修士学位論文

題目

コメント中のライセンス記述を用いた
大規模ソースファイル集合の分類手法

指導教員
井上 克郎 教授

報告者
真鍋 雄貴

平成 20 年 2 月 8 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻 ソフトウェア工学講座

平成 19 年度 修士学位論文

コメント中のライセンス記述を用いた大規模ソースファイル集合の分類手法

真鍋 雄貴

内容梗概

ソースファイルをはじめとするソフトウェア部品の再利用はソフトウェア開発を効率化すると言われている．近年のオープンソースソフトウェア開発の活発化により，開発者は，大規模なソースファイル集合から開発中のソフトウェアに必要なソースファイルを取得することができる．しかし，ソースファイルを再利用するためには，その著作権者により定められたライセンス（利用条件）に従う必要がある．ライセンスの中には，ソースファイルの公開を義務付けるなど，組み込む先のソフトウェアに影響を与えるものも存在する．そのため，ソースファイルをソフトウェアに組み込む前に，開発者はそのライセンスを確認しなければならない．もし問題のあるライセンスならば，開発者は別のソースファイルを取得する必要がある．この手間は，開発者が利用できないソースファイルを取得しないよう，再利用の候補すべてのライセンスをあらかじめ特定しておくことで削減できる．しかし，大規模なソースファイル集合に対するライセンスの特定を手作業でおこなうことは現実的では無い．

そこで，開発者がソースファイルを再利用するときにライセンスを確認する労力を減らすために，大規模なソースファイル集合をライセンスに基づき分類する手法を提案する．ライセンスをラベルとしたカテゴリにソースファイルを分類することで，開発者が利用できない部品を取得することを防ぐ．本手法は，複数のソースファイルのコメントに共通する文字列（共通文字列）を抽出し，その文字列に基づきカテゴリの生成およびソースファイルの分類を行う．ライセンスの指定にはコメントが用いられることが多いため，共通文字列を抽出することにより効率的な分類が可能であると考えられる．また，本手法を評価するために複数のソフトウェア集合を対象とした実験を行なった．実験では，分類手法が完了するまでに要した時間や手法の手順の進行に伴う分類率の変遷などにより，本手法の有効性を示した．

主な用語

再利用 (Reuse)
ソフトウェアライセンス (Software License)

目 次

1 はじめに

ソフトウェア開発コストの削減に有効な方法の１つとして，ソフトウェアの再利用が挙げられる．ソフトウェアの再利用とは既存のソフトウェア部品を同一システム内や他のシステムで利用することである [14]．再利用の対象となるソフトウェア資産として，本研究ではソースファイルに注目する．開発者が再利用を行うためには，まず必要なソースファイルを膨大な量のソフトウェアに含まれるソースファイルから選択して取得する必要がある．開発者が利用可能なソフトウェアとしては，過去に開発者が自ら開発したソフトウェアに加え，SourceForge.net [12] 等で開発されるオープンソースソフトウェアがある．

このような膨大な量のソフトウェアからのソースファイルの取得を支援するシステムとして，ソースファイル検索システムがある [7] [9] [32]．ソースファイル検索システムにはオープンソースソフトウェアなどから得られたソースファイルが蓄積されており，開発者は検索クエリを入力することにより必要なソースファイルを取得することができる．

しかし，取得したソースファイルを利用するためには各ソースファイルの使用条件（ライセンス）を特定し，開発中のソフトウェアの使用条件と矛盾しないかどうか確認しなければならない．また，確認した結果，矛盾が生じる可能性のあるソースファイルは再利用する事はできず，開発者は再び別のソースファイルを取得し直さなければならない．それらにかかる労力は再利用の妨げとなっている．開発者が取得したソースファイルのライセンスを確認するためには，ソースファイルやソースファイルに関連する文書を読む必要がある．また，近年のオープンソースソフトウェア開発の普及にともない，多様なライセンスが出現しており [10]，開発者がライセンスを特定する作業を困難にしている．

このような開発者の労力は，ソースファイルを取得するのと同時にライセンスを知ることが出来れば不要である．そのためには，ソースファイル検索システムに蓄積されたソースファイルがライセンスごとに分類されていればよい．しかし，ソースファイルの数は膨大であり，またライセンスも多様であるため，ソースファイルの分類を全て人手で行うのは現実的では無い．

そこで，本稿では，大規模なソースファイル集合に対して適用可能なライセンス記述に基づくソースファイル分類手法を提案する．私は，ソースファイルに対するライセンスの指定（ライセンス記述）がそのソースファイルのコメントに記述されることが多く，また，その記述には共通する文字列表現が用いられる場合が多いことに着目した．そこで，提案手法は，コメント中での出現頻度の高い文字列（共通文字列）を抽出し，抽出された文字列に対してライセンスを結び付けることにより，多数のソースファイルをライセンスごとに効率的に分類できる．

提案手法の有効性を検証するために適用実験を行い，ライセンス特定の効率，時間を尺度

とした利用者の労力，共通文字列の個数を尺度とした利用者の労力の 3 つの観点で評価した．まず，利用者が読んだ共通文字列の数と，ライセンスが特定されたソースファイルの数を計測した結果，少数の共通文字列を読むだけで大部分のソースファイルを分類できることを確認した．続いて，手順が終了するまでに要する時間を評価した結果，提案手法が実用的な時間で終了することが示された．最後に，手順が終了するまでに利用者が読む必要のある共通文字列数より，利用者の労力を評価した結果，共通文字列のフィルタリングなどによる改善の余地があると分かった．

以降，2 節では本稿で提案する手法の背景として，ソフトウェア部品の再利用，ソフトウェア部品検索システム，ソフトウェアライセンスについて説明する． 3 節では 提案するライセンス記述に基づく大規模ソースファイル集合分類手法について述べる．4 節では提案手法の実装について述べる． 5 節では提案手法の評価のため，実際のオープンソースソフトウェアを実験対象とした評価実験の内容と結果を説明し，実験結果についての考察を述べる．6 節では関連研究として既存のライセンス特定手法，ソースファイルをライセンスごとに分類する手法について説明する．7 節ではまとめと今後の課題を述べる．

2 背景

2.1 ソフトウェア部品の再利用

ソフトウェアの再利用とは既存のソフトウェア部品を同一システム内や他のシステムで利用することである [15]．ソフトウェア部品とは，クラスや関数，ソースコードやドキュメントなどのソフトウェアの開発過程で生成される成果物のことを指す．以降，ソフトウェア部品を単に部品と呼ぶ．部品の再利用により，ソフトウェア開発にかかるコストを削減することが出来ると言われている．

一般に，再利用のプロセスは以下の 3 つの段階により構成される [22].

部品の取得: 再利用可能な部品を検索などにより取得する

部品の評価: 得られた部品が，開発者の要求を満たすかどうか評価する

部品の適応: ソフトウェアに組み込む際に，要求に対して部品を適応させる．

部品の取得 部品の再利用プロセスにおいて，大量の部品の中から効率よく目的に応じた部品を発見するために，ソフトウェア部品検索システムと呼ばれるシステムが用いられる [18]. SPARS-J [19,32] は Java クラスを部品とし，ソースコードの全文検索を行うシステムである．SPARS-J は，依存や類似といった部品間の関連を解析することにより，重要な部品を優先的に取得することが出来る Component Rank と呼ばれる手法を実装している． Sourcerer [13] は，SPARS-J と同様に Java クラスを部品としてソースコードの全文検索を行うシステムであるが，Fingerprint と呼ばれる部品の特徴による検索が可能である．Google Code Search [7] は複数のプログラミング言語に対応した検索システムであり，大量のソフトウェア部品に対し，正規表現などによる柔軟な検索を行える．これらのシステムはいずれも検索対象の部品としてオープンソースソフトウェアを収集しており，WWW を通じて検索システムを利用することが出来る．また，大須賀ら [31] は，インターネットを通じて利用可能なソフトウェア部品検索システムから，横断的に検索する手法を提案している．

以上のシステムはキーワードをクエリとする全文検索である．これに対し，SPARTACAS [23] は必要な部品の仕様の形式的な記述をクエリとして部品を取得する．SPARTACAS では，クエリとして指定した仕様に完全に一致する部品に加え，部分的に一致する部品の組み合わせを検索する．Park の手法 [25] では，入出力の振る舞いを入力として部品を検索する．また，CodeBroker [30] は開発環境に統合されたシステムであり，開発者の編集しているソースコードからドキュメントコメントやメソッドに利用される文字列を取得することで，類似した部品を自動的に検索する．

