The University of Osaka
Institutional Knowledge Archive

Title	MUDABlue : ソフトウェアリポジトリ自動分類システム
Author(s)	川口, 真司; ガーグ, パンカジ; 松下, 誠 他
Citation	電子情報通信学会論文誌D-I. 2005, J88-D-I(8), p. 1217-1225
Version Type	VoR
URL	https://hdl.handle.net/11094/26587
rights	©（社）電子情報通信学会 2005
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

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[†a)]　　パンカジ　ガーグ[††]　　松下　誠[†]　　井上　克郎[†]

MUDABlue: Automatic Software Repository Categorization System

Shinji KAWAGUCHI[†a)], 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 の特長は以下の四つである．(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ー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ー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及び分散ソフトウェアの一般化に伴って，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] をはじめ，様々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，バイナリー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 万を超えており，現在も急速に増加し続けている．更に，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ー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ーに分類されるのか，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

† 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
a) E-mail: s-kawagt@ist.osaka-u.ac.jp

(注1)：IR: Information Retrieval.

LSA）を用いて，カテゴリー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 は単語間の意味的つながりを統計的手法を用い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ーを割り当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] をもとにした 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ー集合を定義する工程と，(2) 各ソフトウェアをカテゴリーに割り当てる工程の二つからなる．

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，カテゴリー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) のカテゴリー集合を作成する工程の自動化も重要であると考える．カテゴリー集合の定義は，カテゴリー結果に重大な影響があること，そしてカテゴリー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，最も特徴的な側面のみしか抽出することができない [6].

更に，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方が，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) カテゴリー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ー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 点を挙げることができる．

3. MUDABlue

MUDABlue システムは大きく分けてカテゴリー抽出，分類を行う分類部と，分類結果を利用してカテゴリー，ソフトの検索を行うカテゴリー描画部とで構成される．本章では，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.** で述べた三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 を用いたカテゴリー抽出方法の概要，及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 は文書群の中から，単語間の潜在的な関連を考慮して，単語間若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 は同一カテゴリー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ほか，人間の知識獲得過程の理解等，様々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付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 カテゴリー抽出手法

ソフトウェアの集合からカテゴリーを抽出するために，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付けることが可能であるが，通常，識別子にはその

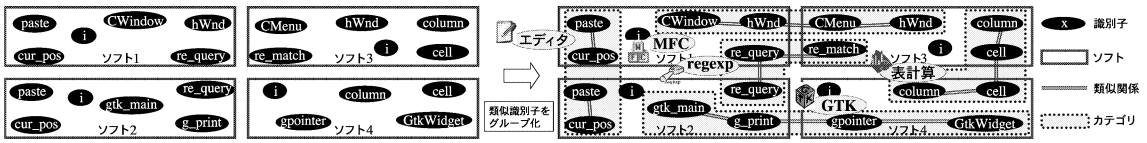


図 1　ソースコード内の識別子の関連からカテゴリーを抽出
Fig. 1　Retrieve categories from source code using relationships among identifiers.

動作や役割に応じた名前が付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付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].

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ーと定義する（図 1）．そして，もしソフトが，そのカテゴリーに含まれる識別子を一つでももっているなら，そのカテゴリーに所属するものとする．図 1 はカテゴリー抽出過程の例である．図 1 では，“`cut, copy, paste`” を “エディタ” カテゴリーとして，“`CWindow, CMenu`” を “MFC” カテゴリーとして抽出している．このとき，Software1 は “エディタ” カテゴリーと “MFC” カテゴリーに属する．

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ー抽出を行う．図 2 に MUDABlue 分類手法のデータフローを示す．本手法は七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取り除く．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 を適用する前に，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，及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，カテゴリーを抽出

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ーとする．

（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，カテゴリー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，どれか一つでももっているソフトの集合である．

（7）　カテゴリータイトルの作成

この段階で，カテゴリーと各カテゴリーに属するソフトの集合が得られた．最後に各カテゴリー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出し，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 個取り出し，それをタイトルとする．

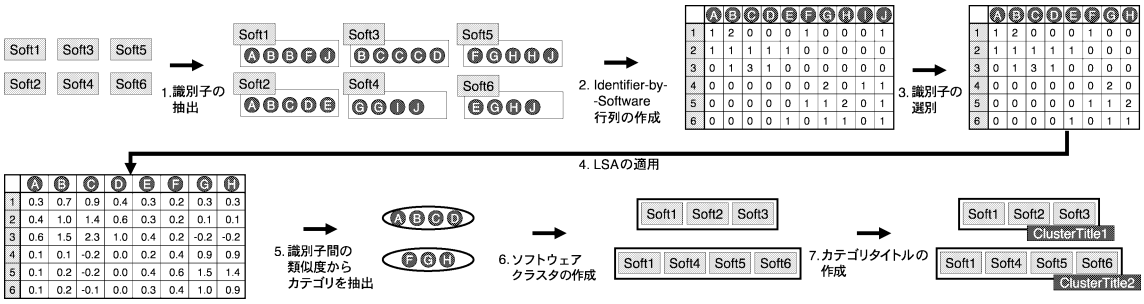


図 2　カテゴリー抽出アルゴリズム
Fig. 2　Algorithm of retrieving categories.

