コードクローン研究における産学連携の取り組み

若　　者　　肥後芳樹*

Academic-Industrial Collaboration on Code Clone Research

Key Words：Code clone, Software Maintenance, Academic-Industrial Collaboration

1．はじめに

筆者がソフトウェア工学の研究に携わるようになって今年で8年目．この8年，時間がとても早く流れるように感じ，充実した研究生活を送ることができました．これもひとえに，大学院時代に所属した研究室および現在の所属研究室の先生方や仲間に恵まれたからだと感謝しています．このコラムでは筆者がこれまで研究の主眼をおいてきたコードクローンについて，産学連携の取り組みの様子とその成果について紹介したいと思います．

2．コードクローンとは

コードクローンとは，プログラムのソースコード中に存在する重複コードを指す．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原因の1つといわれている（近年，コードクローンはソフトウェア工学の中で最も注目を集めている分野の1つであり，2006年前後から大量にコードクローンに関係する論文が発表されている）．たとえば，あるコード片に対して修正を行う場合，もしそのコード片のコードクローンが存在する場合は，それらについても同様の修正の是非を検討しなければならない．このような作業は，システムがある程度以上大きい場合は非常に煩雑であり，また修正漏れによる新たなバグの混入の危険もある．そのため，システム中に存在するコードクローンを自動的に効率よく検出することが必要である．これまでにさまざまなコードクローン検出ツールが開発されており，筆者の研究グループでもコードクローン検出ツールCCFinder（産業技術総合研究所の神谷年洋氏によって後継機CCFinderXが開発されている，http://www.ccfinder.net/）や分析ツールICCA（ICCAウェブページ，http://sel.ist.osaka-u.ac.jp/icca/）を開発し，大学などの研究機関および企業などの実際のソフトウェア開発現場などに配布してきた．

3．産学連携の取り組み

筆者らはツールの開発者（大学）と利用者（主に産業界）の意見交換の場としてコードクローンセミナーを開催してきている．第一回セミナーは2002年11月に開催し，これまでに東京で3回，大阪で3回の計6回開催している．セミナーの内容は毎回異なるが，ツールのデモンストレーションと利用法講座や，コードクローン情報の利用法，また実際にツールを使用している企業の事例報告など，その内容は多岐にわたる．コードクローンセミナーは現場の生の声が聞けるという利点があるが，頻繁に行う

図1　コードクローンセミナーの様子

*Yoshiki HIGO
1980年4月生
大阪大学・大学院情報科学研究科（2006年）
現在、大阪大学大学院情報科学研究科
助教　博士（情報科学）　プログラム解析，ソフトウェア保守
TEL：06-6879-4112
FAX：06-6879-4114
E-mail：higo@ist.osaka-u.ac.jp

のは難しい．また，全てのユーザに出席していただくことも現実的ではない．このことから，メーリングリストを運用している．利用者はメーリングリストを用いてツールの利用法に関する質問や新機能の要望などを行い，また開発者は，ツールのバージョンアップや時期セミナーの開催日程などを利用者に告知している．

４．産業ソフトウェアの分析事例

ここでは，開発したツールを，情報処理推進機構（IPA）ソフトウェア・エンジニアリング・センター（SEC）の先進ソフトウェア開発プロジェクトにおいてベンダー５社が開発しているプローブ情報システム（プローブ情報システムとは，センサーなどの計測機器が収集したデータを，ネットワークを通じて中央センターに転送し，その情報を分析・蓄積・加工などすることによってさまざまな有用な情報を提供するシステムである）に対して適用した結果を紹介する（実験の詳細は論文 [1] を参照されたい）．このプロジェクトでは，各社は個別に開発を行っており，プロジェクトマネージャでもソースコード，開発工数，開発体制（外部委託先，要員数等）に関しては知りえない状況であった．プロジェクトマネージャが主催する進捗会議では，各社のマネージャから各工程の進捗割合と，予定日数のずれが報告されるのみであった．このようなブラックボックスとなっているソースコードの特徴を把握するためにコードクローン分析を行った．

コードクローンの検出は単体テスト終了後，結合テスト終了後の２度行った．全社合計の開発規模は数十万ステップであり，結合テスト後は単体テスト後に比べ約２万ステップ増加していた．このシステムは C/C++ 言語を用いて開発されている．コードクローン分析は各社が開発したソースコードに対して個別に行った．紙面の都合上，全ての分析結果を示すのは困難であるので，一部分のみを紹介する．なお，4.2 ～ 4.3 節の各分析は，結合テスト後のコードに対する適用結果である．