ソフトウェア部品の再利用は，従来，開発現場それぞれにおいてソフトウェアそのものと並行して再利用可能な部品を計画的に開発することで実践されてきた [20]．ドメイン分析により，それぞれの開発現場に適した部品が開発され，再利用される．近年注目されている Software Factories と呼ばれる手法は，この概念を拡張したものであると言える [16]．しかし，再利用を実践する為に開発プロセスを変更する必要があり，再利用を実践する為のコストが高いことが問題点とされてきた．

このような計画的な再利用に対し，開発者が必要に応じて部品を検索して取得する，軽量な再利用が注目されている [17]．再利用対象となる部品は，開発現場に蓄積されたソフトウェア資産に加え，インターネットを通じて得ることが出来る．近年のオープンソースソフトウェア開発の活発化により，誰でも高品質な部品をインターネットを通じて得られるようになったことから，この方式の再利用が活発化してきている．例えば，SourceForge.net [12] には，約 170,000 のプロジェクトが登録されており，FreeBSD [5] のソフトウェアパッケージ管理システムである Ports には約 20,000 のパッケージが存在する[1]． 開発者は，オープンソースソフトウェアの部品をソフトウェア部品検索システムもしくは WWW の検索システムを用いて検索し，取得することが出来る．

部品の評価　上記で述べたとおり，部品は様々な手法により取得することが出来るが，部品が開発者の要求を満たさない場合や，部品が何らかの制約条件をもつ場合には，必ずしも利用できるとは限らない．部品が利用できない場合には，別の部品を改めて取得する必要がある．例えば，キーワード検索を用いるシステムでは，クエリに用いるキーワードを適切に選択することが難しく，要求を満たす部品を取得するために繰り返し検索が必要であると言われている [30]．また，部品の品質が要求を満たさない場合が挙げられる．再利用対象の部品は，特別な契約が取り交わされている場合を除き，再利用する時の動作を保証していない場合が多い [15]．また，ライセンスにより部品の利用に制約が設けられている場合が挙げられる．部品は知的財産物として保護されており，著作権者の許諾無しに利用することは出来ない．以下，ライセンスに関して説明する．

ソフトウェアライセンス　ソフトウェアは一般に著作権により保護されていることが多く，ソフトウェアを利用する際には，著作権者とライセンス契約 (license agreement) を結ぶ必要がある [15, 24]．ソフトウェアを利用する場合と同様に，部品を再利用する場合に関しても，著作権者とライセンス契約を結び対象となるソフトウェアの利用に関する権利と義務について明らかにする必要がある．本稿では，このような明文化された権利および義務を**ソフトウェアライセンス（ライセンス）**と呼ぶ． ソフトウェアライセンスには，プログラム

[1] 2007 年 2 月現在

（ソースコード）の複製や改変，再配布等，利用上の制限が含まれる．

ソースコードの開示や再配布に関し，Open Source Initiative [10] により定められた一定の基準を満たすライセンスにより配布されているソフトウェアはオープンソースソフトウェアと呼ばれる．OSI により認定されたライセンスは 68 種類[2]存在する．代表的なものとして以下のようなライセンスが挙げられる．

GPL (GNU Public License): Linux カーネルなどに用いられるライセンスであり，ソフトウェアを配布する際にソースコードを開示することを義務づけている．また，GPL ライセンスの部品を組み込んだソフトウェアは，GPL ライセンスで再配布される必要がある

LGPL (GNU Lesser Public License): 基本的に GPL と同じであるが，部品そのものに変更を加えない場合には，部品を組み込んだソフトウェアのライセンスに制限が存在しない．

ASL (Apache Software License): 主に Apache Software Foundation [1] で用いられている，Apache HTTP Server などに用いられているライセンスである．GPL と異なり，部品を再利用した場合にも，ソースコードの開示は義務づけられない．

オープンソースソフトウェアの部品に関するライセンス契約は，著作権者と再利用者が直接交渉する代わりに，部品を利用した時点で契約に同意したと見なすことが多い．部品に対するライセンスの指定方法としては，主に以下の 3 通りの方法が用いられることが多い．

1. 部品のソースコード中に，コメントなどで直接ライセンスが記述される．
2. 部品のソースコード中に，コメントなどでライセンスの記述されたファイルへのポインタ（ファイルパスや URL など）が記述される．例を図 2.1 に示す．著作権表示およびライセンスの名前に加え，ライセンスの本文が存在するインターネット上の URL が示されている．
3. 部品と同じ配布パッケージ中に含む文書中にライセンスが記述される．README ファイルと呼ばれる，利用者が最初に読むべき文書に記述されている場合もあれば，COPYING，LICENSE など，ライセンスが記述されていることが分かるような名前を持つファイルに記述されている場合もある．

プロジェクトによっては，開発者の規約により，上記の 1. もしくは 2. を強制するものも多い．例えば，Apache Software Foundation では，開発者に対してソースコードのヘッダ部分にライセンスを指定する記述を含むコメントを記述することを求めている [2]．この理由と

[2]2007 年 2 月現在

しては，オープンソースソフトウェアの部品は単体で再利用されることが多く，ソースコード中にライセンスに関する記述が無ければその特定が困難であることが考えられる．

部品の適応　取得された部品が開発者の要求に完全に合致しない場合，部品はソフトウェアに組み込む前に目的に適応させられる．適応は，部品の再利用のされ方で異なる．ここで，部品の再利用は以下の 3 種類に分類することができる [28].

ブラックボックス再利用　開発者は，部品の実装や内部仕様を見ず，インターフェースや仕様書のみを見て部品を再利用する．部品に対する変更は行わない．例えば，COTS(Commertial off-the-shelf) 部品の再利用が挙げられる．

グラスボックス再利用　開発者は，ブラックボックス再利用と同様，部品を変更せずに再利用する．しかし，インターフェースに加え，部品の実装や内部仕様を知ることができ，部品に対してより深く理解することが出来る．例えば，Java SE [8] など，実装の公開された標準 API の利用が挙げられる．

ホワイトボックス再利用　開発者は，部品の実装を知ることができ，必要に応じて実装やインターフェースを変更して再利用する．例えば，ソースコードの全文検索を行う部品検索システムを用いた軽量な再利用が分類される．

ブラックボックス再利用およびグラスボックス再利用では，再利用する部品を組み合わせる為の部品を新たに作成することで，部品を適応させる．ホワイトボックス再利用では，上記の方法に加え，部品そのものを変更することで適応させられる．部品を適応させることにより，より多くの部品が再利用可能となる．しかし，Selby [27] の調査によると，部品に対し大規模な修正を加えて再利用を行う場合，新規に作成する場合と比べ，欠陥を修正，もしくは分離するのにかかる労力が大きい．そのため，要求との差異が小さい部品を取得する必要がある．

2.2　再利用における問題

2.1 節で述べたように，ソフトウェア部品検索システムを用いることで，開発者は膨大な量の部品を取得することが出来る．しかし，開発者の要求を満たす部品は取得した部品の一部であるため，開発者は取得した部品一つ一つを評価し，利用可能な部品を選び出すために労力を払わなければならない．

部品に対する要求の例としては，部品の持つ機能に加えて，部品の性能や信頼性などの品質，部品のライセンスなどがあり，再利用される部品はこれらの要求を全て満たしていなければならない．もし，機能や品質についての要求が満たされない場合には，部品を組み込む

Copyright [yyyy] [name of copyright owner]
Licensed under the Apache License, Version 2.0 (the ”License”);
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an ”AS IS” BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

図 1: Apache License 2.0 のライセンス記述

ことが出来ないか，組み込むことが出来ても部品が組み込まれたソフトウェアの動作に支障をきたすといった問題が起きる．一方，ライセンスについての要求が満たされない場合であっても，部品をソフトウェアに組み込むことは可能であり，ソフトウェアの動作に悪影響を及ぼすことはない．

しかし，ライセンスについての要求を満たさない部品を組み込んでしまった場合，法的な問題を引き起こすことがある．例えば，ソースコードを公開していないソフトウェアに GPL ライセンスで配布されている部品を意図せずに組み込んでしまっていた場合，そのソフトウェア全体のソースコードを公開し， GPL ライセンスで配布しなければならなくなる可能性がある．また，前述のように，ライセンスに対する要求を満たしていない場合であってもソフトウェアの動作には支障がないため，問題に気付くのが難しいという特徴がある．

