Vol.2014-SE-184 No.8
Vol.2014-EMB-33 No.8
2014/5/20

識別子名のタグクラウドを用いたコードクローン理解支援ツールの開発

佐野 真夢[1,a)]　崔 恩瀞[1]　山中 裕樹[1]　吉田 則裕[2]　井上 克郎[1]

概要：コードクローンとは, ソースコード中に存在する互いに一致または類似した部分を持つコード片のことであり, ソフトウェア保守を困難にする要因の 1 つと言われている. ソースコード中のコードクローンの存在位置を示したクローン散布図を用いて, コードクローンを多く含むコンポーネントを選択できるが, 各コードクローンのソフトウェア中における役割を理解するには実際のコードを読む必要がある. コードクローンの処理内容を直観的に理解できれば, 重要なコードクローンを把握し, 着目するコードクローンを効率良く選択できると考えられる. 本研究では, 識別子名を抽出し, それをキーワードとしたタグクラウドを用いたコードクローン理解支援ツールを開発した. 本研究における識別子名のタグクラウドはコードクローンの処理内容の直観的な理解を目的としている.

キーワード：コードクローン，ソフトウェア保守，タグクラウド

Developing a Tool for Understanding Code Clone Using Tag Cloud of Identifier Name

Abstract: Code Clone is a code fragment that is identical or similar to other code fragments in source code. Clones have been known to be a contributing factor for difficulties in software maintenance. To focus on relevant clones, visual techniques such as scatter plots have been used for location identification. However, it is often necessary to read the source code contents a code clone to properly understand its function. Proper understanding ensures effective identification of relevant clones. In this study, we developed a tool for understanding code clones using identifiers in the source code. Our tool uses a tag cloud to visualize extracted identifiers from clones. We show that tag clouds can be used to intuitively understand the function of code clones.

Keywords: Code Clone, Software Maintenance, Tag Cloud

1. まえがき

ソフトウェアの保守工程における大きな問題の 1 つとしてコードクローンが指摘されている. コードクローンとは, ソースコード中に存在する互いに一致または類似した部分を持つコード片のことであり, 既存コードのコピーアンドペーストなどの様々な原因により生じる [4].

コードクローンに対する保守作業の 1 つとして集約が挙げられる. 集約とは, 互いにコードクローンとなっている複数のコード片を 1 つのメソッドなどにまとめることである. 集約を行うことで, 保守対象となるコードクローンを除去することが可能である.

ソフトウェアの保守を容易にするためには, コードクローンの集約を行うことが望ましいと考えられる. しかし, ツールによって自動検出されたコードクローンの中には, 性能改善などの様々な理由により意図的に除去されていないコードクローンが数多く存在する [6]. そのため, 全てのコードクローンが集約の対象になるとは限らない. コードクローンを集約すべきかどうかを判断するには, 実際にソースコードを読む必要があるが, 全てのコードクローンを読むことは非効率的である. このような問題を解決し, コードクローンの効率良い分析を支援するために様々な可

[1] 大阪大学
Osaka University
[2] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[a)] m-sano@ist.osaka-u.ac.jp

視化手法が提案されている.

既存のコードクローン可視化手法の 1 つとしてクローン散布図が挙げられる. クローン散布図とは, ソースコード上に存在するコードクローンの位置を示した散布図のことであり, コードクローン分析ツール Gemini[12] などで利用されている. クローン散布図を用いることで, ソフトウェア中におけるコードクローンが多く存在するコンポーネントを直観的に理解できる. また, そのコンポーネントのみに着目することで, 分析するコードクローンの数を減らすこともできる. しかし, クローン散布図には, 実際のソースコードに関する情報は反映されていない. そのため, コードクローンのソフトウェア中における実際の役割を理解することは困難である.

そこで, 本研究ではコードクローン理解支援を目的として, 変数名などの識別子名を利用した可視化手法を提案した. 具体的には, 開発者がクローン散布図中から興味のあるコンポーネントを選択すると，そのコンポーネント中のコードクローンに含まれる識別子名をキーワードとしたタグクラウドを生成するコードクローン理解支援ツールを開発した. タグクラウドは，テキストデータの視覚的表現の 1 つであり，指定された領域に多くのキーワードが配置され，重要度の高いキーワードが大きく表示されるという特徴を持つ. 本ツールでは，タグクラウドを散布図とソースコードの間に導入したため，開発者が効率的に興味のあるコードクローンを探索できると期待される. 識別子名をキーワードとする理由は, “ソフトウェア開発者はプログラムを読む際, 識別子名の意味からプログラム要素の役割を推測している [9], [13]”ためである. さらに, 本研究では, コードクローンの分析経験を持つ専門家が記述したコードクローンの説明文に基づく有用な識別子名リストを利用して, 本手法により抽出されたコードクローンに含まれる識別子名が有用であることを確認した.

以降, 2 節では本研究に関連する研究について説明する. また, 3 節では本研究で開発したコードクローン理解支援ツールについて説明し, 4 節では本研究で行った評価実験について述べる. そして, 5 節で本研究のまとめと今後の課題について述べる.