3.2　カテゴリー描画手法

ソフトウェアリポジトリは一般に大規模であり，得られるカテゴリー数もそれだけ多くなるため，得られたカテゴリーを分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) カテゴリートリー，(3) Unifiable Cluster Map（UCM）の三つのビューを提供する．

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，カテゴリー及びソフトを検索可能である．カテゴリー検索時には，カテゴリータイトル，及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ーを表示する．またソフト検索時にはソフト名，若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　カテゴリートリー

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

本論文では，カテゴリー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，カテゴリー間の類似度 $or'(A,B)$ 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ートリーとして表示する．

Category	Elements
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

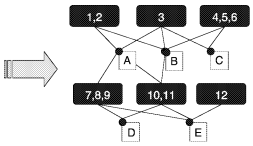


図 3　Cluster Map の例
Fig. 3　Example of cluster map.

3.2.3　Unifiable Cluster Map

MUDABlue で抽出したカテゴリーの描画には，(1) 複数カテゴリーへの所属を効率良く描画できること，(2) 十分なスケーラビリティをもっていること，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，我々はカテゴリー描画手法として Unifiable Cluster Map を定義した．Unifiable Cluster Map は Cluster Map [3] をもとにした手法である．

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．カテゴリーは黒い丸で表し，その丸から延びる辺によって所属しているアイテムを示す．例えばカテゴリー 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 はカテゴリー B にも属するため，カテゴリー 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，所属するカテゴリー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，もしカテゴリーが数十個を超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「カテゴリーの融合・展開」という操作を追加した Unifiable Cluster Map を定義，実装した [14]．UCM ではカテゴリーを融合させることで，

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

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

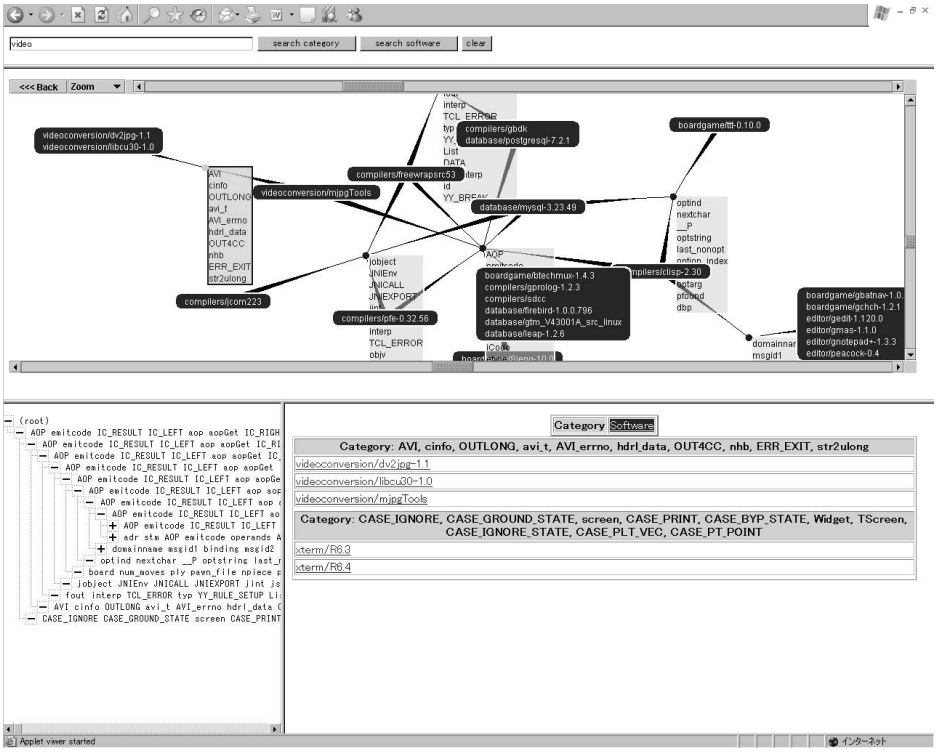


図 4　“video” キーワードで検索
Fig. 4　Searching with keyword “video.”

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ーを再び分離することもできる．ソフトウェアクラスタと，結合したカテゴリー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ー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，初期状態では先ほど定義したカテゴリー階層の上位 7 個のカテゴリーのみが表示され，その他のカテゴリー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 から五つのカテゴリー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ー，及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ーを検索した画面である．この画面から，動画編集に関す

表 2　MUDABlue 分類結果（抜粋）
Table 2　MUDABlue result (excerpt).