このため，開発者は，再利用の際には必ず部品のライセンスを調査し，それに違反しないよう注意しなければならない．しかし，2.1 節で述べたようにライセンスの種類は多く，また，同じライセンスであってもその指定方法は一定ではないため，部品のライセンスを調査することは簡単ではない．

以上のことから，ソフトウェア部品検索システムで取得される部品には，その部品がどのライセンスに従って配布されているかという情報が付属していることが望ましいと考えられ

る．しかし，上述のようにライセンス特定の作業は複雑であるため，特定作業の自動化は困難である．また，ソフトウェア部品検索システムには膨大な量の部品が登録されているため，ソフトウェア部品検索システムに登録された部品一つ一つのライセンスを人手で調べることは非現実的である．

3　ソースファイル分類手法

本節では，2.2 節で述べた再利用の際に起こる問題を解決することを目的として，ソフトウェア部品検索システムに登録された部品が何のライセンスによって配布されているかを特定する手法を提案する．手法はライセンスが未知である部品の集合を入力とし，それぞれの部品が何のライセンスによって配布されているかを調べ，部品のライセンス毎に部品集合を分割して出力する．以下，分割された集合のそれぞれをカテゴリと呼ぶ．カテゴリにはラベルとしてライセンス名が付けられている．ライセンスが判明しなかった部品は「ライセンス不明」というラベルの付けられた特別なカテゴリに所属するものとする．

しかし，この問題を計算機により自動的に実行することは 2.2 節で述べたように困難である．そこで，提案手法は，人間と機械の協調により部品のライセンスを効率的に特定することを目指す．

3.1　ソースファイルの分類

本研究では，ソースファイル中にコメントとして記述する方法において，共通した記述が用いられることが多い点に着目する．私はこの着目点に基づき，複数のソースファイルのコメントに共通して含まれる文字列を抽出することにより，ライセンスを指定する記述が抽出されると考えた．ここで，ライセンス記述とは，ソースファイルの利用条件を示す文字列，もしくは，それを記述しているファイルがある場所を示す文字列とする．

本稿では，複数のソースファイルのコメントに共通して含まれる文字列を共通文字列と呼ぶ．ソースファイルからどのような共通文字列が抽出されるかを表す例を図 2 に示す．図 2 では，3 つのファイルからそれぞれコメントの部分が抽出され，そして，3 つのコメントに共通して現れている「All rights reserved. aaa」が共通文字列として抽出されている．この例の場合，利用者が共通文字列「All rights reserved. aaa」を読み，この共通文字列が示すライセンスを特定することが出来れば，3 つのソースファイルに対して，それらソースファイルのライセンスを特定することができる．ここで，共通文字列はライセンス記述を含まない場合がある．抽出された共通文字列は以下の 3 種類に分類される．

1. ライセンス記述を含み，その記述が示すライセンスを特定することが可能である
2. ライセンス記述を含み，その記述が示すライセンスを特定することが不可能である
3. ライセンス記述を含まない．

共通文字列の分類は，共通文字列に含まれる，ライセンス名やライセンス記述を含むファイルへの参照，URL，列挙されている条項に基づいて判定する．図 3 は 1. の例である．この

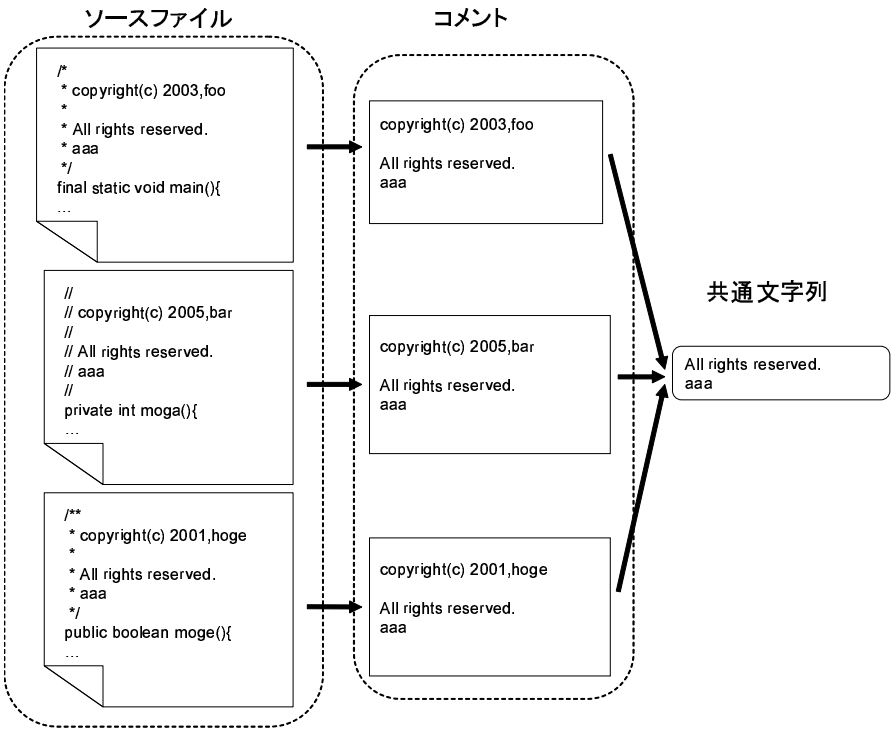

図 2: 共通文字列の抽出

図には GNU Lesser General Public License Version 2.1 [6] のライセンス記述が含まれていると特定できる．図 3 は 2. の例である．2. は，複数のライセンスを指定する記述の部分文字列となっている共通文字列が該当する．この例では，GNU に関連するライセンスのライセンス記述の一部と推測できるが，どのライセンスか特定できない．図 5 は 3. の例である．この例ではライセンスと無関係な文字列が共通文字列として抽出されている．

このような共通文字列は大量に抽出されるため，利用者がどの共通文字列を読むべきかを決める基準が必要となる．本研究ではその基準として，未特定ファイル数を提案する．

未特定ファイル数とは，ある共通文字列を含むソースファイルのうち，ライセンスをカテゴリ名とするカテゴリに分類されておらず，ライセンスが判明していないソースファイルの数である．R を，ソースファイル f と f のライセンス l との関係 (f,l) を要素とする集合であるとすると，共通文字列 p の未特定ファイル数は以下の notClassifiedFiles 関数により求められる．

$$\text{unknownLicenseFiles}(p) = |\{f | \forall l, (f,l) \notin R \wedge f \in \text{relatedFile}(p)\}|$$

relatedFile(p) は p を含むソースファイルの集合である．例として，図 7 のような対応関係があるとする．図 7 では，ソースファイルとコメント，共通文字列の対応関係の一例を示している．この図において，矢印は対応関係を表し，A→B は A は B に含まれる文字列であるこ

```
This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this
```

図 3: ライセンス記述を含み，その記述が示すライセンスを特定することが可能である共通文字列の例

```
is distributed in the hope that it will be useful,
   but WITHOUT ANY WARRANTY; without even the implied warranty of
   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
```

図 4: ライセンス記述を含み，その記述が示すライセンスを特定することが不可能である共通文字列の例

とを示す．この図 7 の例では，$relatedFile(p_2) = \{f_1, f_2, f_3, f_4\}$ である．

また，ライセンスに分類されたソースファイルとは，ソースファイルが含む共通文字列のうち，利用者により共通文字列が示すライセンスが特定された共通文字列が 1 つ以上存在するソースファイルとする．未特定ファイル数は，利用者がその共通文字列が示すライセンスを特定することができれば，新たにライセンスをカテゴリ名としたカテゴリに分類されるソースファイルの数を意味する．

提案手法では，あるソースファイルが含む共通文字列のうち，ひとつの共通文字列に対し

```
Fetch the next object in the iteration

      @return The next object

      Returns true if and only if there are more objects available
      via the <code>next()</code> method

      @return The next object
```

図 5: ライセンス記述を含まない共通文字列の例

てライセンスを入力する際，利用者により共通文字列が示すライセンスが特定された共通文字列が既に存在し，かつそれらのライセンスが異なる状況が発生する場合がある．この状況をライセンスの多重割り当てとする．ファイル f において，共通文字列 p_1，p_2 によるライセンスの多重割り当ての判定は以下の conflict 関数により求められる．ライセンスの多重割り当ては，本研究の仮定である 1 つのファイルは 1 ライセンスに分類されることに反する．そのため，提案手法ではライセンスの多重割り当てを解消する処理を行う．

$$\text{conflict}(f, p_1, p_2) = \exists f \in relatedFile(p_1) \exists f \in relatedFile(p_2)((p_1, l_n), (p_2, l_m) \in R \wedge l_n \neq l_m)$$