2. 関連研究

2.1 コードクローン

コードクローンとは, ソースコード中に存在する互いに一致または類似した部分を持つコード片のことである. 一般的に, コードクローンの存在はソフトウェアの保守を困難にすると言われている [4]. そのため, コードクローンの存在を知ることは重要であるが, 大規模ソフトウェアの場合, 開発者がソフトウェアに含まれる全てのコードクローンを見つけることは非現実的である. 従って, 一般的に, ソフトウェア開発では検出ツールを用いてコードクローンを見つけている [5].

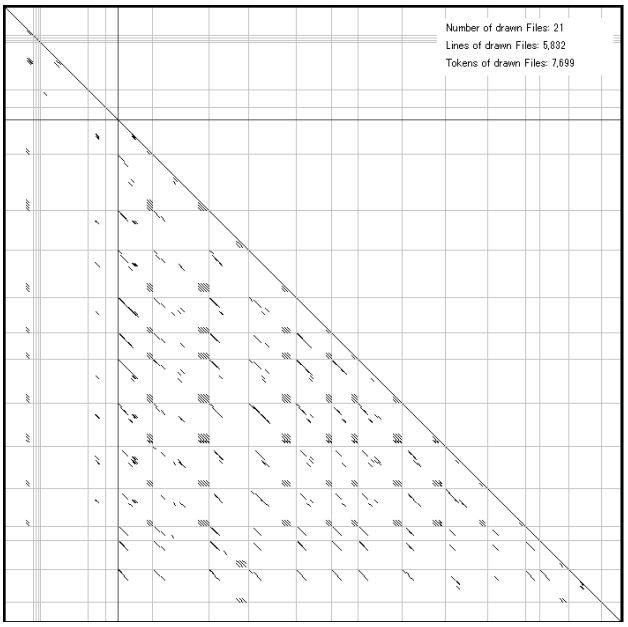


図 1 Gemini におけるクローン散布図の例
Fig. 1 Example of scatter plot in Gemini.

一般的に, 互いにコードクローンとなるコード片の対をクローンペアと呼び, クローンペアにおいて推移関係が成り立つコードクローンの集合をクローンセットと呼ぶ.

コードクローン検出手法の 1 つとしてソースコードの字句解析に基づく手法 [1], [7], [10] が存在する. この手法では, ソースコード中にある同一の文字列を検索することでコードクローン検出を行う.

本研究では, 字句解析ベースのコードクローン検出ツール CCFinder[5] を利用してコードクローン検出を行う. CCFinder は高いスケーラビリティを有しており, 大規模なソフトウェアに対しても実用的な時間でコードクローンを検出できる. また, 空白やタブの有無などのコーディングスタイルを除いて完全に一致するコードクローンだけでなく, 変数名などのユーザ定義名や変数の型などの一部の予約語のみが異なるコードクローンも検出することができるという特徴を備えている. 実際に, CCFinder は様々な大規模ソフトウェアに適用され, その有用性が確認されている [8].

2.2 クローン散布図

既存のコードクローン可視化手法の 1 つとしてクローン散布図が挙げられる. クローン散布図とは, 分析対象となるソースコード上に存在するコードクローンの位置を示した散布図のことであり, コードクローン分析ツール Gemini[12] などで利用されている. **図 1** は Gemini におけるクローン散布図の例である. 図 1 では, コードクローンは斜めの線分としてクローン散布図に描画されている. また, ファイルやディレクトリ単位で区切りが引かれるため,

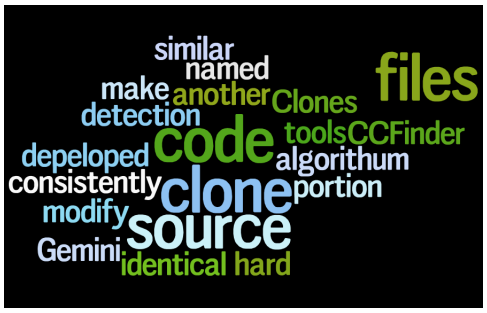


図 2　自然文におけるタグクラウドの例
Fig. 2　Example of tag cloud in natural language text.

ソフトウェア中のどの部分に, どの程度のコードクローンが含まれるのかを直観的に理解できる.

しかし, クローン散布図には, 識別子名などの実際のソースコードに関する情報は反映されていない. そのため, コードクローンのソフトウェア中における役割を理解するには, 実際にソースコードを読む必要があるが, 全てのコードクローンに関するソースコードを読むことは非効率的である. 開発者は識別子名の意味からプログラム要素の役割を推測するため [9], [13], 識別子名を利用することで, コードクローンの役割を直観的に理解し, 注目したいコードクローンをより効率的に見つけ出すことができると考えられる.

2.3　タグクラウド

タグクラウドとは, 文書からキーワードを抽出し, そのキーワードを画面上に表示する視覚的表現の 1 つである. タグクラウドの厳密で普遍的な定義は存在していないが, 一般的には, 表示されるキーワードの大きさは, 対象文書における重要度や関心度を表す指標に基づいて定められており, 相対的に大きく表示されているキーワードは, より重要度や関心度の高いキーワードであることを意味している. **図 2** は, タグクラウド生成 Web サービス Wordle[*1] を用いて生成した自然文に対するタグクラウドの例である. 図 2 において, 他のキーワードよりも相対的に大きく表示されている “code”や “clone”などのキーワードは, 元の自然文で頻出する単語であり, その文章の特徴を表すキーワードである可能性が高いことを示している.

