EASE PROJECT
NEWS LETTER
Volume 4
2005 年 12 月発行

エンピリカルソフトウェア工学の研究と実践　―コードクローンを例に―

大阪大学　大学院情報科学研究科　教授　井上 克郎

１．はじめに

今までに何度かニュースレターの記事としてコードクローンの話が取り上げられている。今回は、このテーマに我々が関心を示すきっかけ、産学連携の様子、今後の発展に関して、比較的自由な観点で述べる。

２．コードクローンとは

コードクローン（以降単にクローンと呼ぶ）に関して、近年、国内外で研究成果が報告されるようになってきており、その実用性も注目されるようになりつつある。しかし、クローンとは何か、を明確に定義しているものは少ない。

「クローンとはソースコード中に存在する同一、または類似したコード片のことである。」これは、ある学生がとある原稿で使用した定義であるが、同一や類似の対象が不明で定義になっていない（指導が悪い）。「同一または類似したコード片を有するコード片のことをクローンと呼ぶ。」だいぶましになっているが、より厳密に定義しようとすると苦労する。囲みの定義は、系列の問題としてクローンを定義している。ここで「類似」は、単に空白やコメントの除去、異なる識別子の吸収などを意味する場合が多いが、明確簡潔に表記することは難しく、実際のツールの動作で規定することになる。

図１に、クローンの簡単な例を示す。太字の３行から８行と 111 行から 116 行のコード片はお互いクローン関係であり極大クローンである。色の付いている語には差異があるが、識別子の違いでありこれらの違いは吸収して類似したものとしている。

このようなクローン検出を同一ファイル内や他のファイルとの間で行うと、通常のプロダクトの分析とは一風変わった結果が得られる。図２に３つの Unix OS のカーネルソースコードに対してクローン検出を行った結果の図（散布図と呼ぶ）を示す。縦軸と横軸はそれぞれ各ファイルを連接した行で、各点は 30 トークン以上の長さをもつクローンである。この図は、原理的には対角線に必ず点が存在し、対角線に対して線対称になるが、ここで

```
1. static void foo() throws RESyntaxException {
2.   String a[] = new String [] { "123,400", "abc", "orange 100" };
3.   org.apache.regexp.RE pat = org.apache.regexp.RE("[0-9,]+");
4.   int sum = 0;
5.   for (int i = 0; i < a.length; ++i)
6.     if (pat.match(a[i]))
7.       sum += Sample.parseNumber(pat.getParen(0));
8.   System.out.println("sum = " + sum);
9.    System.out.println("end"); }
   ...
110. static void goo(String [] a) throws RESyntaxException {
111.   RE exp = new RE("[0-9,]+");
112.   int sum = 0;
113.   for (int i = 0; i < a.length; ++i)
114.     if (exp.match(a[i]))
115.       sum += parseNumber(exp.getParen(0));
116.   System.out.println("sum = " + sum);
117. }
```

クローンとクローン検出：

ある系列中に存在する２つの部分系列α、βが同一または類似しているとき、C（α，β）と書き、αはβのクローンである言う。通常、Cは、反射、推移、対称律が成り立つ同値関係とする。また、αの同値類をαのクローンクラスと言う。

任意のα、βに対してC（α，β）ならば、αの任意の部分系列α'に対し、C（α'，β'）となるβの部分系列β'が存在する。また、α、βをそれぞれ真に含む任意の系列α''、β''（ただしα''≠β''）に対してC（α，β）かつ¬C（α''，β''）ならば、（α、β）を極大クローンペアと呼ぶ。ある部分系列αに対し、別の部分系列βがαと極大クローンペア（α、β）を構成するとき、αもしくはβを単に「クローン」と呼ぶ。

系列Sが与えられたとき、S中の極大クローンペアを全て発見することを、クローン検出と言う。通常、クローン検出ツールと呼ばれるツールは、これを目的としている。ただし、ある一定の長さ以上の極大クローンペアのみを出力するようにするのが普通である。短いクローンは、多数発見されることが多いが、その意味や存在には、興味のない場合が多いからである。

図１　クローンの例

は、対角線上の点や右上半分の点は省略している。この図を見れば分かるように、先祖を同じとするFreeBSDとNetBSDとの間には多数の点があり、Linuxとそれらの間には点が少ない。このように、プロダクトの間の関連や歴史を知ることができる。

