ソフトウェア部品検索システムを対象とするソフトウェアライセンス特定手法

真鍋　雄貴[†]　市井　誠[†]　早瀬　康裕[†]　松下　誠[†]　井上　克郎[†]

大阪大学大学院情報科学研究科[†]

1. はじめに

著者らが所属している研究グループでは，ソフトウェアの再利用を目的としたソフトウェア部品検索システム SPARS-J[1]を開発した．SPARS-J はソフトウェア部品を大規模ソースファイル群から抽出するという特徴がある．再利用をおこなう際には部品のソフトウェアライセンス（以降，単にライセンスとする）に従わなければならない．しかし，現在の SPARS-J ではライセンスを扱っておらず，SPARS-J の利用者が別途ライセンスを調べる必要がある．また，ライセンスの種類は多く，OSI[2]により承認されたオープンソースの為のライセンスに限定しても 50 種類以上存在するため，機械的な特定は困難である．このため，大規模ソースファイル群のライセンスを少ない手間で特定する手法が必要となる．

Tuunanen ら[3]は，正規表現を用いてライセンスを特定する手法を提案している．この手法は，ライセンスの特定に用いる正規表現の文字列を，ライセンスの特定が行われていないソースファイル（以下，単にファイルとする）を見て作成する必要がある．また，検索結果のライセンスを表示するソフトウェア部品検索システムとして Google Code Search[4]があるが，ライセンスが特定できていないファイルが多い．

本稿では，複数のファイルのコメント間に共通して表れる文字列を用いてライセンスを特定する手法を提案する．本手法ではシステムが自動的に抽出する文字列を読むことでライセンスを特定するため，管理者が正規表現の文字列などを作成する必要がない．また，多くのファイルに共通する文字列から順に処理するため，効率的にライセンスを特定できる．

2. ソフトウェアライセンス特定手法

2.1 提案手法の概要

本稿では，ファイルのブロックコメントを用いたライセンス指定に注目する．ブロックコメントとは複数行にわたり記述できるコメントを指す．以下，本稿ではブロックコメントを単にコメントと表記する．提案手法では以下の経験則に基づき，複数のコメントに共通して現れる文字列がライセンスの指定であるとみなす．ファイルには，単体での再利用を考慮し，ライセンスを指定する記述がコメントに含まれている場合が多い．また，同一のソフトウェアに含まれる同一のライセンスのファイルには，ライセンスの指定に統一された記述が用いられることが多い．

複数の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び，提案手法では共通文字列に対してライセンスの特定をおこなう．ファイルに対するライセンス指定の記述は共通文字列として抽出される．共通文字列により指定されるライセンスを特定することで，複数のファイルのライセンスを同時に特定できる．

効率的にライセンスの特定をおこなうためには，多くのファイルに含まれ，ライセンスが特定されていない共通文字列を調べればよい．共通文字列 A を含み，ライセンスの特定されていないファイルの数を共通文字列 A の未特定ファイル数とする．未特定ファイル数は，共通文字列 A の示すライセンスを特定することにより，新たにライセンスが特定されるファイルの数を示す．

また，共通文字列が示すライセンスはシステムの管理者が手作業で確認し，指定する．なぜなら，機械的に特定し誤りが生じた場合，システムの利用者に不利益が生じるためである．

2.2 提案手法の処理の流れ

提案手法の処理を以下に示す．また，Step1 から Step4 の流れを図 1 に示す．以下，特に注意がない場合はシステムによる自動的な処理とする．

Step1．ファイルからコメントを抽出する．

Step2．コメントから共通文字列を抽出する．

Step3. 各共通文字列の未特定ファイル数を計算し，未特定ファイル数を基に降順ソートをおこなう．

Step4．システムの管理者は Step3 の結果の上位から順に確認し，ライセンスを特定する．特定

An approach to identifying software licenses for software component retrieving system.

† Yuki MANABE ,† Makoto ICHII, † Yasuhiro HAYASE, † Makoto MATSUSHITA, † Katsuro INOUE