開発者は識別子名の意味を元にプログラム要素の役割を推測することから [9], [13], 識別子名をキーワードとし, コードクローンに対してタグクラウドを適用することで, そのクローンの特徴を把握できると考えられる. しかし, コードクローンに対してタグクラウドを適用した事例はまだ見られない. そこで, 本研究では, タグクラウドを用いてソースコード中の識別子名を可視化することで, コードクローンの理解支援を図る. タグクラウドを用いる利点として, 以下の 2 つが挙げられる.

小さな領域に多くのキーワードを表示できる
キーワードの相対的な重要度を直観的に理解するためには, 全てのキーワードが画面内に収まることが望ましい. しかし, 多くのキーワードが抽出された場合, 単純なリストなどでは全てのキーワードが画面内に収まらない可能性が高いと考えられる. タグクラウドを利用することで, より多くのキーワードを 1 つの画面内に収める事が可能となる.

重要度を直観的に理解しやすい
タグクラウド上に表示されるキーワードの大きさは, その重要度が高いほど相対的に大きく表示される. そのため, どのキーワードがより重要であるのかを直観的に理解することが可能となる.

[*1] http://www.wordle.net/

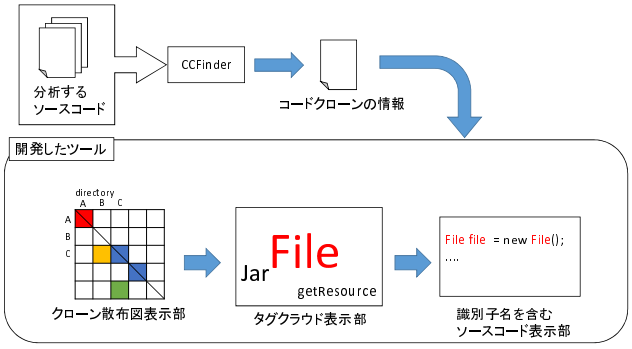


図 3　ツールの概要
Fig. 3　Overview of the proposed tool.

3.　開発ツール

図 3 は本ツールの全体構成を表す概要図である. 本ツールは, CCFinder の出力結果を入力として動作する. そのため, ユーザは分析するソースコードを CCFinder に適用した結果（コードクローンの情報）を事前に準備する必要がある.

本ツールを実行すると, 始めに分析するソースコード中の識別子名の抽出が行われる. その後, 抽出した識別子名の情報と入力したコードクローンの情報（CCFinder の出力結果）に基づいて, クローン散布図の生成を行う. 本ツールでは, “クローン散布図表示部”, “タグクラウド表示部”, “識別子名を含むソースコード表示部”の 3 つの分析環境を提供しており, 以下のような手順で利用する.

手順 1.　最初に表示されるクローン散布図を用いて, コードクローン密度の高いディレクトリ対を探す. 本ツールのクローン散布図は一般的なものとは異なり, ディレクトリ単位のコードクローン密度を示している.

手順 2.　手順 1. で見つけたディレクトリ対を選択することで, そのディレクトリ対に応じた識別子名のタグクラウドが表示される. このタグクラウドを用いて, 選択したディレクトリ対におけるコードクローンの大まかな役割を把握する.

手順 3.　手順 2. で関心のある識別子名が存在する場合は,

Vol.2014-SE-184 No.8
Vol.2014-EMB-33 No.8
2014/5/20

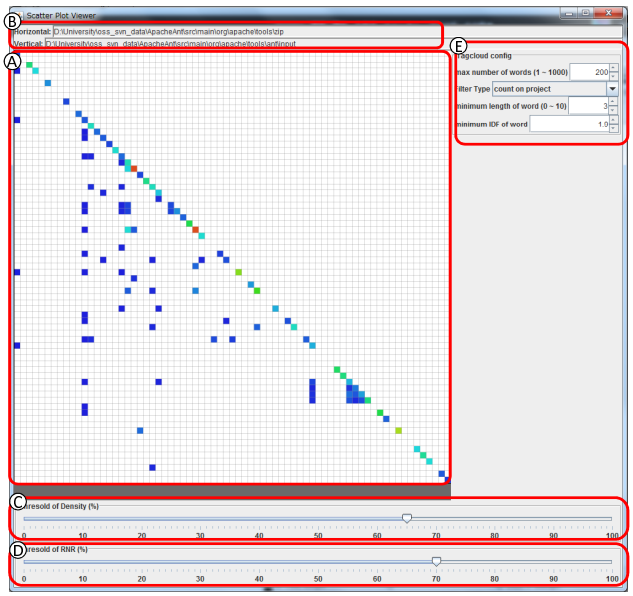


図 4　クローン散布図表示部の例
Fig. 4　Example of scatterplot view.

