システム開発における OSS と XML に関する検討

A Study on OSS and XML in System Development

前田和昭

中部大学　経営情報学部

Kazuaki Maeda,　Email: kaz@acm.org

College of Business Administration and Information Science

Chubu University

概要

ビジネスのいろいろな分野で OSS（オープンソース・ソフトウェア）と XML が注目され，積極的な利用が進んできている．本稿では，ソースプログラムから設計情報を抽出する逆エンジニアリングツールの開発の経験を踏まえ，OSS の再利用とその再利用のための XML による情報出力について述べる．

1　OSS について

最近，オープンソース・ソフトウェア（以後，OSS と略記する）に関する書籍や報告書が数多く出版され，マスコミでも話題になることが多くなってきた．日本国内では，例えば，2003 年 1 月に日本総合研究所による報告「オープンソースソフトウェアのセキュリティ確保に関する調査」が Web 上で公開され [1]，続いて 2003 年 8 月には，経済産業省の企画のもとでまとめられた報告書「オープンソース・ソフトウェアの現状と今後の課題について」が Web 上で公開された [2]．また，2003 年 3 月に 1 回目が行われた Asia Open Source Software Symposium[3] の開催回数はすでに 6 回に及んでいる（2005 年 10 月現在）．今までビジネスには縁遠かった OSS が，ようやく表舞台で活動を始め，全世界でソフトウェアビジネスが変わろうとしているように思える．

現在の一般的な商用ソフトウェアでは，オブジェクトコードのコピーのみが利用者に渡されている．企業が高価な人件費をかけて作り上げたソースコードの中には，社外には漏らしたくない重要なアイディアや，生産性を上げるために長年に渡って苦労して開発したプログラム部品が含まれる．したがって，競合する他社に対して優位性を維持するためには，ソースコードを隠しておくのが当然とされてきた．

OSS は，これまでのビジネスの慣例とは全く異なり，オブジェクトコードのコピーを無料（または安価）で公開し，かつ，ソースコードを無料公開する．OSS を入手すると，以下の a〜f が可能になる．これら a〜f では，OSS の活用が容易な順に並べた．a の場合は，そのままインストールするだけなので比較的容易であるが，f の場合は，かなりの知識・経験・時間を使った複雑な作業が必要となる．

a. ソフトウェアを無料で使用する

最近のコンピュータ関連雑誌には CD-ROM が添付されていることが多く，OSS が付録としてついてくる．例えば，Linux, OpenOffice.org, Eclipse などが CD-ROM 中に入っているので，これらを自宅のコンピュータにインストールすれば，ワープロやソフトウェア開発ツールを無料で使用することができる．また，知人に OSS のコピーを渡すことが許されていることも多い．

b. ソースコードを閲覧する

熟練したソフトウェア開発者が記述したソースコードには，貴重なノウハウがたくさん詰まっている．そのため，昔からソースコードはプログラミングの教科書であると言われ，実力を向上させるために重要な役割を果たしてきた．

c. 不足した機能を追加する

ソフトウェア開発者は，不便な現状を改善して自分の生産性を上げるために，ソフトウェアを開発することが多い．もし，他人が開発した便利なソフトウェアをソースコードと一緒に入手できたとして，自分にとって足りない機能があれば，その機能を自分で作成・追加して，さらに使い続けることができる．

d. プログラムを再利用する

優れた OSS の開発活動が止まってしまっても，ソースコードが OSS として公開されていれば，それまでに開発されたソフトウェアを引き継いで，OSS の開発を再び立ち上げることができる．例えば，Web/メール専用サーバ機を販売していた Cobalt 社の例がある [4]．Cobalt 社の製品は，小型筐体のデザインが魅力的で，システム管理を簡素化するために独自で開発した優れたソフトウェアが搭載され，サーバ管理者などに人気があった．その後，2000 年に Sun Microsystems 社が Cobalt を買収したが，自社内のビジネス戦略のため 2003 年に販売中止を決定した．Sun Microsystems 社は，Cobalt で人気があった Web ベースの管理システムのソースコードを一般に公開したため，その後ユーザグループがプロジェクトを引き継いでいる．

OSS の一部分だけであっても現在進行中の製品開発プロジェクトにうまく取り込み再利用ができれば，プログラム開発の工数を減らすことができる．再利用可能なライブラリとして OSS を提供している場合は問題ないが，それ以外の場合に OSS の一部分だけを取り出し再利用する作業は非常に難しい．

e. 別のプラットフォームに対応する

プログラムを開発するときには，ある特定のプラットフォーム（例えば Linux）上で開発とテストを行う．開発の途中もしくは開発後に，そのプログラムを他のプラットフォーム（例えば Windows）に移植することで複数のプラットフォームに対応したプログラムができあがる．

f. 障害に対応する

商用で販売されている製品であっても，時々正常とは思えない動作をすることがある．「不正アドレスをアクセスしました」というパネルが出たり，画面が突然青くなったり，最悪の場合コンピュータの反応が全くなくなることさえある．商用のプログラムのソースコードは非公開となっているため，どんなにコンピュータに詳しいプログラム開発者であったとしても，その障害を取り除くことは極めて難しい．しかし，ソースコードが公開されている場合，知識と経験豊富なプログラム開発者であれば，その原因を調べ，ソースコードを修正することで障害を取り除くことができる．また，インターネット上でウィルスやワームの活動が確認される前に，トラブルの元になる部分を発見し修正することも，非常に難しい作業であるが可能である．

