コードクローンとロジカルカップリングを用いた文字列検索ツールの出力順序の並べ替え

佐飛 祐介[1,a)]　村上 寛明[1,b)]　肥後 芳樹[1,c)]　楠本 真二[1,d)]

概要：ソフトウェアの開発において, 複数のコード片を同時に変更しなければならない状況がしばしば発生する．そのような同時に変更すべきコード片を漏れなく見つけるために文字列検索ツールがよく使用される．ある企業では独自に文字列検索ツールを開発しており，ソフトウェアの開発で使用している．しかし，そのツールの検索結果の表示順はファイルパスの辞書順であり，同時に変更が必要となる可能性が高い順ではない．開発者は同時に変更すべきコード片を漏れなく見つけるために検索結果を上位から 1 つずつ目視で確認している．そのため，検索結果にて確認すべき項目が多くなると集中力の低下によって見落としが発生する可能性が高くなることが課題となっている．本研究ではコードクローンとロジカルカップリングを用いて文字列検索ツールの出力順序を並べ替えることにより，見落としの発生頻度を少なくすることを試みた．企業内で開発されているソフトウェアに対する実験の結果，コードクローンおよびロジカルカップリングそれぞれにおいて検索によって 50 以上のコード片が見つかった場合は，それらの内の変更が必要であったものを上位に並べ替えられていたことを確認した．しかし見つかったコード片の数が 50 未満の場合は，変更が必要なものを下位に並べ替えてしまうこともあった．また，並べ替えを実装したツールについてシステムユーザビリティスケールに基づく評価を行った．評価の結果，ツールには導入が簡単という利点があることが分かった．しかし，ユーザビリティは一般的なツールの平均より少し低い結果となった．

1. はじめに

ソフトウェアの開発においては，コーディングやバグ修正といった，複数のコード片を変更する作業が多くを占める．コード片を変更する作業において，あるコード片を変更する際に，他の複数のコード片にも同時に変更しなければならない状況がしばしば発生する．そのためには，開発者は変更の必要がある全てのコード片を把握する必要がある．もしそのようなコード片を把握していなければ，見落としが発生する恐れがある．見落としが発生すると後のバグの原因となったり，そのバグ修正が必要となったりするなど，その後のソフトウェアの開発に悪影響を及ぼす．

このような見落としを防ぐために，grep 等の文字列検索ツールがしばしば用いられる [1]．あるコード片を変更する際にそのコード片に関連する文字列を文字列検索ツールに入力するとソースコードの中から入力された文字列と同じ文字列があるコード片が出力される．検索結果を 1 つずつ目視で確認することで，同時に変更の必要があるコード片を漏れなく見つけることができる．

ある企業では独自に文字列検索ツールを開発している．その文字列検索ツールには，一般的に用いられている文字列検索ツールには無い機能が実装されている．例えば，検索結果を Excel ファイルとして出力する機能がある．開発者はバグ修正や仕様変更に伴う複数のコード片の変更を行う際に，この文字列検索ツールを使用し，検索結果に対して 1 つずつ目視で同時に変更が必要であるか否かを判断している．しかし，検索結果が表示される順番はファイルパスの辞書順であり，同時に変更が必要となる可能性が高い順ではない．そのため確認すべき項目が膨大になった場合，検索結果の下部に変更が必要なコード片があると集中力の低下により見落としが発生することが課題となっている．

本研究ではコードクローンとロジカルカップリングを利用して，文字列検索ツールの出力順序を並べ替えることにより，見落としの発生頻度を少なくすることを試みた．[*1] 並べ替えではまず検索対象ソフトウェアのソースファイル群を解析しコードクローンの検出を行う．もしくは検索対

[1] 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
[a)] y-sabi@ist.osaka-u.ac.jp
[b)] h-murakm@osaka-u.ac.jp
[c)] higo@osaka-u.ac.jp
[d)] kusumoto@osaka-u.ac.jp

[*1] 著者らは ICPC2015 で採録された論文で，コードクローンとロジカルカップリングが企業のソフトウェアに対しても有効であることを示している [2]．本論文ではそれに加えて，ユーザビリティの評価を行い，その結果および考察を加えている．