	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

るソフトである “dv2jpg” や “libcu30,” “mjpgTools” が提示されていることが分かる．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ー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ー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ー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ー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ーの抜粋である．各行が一つのカテゴリーを表しており，各列は左からカテゴリータイトル，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 個のカテゴリーが得られた．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ーであった．残り 11 個はそのどちらでもなく，意味のないカテゴリーであった．表 2 では，カテゴリー No.1, 2, 4, 5, 6, 7, 10 が用途によるカテゴリーであり，カテゴリー

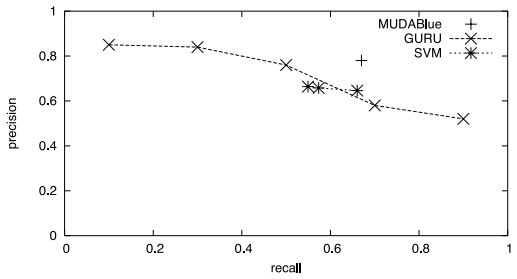


図 5　適合率–再現率
Fig. 5　Precision-recall graph.

No.3, 8, 9 が新しいカテゴリーの例である．新しいカテゴリーは No.3（YACC カテゴリー），No.8, 9（GTK カテゴリー），No.14（SSL カテゴリー），No.22（regexp カテゴリー），No.25（JNI カテゴリー），No.29, 33, 40（Win32 API カテゴリー），No.30（getopt カテゴリー），No.32（Python/C カテゴリー）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・再現率も併せて示す．

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分かる．実際に MUDABlue の結果の F 値が 0.72 であるのに対して，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分かる．

5. 考　　察

5.1 カテゴリー抽出手法

本論文では，MUDABlue が用途による分類に合わせて，ライブラリによる分類も適切に行えることを示した．実験において，MUDABlue は GTK や yacc カテゴリー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．

先ほどの実験では，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．更に MUDABlue には **2.** で述べた三つの特性，すなわち (1) カテゴリー集合の自動抽出，(2) 多重従属の許容 (3) ソースコードのみに依存を備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ーは，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出してしまうと著しく精度が下がるからである．また，カテゴリー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 カテゴリー描画手法

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合わせて，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合わせることで，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 をもとに 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

本章では，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取り上げる．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.** で取り上げている．

(注2)：SVM: Support Vector Machine.

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も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．

このほか，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当てたり [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 の設計及び評価について述べた．MUDABlue は決められたカテゴリーにそってソフトウェアを分類するのではなく，カテゴリー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ー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．我々が行った実験では，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併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．例えば，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, vol.25, pp.259–284, 1998.

[3] C. Fluit, M. Sabou, and F. van Harmelen, “Supporting user tasks through visualisation of light-weight ontologies,” in Handbook on Ontologies in Information Systems, ed. S. Staab and R. Studer, pp.415–434, 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. Softw. Eng., vol.17, no.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, pp.632–638, Edmonton, Alberta, Canada, 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), pp.195–200, 2003.

[7] T.K. Landauer and S.T. Dumais, “A solution to Plato’s problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge,” Psychological Review, vol.104, pp.211–240, 1997.

[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., vol.41, no.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), pp.125–135, Portland, OR, 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, 2nd ed., Microsoft Press, 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), pp.184–193, 2004.

[15] W.B. Frakes and T. Pole, “An empirical study of representation methods for reusable software components,” IEEE Trans. Softw. Eng., vol.20, no.8, pp.617–630, 1994.

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

[17] 酒井恵光，山口和紀，川合　慧，“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，vol.40, no.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.

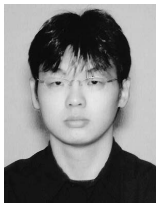
[19] A. Chan and T. Spracklen, “Feature indicators: A self-organizing approach to legacy code,” Proc. Int. Conf. on Artificial Intelligence (IC-AI’2000), pp.1449–1454, Las Vegas, NV, USA, 2000.

[20] Y. Ye and G. Fischer, “Supporting reuse by delivering task-relevant and personalized information,” Proc. 24th Int. Conf. on Software Engineering (ICSE 2002), pp.513–523, Orlando, Florida, USA, 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), pp.93–102, 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 17 年 1 月 17 日受付，3 月 24 日再受付）



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 インド技術大計算機科学部卒．1989 南カルフォルニア大大学院計算機科学研究科博士課程了．同年ヒューレットパッカード（株）入社．2002 同社退社．同年 Zee Source 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 59〜61 ハワイ大マノア校・情報工学科・助教授．平元阪大・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．