衝突の解消は以下の 2 つの方式で行う．

1. どちらか一方のライセンスをソースファイルに適用する
2. 2 つのライセンスを一つにまとめてデュアルライセンス（利用者が 2 つのライセンスのうちどちらか一方を選択することができるライセンス）が適用されているとする

1. の場合は衝突の原因となっている共通文字列のうち一方を選択する．そして，選択されなかった共通文字列を p とし，衝突が発生しているファイルを f とする時，$relatedFile(p)$ から衝突が起きている f を除去する．

2. の場合は衝突の原因となっている共通文字列を p_1, p_2 とし，衝突が発生しているファイルを f とする時，$relatedFile(p_1), relatedFile(p_2)$ から f を除去する．そして p_1, p_2 を集約する仮の共通文字列 p_{new} を作成し，relatedFile(p_{new}) に f を追加する．

提案手法では，利用者が，共通文字列が示すライセンスを特定する．理由は，結果の信頼性を重視し，ソースファイルとライセンスの関係に誤りを含まないようにするためである．

パターンマッチなどの方法により，その記述がどのライセンスを指定するかの解釈を自動的に行うことも考えられるが，完全に誤りを防ぐことは困難である．得られたライセンスの特定結果に誤りを含む場合は，部品を再利用するときに改めてライセンスが正しいか調査する必要があり，部品を再利用するための開発者の労力を軽減するという本研究の目的を達成できない．

3.2 分類の手続き

提案手法は，ソースファイル集合 F を入力とし，ソースファイル f とライセンス集合 L に属するライセンス l の関係 (f,l) の集合 R を得る対話的な手続きである．なお，L は手続き開始時には空集合である．

具体的なライセンス特定の手順を図 6 に示す．以下，図 6 の手順について詳細に説明する．

3.2.1 コメントの抽出

ソースファイル集合 F を入力として，連結コメントの集合 C を抽出する (図 6，行 1)．連結コメントとは，ソースファイルに含まれる全てのコメントを 1 つの文字列に連結した文字列である．この処理では，ひとつのソースファイル中の全てのコメントはひとつの連結コメントとして抽出される．そのため，F に属する各ソースファイルからそれぞれ 1 つの C に属する連結コメントが抽出され，F の各要素と C の各要素は 1 対 1 に対応する．

3.2.2 共通文字列の抽出

連結コメント集合 C を入力として共通文字列集合 P を抽出する (図 6，行 2)．P に属する共通文字列は C に属する複数の連結コメントから抽出されるため，P の要素 1 つに対し．C の要素が複数対応する．一方，C に属する連結コメント 1 つに複数の共通文字列が含まれている場合があるため，C の要素 1 つに対し．P の要素が複数対応する．したがって，C の要素と P の要素には多対多の対応関係がある．

3.2.3 手作業によるソースファイルの分類

利用者が共通文字列を読み，共通文字列にライセンスを指定する記述が含まれると利用者が判断する場合，ライセンスを入力することにより，読んだ共通文字列を含むソースファイルを分類する．まず，利用者は表示された共通文字列を読む (図 6，行 6)．次に，利用者は共通文字列がライセンスを指定する記述を含むかどうか入力する (図 6，行 7)．最後に，共通文字列がライセンスを指定する記述を含んでいると利用者が入力した場合 (図 6，行 8)，利

用者は指定する記述がどのライセンスを示すか特定し，そのライセンスを入力する (図 6，行 9)．ライセンスが入力されると，手続きは利用者が入力したライセンスをライセンス集合 L に追加する (図 6，行 10)．そして，利用者が入力したライセンスと利用者が読んだ共通文字列に対応するソースファイル集合との関係が R に追加される (図 6，行 11) ことにより，読んだ共通文字列に対応するソースファイルが分類される．

3.2.4 衝突の解消

ライセンスの入力により発生した衝突を解消する．ライセンス入力後，衝突が発生しているか検出する (図 6，行 14)．衝突が発生している場合，利用者に衝突の発生を知らせる (図 6，行 15)．次に，利用者は衝突が発生している p_{max} と p のどちらかのライセンスを適用するか，両方無効化することを選択する (図 6，行 16)．p_{max} と p のどちらか一方のライセンスをソースファイルに適用する場合は，選択しなかった共通文字列とファイル f の対応関係を除去する (図 6，行 18，20)．p_{max} と p のライセンスのデュアルライセンスをソースファイルに適用する場合は p_{max}，p とファイル f の対応関係を除去する (図 6，行 22，23)．そして，p と p_{max} を集約した仮の共通文字列 p_{new} を作成する (図 6，行 24)．ここで，createNewString(p_1, p_2) は p_1 と p_2 の文字列を集約した仮の共通文字列を返す関数である．そして，p_{new} と f の対応関係を作成する (図 6，行 25)．

1. ソースファイル集合 F から連結コメント集合 C を抽出
2. C から共通文字列集合 P_0 を抽出
3. $P = P_0$
4. **while**($P \neq \phi$)
5. 集合 $\{p' | p' \in P \wedge \mathrm{unknownLicenseFiles}(p') = \max_{p \in P} \mathrm{unknownLicenseFiles}(p)\}$ の要素を 1 つ選び，$p_{\max}$ とする
6. 利用者は $p_{\max}$ を読む
7. 利用者が $p_{\max}$ が特定可能なライセンスを指定する記述を含むか真偽値を入力
8. **if**(7. で入力された値が真である)
9. 利用者はライセンス l を入力
10. $L \leftarrow L \cup \{l\}$
11. $R \leftarrow R \cup \{(f', l) | f' \in relatedFile(p_{max})\}$
(* ライセンスの多重割り当ての検出 *)
12. **for each** $f \in F$
13. **for each** $p \in \{p | p \in relatedFile(f) \wedge (p, l) \in R \wedge l \in L\}$
14. **if** conflict(f, p_{max}, p)
15. 利用者に p_{max} と p のライセンスが衝突していることを知らせる
16. p_{max} または p のライセンスを適用するか，p_{max}と p のライセンスのデュアルライセンスを適用するかを選択
17. **if**(16. で p_{max} のライセンスを選択した)
18. $relatedFile(p) \leftarrow relatedFile(p) \backslash \{f\}$
19. **else if**(16. で p のライセンスを選択した)
20. $relatedFile(p_{max}) \leftarrow relatedFile(p_{max}) \backslash \{f\}$
21. **else**(16. で p_{max}と p のライセンスのデュアルライセンスを選択した)
22. $relatedFile(p) \leftarrow relatedFile(p) \backslash \{f\}$
23. $relatedFile(p_{max}) \leftarrow relatedFile(p_{max}) \backslash \{f\}$
24. $p_{new} = createNewString(p_{max}, p)$
25. $relatedFile(p_{new}) \leftarrow relatedFile(p_{new}) \cup f$
26. **end if**
27. **end if**
28. **end for**
29. **end for**
30. **end if**
31. $P \leftarrow P \backslash (\{p | \mathrm{unknownLicenseFiles}(p) = 0\} \cup \{p_{\max}\})$
32. **end while**