この他、クローン分析はいろいろな用途がある。ソフトウェアのデバッグ、保守、分析（履歴、依存性、類似性等）、著作権管理、剽窃検出などに利用されている[1,2]。

３．研究のきっかけ

クローン解析のきっかけは、某大手ソフトウェア会社D社との雑談である。何十年もメンテし続けてきた巨大ソフトの管理が手に負えなくなった、特に、コピーされたモジュールが散在しており、一つの変更がどこに影響を与えているか､どこがコピーされたものか分からず困っている、というものだった。この話は、1999年に、私が併任していた奈良先端大で聞いたのだが、これを阪大で当時、博士２年の学生だった神谷年洋君（現・産総研）に話したところ面白そうだということでやってみることになった。同時に、DNA塩基配列の問題を文字列の照合問題として研究している情報科学科の松田助教授（現教授）に、使えそうなアルゴリズムとして、サフィックス木とその教科書を教えてもらった[3]。

その後、神谷君は、字句解析と差異吸収規則を組合せ、サフィックス木でクローン検出を効率よく行う手法を考案し、驚異的なスピードで実装を始めた。そして実装開始から数週間でプロトタイプシステムが動き始め、デバッグを兼ねて、いろいろなソースプログラムにかけ始めた。日々、神谷君、楠本助教授（現教授）と共に、出力結果の検討を行い、このツールの切れ味の良さを確信した。ちょっといけそう、となると、あれも分析したい、これもかけてみたい、と要求がどんどん膨らんで、プログラムのチューニングをすると共に、CPUやメモリの増強に、がんがん資金をつぎ込んだ。そして、数ヵ月後の2000年初頭ごろには、図２のような散布図を出せるようになった。

この結果は非常に面白い、ぜひ論文に書こう、特許も書こうとなった。論文を書くにあたって、過去の同様の研究を調べてみると、海外で似たような動機のいろいろな研究が行われていることが分かってきた。ただ、それらは、方式が異なり、効率・精度のバランスが悪いものであった。それらと勝負しても勝てると思われたので、この分野での一番の論文誌であるIEEE TSE (Transactions on Software Engineering)に投稿することにした。2000年７月に初版を投稿し、翌年１月に修正要求、さらに８月に修正要求が来て、結局2001年９月に採録となって、実際に掲載されたのは2002年７月号であった[4]。特許は、2001年７月に科学技術振興事業団JSTを通じて出願手続きを行った[5]。

４．CCFinderの発展

当初、このツールはDupSと称していたが、途中からCCFinderと改名した[9]。最初は、Cプログラムにしか

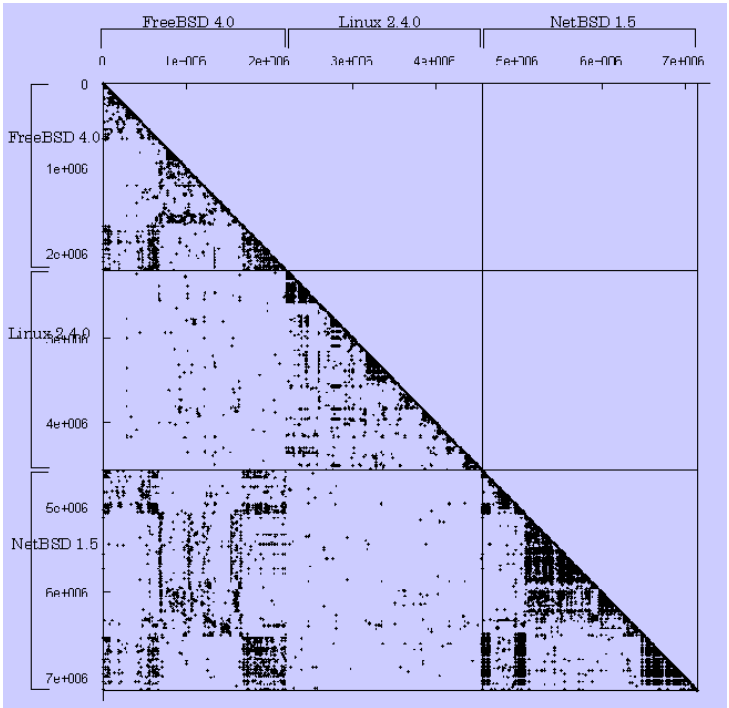

図２　３つのUNIX　OSのカーネルソースコードのコードクローンの散布図

対応しなかった。しかし、字句解析部を言語仕様に合わせるだけで別の言語に適用可能なので、今ではC, C++, Cobol, Java, Lisp などの多くのプログラミング言語や、テキストに適用できるようになった。

ある程度このツールが実用的になった段階で、きっかけとなったD社の資産に適用することにした。その結果、多くの予想されたクローンの確認と共に、予想外のクローンも発見され、この種の分析の重要性を再認識した[6]。

CCFinderはテキスト出力を主とした軽いツールを目指しているが、出力を直感的に理解できるようなGUIが普及に必須と考え、Geminiというクローン可視化用ポストプロセッサをM2の学生の植田泰士君（現JAXA）が開発してくれた[7]。さらに、クローンの除去やクローンをまとめるリファクタリングの支援ツールAriesを博士課程の学生である肥後芳樹君が作り現在も改良中である[8]。これら一連のツールの完成で、CCFinderの普及にも拍車がかかった。