４．１．履歴分析

単体テスト後と結合テスト後のクローン検出量の変化を調査した．表１は単体テスト後，結合テスト後の検出された重複コード片の数とシステムの重複度を表している．Ｙ社の開発部分に，単体テスト後に比べて結合テスト後に多くのコードクローンが含まれていた．通常，単体テスト後には新規で実装される機能はないため，単体テスト後と結合テスト後のクローン検出量にあまり差はない，と予測していた．Ｙ社が開発したファイルの重複度分布状態のグラフを図２に示す．この図からわかるように，結合テスト後は重複度が非常に高いファイル（81%～100%）の数が増えていた．Ｙ社の開発者にインタビューを行ったところ，これらのファイルは，結合テストを行う前に，新しい機能を実装するために社内で構築したライブラリコードを流用した部分であった．ライブラリのファイル間で多くのコードクローンを共有しているため，重複度は非常に高いが，これまでに数多くのソフトウェアで運用されており，信頼できるコードであるとの回答を得た．

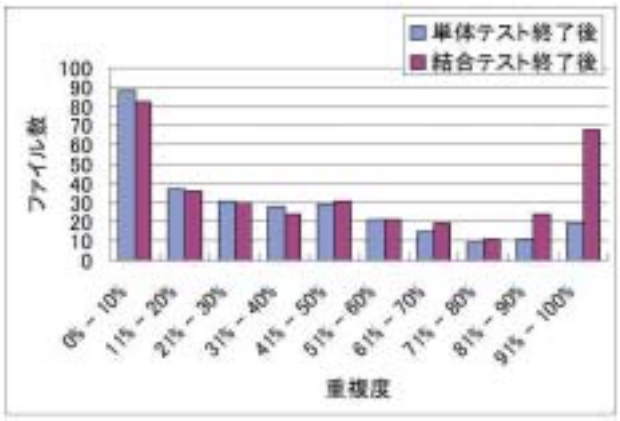

図２　Ｙ社の重複度の変化

表１　検出されたコードクローンの量

	単体テスト後		結合テスト後	
	コード片数	重複度	コード片数	重複度
Ｖ社	259	33.9%	259	33.4%
Ｗ社	369	27.3%	379	26.2%
Ｘ社	4,483	55.3%	4,768	50.8%
Ｙ社	6,747	42.6%	7,628	46.0%
Ｚ社	2,450	56.2%	2,505	56.3%

[1] 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 年 2 月

４．２．特徴的なコードクローンの分析

４．２．１．大きなコードクローン

ある社のソースコードから検出された最長のコードクローンは長さが154行であった．このコードクローンはそれぞれ，AAAXXXBBB.cppとAAAYYYBBB.cppというファイルに含まれていた．AAAYYYBBB.cppのコードクローンで呼び出されている関数の名前の一部がXXXであり，また該当箇所のコメントにもXXXが含まれていた．このことから，このコードクローンは，AAAXXXBBB.cppからAAAYYYBBB.cppにコピーアンドペーストされ，その修正を忘れたのではないかと思われる．

４．２．２．同形コードの多いコードクローン

全社においてもっとも同形コードが多かったコードクローンは，データの整合性をチェックし，もし間違っていればエラーを出力する，という処理の部分であった．データの種類は各社間で違いがあるものの，処理内容は全社で共通していた．

４．２．３．多くのファイルにまたがって存在するコードクローン

ある社のサブシステムから検出されたコードクローンは8個のソースファイルにまたがって存在していた．このコードクローンは，文字列の終端にNULLがあるかをチェックし，もしなければ追加する処理を行っている関数であった．関数全体が重複しているため，ユーティリティパッケージなどへの移動などで，簡単に集約可能である．

４．３特徴的なファイルの分析

４．３．１．多くのコードクローンを含んでいるファイル

ある社が開発したサブシステムに，358個のコードクローンを所有しているファイルが存在した．コードクローンは特定の処理をしている部分に集中しているわけではなく，さまざまな部分がファイル内クローンもしくはディレクトリ内ファイル間クローンになっていた．とくに問題があると思われるコードクローンは特定されなかったが，このファイル事態が非常に多くの処理を行っており，その保守性が疑われた．

４．３．２．重複度が高いファイル

ある社のサブシステムに重複度が96％であるファイルが2つ検出された．一方はある処理をオンライン時に行い，他方は同様の処理をオフライン時に行うファイルであった．開発者にこのコードクローンの存在を報告したところ，設計時からオンライン時とオフライン時のコードは別々に実装すると決めており，存在が確認されているコードクローンであった．

４．３．３．多くのファイルとコードクローンを共有しているファイル

ある社のサブシステムに，他の13個のファイルとコードクローンを共有しているファイルが存在した．このファイルにはさまざまな入力処理が含まれており，それぞれの処理が別ファイルとコードクローンになっており，結果として13個ものファイルとコードクローンを共有していた．コードクローンになっている部分は，対象ソフトウェアで汎用的な処理であり，処理内容も単純であるため，特に問題ないと思われた．

５．おわりに

このコラムでは，筆者の産学連携の取り組みと，産業ソフトウェアに対するコードクローン分析の適用事例を紹介した．今後も実際のソフトウェア開発現場で役立つような実用性の高い研究を進めていきたいと思う．