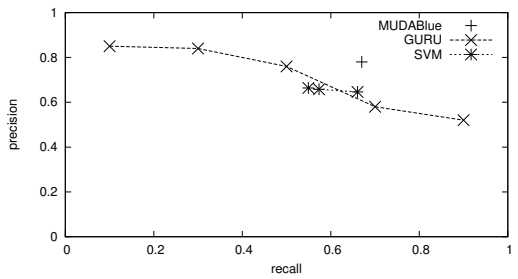


図 5　適合率-再現率
Fig. 5　Precision-recall graph

5. 考　　察

5.1 カテゴリ抽出手法

本論文では，MUDABlue が用途による分類にあわせて，ライブラリによる分類も適切に行なえることを示した．実験において，MUDABlue は GTK や yacc カテゴリなどを自動的に抽出している．MUDABlue は人による知識入力が必要ないため，新しいライブラリを使うソフト群が追加されても，これを自動的に抽出することが期待できる．

さきほどの実験では，適合率，再現率において MUDABlue は既存の研究に比べてひけをとらないことを示した．さらに MUDABlue には 2. で述べた 3 つの特性，すなわち (1) カテゴリ集合の自動抽出，(2) 多重従属の許容 (3) ソースコードのみに依存を供えているという大きな違いがある．

ただし，MUDABlue が導出するカテゴリは，全体として細かいクラスタになる傾向がある．4. での実験では一つのクラスタに含まれるソフト数は平均 2.6 個となっている．これは不要識別子のフィルタリングが不十分なために，大きな単位でクラスタを切りだしてしまうと著しく精度が下るからである．また，カテゴリタイトルも解釈の難しいタイトルが生成される場合がある．これらの問題に対処するには，どちらも MUDABlue の利点を減じてしまうが，ソースコードだけでなくドキュメントも入力として利用することが考えられる．また，大規模なコーパスを前もって作成してもらう，またはタイトルを人に直接入力してもらう等，人手をかける方法も考えられる．

5.2 カテゴリ描画手法

3.3 での利用例を通じて，MUDABlue が複数の閲覧検索手法を組みあわせて，より実用的な検索を実現していることを示した．Frakes ら [15] はいくつかの閲覧手法の比較を実証的に行なっている．その結果，全体的には検索手法ごとの適合率，再現率には有意差が認められないこと，しかし検索手法によって実際に提示されるアイテムは異なると結論づけている．そのため，本システムでは両タイプの検索手法を組みあわせることで，効率的な閲覧を可能にしている．

非排他的集合の関係を描画するために，我々は Cluster Map を元に Unifiable Cluster Map を実装した．そのほかにも InfoCrystal [16] もそのような手法の一つである．また，酒井ら [17] や Allan ら [18] の手法のように，各アイテムを所属する集合に応じて何らかの空間にマッピングする手法も存在する．

6. 関 連 研 究

本節では，IR 手法をソフトウェア工学に応用している研究についてとりあげる．ソフトウェアの自動分類に関する研究，非排他的集合の描画に関する研究については 5. でとりあげている．

IR 手法を用いてソースコードから何らかの有用な情報を抽出する研究として，ソフトウェアクラスタリングに関する研究がある．ソフトウェアクラスタリングとは，ソフトウェアの理解支援のために，一つのソフトウェアを機能単位で分割する手法である．ソフトウェアクラスタリングの研究については，LSA を使う研究 [9] のほか，自己同一化マップを利用する研究 [19] がある．これらの研究では，ファイルや関数を単位として IR 手法を用いて分類を行うことでソフトウェアクラスタリングを実現している．

また，再利用可能なコンポーネントを取得するために，IR 手法を用いる手法も提案されている．CodeBroker [20] は Java に対応したコンポーネント取得システムの一つである．CodeBroker は開発者がソースコードに記述した JavaDoc を自動的に取得し，LSA を用いて類似したコメントを持つメソッドを提示する．

Marcus ら [10] はソースコードと設計書などの開発文書の間の関連を自動的に抽出する手法を提案している．ソースコード内のコメントと設計文書との類似度を計算して，ソースコードに対応する開発文書を提示したり，設計書の記述に対応する実装を提示する．