図 6: 提案手法におけるソースファイルのライセンスを特定する手続き

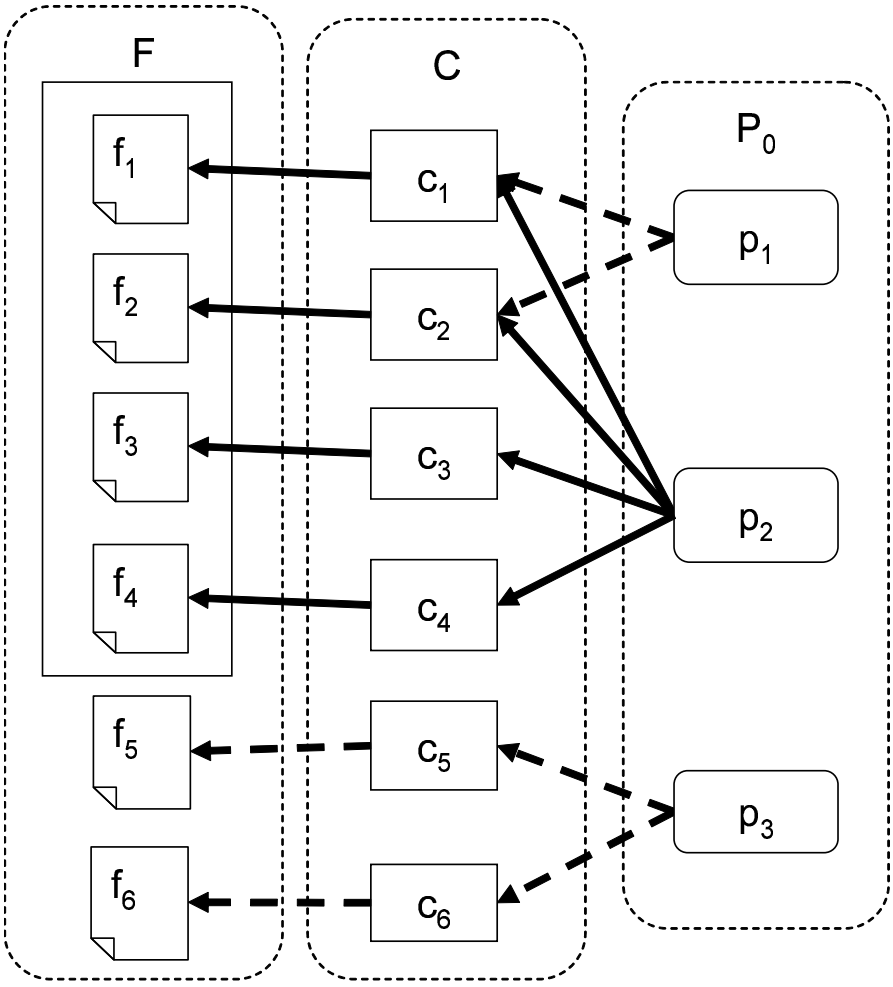

図 7: 共通文字列とソースファイルの対応

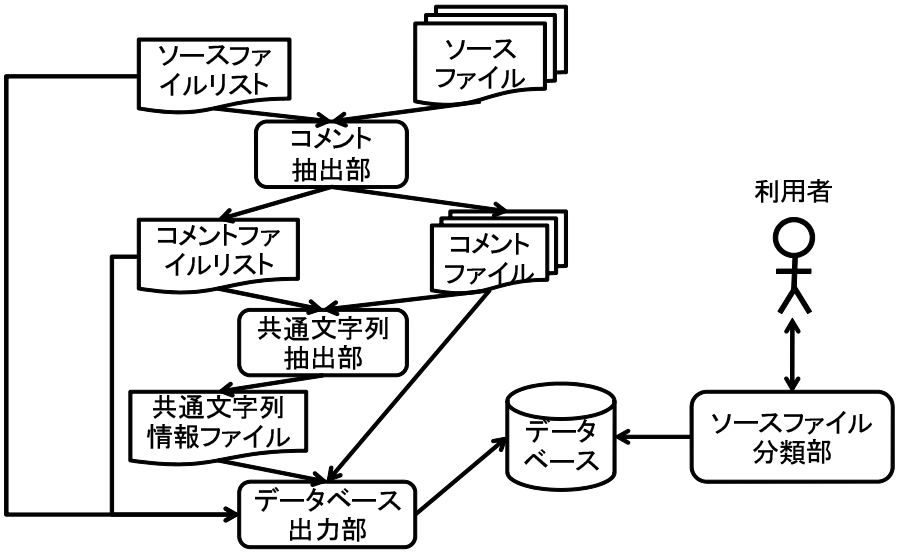

図 8: システム構成図

4 提案手法の実装

本節では提案手法を実装について説明する．4.1 では実装したシステムの構成について説明する．4.2 では実装したシステムの構成に含まれるデータベースについて説明する．4.3 ではそれぞれ，システムを構成するモジュールの概要について説明する．

4.1 システムの構成

実装したシステムは，本研究で提案しているソースファイル分類手法に基づき，利用者とシステムとの相互作用によりソースファイルの分類を行うシステムである．実装したシステムの構成は図 8 の通りである．本システムは入力として，分類対象となるソースファイルと，それらのソースファイルのパスを記述したソースファイルリストを使用する．出力結果はデータベースに保存される．

4.2 データベース

本システムではソースファイル分類に必要な情報はデータベースに格納する．データベースシステムとして本システムでは PostgreSQL [11] を使用している．データベースにはファイル情報，共通文字列情報，ファイル-共通文字列関係情報，ライセンス情報に対応するテーブルと未分類ファイル数を表示するビューを作成した．各テーブルとビューの詳細は以下で述べる．

- テーブル

file 分類の対象となるソースファイルの情報を格納するテーブル．テーブルのフィールド以下の通りである．ファイル ID はソースファイルに一意に付与される整数である．ファイルパスはソースファイルが存在するパスである．コメントファイルパスはソースファイルに対応するコメントファイルが存在するパスである．

- ファイル ID
- ファイルパス
- コメントファイルパス

pattern 共通文字列情報を格納するテーブル．テーブルのフィールドは以下の通りである．共通文字列 ID は共通文字列に一意に付与される整数である．共通文字列は共通文字列そのものである．ライセンス ID は後述するライセンステーブルにおいてライセンスに一意に付与される数値である．処理済みフラグはソースファイル分類部において，その共通文字列が表示されたかどうかを示す論理型の値である．

- 共通文字列 ID
- 共通文字列
- ライセンス ID
- 処理済みフラグ

patternrelate ある共通文字列がどのファイルに含まれるかという情報を格納するテーブル．テーブルのフィールドは以下の通りである．共通文字列 ID は共通文字列テーブルで共通文字列に対し，一意に付与される整数値である．ファイル ID はファイルテーブルでソースファイルに対し，一意に付与される整数値である．無効化フラグはファイル ID に対応するソースファイルに共通文字列 ID に対応する共通文字列が含まれることを無効化しているかを示す論理型の値である．

- 共通文字列 ID
- ファイル ID
- 無効化フラグ

license ライセンスを格納するテーブル．テーブルのフィールドは以下の通りである．ライセンス ID はライセンスに一意に付与される整数値である．ライセンスは利用者が入力したライセンスである．

- ライセンス ID

- ライセンス名

- ビュー

  undefinedpattern まだ利用者に表示されていない未特定ファイル数が 0 でない共通文字列の一覧を表示するためのビュー．ビューのフィールドは以下の通りである．

  - 共通文字列 ID
  - 未特定ファイル数
  - 共通文字列

4.3 システムを構成する各モジュールの概要

本節ではシステムを構成する各モジュールの概要について説明する．

コメント抽出部 コメント抽出部は各ソースファイルに含まれるコメントを抽出する図 6 の 1. に対応するモジュールである．このモジュールは入力としてソースファイルとソースファイルのパスが列挙されたソースファイルリストを入力とし，コメントファイルとコメントファイルリストを出力する．抽出されたコメントはコメントファイルとして保存する．本モジュールはドキュメントコメント，ブロックコメント，ラインコメントにあるコメントの形式に対応している．

共通文字列抽出部 共通文字列抽出部では入力されたコメントファイルから共通文字列を抽出する図 6 の 2.,3. に対応するモジュールである．共通文字列の抽出には CCFinder を使用した [21]．抽出された共通文字列の情報として以下の情報が含まれる．

- ファイル ID
- コメントファイルのパス
- 共通文字列の開始位置 (行番号・カラム)
- 共通文字列の終了位置 (行番号・カラム)

データベース出力部 データベース出力部ではソースファイルリストと共通文字列情報ファイルとコメントファイルを入力とし，データベース中の各テーブルに対応する情報を格納する．

ソースファイル分類部　ソースファイル分類部は，利用者とシステムの相互作用によりソースファイルを分類する図 6 の 4.-31. に対応するモジュールである．このモジュールは利用者の入力に基づき，データベースを操作することでソースファイルの分類を行う．分類の結果は SQL 文を用いて，各テーブルを組み合わせることにより得ることができる．

5　適用実験

提案手法の有効性を評価するため，オープンソースソフトウェアに含まれるソースファイルの集合を用いた適用実験を行った．

5.1　提案手法の有効性評価方法

本実験では，ソースファイル分類の効率，時間を尺度とした利用者の労力，共通文字列の数を尺度とした利用者の労力の 3 つの観点から提案手法を評価する．

実験では複数のオープンソースソフトウェアに含まれるソースファイルの集合 4 組に対して，提案手法を適用した．それぞれの実験対象を構成しているソフトウェア，それらのソフトウェアが基づくライセンス，対象ソースファイル数を表 1 に記す．なお，実験では Java ソースファイルのみを対象とし，ソフトウェアが基づくライセンスは SourceForge のプロジェクトデータに従っている．実験対象 1，実験対象 2，実験対象 4 は異なったソースファイル数で構成されており，ソースファイル数が増えた場合の利用者が共通文字列を読む労力や，ライセンスの特定にかかる時間の変化を評価することができる．また，実験対象 2 と実験対象 3 はソースファイル数に大きな差は無いが，互いにライセンスが重複しないソフトウェアで構成してあり，ライセンスの違いが提案手法に与える影響を確認することができる．