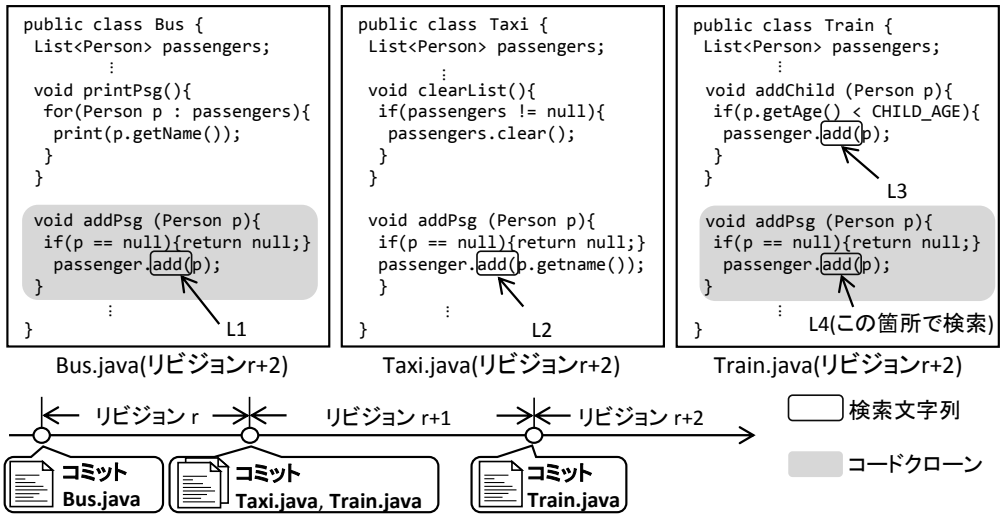


図 1　並べ替えを行うソースファイル群の例
Fig. 1　Example of Source Files for Reordering

象ソフトウェアのリポジトリを解析しロジカルカップリングの抽出を行う．次に，検索を実行した際にあらかじめ検出したコードクローンもしくは抽出したロジカルカップリングを用いて，ツールの出力順序を同時に変更が必要となる可能性が高い順に並べ替える．

並べ替えを用いるメリットとして以下のものがある．

- 多数の検索結果が出力された場合，集中力の低下による見落としの発生する頻度が少なくなる．
- コードクローン，ロジカルカップリングを検出，抽出するため，依存関係のあるコード片を把握しやすい．

また，並べ替えを用いるデメリットとしては前処理および並べ替えを行う必要があるため，並べ替えを実装しないツールと比較して実行速度が遅くなることが挙げられる．

2. 準備

本研究で用いる技術および用語について述べる．

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ているコード片である [3]．一部のコードクローンはソフトウェアの保守を困難にするといわれている [4]．開発を行っているソフトウェアの規模が膨大になると，コードクローンの数も膨大になる．そのため，開発者が全てのコードクローンを把握するのは困難である．そのような理由から，コードクローンを自動的に検出する研究が進められている [5]．

2.2 ロジカルカップリング

ソフトウェア開発において，プログラムの一部が変更された際に同時に変更しなければならない箇所がある．そのようなプログラム要素 (ファイル，メソッド等) 間には論理的結合関係が存在し，その関係をロジカルカップリングという [6]. ロジカルカップリングを有するプログラム要素は同一開発者によって極めて短時間，もしくは同時に変更されることが多い [7]．そのため，ソフトウェアの開発履歴から開発者および変更時刻を解析することによって，ロジカルカップリングを抽出することが可能となる．

2.3 文字列検索ツール

ソフトウェア開発において，あるコード片を変更する際に他の複数のコード片にも変更が必要か否かを検討する必要がある．同時に変更を加えるコード片を見つけるために文字列検索ツールがしばしば用いられる [1]．文字列検索ツールは入力として文字列を与えると，ソースコード中から同じ文字列を含むコード片を出力する．ソフトウェアの開発においてコード片を変更する際に，そのコード片に関連する文字列を文字列検索ツールに入力し，見つかったコード片を 1 つずつ目視で確認し，同時に変更が必要であるか否かを判断する．このように文字列検索ツールを使用することで，見落としの削減が期待できる．

3. 並べ替えの方法

本研究で用いた並べ替えの方法について説明する．

3.1 概要

本研究では文字列検索ツールの出力結果をコードクローンとロジカルカップリングそれぞれを用いて並べ替える．並べ替えの基準にはコードクローンを用いたものとロジカルカップリングを用いたものの 2 種類あり，それぞれについて以下で説明する．