この他，SVM を利用してバグ追跡システムに新しく登録された不具合の担当者を自動的に割りあてたり [21]，プログラム中に潜在的に含まれるバグを自動的に発見する手法 [22] が提案されている．

7. む す び

本論文では，ソフトウェアリポジトリ自動分類システム MUDABlue の設計および評価について述べた．MUDABlue は決められたカテゴリにそってソフトウェアを分類するのではなく，カテゴリそのものもソースコードから抽出して分類を行う．そのため MUDABlue は事前の知識入力を必要としない．また，MUDABlue システムは取得したカテゴリに基づいてソフトウェアリポジトリを閲覧するインタフェースも備えている．我々がおこなった実験では，用途による分類のみならず依存ライブラリやアーキテクチャによる分類もあわせて行えることを示し，また MUDABlue の既存研究に対する優位性も示した．

MUDABlue は完全に自動化されたシステムである．自動化は大規模なソフトウェアリポジトリに対して適用する際には重要な特性であるが，現在のところ人の入力も活用することで改善が期待できる部分もある．たとえば，タイトルの理解しやすさの向上はそのような部分の一つである．

謝辞

本研究の一部は文部科学省「e-Society 基盤ソフトウェアの総合開発」の委託に基づいて行われた．

文　献

[1] SourceForge.net. `http://sourceforge.net/`.

[2] T. K. Landauer, P. W. Foltz and D. Laham: “An introduction to latent semantic analysis”, Discourse Processes, **25**, pp. 259–284 (1998).

[3] C. Fluit, M. Sabou and F. van Harmelen: “Supporting user tasks through visualisation of light-weight ontologies”, Handbook on Ontologies in Information Systems (Eds. by S. Staab and R. Studer), Springer-Verlag (2003).

[4] Y. S. Maarek, D. M. Berry and G. E. Kaiser: “An information retrieval approach for automatically constructing software libraries”, IEEE Trans. Software Engineering, **17**, 8, pp. 800–813 (1991).

[5] S. Ugurel, R. Krovetz, C. L. Giles, D. M. Pennock, E. J. Glover and H. Zha: “What’s the code? automatic classification of source code archives”, Proc. 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Edmonton, Alberta, Canada, pp. 632–638 (2002).

[6] S. Kawaguchi, P. K. Garg, M. Matsushita and K. Inoue: “Automatic categorization algorithm for evolvable software archive”, Proc. 2003 Int. Workshop on Principles of Software Evolution(IWPSE 2003) (2003).

[7] T. K. Landauer and S. T. Dumais: “Latent Semantic Analysis and the Measurement of Knowledge”, Educational Testing Service Conf. on Natural Language Processing Techniques and Technology in Assessment and Education, princeton (1994).

[8] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas and R. A. Harshman: “Indexing by latent semantic analysis”, J. Am. Soc. Inf. Sci., **41**, 6, pp. 391–407 (1990).

[9] J. I. Maletic and A. Marcus: “Using latent semantic analysis to identify similarities in source code to support program understanding”, Proc. 12th IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI’00), pp. 46–53 (2000).

[10] A. Marcus and J. I. Maletic: “Recovering documentation-to-source-code traceability links using latent semantic indexing”, Proc. 25th Int. Conf. on Software Engineering(ICSE2003), Portland, OR, pp. 125–135 (2003).

[11] B. W. Kernighan and R. Pike: “The Practice of Programming”, Addison-Wesley (1999).

[12] A. Hunt and D. Thomas: “The Pragmatic Programmer: From Journeyman to Master”, Addison-Wesley (1999).

[13] S. McConnell: “Code Complete”, Microsoft Press, 2nd edition (2004).

[14] S. Kawaguchi, P. K. Garg, M. Matsushita and K. Inoue: “Mudablue: An automatic categorization system for open source repositories”, Proc. 11th Asia-Pacific Software Engineering Conf.(APSEC2004) (2004).

[15] W. B. Frakes and T. Pole: “An empirical study of representation methods for reusable software components”, IEEE Trans. Software Engineering, **20**, 8, pp. 617–630 (1994).