† Graduate School of Information Science and Technology, Osaka University

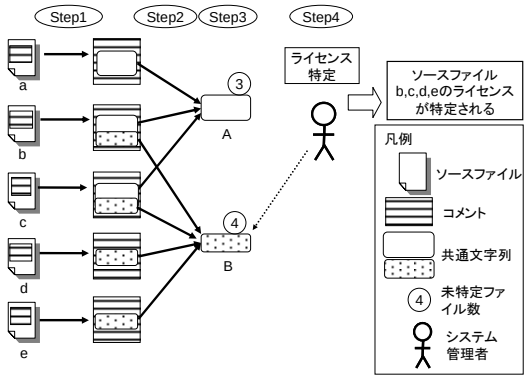

図 1 提案手法の処理の流れ

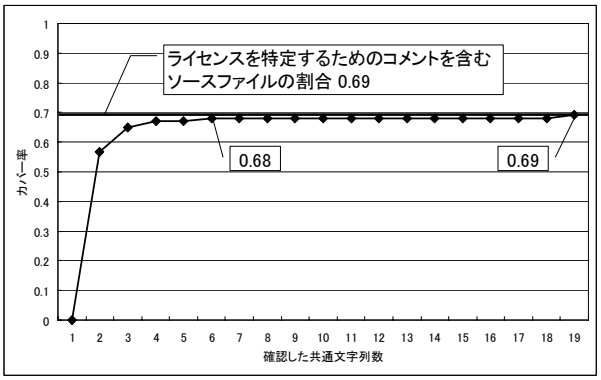

図 2 提案手法によるライセンス特定のカバー率

不可能であった場合は，特定可能な共通文字列が見つかるまで，順に共通文字列を確認する．

Step5．未特定ファイル数が 0 でなく，ライセンス特定不可能でない共通文字列が残っている場合は Step3 に戻る．残っていない場合は終了する．

3. 適用実験

3.1 実験概要

実験ではソフトウェアのファイル群に対し提案手法を適用した場合のライセンス特定のカバー率を調べる．本実験におけるカバー率は以下の式で表される．

$$\text{カバー率} = \frac{\text{ライセンスが特定されたファイル数}}{\text{ソフトウェアに含まれる全ファイル数}}$$

実験のため，手法を実装したツールを作成した．また，Step2 の共通文字列の抽出には CCFinder[5] を用いた．

実験対象として Gaim[6]のファイルを用いる．Gaim は 182 個の C 言語で書かれたファイルを含み，そのうち 126 個でコメントにより 3 種類のライセンスが指定され，全ファイルに対するこれらのファイルの割合は 0.69 である．

3.2 実験結果

Step2 で 141 個の共通文字列が抽出された．また，少なくとも 1 個の共通文字列を含むファイル数は 130 個であった．

次に，Step3～Step5 によるライセンス特定のカバー率と，ライセンスが指定されているファイルの割合を図 2 に示す．6 個の共通文字列を確認することでカバー率が 0.68 となり，ライセンスの指定されたファイルの 98%を特定できた．さらに，3 種類のライセンス全てを特定した．一方，1 番目と 5 番目に確認した共通文字列はライセンスを指定する記述の一部であったが，ライセンスを特定出来なかった．全ての処理が終了するまでに 19 個の共通文字列を確認し，カバー率は 0.69 となった．

3.3 実験の考察

実験では，6 個の共通文字列を確認することで，ライセンス指定の記述を含むファイルのうち，ほぼ全てに対してライセンスを特定できた．[3] では同様のカバー率を達成するために，7 種類の正規表現の文字列を作成する必要がある．本手法はシステムにより自動的に共通文字列を抽出するため，システムの管理者の負担をより軽減できると考えられる．

また，上位にライセンスを特定できない共通文字列が現れた．これは，ライセンスを指定する文字列の一部が異なっており，その共通部分のみが抽出されたためである．このようなライセンスの特定に至らず，また，差異の部分を含まない共通文字列は，差異の部分を含む共通文字列よりも未特定ファイル数が大きいため，上位に現れた．

4. 今後の課題

今後の課題は，実際のソフトウェア部品検索システムで検索対象となるような大規模なソースファイル群に対する適用実験と，本手法のソフトウェア部品検索システム SPARS-J に対する実装である．

参考文献

[1] Inoue, K., Yokomori, R., Yamamoto, T., Matsushita, M., Kusumoto, S.: Ranking significance of software components based on use relations, *IEEE Trans. Softw. Eng.* Vol. 31, No. 3, Pp.213 – 225 (2005)

[2] Open Source Initiative: http://www.opensource.org/

[3] Tuunanen, T., Koskinen, J., Kärkkäinen, T.: Retrieving open source software licenses, *Proc. OSS 2006*, Pp.35 – 46 (2006)

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

[5] Kamiya, T., Kusumoto, S., Inoue, K.: CCFinder: a multilinguistic token-based code clone detection system for large scale source code, *IEEE Trans. Softw. Eng.* Vol. 28, No. 7, Pp.654 – 670 (2002)

[6] Gaim: http://gaim.sourceforge.net