3.2 コードクローンを用いた並べ替え

入力は以下の通りである．

- コードクローンとみなす最小一致トークン数
- 検索対象ソフトウェアのソースファイル群
- 検索を行った文字列を含むファイルのパス
- 文字列検索を行った結果

出力は並べ替えが行われた後の検索結果である．並べ替えは以下に示す 2 つの Step で行われる．

Step-A1: コードクローンの検出

Step-A2: 検索結果の並べ替え

実行速度を速くするため，Step-A1 のコードクローンの検出は検索を行う前に 1 回行う．それ以降の検索では初回に検出したコードクローンを用いて並べ替えを行う．

並べ替えの順はそのコード片が含まれるクローンセットの数の降順である．また，クローンセットの数が同数の場合はファイルパスの辞書順に並べ替える．

図 1 を用いてコードクローンを用いた並べ替えの例を説明する．図 1 は 3 つのファイルに対して *Train.java* 内の “add(” という文字列で検索を実行した例である．検索を実行した結果，この例では 4 箇所のコード片が出力される．それぞれを L1, L2, L3, L4 と名付ける．一般的な文字列検索ツールは結果の出力順序が文字列を含むファイルのパスの辞書順となっているため，L1 → L2 → L3 → L4 という順番に出力される．図 1 において，ハイライトしているコード片は互いにコードクローンである．そのためコードクローンに含まれる L1, L4 は上位に並べ替えられ, 並べ替え後は L1 → L4 → L2 → L3 という順番に出力される．

3.3 ロジカルカップリングを用いた並べ替え

入力は以下の通りである．

- ロジカルカップリングとみなす変更時刻の差の閾値
- 検索対象ソフトウェアのリポジトリ
- 検索を行った文字列を含むファイルのパス
- 文字列検索を行った結果

出力は並べ替えが行われた後の検索結果である．並べ替えは以下の 2 つの Step で行われる．

Step-B1: ファイル単位のロジカルカップリングの抽出

Step-B2: 検索結果の並べ替え

まず，Step-B1 で検索対象のソフトウェアのリポジトリを解析し，ファイル単位のロジカルカップリングを抽出する．

本研究でのロジカルカップリングの定義は以下の 2 つの条件を満たすファイルのペアである．

(1) 2 つのファイルにおける変更時刻の差が，入力された閾値以下である．

(2) 2 つのファイルが同じ開発者によって変更されている．

本研究ではロジカルカップリングの抽出を高速に行うため，ロジカルカップリングの単位をファイルとした．また，実行速度を速くするため Step-B1 のロジカルカップリングの抽出は検索を行う前に 1 回行う．それ以降の検索では初回に抽出したロジカルカップリングを用いて並べ替える．

検索文字列：“add(” 検索日時：14/04/01 12:00

順位	ファイルパス	行数	変更の有無
1.	/home/A.java	15行目	無
2.	/home/B.java	86行目	有
3.	/home/C.java	46行目	無
4.	/home/D.java	13行目	無
5.	/home/E.java	36行目	有

図 2 Excel ファイルの例
Fig. 2 Example of Excel File

Step-B2 では検索を行った文字列を含むファイルと同時変更された回数の降順になるように並べ替える．ただし，検索を行った文字列を含むファイル内に見つかったコード片は最も上位に並べ替えられる．同時変更された回数が同じ場合はファイルパスの辞書順に並べ替えられる．

図 1 の例において，ソフトウェアのリポジトリを解析すると *Train.java* と *Taxi.java* が同時に変更されているため *Train.java* と *Taxi.java* はロジカルカップリングの関係にある．そのため，*Taxi.java* 内に見つかった L2 は上位に並べ替えられる．また，検索を行った *Train.java* 内に見つかった L3 と L4 は最も上位に並べ替えられる．従って，並べ替え後は L3 → L4 → L2 → L1 という順番に出力される．

4. 並べ替えの評価実験

本研究にて行った並べ替えの評価実験について説明する．

4.1 目的

この実験の目的は，コードクローンやロジカルカップリングを用いた複数のコード片の同時変更支援が企業内のソフトウェアに対して有効であるかを確かめることである．そのため企業で使用されているツールに並べ替えの機能を実装し，実験を行った．実験対象は企業で開発されている 3 つのソフトウェアである．表 1 に詳細を示す．