[16] A. Spoerri: “Infocrystal: a visual tool for information retrieval”, Proc. 2nd Int. Conf. on Information and Knowledge Management, Washington, D.C., United States, pp. 11–20 (1993).

[17] 酒井, 山口, 川合：“図形オブジェクトの遠隔度に基づく階層集合の可視化モデル”, 情処学論, **40**, 9, pp. 3455–3470 (1999).

[18] J. Allan, A. V. Leouski and R. C. Swan: “Interactive cluster visualization for information retrieval”, Technical Report IR-116, Center for Intelligent Information Retrieval, University of Massachusetts, Amherst (1997).

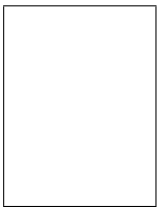
[19] A. Chan and T. Spracklen: “Discoverying common features in software code using self-organising maps”, Proc. Int. Symposium on Computational Intelligence (ISCI’2000), Kosice, Slovakia (2000).

[20] Y. Ye and G. Fischer: “Supporting reuse by delivering task-relevant and personalized information”, Proc. 24th Int. Conf. on Software Engineering(ICSE 2002), Orlando, Florida, USA, pp. 513–523 (2002).

[21] G. Lucca, M. D. Penta and S. Gradara: “An approach to classify software maintenance requests”, Proc. Int. Conf. on Software Maintenance (ICSM’02), Montreal, Quebec, Canada (2002).

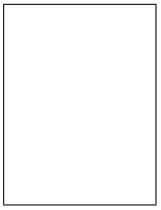
[22] Y. Brun: “Software fault identification via dynamic analysis and machine learning”, Master’s thesis, MIT Department of Electrical Engineering and Computer Science, Cambridge, MA (2003).

（平成 xx 年 xx 月 xx 日受付）



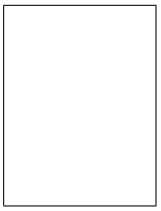
川口　真司

平 13 阪大・基礎工・情報卒．現在同大大学院情報科学研究科博士後期課程在学中．ソフトウェアプロセスの研究に従事．情報処理学会会員．



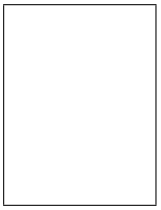
パンカジ ガーグ

1984 インド技術大学計算機科学部卒．1989 南カルフォルニア大学大学院計算機科学研究科博士課程修了．同年 ヒューレットパッカード株式会社入社．2002 同社退社．同年 ZeeSource 設立．オープンソースソフトウェア開発の研究に従事．ACM 会員



松下　　誠

平 5 阪大・基礎工・情報卒．平 10 同大大学院博士課程了．同年同大・基礎工・情報・助手．平 14 阪大・情報・コンピュータサイエンス・助手．平 17 阪大・情報・コンピュータサイエンス・助教授．博士 (工学)．ソフトウェアプロセス，オープンソースソフトウェア開発の研究に従事．



井上　克郎　（正員）

昭 54 阪大・基礎工・情報卒．昭 59 同大大学院博士課程了．同年同大・基礎工・情報・助手．昭 59〜昭 61 ハワイ大マノア校・情報工学科・助教授．平 1 阪大・基礎工・情報・講師．平 3 同学科・助教授．平 7 同学科・教授．平 14 阪大・情報・コンピュータサイエンス・教授．博士 (工学) ソフトウェア工学の研究に従事．情報処理学会，日本ソフトウェア科学会，IEEE，ACM 各会員．

Abstract Recently, largescale software repositories have become feasible. Such repositories store software projects with their source code, documents, and so forth. To leverage repositories, categorization of software systems is essential, however, determining what projects are related to each other is often a hard problem. Hence, we propose MUDABlue, automatic software repository categorization system. MUDABlue has four major aspects: 1) it relies on source codes only, 2) it determines category sets automatically, 3) it allows a software system to be a member of multiple categories, and 4) it depicts categorizations graphically. We demonstrate MUDABlue's effectiveness by comparing with existing research tools.

Key words Software acquisition, Reusable libraries, Software libraries, Repositories, Clustering.