それを選択することでその識別子名を含む実際のコードクローンを表示できる．これにより，コードクローンに関する詳細な調査を行うことができる．

以降，本節では各分析環境の詳細について説明する．

3.1　クローン散布図表示部

本ツールが提供するクローン散布図は Live scatterplot[2] を参考にしている．Live scatterplot は，2.2 節で述べた一般的なクローン散布図とは異なり，ディレクトリなどのソフトウェア要素の単位に対するコードクローン密度を示す．また，大規模ソフトウェアの場合，一般的なクローン散布図には，コードクローンを表す線分を多数描画するため生成に時間がかかる，線分が密集する場所では可視性が悪くなるなどの問題がある．しかし，Live scatterplot はディレクトリなどの大きな単位で描画を行うため，これらの問題を大きく緩和し，より大規模なソフトウェアに適用できると考えられる．

図 4 は，Ant[*2] に本ツールを適用した結果，実際に表示されたクローン散布図である．図 4 の各記号箇所の説明を以下に示す．

A. コードクローン密度を示したクローン散布図
分析対象を含む全てのソースコードに対するコードクローン密度を示した散布図である．縦軸と横軸は，左上端を原点としてそれぞれ同じ順序でディレクトリの列が並んでいる．また，縦軸と横軸のディレクトリ間にまたがるコードクローンの密度に基づき，各マスが着色される．なお，本ツールにおけるコードクローン密度は，ソースコード全体のトークンのうち，コードクローンに含まれるものの割合を示している．

[*2] http://ant.apache.org/

図 5　コードクローンの密度に対する散布図の色の変化
Fig. 5　Gradation of scatter plot visualizing density of code clone.

また，本ツールの散布図では，コードクローン密度を図 5 に示すグラデーションで表す．コードクローン密度が低い場合は青色で表し，密度が高くなるに従って赤色に変化していく．赤色で表されるコードクローン密度の閾値は図 4 の C. で設定でき，閾値以上の密度は全て赤色で表される．

B. 縦軸・横軸方向ディレクトリの絶対パス
図 4 の A. において，マウスカーソルがあるマスに対応した縦軸・横軸方向ディレクトリの絶対パスを表示する．

C. クローン密度の閾値を設定するスライダー
図 4 の A. のマスの色を決める際に使用する，赤色で表示する密度の閾値を設定できる．

D. RNR の閾値を設定するスライダー
RNR とは，コードクローンの非繰り返し度数を表すメトリクスのことであり，RNR が低い場合，そのコードクローンは print 文の連続などの類似した処理の繰り返しを多く含むことを意味する [3]．一般に，RNR が低いコードクローンは重要度の低い場合が多いため，本ツールでは，RNR が設定した閾値以下のコードクローンを分析対象から除外できる．

E. 識別子名のタグクラウドのパラメータ設定
3.2 節で述べる識別子名のタグクラウドを生成するためのパラメータを設定できる．設定可能なパラメータを以下に示す．

表示上限数
タグクラウドに表示できるキーワードの上限数である．

表示優先順位の指標
タグクラウドに表示するキーワードの優先順位を定める指標である．本ツールでは，識別子名の出現回数の他に，TF-IDF[11] も選択できる．TF-IDF とは，文書中に出現する単語に関する重み付け手法のことであり，出現頻度（TF : Term Frequency）と非一般度（IDF : Inverse Document Frequency）の 2 つの指標を掛け合わせることで重みを計算する．

本ツールでは，文書をソースコード，単語を識別子名と考えて TF-IDF の指標を計算する．本ツールにおいて，識別子名 i の TF_i, IDF_i は以下の式で計算する．以下の式で使用する記号は表 1 にまとめている．

$$TF_i = \frac{n_i}{N},\ IDF_i = \log \frac{|D|}{|d_i|}$$

識別子名 i の TF_i は, 分析対象のソースコード群 s 中における i の出現頻度であり, IDF_i は, あるソースファイルの集合 S における, ファイル単位での i の希少さを意味している. また, S は s を内包した集合であり, 例えば, S はソフトウェアの全てのソースファイル, s はソフトウェアの 1 つのディレクトリに属するソースコードなどがある. なお, 本ツールでは, s を 2 つのディレクトリ（同じディレクトリでも可）に属するソースファイルとしている.

表示する識別子名の長さの下限

不要なものとして除去する識別子名の長さの閾値である. 詳細は 3.2.2 節で述べる.

表示する識別子名の IDF の下限

不要なものとして除去する識別子名の IDF の閾値である. 詳細は 3.2.2 節で述べる.

本ツールにおけるクローン散布図の主な目的は, 注目するディレクトリを削減することである. 一般的に, コードクローン密度の高いディレクトリ対は保守作業が困難であり, 分析する必要性が大きいと考えられる. 本ツールの散布図により, コードクローン密度の高いディレクトリ対を直観的に把握できるため, そのようなディレクトリ対におけるコードクローンに着目することで, 分析の時間的効率を向上できると考えられる.

3.2 タグクラウド表示部

3.1 節で述べた散布図の各マスを選択することで, それに対応したディレクトリ対における識別子名のタグクラウドが表示される. 縦軸と横軸が同じディレクトリの場合は, 単一ディレクトリ内が対象となる.