4.2 方法

この実験を行うにあたり，以下のデータを使用した．

- その企業で開発されている 3 つのソフトウェアのリポジトリ. 表 1 に詳細を示す．
- ソフトウェア α〜γ に対して行われた過去の検索において出力された Excel ファイル．

表 1 において過去の検索件数を表す対象検索数はコードクローンとロジカルカップリングの間で異なっている．その理由は, ロジカルカップリングの抽出は全てのファイルを対象としているのに対し，コードクローンの検出は Java ファイルのみを対象にしているからである．

Excel ファイルの例を図 2 に示す．Excel ファイルには検

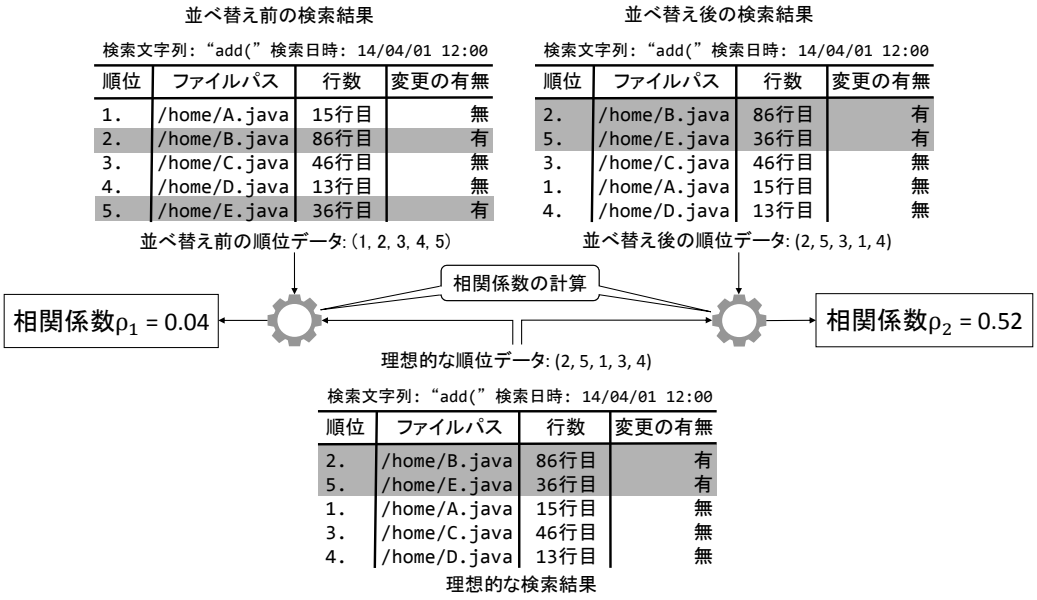


図 3 相関係数の計算例

Fig. 3 Calculation Example of Correlation Coefficients

索文字列，日時そして検索結果として見つかったコード片それぞれについての順位，コード片が見つかったファイルのパス，目視による判断での変更の有無が記載されている．Excel ファイルは 1 回検索が行われるたびに生成されるため，検索を行った回数だけ Excel ファイルが存在する．また，実験には Excel ファイルから取得した順位データを利用する．順位データとは，Excel ファイルに含まれるコード片の順位を上位から順に並べた整数の配列である．図 2 の例では (1, 2, 3, 4, 5) が順位データとなる．並べ替え前の順位データおよび並べ替え後の順位データを比較することにより，並べ替えが有効であるかを確認する．

著者らはこれらのデータを用いて以下の Step で実験を行った．

Step-1: 企業から提供して頂いた Excel ファイルから並べ替え前の順位データを取得する．

Step-2: 検索が行われた当時のスナップショットをリポジトリから取得する．

Step-3: 取得したスナップショットを検索対象として，並べ替え後の検索結果とその順位データを取得する．

Step-4: Excel ファイルにおいて変更が行われたファイルを全て上位に並べ替えた理想的な順位データを取得する．理想的な順位データについては後述する．

Step-5: 相関係数を用いて Step-1, 3 で得られた順位データと Step-4 で取得した理想的な順位データをそれぞれ比較する．

上記の Step を全ての Excel ファイルに対して適用する．

表 1 実験対象ソフトウェア

Table 1 Target Software Systems

名前	コミット数	開発者数	対象検索数	
			コードクローン	ロジカルカップリング
α	約 26,000 回	45 人	11 回	18 回
β	約 700 回	13 人	11 回	28 回
γ	約 260 回	9 人	6 回	11 回