提案手法の効率を評価するため，特定率を評価する．特定率はライセンスをカテゴリ名とするカテゴリに分類されたソースファイル集合を $F_{\text{identified}}$，ライセンス特定の対象とするソースファイル集合を F としたとき，以下のように定義される．

$$\text{特定率} = \frac{|F_{\text{identified}}|}{|F|}$$

次に，ソースファイル分類に要した時間を測定する．

最後に，利用者が何個の共通文字列を見る必要があるかを測定する．

実験は，4 節で説明したシステムを使用し，著者が表 2 で示す計算機環境を用いて行った．

5.2　測定結果

5.2.1　評価 1:特定率の測定

実験対象 1〜4 に提案手法を適用したときの，特定率の推移をそれぞれ図 4〜7 に示す．各グラフにおいて，実線は特定率の推移，破線は，実験対象のソースファイルに占めるライセンス記述を含むソースファイルの割合を示している．

まず，全ての実験対象において，少数の共通文字列を読んだ時点でライセンス記述を含むソースファイルの割合と近い特定率を達成していることがわかる．実験対象 1 では 4 個，実

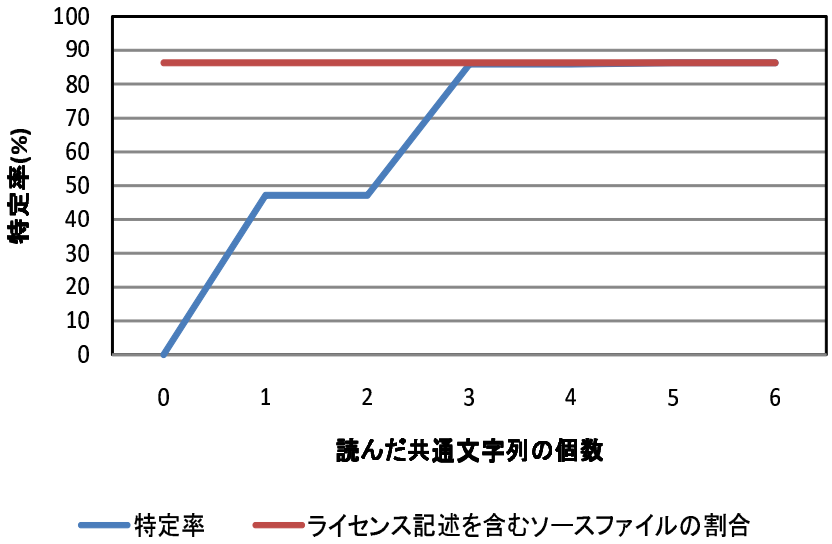

図 9: 実験対象 1 を対象とした特定率の推移

験対象 2 では 5 個，実験対象 3 では 3 個，実験対象 4 では 4 個の共通文字列を利用者が読んだ時点で，ライセンス記述を含むソースファイルのうち，90%以上のソースファイルを分類できている．

次に，手法が終了した時点での特定率はライセンス記述を含むソースファイルの割合とほぼ同値となっているが，上限には達していない．ライセンス記述を含むソースファイルの割合は実験対象 1 で 86.3%，実験対象 2 で 88.2%，実験対象 3 で 38.3%，実験対象 4 で 62.8%であるのに対し，手法が終了した時点での特定率は，実験対象 1 では 86.3%，実験対象 2 では 88.1%，実験対象 3 では 38.2%，実験対象 4 では 61.5%であった．

また，図 10 の 10 個目以降のように，各グラフで特定率の上昇が見られない部分が広範囲にわたって存在する．これらの部分では 3.1 での共通文字列の分類において，2. と 3. に分類される共通文字列が手続きにより表示されていた．

5.2.2　評価 2:所要時間の測定

提案手法による対象ファイル数と所要時間の関係は，表 3 の通りである．実験結果より，8000 ファイル程度のソースファイル集合に対して，利用者は 1 時間以内で対象のソースファイルを分類することができた．また，ソースファイル数 2000 以上の 3 つの実験対象に対して提案手法を適用した場合，手作業によるソースファイルの分類 (図 6，行 4-31) にかかる所要時間は手法全体の所要時間の 50%以上を占めた．

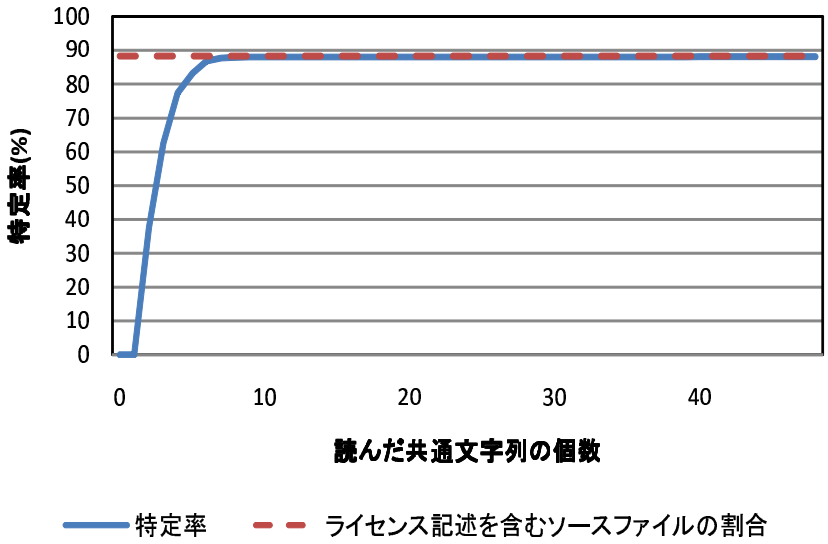

図 10: 実験対象 2 を対象とした特定率の推移

5.2.3　評価 3:手法が終了するまでに表示された共通文字列の個数の測定

提案手法による対象ファイル数と手法が終了するまでに表示された共通文字列の個数の関係は，表 4 の通りである．ソースファイル数が多い場合，手法により提示される共通文字列は多くなっていた．また，ソースファイルの分類に使用できなかった共通文字列がその大半を占めていた．例えば，単なる著作権表記や，複数のライセンスに共通する表記が挙げられる．ライセンスの特定に使用できない共通文字列をフィルタリングすることで，利用者の労力を削減することが出来ると考えられる．

5.3　考察

特定率はライセンス記述を含むソースファイルの割合とほぼ同値であった．この結果より，提案手法によりライセンス名をカテゴリとするカテゴリに分類できていないファイルが存在するが，その数はごくわずかであるため，実用上は問題ないと考えられる．また，全ての実験対象において，4 個程度の共通文字列を読んだ段階で，ライセンス記述を含むソースファイルのうち 8 割以上が特定できている．このため，提案手法を用いた場合，ライセンスを記述を含まないソースファイルを除く大部分のファイルを少数の共通文字列で分類できている．

提案手法では，利用者は表示された共通文字列全てを読まなくてはならないため，ライセンス記述を含まない文字列を読むために多くの労力を割いていた．このような共通文字列を除去することにより，利用者の労力をさらに減らすことができると考えられる．この目的の