図 6 は, 図 4 において, あるディレクトリ対を選択した際に実際に表示された識別子名のタグクラウドである. 図 6 において, コードクローンに含まれる識別子名は赤色, 含まれない識別子名は黒色で着色されている.

本ツールにおける識別子名のタグクラウドの主な目的は, ディレクトリ（または, ディレクトリ対）におけるコードクローンの実際の役割を大まかに把握することである. コードクローンの大まかな役割を直観的に把握することで, 関心の無いコードクローンを除外し, その結果, 実際に読む必要のあるソースコードの量を減らすことができると考えられる.

また, 本ツールでは識別子名のタグクラウドと共に識別子名の情報テーブルを表示する. **図 7** は, 図 6 と共に実際に表示された識別子名の情報テーブルである. このテーブルには, 共に表示されたタグクラウドのキーワードに対応した識別子名に関する情報（ソースコード全体での出現回数など）が記載されている.

本ツールにおける識別子名の情報テーブルは, 識別子名同士の厳密な比較のために利用できる. 例えば, タグクラウドでは相対的に大きく表示されているキーワードが重要であることを直観的に理解できるが, 同程度の大きさのキーワード同士を比較する場合, 目視では正確さに欠ける可能性がある. そのような場合, 識別子名の情報テーブルに記載された具体的な数値を参考にして, キーワード（識別子名）の厳密な比較を行う.

以降, 本節では, 本研究で提案する識別子名のタグクラウドの詳細な仕様について説明する.

表 1 提案手法における TF-IDF の計算式の記号の意味
Table 1 Meaning of each symbol used by the expression of TF-IDF in the proposed approach.

記号	意味
N	分析対象中に出現する重複を許した識別子名の総数
n_i	分析対象における識別子名 i の出現回数
$\|D\|$	分析対象を内包するソースファイル集合のファイル数
$\|d_i\|$	$\|D\|$ に属し, かつ, 識別子名 i を含むファイル数

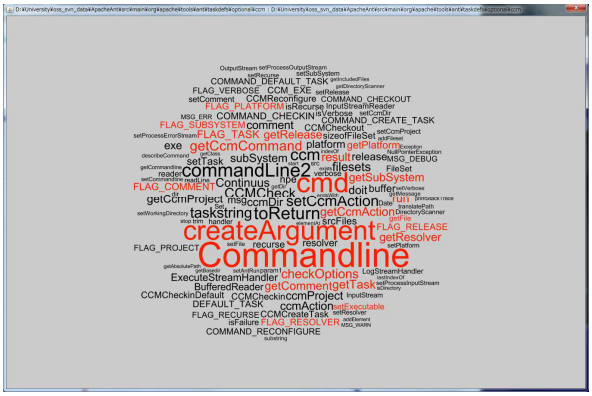
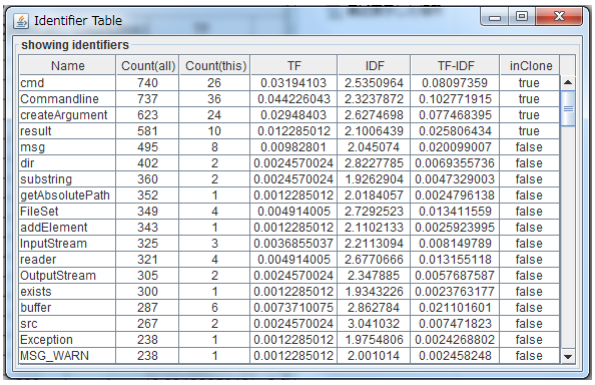


図 6 タグクラウド表示部の例
Fig. 6 Example of tag cloud view.


Identifier Table

showing identifiers

Name	Count(all)	Count(this)	TF	IDF	TF-IDF	inClone
cmd	740	26	0.03194103	2.5350964	0.08097359	true
Commandline	737	36	0.044226043	2.3237872	0.102771915	true
createArgument	623	24	0.02948403	2.6274698	0.077468395	true
result	581	10	0.012285012	2.1006439	0.025806434	true
msg	495	8	0.00982801	2.045074	0.020099007	false
dir	402	2	0.0024570024	2.8227785	0.0069355736	false
substring	360	2	0.0024570024	1.9262904	0.0047329003	false
getAbsolutePath	352	1	0.0012285012	2.0184057	0.0024796138	false
FileSet	349	4	0.004914005	2.7292523	0.013411559	false
addElement	343	1	0.0012285012	2.1102133	0.0025923995	false
InputStream	325	3	0.0036855037	2.2113094	0.008149789	false
reader	321	4	0.004914005	2.6770666	0.013155118	false
OutputStream	305	2	0.0024570024	2.347885	0.0057687587	false
exists	300	1	0.0012285012	1.9343226	0.0023763177	false
buffer	287	6	0.0073710075	2.862784	0.021101601	false
src	267	2	0.0024570024	3.041032	0.007471823	false
Exception	238	1	0.0012285012	1.9754806	0.0024268802	false
MSG_WARN	238	1	0.0012285012	2.001014	0.002458248	false

図 7 本ツールにおける識別子名の情報テーブル
Fig. 7 Table of information on identifier name in the proposed tool.