4.3 理想的な順位データ

図 3 において，並べ替え前の検索結果から理想的な順位データを取得する例を説明する．図 3 では並べ替え前の検索結果から，変更が有となっている箇所を全て上位になるよう並べ替えている．図 3 の例では 2, 5 位を全て上位に並べ替え，その結果から理想的な順位データを取得する．図 3 の例では理想的な順位データは (2, 5, 1, 3, 4) となる．

4.4 相関係数

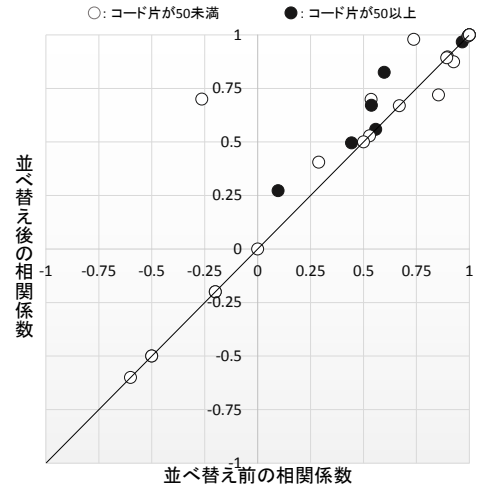
この実験ではスピアマンの順位相関係数を用いて順位データ間の相関を求める．

並べ替え前の順位データと理想的な順位データ，および並べ替え後の順位データと理想的な順位データについて相関係数を求める．そのため，相関係数が 1 に近いほど検索結果が理想的な検索結果に近いことになる．以下，並べ替え前の順位データと理想的な順位データとの相関係数を**並べ替え前の相関係数**，並べ替え後の順位データと理想的な順位データとの相関係数を**並べ替え後の相関係数**と呼ぶ．

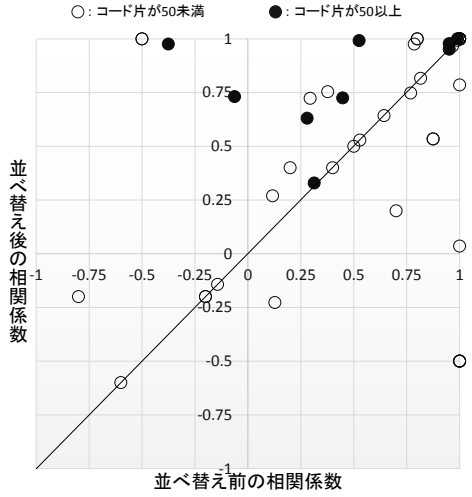
4.5 結果

4.2 節の実験方法を 3 つのソフトウェアに適用した結果を図 4 に載せる．

横軸に並べ替え前の相関係数，縦軸に並べ替え後の相関



(a) コードクローンを用いた並べ替え



(b) ロジカルカップリングを用いた並べ替え

図 4 相関係数の変化

Fig. 4 Change of Correlation Coefficients

係数をとっている．また，塗りつぶされているマーカーは見つかったコード片が 50 以上であったことを示す．図 4 に表示している直線よりも上部に表示されているマーカーは相関係数が増加し，下部に表示されているマーカーは相関係数が減少している．

並べ替えにより相関係数が増加した件数，減少した件数，変化しなかった件数を表 2 に示す．括弧内のものは見つかったコード片が 50 以上の場合を示す．表 2(a) からコードクローンを用いた並べ替えでは，増加した件数が減少している件数よりも多い．また，表 2(b) からロジカルカップリングを用いた並べ替えでは，増加した件数は減少している件数よりも多い．以上から, 並べ替えにより相関係数が減少した件数よりも増加した件数の方が多いことがわかる．

4.6 考察

表 2(a)(b) よりコードクローンを用いた並べ替えにおいてもロジカルカップリングを用いた並べ替えにおいても見つかったコード片が 50 以上の場合，並べ替えによって相関係数が減少する事例が無いことがわかる．このことより，多数の検索結果が出力された場合，集中力の低下による見落としの発生する頻度が少なくなることがわかる．

最後に，本実験は全て企業内で行った．限られた時間内で実験を行わなければならず，また，実験対象ソフトウェアを持ち帰る事もできなかったため，個々の結果に対する詳細な分析は行うことができなかった．