5. CCFinderの普及

D社の資産への適用は前にも述べたが、その後、いろいろな会社で、評価用として使われるようになった。我々は、ツールの出来栄えは、それなりに自信はあったが、実際の現場で使いものになるか、大いに不安であった。そのため、企業といろいろな交流の場で、宣伝活動を行った。付き合いのあるいろいろな企業に行っては、売り込んで、使ってもらい、改良のきっかけとした。ただ、クローン分析のみを目的とした共同研究は、まだわずかである。これは、このツールの位置づけが、保守の一部の作業でのみ使われる、ということに起因してよう。面白そうだが、もっと他にやらなければならない重要なことがたくさんある、という感じである。

広報の一環として、プレスリリースを、IEEE TSEへの論文の掲載時期に同期して行った（2002年5月）。その結果、読売新聞と日経産業新聞に記事が掲載された。

これらの広報活動とは別に、きちんと技術的内容を説明し、ツールを実際に使ってもらう機会が必要と考え、「ソフトウェア工学工房」という枠組みを作り、その中で、コードクローンのセミナーを繰り返し行うことにした。

ソフトウェア工学工房とは、医学部の付属病院を参照したモデルで、ソフトウェア工学技術の普及と向上のため、大学と産業界が連携する場を設けて、そこでお互いのニーズや知識を交換し、現実の諸問題を解決することを目指している（図3）。

現在のEASEプロジェクトの千里ラボやドイツのフランホーファーIESE(Institute for Experimental Software Engineering)は、このモデルを実現したものと言えよう。実際にシステムを作る能力、資金はなかったので、交流の場としてのバーチャルな組織を目指した。結局、実践的なソフトウェア工学技術に関してセミナー開催を行うこととし、そのテーマとしてクローン技術を選んだ。2002年11月を第1回として、現在までに計5回、年に2度程度、セミナー開催を続けている。毎回20-50名程度の参加があり、実際にCCFinderや関連ツールを使用、技術を体験してもらっている（写真1）。この活動には、本EASEプロジェクトを始め、大阪大学のIT連携フォーラムOACISや情報科学研究科の21世紀COEプログラムのご支援を頂いている。

このような普及活動の成果もあり、現在、内外100組織程度でCCFinderが利用されるようになってきた。海外の研究機関では、クローン分析のツールとしては、CCFinderが標準的なものになってきており、ICSE(International Conference on Software Engineering)やFSE(Foundations on Software Engineering)などのソフトウェア工学関係の国際会議の多くの論文で引用されている。国内でも、試用、評価用として多くの組織で利用されており、商用利用を検討している組織もある。

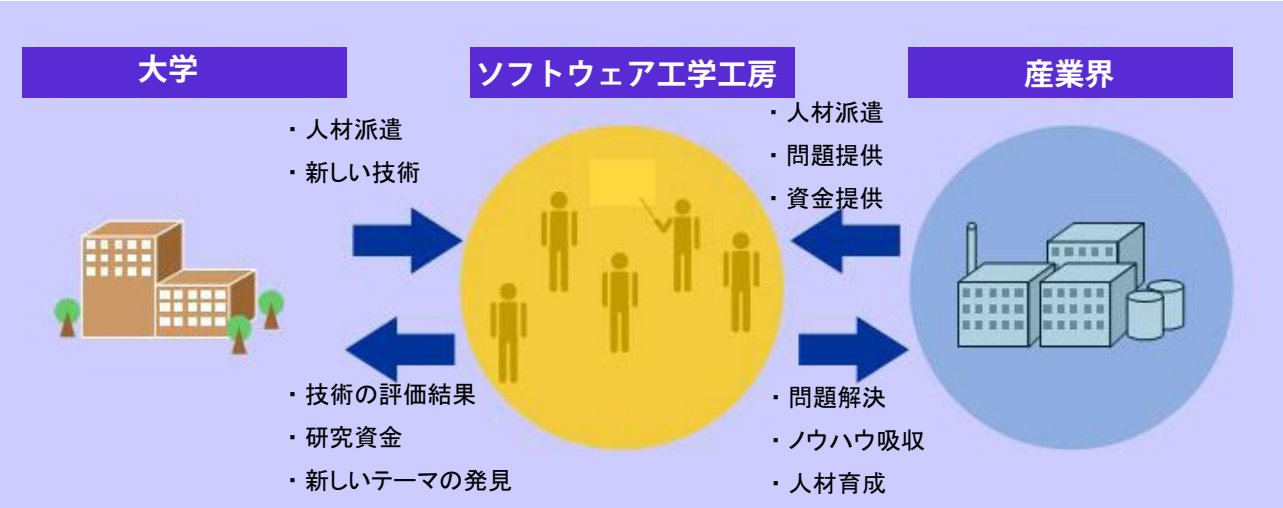

図3　ソフトウエア工学工房モデル

写真１　コードクローンのセミナーの様子

これらの利用例を見ると、おおむね、社内のプロダクトの評価に用いている組織が多いが、下請けからの納品物件のチェックにも有用であるとの話もある。また、大学のプログラミング演習の成果物の検査にも非常に有効である。また、2001 年 10 月の大阪地裁の織布情報作成プログラム事件判決では、鑑定に CCFinder が用いられたが、このように著作権のチェックにも利用できる。