3.2.1 キーワードの色と大きさ

コードクローンに含まれるキーワードは, 含まれていないものと区別できるように着色を行う. これにより, コードクローンと直接関係のある識別子名を直観的に判断できる.

なお, 本ツールでは, 単一ディレクトリ内, または, 2 つのディレクトリ間にまたがるコードクローンを対象とする. すなわち, クローン散布図表示部で選択したマスに応じたディレクトリ対におけるコードクローンに含まれるキーワードを着色する.

また, タグクラウドのキーワードの大きさは, 各キーワードに対応した識別子名の TF-IDF の値に基づいて決定する. TF-IDF の値が高い識別子名は, 相対的に大きなサイズのキーワードとして表示される.

3.2.2 不要なキーワードの除去

全ての識別子名をキーワードとして表示するのは非現実的であり, また, タグクラウドの可視性を損なうと考えられる. そこで, 本ツールではコードクローン理解支援に有益である可能性の低い識別子名を除去するために, 以下の指標によるキーワードのフィルタリングを行う.

識別子名の長さ

"it" などの 2 文字以下の識別子名は, ほとんどの場合その意味を推測できないと考えられる. 従って, 一定の長さ未満の識別子名はキーワード候補から除外する.

識別子名の IDF

IDF 値の低い識別子名は多数のファイル中に見られる一般的な識別子名である可能性が高く, コードクローンを特徴付けるには不向きと考えられる. 従って, 閾値未満の IDF を持つ識別子名はキーワード候補から除外する.

また, 大規模ソフトウェアの場合, フィルタリングを行ってもなお非現実的な量の識別子名が残ると考えられる. そこで, 本ツールでは識別子名の出現回数が多いものを優先的にタグクラウドのキーワードとして表示する.

なお, 識別子名のスペルの大文字・小文字のみが異なる場合, その違いは識別子名の意味に影響しないと考えられるため同じ識別子名として扱う. また, "getValue" などの複数の英単語で構成された識別子名もそのままの 1 つのキーワードとする. これは, 1 つの識別子名を単語に分割した場合, 元の識別子名と異なる意味を推測する可能性が考えられるためである. 例えば, "addElement" と "getValue" という識別子名を "add", "element", "get", "value" の 4 つのキーワードに分けた場合, 元の識別子名が判断できないため, "addValue" などの存在しない識別子名を推測する可能性がある.

3.3 識別子名を含むソースコード表示部

識別子名のタグクラウドのキーワードがコードクローンに含まれる場合, そのキーワードに対する識別子名を含む実際のコードクローンを表示できる. **図 8** は, 図 6 のキーワード "createArgument" に対する実際のコードクローンの例である. 図 8 における各番号が指す箇所の説明を以下に示す.

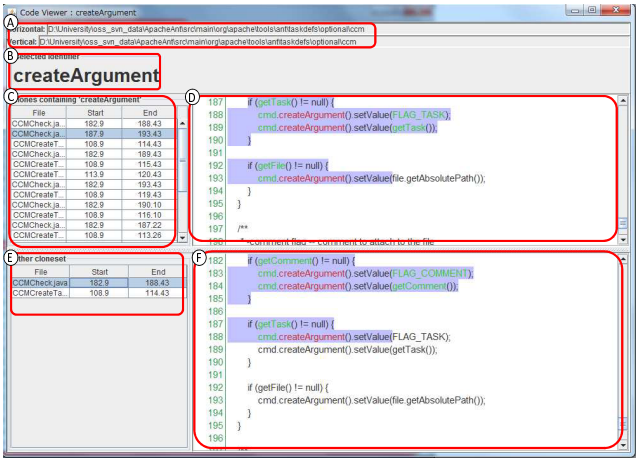


図 8 識別子名を含むソースコード表示部の例

Fig. 8 Example of view that shows source code including an identifier name.

A. ディレクトリ対の絶対パス

識別子名のタグクラウドが対象とするディレクトリ対の絶対パスである.

B. 選択した識別子名

ユーザが選択した識別子名を示す.

C. 識別子名を含むコードクローン一覧

対象ディレクトリ対において, 選択した識別子名を含むコードクローンの一覧である.

D. 識別子名を含むクローンのソースコード

図 8 の C. で選択したコードクローンを含むソースコードを表示する. 選択したコードクローンに該当する場所を青色の背景で強調し, 選択した識別子名を赤色, その他のタグクラウドに表示されている識別子名を緑色の文字で表示する.

E. 同クローンセット内のコードクローン一覧

図 8 の C. で選択したコードクローンと同じクローンセットに属する他のコードクローンの一覧である.

F. 同クローンセット内のクローンのソースコード

図 8 の E. で選択したコードクローンを含むソースコードを表示する. 図 8 の D. と同様, 選択したコードクローンや識別子名を強調表示する.

本ツールにおいて, ユーザが選択した識別子名を含むコードクローンを表示する目的は, 関心のあるキーワードを含むコードクローンの実際のコード片を確認することである.

4. 評価実験

本節では, 本ツールが抽出するタグクラウドのキーワードの有用性に関する評価実験について説明する.