5. ユーザビリティの評価実験

並べ替える機能を実装した文字列検索ツールを実際に企業の開発者に利用して頂いた．2 ヶ月ほど企業の開発者に利用して頂いた後，システムユーザビリティスケール [8] に基づいてユーザビリティの評価を行った．ここで，企業の開発者に利用して頂いた文字列検索ツールはロジカルカップリングおよびコードクローンの両方を用いた並べ替えを実装している．両方を用いた並べ替えでは，まずロジカルカップリングを用いて並べ替えを行い，同時変更された回数が同じものに対してコードクローンを用いて並べ替えを行う．同時変更された回数およびクローンセットの数が同じものについてはファイルパスの辞書順に並べ替える．

5.1 方法

評価の手順はまずツールの利用者に表 3 に示す 10 個の項目に対して 1(全くそう思わない)〜5(強くそう思う) の 5 段階評価のアンケートを行ってもらう．次にアンケートの結果に対してユーザビリティの評価値を算出する．10 項目の内，奇数番の項目は回答から 1 を引いた値，偶数番の項目は 5 から回答を引いた値として 0〜4 の値に変換し，これらの値の合計に 2.5 を掛けた値を評価値として利用する．評価値は 0 から 100 の値をとり，100 に近いほどそのツールのユーザビリティが高いことを示している．

5.2 結果

15 名の開発者に対してユーザビリティの評価のアンケー

表 2 各ソフトウェアにおける相関係数の変化

Table 2 Change of Correlation Coefficients on Each Software

(a) コードクローン

	増加	変化なし	減少	検索回数
ソフトウェア α	4 件 (1 件)	8 件 (3 件)	0 件 (0 件)	11 件 (4 件)
ソフトウェア β	0 件 (0 件)	4 件 (0 件)	2 件 (0 件)	6 件 (0 件)
ソフトウェア γ	4 件 (3 件)	7 件 (0 件)	0 件 (0 件)	11 件 (3 件)
合計	8 件 (4 件)	19 件 (3 件)	2 件 (0 件)	28 件 (7 件)

(b) ロジカルカップリング

	増加	変化なし	減少	検索回数
ソフトウェア α	8 件 (3 件)	6 件 (1 件)	4 件 (0 件)	18 件 (4 件)
ソフトウェア β	4 件 (1 件)	2 件 (0 件)	5 件 (0 件)	11 件 (1 件)
ソフトウェア γ	7 件 (3 件)	19 件 (1 件)	2 件 (0 件)	28 件 (4 件)
合計	19 件 (7 件)	27 件 (2 件)	11 件 (0 件)	57 件 (9 件)

トを行った．表 4 にアンケートの結果に数値変換処理を施した値を回答者ごとに示す．

表 4 より平均である 61.7 という数値が並べ替えを実装したツールのユーザビリティとなる．

5.3 考察

文献 [8] では 2,324 件の調査において評価値の平均は 69.5 であると報告されている．一方，本実験で得られた評価値の平均は 61.7 である．文献 [8] では母平均の推定を行うことができなかったため，有意差があるかは判断できないが，得られた評価値から並べ替えを実装したツールは一般的なツールに比べてユーザビリティが低いということがわかる．その理由としては，検出，抽出したデータを用いて並べ替えを行うため，並べ替えを行わずツール単独で使用した場合と比較して実行速度が遅くなってしまうということが挙げられる．実際，アンケートに設けた自由記述の項目でも，15 名中 7 名の使用者から実行速度が遅くなっているという意見を頂いているため，実行速度が遅くなっていることがユーザビリティの低下と関連があると考えられる．しかし，項目 2 や項目 7 に着目すると平均が 2.7 や 2.8 とやや高く，容易に利用でき，導入も簡単であるということが並べ替えを実装したツールの長所として挙げられる．

表 3 システムユーザビリティスケールのアンケート項目
Table 3 Questionnaires of System Usability Scale

項目	アンケート内容
1	このツールをしばしば使いたいと思う
2	このツールは不必要なほど複雑と感じた
3	このツールは容易に使えると思った
4	このツールを使うのに専門家のサポートが必要と感じた
5	このツールにある様々な機能が良くまとまっていると感じた
6	このツールでは，一貫性の無いところが多くあったと感じた
7	大抵のユーザは，このツールの使用方法について，素早く学べるだろう
8	このツールはとても扱いにくいと思った
9	このツールを使うのに自信があると思った
10	このツールを使い始める前に多くのことを学ぶ必要があった