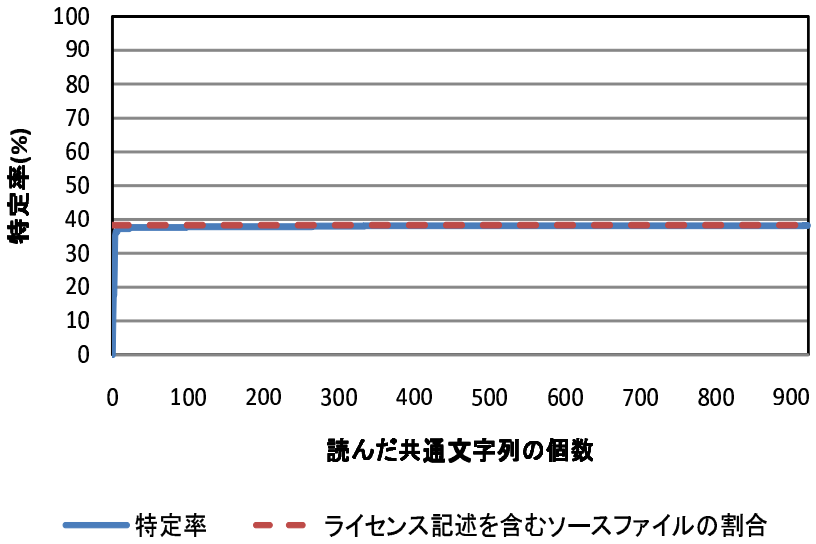

図 11: 実験対象 3 を対象とした特定率の推移

ために，現在，機械学習による文書分類手法 [26] を用いることを考えている．

適用実験では，最大で 8000 ファイル程度のソースファイル集合を実験対象として用いた．この実験対象のソースファイル数は，現在利用可能なソースファイル検索システムが検索対象とするソースファイル数と比べて小さい．具体的には，Koders は約 330000 個，SPARS-J のデモシステム [3] は約 300000 個の Java クラスを検索対象としている [18]．しかし，実験対象は，広く用いられているライセンスを含む様々なライセンスのソフトウェアで構成されており，ライセンスの種類に関しては現実を反映しているといえる．今後，大規模なソースファイル集合に対して手法を適用することにより．より実用的な観点での評価を行うことが必要である．

また，適用実験では提案手法の有効性をソースファイル分類の効率，時間を尺度とした利用者の労力，共通文字列の数を尺度とした利用者の労力の 3 つの観点から評価した．しかし，適用実験は提案手法を利用することが，完全に手作業で行う場合や関連手法に比べ，どのような長所や短所があるか示せていない．そのため，今後の課題として，提案手法と，関連手法や手作業との比較実験を行うべきである．

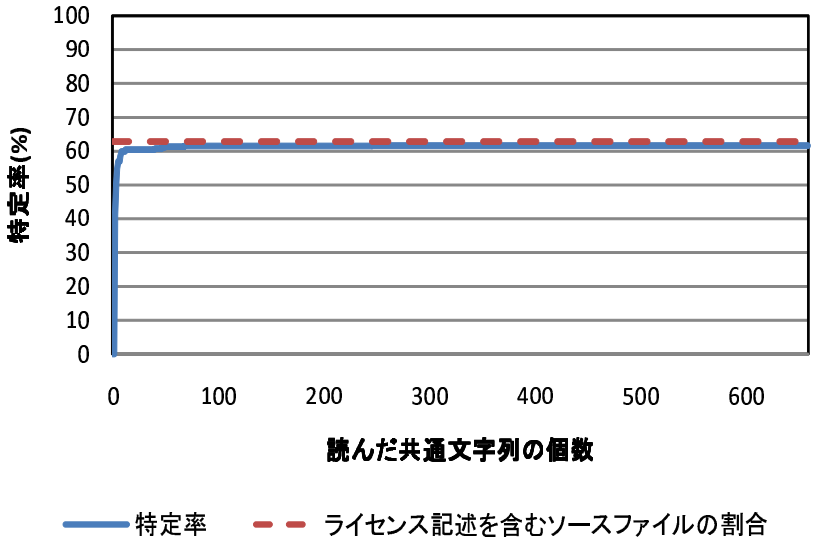

図 12: 実験対象 4 を対象とした特定率の推移

表 1: 実験対象

ソフトウェア名	配布ライセンス	ファイル数
実験対象 1		
Freemaker	BSD License	318
galleon	GPL	236
	合計	554
実験対象 2		
JTrac	Apache License V2.0	156
JUNG	BSD License	647
jEdit	GPL	520
FindBugs	LGPL	818
	合計	2141
実験対象 3		
JMRI	Artistic License	985
Junit	Common Public License	173
RSSOwl	Eclipse Public License	413
mged	MIT License	467
XUI	MPL 1.1	310
	合計	2348
実験対象 4		
Artifactory	Apache License V2.0	158
Spring Framework	Apache Software License	3086
Jetty	Artistic License	461
HSQL Database Engine	BSD License	131
Robocode	Common Public License	172
Bioclipse	Eclipse Public License	1288
ZK	GPL	804
Hibernate	LGPL	1556
Jester	MIT License	55
JGAP	MPL 1.1, LGPL	108
	合計	7819

表 2: 実験で使用した計算機環境

OS	FreeBSD 6.2RELEASE
CPU	Intel Xeon 5160 3.0GHz × 2
搭載メモリサイズ	3G バイト

表 3: 所要時間

	ファイル数	コメントの抽出 (秒)	共通文字列の抽出 (秒)	データベースへの出力 (秒)	手作業によるライセンスの特定 (秒)
実験対象 1	554	12.3	23.8	7.0	74
実験対象 2	2141	33.4	81.2	47.3	489
実験対象 3	2348	32.0	50.0	33.5	1371
実験対象 4	7819	116.6	868.1	170.0	1278

表 4: 手法が終了するまでに表示された共通文字列の個数

	ファイル数	ソースファイルの分類に用いた共通文字列	ソースファイルの分類に用いなかった共通文字列
実験対象 1	554	3	3
実験対象 2	2141	10	38
実験対象 3	2348	10	912
実験対象 4	7819	15	643

6 関連研究

本節ではソースファイルをライセンスごとへの分類をする手法や，ソースファイルのライセンスを明らかにするライセンス特定手法について述べる．蓄積しているソースファイルがライセンスごとに分類するソースファイル検索システムとして，Koders [9] や Google Code Search [7] などがある．Google Code Search ではコメントまたは COPYING，LICENSE など，特定の名前をもつファイルからライセンス情報を取得する．Koders や Google Code Search が対象とするソースファイルの数は多いが，登録されている全ソースファイルに対してライセンスを正しく特定できている割合は低い．提案手法は正しいライセンスのカテゴリに分類されたファイルの割合は高いが，ライセンスの特定にソースファイルのコメントのみを利用しているため，ライセンスを別のファイルに記述する方法には対応していない．

Tunnanen らは正規表現を用いたライセンスを特定する手法を提案している [29]．この手法ではライセンス記述に適合する正規表現を作成し，各ソースファイルがどの正規表現に適合するかにより，ソースファイルが基づくライセンスを特定する．この手法の利用者は，ライセンスが特定されていないソースファイルを見て正規表現を作成する必要がある．

Fossology [4] はソースファイルのライセンス特定を行うソフトウェア解析ツールである．このツールでは事前に用意されたライセンス記述とソースコード中のコメントとの文字列比較によりライセンスの特定を行う．文字列比較では，一部一致しない箇所がある場合でも，その前後の一致する部分の検出ができている．

これらの手法では利用者が事前準備を行う必要があるが，提案手法では利用者が事前準備を行う必要がない．そのため，利用者がライセンスの特定を行うのにかかる労力は小さいと考えられる．

7 むすび

本研究ではソースファイルのコメント中に含まれるライセンス記述を用いたソースファイル分類手法を提案した．提案した手法はライセンス記述が複数のソースファイルのコメントに共通して現れることが多いことに着目し，同じライセンス記述を持つソースファイルを一度に分類することができる．オープンソースソフトウェアに含まれるソースファイルの集合に対する適用実験により，少ない労力でライセンスを特定することが可能であることを示した．

今後の課題として，関連手法との比較実験が必要である．また，利用者がより小さい労力で正しくライセンスが特定できるよう，手法の改善に取り組む予定である．具体的には，機械学習を用いた共通文字列のフィルタリングを考えている．また，提案手法をより大規模なソースファイル集合に対して適用可能にし，ソースファイル検索システムに組込むことも重要な課題である．

謝辞

本研究の全過程を通して，常に適切な御指導を賜わりました大阪大学大学院情報科学研究科コンピュータサイエンス専攻 井上 克郎 教授に心より深く感謝致します.

本研究において，常に適切な御指導および御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻 松下 誠 准教授に深く感謝致します.

本研究において，常に適切な御指導および御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻 石尾 隆 助教に深く感謝いたします.

本研究において，常に適切な御指導および御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻 早瀬 康裕 特任助教に深く感謝いたします.

本研究において， 常に適切な御指導および御助言を頂きました大阪大学大学院情報科学研究科コンピュータサイエンス専攻 市井 誠 氏に深く感謝いたします.

最後に，その他様々な御指導，御助言等を頂いた大阪大学大学院情報科学研究科コンピュータサイエンス専攻井上研究室の皆様に深く感謝致します.

参考文献

[1] Apache software foundation. http://apache.org.

[2] Asf source header and copyright notice policy. http://www.apache.org/legal/src-headers.html.

[3] demo.spars.info. http://demo.spars.info/.

[4] Fossology. http://fossology.org/.