本稿で述べる逆エンジニアリングツールを作成するときには，その過程で上記 c と d に相当する作業について考えながら作業を進めてきた．

2 XML について

インターネットの発達とコンピュータの飛躍的な性能向上を背景にして，XML (Extensible Markup Language) を使ってデータを表現することが多くなってきた．XML は，データに意味を付記することができるように SGML から派生したマークアップ言語であり，XML で記述されたデータである XML 文書は，人間が読めるようにテキスト形式で表現される．WWW の普及の立役者となった HTML にはない機能をも備え，電子商取引などビジネスで広く活用できるようにいろいろな工夫がなされている．

XML がプラットフォームに依存しないという特徴を使うことで，異種のプラットフォーム（コンピュータ，オペレーティングシステム，プログラミング言語）間でのデータ交換が可能となる．例えば，アプリケーション間でデータを交換したり，データベースからデータを取り出して他の製品で活用したり，またネットワークを介してデータ交換する場面など多くの応用分野が広がってきている．

XML 文書を使い方で分類すると，

- 文書中心 XML 文書 (document centric documents)
- データ中心 XML 文書 (data centric documents)

のどちらかになる [5, 6]．文書中心 XML 文書は，主として人間が読むために作られるもので，規則性が少なく，データの粒度が粗い特徴を持つ．例えば，本や記事や電子メールなどを記述したものがそれにあたる．これに対して，データ中心 XML 文書は，コンピュータでの処理やデータベースに格納するためのもので，規則性があり，データの粒度が細かい特徴を持つ．例えば，文献データや注文伝票などを記述したものがそれにあたる．

本稿で XML 文書について述べる場合は，データ中心 XML 文書のこととする．以下，データ中心 XML 文書のことを，単に XML データと呼ぶことにする．

3　逆エンジニアリングツール

この節では，現在開発を進めている C#プログラムを対象にした逆エンジニアリングツールについて述べる．逆エンジニアリングツールを作成するにあたり，OSS の一部分を再利用すること，また，再利用のために OSS 内部の情報を XML を使って出力することについて検討を行った．

3.1　背景と動機

ソフトウェアの設計情報と実装後のプログラムが一致しない場面をよく見かける．十分に吟味してプログラムを設計・実装したとしても，設計情報を修正しないままソースコードだけを修正することがあると，後の保守で面倒な作業が必要になる．その解決策として，ソースコードから設計情報を生成する逆エンジニアリングツールがいくつか提案されている．

逆エンジニアリングツールの基本的な機能の一例として，ソースコードからクラス図を生成する機能を考えることができる．例えば，図 1 の C#プログラムからは図 2 のクラス図が生成できる．

```
class A {
    public void method1(char c) { }
}
class B : A {
    private void method2(char c) { }
}
class C : A {
    private void method3(char c) { }
}
```

図 1　簡単な C#プログラム

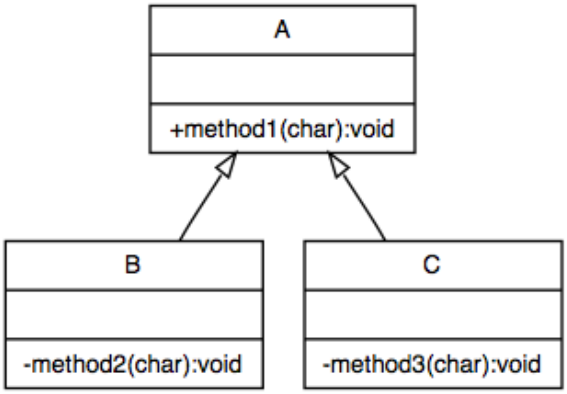

図 2　クラス図

逆エンジニアリングツールを作成するには，複雑な作業が必要となる．ソースコードを解析する逆エンジニアリングツールの作成には，コンパイラのパーザに相当するプログラムが必要である．ところが，最近のプログラミング言語の言語仕様は複雑であり，その言語仕様に従ってパーザを作成するにはかなりの工数を必要とする．さらに，プログラミング言語として人気のある C# や Java は，時が経過するにつれて言語仕様がどんどん拡張されていくため，その時々の言語仕様に準拠するようにパーザを保守していくことは手間のかかる作業となる．