表 4 システムユーザビリティスケールの結果
Table 4 Results of System Usability Scale

回答者	項目 1	項目 2	項目 3	項目 4	項目 5	項目 6	項目 7	項目 8	項目 9	項目 10	評価値
A	1	3	3	3	1	3	3	1	1	1	50
B	1	3	1	1	2	2	3	1	2	3	47.5
C	3	4	2	3	3	2	3	2	2	3	67.5
D	1	4	2	4	1	1	3	1	1	3	52.5
E	0	4	3	3	3	3	3	3	3	2	67.5
F	2	3	2	3	2	3	3	1	2	3	60
G	3	4	4	4	3	4	4	4	3	4	92.5
H	3	2	2	3	2	2	3	3	3	2	60
I	2	3	2	2	2	3	3	2	2	3	60
J	3	3	4	3	3	3	3	3	3	3	77.5
K	1	1	2	1	2	2	1	2	1	1	35
L	1	2	3	3	2	3	1	2	1	2	50
M	3	4	3	3	4	3	4	3	3	2	80
N	3	4	3	3	2	3	4	3	3	3	77.5
O	2	2	1	3	2	2	3	2	2	0	47.5
合計	28	42	34	39	33	36	41	32	30	34	925
平均	1.9	2.8	2.3	2.6	2.2	2.4	2.7	2.1	2	2.3	61.7

6. おわりに

本研究では，文字列検索ツールにおいて変更の必要があるコード片を含むファイルを上位に並べ替えることにより，見落としの発生頻度を少なくすることを試みた．並べ替えを行うにあたって，まず検索対象ソフトウェアのソースファイル群からコードクローンの検出を行う．もしくは検索対象ソフトウェアのリポジトリよりロジカルカップリングの抽出を行う．次に，検出されたコードクローンもしくは抽出されたロジカルカップリングを用いて文字列検索ツールにおける検索結果の並べ替えを行う．

また，並べ替えの有効性を確かめるためにある企業で開発されている 3 つのソフトウェアに対して実験を行った．実験の結果，並べ替えにより変更が必要であるコード片を含むファイルを上位に並べ替えられていることを確認した．また，検索により見つかったコード片が 50 以上の場合は並べ替えがより有効に働くことを確認した．

今後の課題として，ツールの実行速度およびユーザビリティの改善が挙げられる．また，検出するコードクローンやロジカルカップリングの粒度を変更することにより，並べ替えの精度が変わるのかを調査したいと考えている．

謝辞 本研究を遂行するにあたって，様々なご協力を頂いた松岡 宏和さん，豊川 宏美さんに対して，この場を借りて感謝申し上げたい．本研究は，日本学術振興会科学研究費補助金基盤研究 (S)(課題番号：25220003) および文部科学省科学研究費補助金若手研究 (A)(課題番号：24680002) の助成を得た．

参考文献

[1] LaToza et al. Developers Ask Reachability Questions. In *Proc. of the 32nd ICSE*, pp. 185–194, 2010.
[2] Y. Sabi et al. Reordering Results of Keyword-based Code Search for Supporing Simultaneous Code Changes. In *Proc. of the 23rd ICPC*, pp. 289–290, 2015.
[3] 肥後芳樹ら. 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. D, Vol. 91, No. 6, pp. 1465–1481, 2008.
[4] X. Wang et al. Can I Clone This Piece of Code Here? In *Proc. of the 27th ASE*, pp. 170–179, 2012.
[5] 石原知也ら. 大規模なソフトウェア群を対象とするメソッド単位でのコードクローン検出. 情報処理学会論文誌, Vol. 54, No. 2, pp. 835–844, 2013.
[6] H. Gall et al. Detection of Logical Coupling Based on Product Release History. In *Proc. of the 1st ICSM*, pp. 190–200, 1998.
[7] 松村知子ら. ファイルの同時変更パターンと変更差分の分析による論理的結合関係の自動抽出. ソフトウェアシンポジウム 2005, pp. 104–112, 2005.
[8] A. Bangor et al. An Empirical Evaluation of the System Usability Scale. *International Journal of Human-Computer Interaction*, Vol. 24, No. 6, pp. 574–594, 2008.