[5] FreeBSD. http://www.freebsd.org/.

[6] GNU Lesser General Public License, version 2.1. http://www.gnu.org/licenses/old-licenses/lgpl-2.1.html.

[7] Google Code Search. http://www.google.com/codesearch.

[8] Java se. http://java.sun.com/javase/.

[9] koders. http://www.koders.com/.

[10] Open source initiative. http://www.opensource.org/.

[11] Postgresql. http://www.postgresql.org/.

[12] Sourceforge.net. http://sourceforge.net/.

[13] Sushil Bajracharya, Trung Ngo, Erik Linstead, Yimeng Dou, Paul Rigor, Pierre Baldi, and Cristina Lopes. Sourcerer: A search engine for open source code supporting structure-based search. In *Proc. International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'06)*, pp. 25–26, October 2006.

[14] Christine L. Braun. *Reuse, Encyclopedia of Software Engineering*, Vol. 2. John Wiley & Sons, 1994.

[15] Christine L. Braun. ソフトウェア工学大辞典, 再利用, pp. 338–405. 朝倉書店, 1994.

[16] Jack Greenfield and Keith Short. *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Addison Wesley, 2004.

[17] Reid Holmes and Robert J. Walker. Supporting the investigation and planning of pragmatic reuse tasks. In *Proc. 29th International Conference on Software Engineering (ICSE'07)*, pp. 447–456, May 2007.

[18] Oliver Hummel and Colin Atkinson, editors. *Using the Web as a Reuse Repository*, ICSR 06':Proceedings of the 9th International Conference on Software Reuse, 2006.

[19] Katuro Inoue, Reishi Yokomori, Tetsuo Yamamoto, Makoto Matsushita, and Shinji Kusumoto. Ranking significance of software components based on use relations. *IEEE Transactions on Software Engineering*, Vol. 31, No. 3, pp. 213 – 225, 2005.

[20] I. Jacobson, M. Griss, and P. Jonsson. *Software Reuse*. Addison Wesley, 1997.

[21] Toshihiro Kamiya, Shinji Kusumoto, and Katuro Inoue. CCFinder: a multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering*, Vol. 28, No. 7, pp. 654 – 670, 2002.

[22] Hafedh Mili, Fatma Mili, and Ali Mili. Reusing software: Issues and research directions. *IEEE Transactions on Software Engineering*, Vol. 21, No. 6, pp. 528–561, June 1995.

[23] Brandon Morel and Perry Alexander. Spartacas: automating component reuse and adaptation. *IEEE Transactions on Software Engineering*, Vol. 30, No. 9, pp. 587–600, September 2004.

[24] Thomas J. Ostrand. ソフトウェア工学大辞典, データ権, pp. 927–950. 朝倉書店, 1994.

[25] Young Park. Software retrieval by samples using concept analysis. *The Journal of Systems and Software*, Vol. 54, No. 3, pp. 179–183, November 2000.

[26] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, Vol. 34, No. 1, pp. 1–47, 2002.

[27] Richard W. Selby. Enabling reuse-based software development of large-scale systems. *IEEE Transactions on Software Engineering*, Vol. 31, No. 6, pp. 495–510, June 2005.

[28] Arun Sharma, Rajesh Kumar, and P. S. Grover. A critical survey of reusability aspects for component-based systems. In *Proc. World Adacemy of Science, Engineering, and Technology Volume 21*, pp. 411–415, January 2007.

[29] Timo Tuunanen, Jussi Koskinen, and Tommi Kärkkäinen. ASLA: Reverse engineering approach for software license information retrieval. In *CSMR '06: Proceedings of the Conference on Software Maintenance and Reengineering*, pp. 291–294, Washington, DC, USA, 2006. IEEE Computer Society.

[30] Yunwen Ye and Gerhard Fischer. Reuse-conducive development environments. *Automated Software Engineering*, Vol. 12, No. 2, pp. 199–235, 2005.

[31] 大須賀俊憲, 金子伸幸, 山本晋一郎, 小林隆志, 阿草清滋. ソフトウェア統合検索を利用した再利用支援システム. ソフトウェア工学の基礎 XIV 日本ソフトウェア科学会 FOSE2007, 2007.

[32] 横森励士, 梅森文彰, 西秀雄, 山本哲男, 松下誠, 楠本真二, 井上克郎. Java ソフトウェア部品検索システム SPARS-J. 電子情報通信学会論文誌 D-I, Vol. J87-D-I, No. 12, pp. 1060–1068, 2004.

付録

A. 実験時に表示された共通文字列の例

A　実験時に表示された共通文字列の例

本節では，実験時，提案手法を実験対象 2 に適用した際に表示され，ソースファイルの分類に用いた共通文字列を示す．以下の各図は表示された共通文字列を表しており，各図の注釈はその共通文字列が含むライセンス記述が示すライセンスである．

```
This library is free software; you can redistribute it and/or
   modify it under the terms of the GNU Lesser General Public
   License as published by the Free Software Foundation; either
   version 2
```

図 13: GNU Lesser General Public License version 2.1

```
 the JUNG Project and the Regents of the University
   of California
   All rights reserved.

   This software is open-source under the BSD license; see either
   "license.txt" or
   http://jung.sourceforge.net/license.txt for a description
```

図 14: BSD License

```
 This program is free software; you can redistribute it and/or
   modify it under the terms of the GNU General Public License
   as published by the Free Software Foundation; either version 2
   of the License, or
```

図 15: GNU General Public License version 2

```
Copyright 2002-2005 the original author or authors.

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License
```

図 16: Apache License version 2.0

```
This file is part of the BeanShell Java Scripting distribution.
   Documentation and updates may be found at http://www.beanshell.org/

   Sun Public License Notice:

   The contents of this file are subject to the Sun Public License Version
   1.0 (the "License"); you may not use this file except in compliance with
   the License. A copy of the License is available at http://www.sun.com

   The Original Code is BeanShell. The Initial Developer of the Original
   Code is Pat Niemeyer. Portions created by Pat Niemeyer are Copyright
   (C) 2000.  All Rights Reserved.

   GNU Public License Notice:

   Alternatively, the contents of this file may be used under the terms of
   the GNU Lesser General Public License (the "LGPL"), in which case the
   provisions of LGPL are applicable instead of those above. If you wish to
   allow use of your version of this file only under the  terms of the LGPL
   and not to allow others to use your version of this file under the SPL,
   indicate your decision by deleting the provisions above and replace
   them with the notice and other provisions required by the LGPL.  If you
   do not delete the provisions above, a recipient may use your version of
   this file under either the SPL or the LGPL.

   Patrick Niemeyer (pat@pat.net)
   Author of Learning Java, O'Reilly & Associates
   http://www.pat.net/~pat
```

図 17: Sun Public License と GNU Lesser General Public License のデュアルライセンス

```
This work
  has been placed into the public domain. You may use this work in any way and
  for any purpose you wish.

  THIS SOFTWARE IS PROVIDED AS-IS WITHOUT WARRANTY OF ANY KIND, NOT EVEN THE
  IMPLIED WARRANTY OF MERCHANTABILITY. THE AUTHOR OF THIS SOFTWARE, ASSUMES
  _NO_ RESPONSIBILITY FOR ANY CONSEQUENCE RESULTING FROM THE USE, MODIFICATION,
  OR REDISTRIBUTION OF THIS SOFTWARE
```

図 18: public domain

```
  This software is published under the terms of the Apache Software License
  version 1.1, a copy of which has been included with this distribution in
  the LICENSE.txt file
```

図 19: Apache Software License version1.1

```
This library is free software; you can redistribute it and/or modify
  it under the terms of the GNU Library General Public License as published
  by the Free Software Foundation; either version 2 of the License, or
  (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU Library General Public License for more details.

  You should have received a copy of the GNU Library General Public License
  along with this program; if not, write to the Free Software
  Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA
```

図 20: GNU Library General Public License version 2

```
 The Apache Software Foundation.  All rights
   reserved.

   Redistribution and use in source and binary forms, with or without
   modification, are permitted provided that the following conditions
   are met:

   1. Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.

   2. Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in
      the documentation and/or other materials provided with the
      distribution.

   3. The end-user documentation included with the redistribution, if
      any, must include the following
```

図 21: Apache Software License version 1.1

```
This source is under the same license with JUNG.
   http://jung.sourceforge.net/license.txt for a description.
  package edu.uci.ics.jung.visualization;package org.ingrid.nexas.graph;
   Implements the Kamada-Kawai algorithm for node layout
```

図 22: BSD License