現在 CCFinder は、特許の関係もあり、(Windows 上で稼働する) バイナリー形式のみの形態で、研究・評価・試験用システムとして、希望者に無償で配布している。前述のように、商用利用は JST からのライセンスが必要であるが、普及の妨げにならないよう、安価で簡単な手続きで利用できるようにお願いしている。特許申請などせず、オープンソースにしたほうが普及には良かったのでは､という声もある｡ただ､オープンソースにしてしまって、後は知らないよ、では、ソフトウェアの普及は難しい。我々は、いろいろなバグ修正要求や機能拡張要求に対応してきた。権利確保と普及のバランスをどうとるかは、非常に難しい問題で、今でもどうしたら良かったか、これからどうすれば良いのか、自問している。

６．今後について

神谷君は、産総研勤務の下、IPA の未踏ソフトウェアに応募し、CCFinder の次期版 CCFinderX を作成した [10] (彼はこの成果で天才プログラマー／スーパークリエータの称号を得た)。このツールでは、字句解析部をユーザ定義可能な形にして、いろいろな言語に容易に対応できるようにしている。今まで、多種多様な言語への対応要求があり、その一部しか実現できていなかったが、これで、ユーザ自身で、カスタマイズして、適用範囲を広げることができるようになった。また、Eclipse 上で稼動する Gemini の後継 GUI も開発中である。

EASE プロジェクトでも、EPM で蓄積されたデータに対して、CCFinder を適用して、分析する試みを行っている。通常のプロセスやプロダクト分析では得られない結果が出る、などと興味を持っていただける機会も増えてきた。今後、EASE の一連の分析と連携して、対象プロジェクトの特性分析の重要な位置を占めるようになると思われる。これからは、ツールの発展だけではなく、分析の手順や手法の標準化、分析結果の蓄積と評価法の確立などが必要になろう。

クローンに関しては、研究と実践が比較的うまく噛み合った例であろう。今までに、我々の研究室では多くの論文、国際会議をこのテーマで発表してきた。現在も多くの学生がこのテーマに関わっており、今後も研究を継続していくものと思われる。このように、研究活動としては十分満足すべき結果であろう。

一方、実践的な見地からすると、まだまだやるべきことは多い。大学の教員や学生の片手間で、プレスリリース、セミナーなどを行ってきているが、個別の案件に関して、余りフォローできていない。例えば、もう少しマンパワーがあって、営業的な活動ができれば、いろいろな企業のクローン分析をきっかけに、プロセス改善や品質向上の提言ができるようになると思われる。しかし、その実施は現在の大学の枠内では容易ではない。

より自由な組織、例えば、ベンチャー会社、または、ソフトウェア工学工房のような組織において、企業に対してより積極的に分析、提言をやっていければ、普及により一層、弾みが付くこととなろう。

文　献

[1] 神谷　年洋：〝コードクローンとは，コードクローンが引き起す問題，その対策の現状〞，電子情報通信学会誌，平成 16 年 9 月号，pp. 791-797，2004.
[2] 井上克郎，神谷年洋，楠本真二：〝コードクローン検出法〞，コンピュータソフトウェア，Vol. 18，No. 5，pp. 47-54，2001.
[3] D. Gusfield, Algorithms on Strings, Trees, and Sequences, pp. 89-180, Cambridge University Press, 1997.
[4] Toshihiro Kamiya, Shinji Kusumoto, Katsuro Inoue: 〝CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code〞, IEEE Transaction on Software Engineering Vol. 28, No. 7, pp. 654-670, 2002.
[5] 特願 2001-214037：同形パターン検出システム，2001.
[6] 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.
[7] 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 D-I，Vol. 86-D-I，No. 12，pp. 863-871，2003.
[8] 肥後 芳樹，神谷 年洋，楠本 真二，井上 克郎：〝コードクローンを対象としたリファクタリング支援環境〞，電子情報通信学会論文誌 Vol. J88-D-I，No. 2，pp. 186-195，2005.
[9] CCFinder，http://sel.ics.es.osaka-u.ac.jp/cdtools/index.html
[10] CCFinderX，http://www.ccfinder.net/index-j.html