4.1 実験方法

本実験では, 有用なキーワードを事前に定めておき, 本ツールにおけるタグクラウドのキーワードがそれらと一致するか, あるいは推測可能であるかを調べることでその有用性を評価する. 有用なキーワードは, コードクローン分析経験を持つ専門家が書いたコードクローンの説明文（自然文）を元に決定する. また, 有用なキーワードは説明文に出現する名詞と動詞に限定する. ただし, 代名詞や be 動詞などの明らかに有用ではないと考えられる単語は除外する.

本実験において “キーワードが一致した”とは, 有用なキーワードがタグクラウドのキーワードと全く同じ, または, タグクラウドのキーワードの一部に含まれることを示す. 名詞の複数形などの文法的な変化がある場合も一致したとみなす. また, “キーワードが推測可能”とは, タグクラウドのキーワードから有用なキーワードが推測できることを意味し, 例えば, “Windows”と “OS”, “utility”と “util” などが挙げられる.

本実験における評価指標を以下に示す. 以下の評価指標では, タグクラウドのキーワードはコードクローンに含まれるものに限定する. この理由は, 説明文がコードクローンのみを対象しているため, タグクラウドのキーワードも同じ範囲に限定するべきであると考えられたためである.

再現率 本実験では, 有用なキーワードのうち, タグクラウドのキーワードと一致, あるいは推測可能なものの割合を示す.

適合率 本実験では, タグクラウドのキーワードのうち, 有用なキーワードと一致, あるいは推測可能なものの割合を示す.

本実験では, **表 2** に示す Ant の 10 個のディレクトリ対を対象に評価を行った. 表 2 は各ディレクトリ対の相対パスを示しており, 頭の “src/main/org/apache/tools/ant/” は省略されている. また, これらはコードクローン密度が高いものから選択しているが, 最上位の 10 組ではない. この理由は, 他者に説明文を書いてもらう都合上, 多数のコードクローンが存在するディレクトリ対を対象にすることが困難であったためである.

4.2 実験結果と考察

本実験における実験結果を**表 3** に示す. 表 3 からわかるように, 実験結果では, 再現率と適合率は共に平均値が約 0.8 であった. 従って, 平均的に見ると, 本手法が抽出したタグクラウドのキーワードは, ほとんどが有用なキーワードであり, 不要なキーワードは少ないと考えられる. しかし, ディレクトリ対 {util/regexp, util/regexp} などの極端に低い数値を示す結果も存在する. このような結果が生じた要因として, 以下のような可能性が考えらえれる.

- コードクローンの中には処理の一部分などの表現が難しいものが存在する. このようなコードクローンを自然文で説明することは難しく, 結果として説明文を元にした有用なキーワードが不正確となる可能性がある. これは再現率と適合率に影響を与えると考えられる.
- 本ツールでは出現回数の多い識別子名を優先的にタグクラウドに表示するため, ソースコード全体と直接関係の無い処理などの出現回数の少ない識別子名が関係するコードクローンが存在する場合, それを抽出できない. そのため, 再現率が低下する可能性がある.
- 本ツールでは一般性の高いキーワードはタグクラウドに表示されない. ゆえに, コードクローンが文字列の操作などの汎用的な処理の場合, 関連するキーワードを抽出できず, 再現率が低下する可能性がある.
- 有用なキーワードやタグクラウドのキーワードの数自体が少ない場合, 1 つのキーワードに掛かる重みが大きくなり, 指標の値が低下する. 例えば, 有用なキーワードが 3 個の場合, 1 個取得できないだけで再現率は 0.67 になる.

4.3 妥当性への脅威

本実験ではコードクローン分析経験を持つ専門家が書いた説明文を元に有用なキーワードを選定した. しかし, 例えば, Ant に詳しい専門家が書いた説明文を用いると, 説明文の詳細の程度が異なり, 結果が変化する可能性が考えられる. また, 説明文は自然文であるため, 書く人によって個人差が出る可能性が高い. ゆえに, 様々な人が書いた説明文を用いて実験する必要があると考えられる. 同様に, 実験対象を他のディレクトリ対や Ant 以外のソフトウェアにした場合も結果が変化する可能性があると考えられる.

また, 本研究ではコードクローン検出に CCFinder を用いているが, 異なる検出手法を実装したコードクローン検出ツールを用いると結果が変化すると考えられる. 例えば, 識別子名の類似度に基づくコードクローン検出手法 [14] を用いたツールの場合, 内部に出現する識別子名が類似するメソッドが同じクローンセットのコードクローンとして検出される. このような検出ツールを用いた場合, 識別子名ベースの検出手法であることが結果に影響するかもしれない. また, メソッド単位のコードクローンであれば, 説明文を書くことが容易になるため, 上述の問題点が一部解決されると考えられる.

5. まとめと今後の課題

本研究では, ソースコード中に含まれる識別子名を抽出し, 特徴的な識別子名をタグクラウドとして表示することで, コードクローンの実際の役割を直観的に理解する手法を提案した. また, 本手法を実装したクローン散布図ベースのコードクローン分析ツールの開発を行った. そして, コードクローンの説明文に基づく有用なキーワードと比較することで, 本手法により抽出される識別子名が有用なも

表 2　実験対象
Table 2　Pairs of target directories for evaluation.

ディレクトリ対	
taskdefs/optional/clearcase	taskdefs/optional/clearcase
taskdefs/optional/javah	taskdefs/optional/javah
taskdefs/optional/sos	taskdefs/optional/sos
types/spi	types/spi
types/optional/image	types/optional/image
taskdefs/optional/depend/constantpool	taskdefs/optional/depend/constantpool
util/regexp	util/regexp
taskdefs/launcher	taskdefs/launcher
taskdefs/optional/ccm	taskdefs/optional/ccm
types/resolver	types/resolver

表 3　実験結果
Table 3　Results of evaluation.

ディレクトリ対		再現率	適合率
taskdefs/optional/clearcase	taskdefs/optional/clearcase	0.88	0.84
taskdefs/optional/javah	taskdefs/optional/javah	0.90	0.75
taskdefs/optional/sos	taskdefs/optional/sos	0.95	1.00
types/spi	types/spi	0.67	1.00
types/optional/image	types/optional/image	0.93	0.79
taskdefs/optional/depend/constantpool	taskdefs/optional/depend/constantpool	0.71	1.00
util/regexp	util/regexp	0.48	0.67
taskdefs/launcher	taskdefs/launcher	0.80	0.67
taskdefs/optional/ccm	taskdefs/optional/ccm	1.00	0.91
types/resolver	types/resolver	0.67	0.71
平均		0.80	0.83

のであることを確認した.

今後の課題として, まず, 本ツールの有用性に関する評価が必要と考えられる. コードクローンの説明文には個人差がある可能性が高いため, 様々な説明文に対して本手法の評価を行う必要もある. 同様に, 様々なソフトウェアを実験対象に評価を行う必要がある. また, 選択した識別子名の強調表示の範囲をソースコード全体に広げるなどの細かいツールの改善も必要である.

謝辞　本稿の執筆にあたり，貴重なご意見をくださりました大阪大学 大学院情報科学研究科 春名 修介 特任教授，ならびに Raula Gaikovina Kula 特任助教に感謝いたします．本研究は JSPS 科研費 25220003，26730036 の助成を受けたものてす.

参考文献

[1] Basit, H. A. and Jarzabek, S.: Detecting higher-level similarity patterns in programs, *Proc. of ESEC/FSE 2005*, pp. 156–165 (2005).
[2] Cordy, J. R.: Live scatterplots, *Proc. of IWSC 2011*, pp. 79–80 (2011).
[3] 肥後芳樹，吉田則裕，楠本真二，井上克郎：産学連携に基づいたコードクローン可視化手法の改良と実装，情報処理学会論文誌，Vol. 48, No. 2, pp. 811–822 (2007).
[4] 肥後芳樹，楠本真二，井上克郎：コードクローン検出とその関連技術，電子情報通信学会論文誌，Vol. J91-D, No. 6, pp. 1465–1481 (2008).
[5] Kamiya, T., Kusumoto, S. and Inoue, K.: CCFinder: a multilinguistic token-based code clone detection system for large scale source code, *IEEE Trans. Sofw. Eng.*, Vol. 28, No. 7, pp. 654–670 (2002).
[6] Kapser, C. and Godfrey, M. W.: "Cloning considered harmful" considered harmful, *Proc. of WCRE 2006*, pp. 19–28 (2006).
[7] Li, Z., Lu, S., Myagmar, S. and Zhou, Y.: CP-Miner: Finding copy-paste and related bugs in large-scale software code, *IEEE Trans. Sofw. Eng.*, Vol. 32, No. 3, pp. 176–192 (2006).
[8] 門田暁人，佐藤慎一，神谷年洋，松本健一：コードクローンに基づくレガシーソフトウェアの品質の分析，情報処理学会論文誌，Vol. 44, No. 8, pp. 2178–2188 (2003).
[9] Pennington, N.: *Empirical studies of programmers: 2nd workshop*, pp. 100–113, Ablex Publishing Corp. (1987).
[10] Prechelt, L., Malpohl, G. and Philippsen, M.: Finding plagiarisms among a set of programs with JPlag, *Journal of Universal Computer Science*, Vol. 8, No. 11, pp. 1016–1038 (2002).
[11] 徳永健伸：情報検索と言語処理，東京大学出版会 (1999).
[12] 植田泰士，神谷年洋，楠本真二，井上克郎：開発保守支援を目指したコードクローン分析環境，電子情報通信学会論文誌，Vol. 86-D-I, No. 12, pp. 863–871 (2003).
[13] von Mayrhauser, A. and Vans, A.: Identification of dynamic comprehension processes during large scale maintenance, *IEEE Trans. Sofw. Eng.*, Vol. 22, No. 6, pp. 424–437 (1996).
[14] 山中裕樹，吉田則裕，崔　恩瀞，井上克郎：テキストマイニング技術を応用したメソッドクローン検出手法の提案，情報処理学会研究報告，Vol. 2013-SE-182, No. 28, pp. 1–8 (2013).