別のプロジェクトで開発されたパーザを再利用することは難しい．パーザを作成する工数を抑える方法として，ソースコードを公開しているコンパイラのパーザ部分の再利用を考えることができる．一般的に，コンパイラの規模は比較的大きく，また，コンパイラ内部ではいろいろな処理が複雑に絡み合っているため，修正を加えて再利用可能なパーザだけを正しく抜き出すことは，非常に難しい．また，コンパイラ内部で保持される情報を別目的に活用したいと考えても，コンパイラを動作させることを考えて設計されているため，難しい作業になる．

再利用したいソフトウェアが GPL の下でソースコードを公開している場合，その取り扱いに注意が必要となる [7]. GPL の下で公開しているソースコードの一部からパーザを抜き出し，そのパーザを逆エンジニアリングツールの中に埋め込んだ場合，作成した逆エンジニアリングツールのソースコード全てを公開する義務が発生しかねない．もし，逆エンジニアリングツールを製品として販売する場合にソースコード公開の義務が発生すると，ビジネスを進める上で支障をきたす可能性がある．

そこで，C# コンパイラ mcs でパーザ生成系 jay が使われていることに着目し，jay を修正して再利用可能なパーザを生成可能にし，mcs を修正して再利用可能なパーザを動かすための情報を出力することを考えた．C#コンパイラ mcs は，.Net の開発実行環境を OSS で提供することを目的とした Mono プロジェクト [8] の一部である．mcs で使われている jay は，Java のために開発された jay[9] を C#対応に修正したものである．本稿で述べる mcs と jay に対する修正は，パーザを再利用可能にするためのものであり，逆エンジニアリングツールの作成に特化したものではない．mcs と jay の出力だけを逆エンジニアリングツールに取り込むことになるので，mcs と jay が GPL の下で配布されていても，完成した逆エンジニアリングツールは GPL の影響から逃れることができる．

3.2 逆エンジニアリングツールの作成

jay は，プログラミング言語の文法定義を読み込み，その文法定義にしたがってソースコードを解析するパーザを生成する．jay が生成するパーザは，スキャナを呼び出して字句をひとつずつ読み込みながら解析を進めていく．機能を追加した jay（以下，mjay と記す）が生成するパーザは，この解析のトレース情報を XML データとして出力する．

mcs に修正を加えた C#コンパイラの構成を図 3 に示す（修正部分を灰色で表している）．mjay は，未修整の C#文法定義を読み込み，C#パーザと簡易版 C#文法定義を生成する．簡易版 C#文法定義は，オリジナルの C#文法定義内に埋め込まれた不必要な定義を削除したもので，逆エンジニアリングツールのパーザを作成しやいようになっている．mjay が生成した C#パーザは，コンパイル対象の C#プログラムを読みながら，パーザの処理に関するトレース情報を XML データとして出力する．このトレース情報出力機能を制御するために，オリジナルの mcs の主プログラムの一部を修正した．

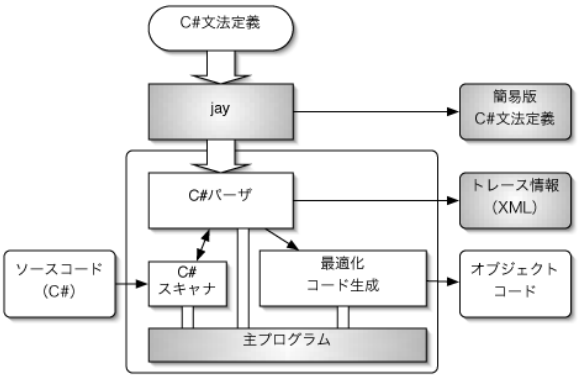

図 3　修正した mcs の構成

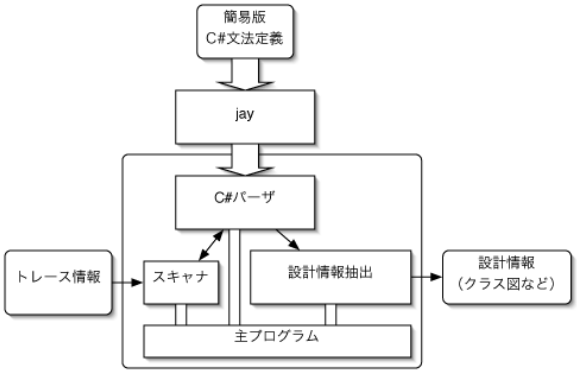

図 4　逆エンジニアリングツールの構成

図 4 に，逆エンジニアリングツール用パーザの生成と，逆エンジニアリングツール内部の構成を示す．図 4 の中の C#パーザは，mcs が出力したトレース情報をスキャナを介して読み込みながら，図 3 にある mcs のパーザの動きとほぼ同じ動きを再現する．GPL で保護されているソースコードは，図 3 に示すコンパイラ内に閉じこめることができるので，GPL が逆エンジニアリングツールに影響を及ぼすことはない．また，C#の言語仕様が拡張されたときは，Mono プロジェクトのコミュニティが，拡張された仕様を満足する mcs を実装してくれれば，新しい簡易版 C#文法定義をいつでも入手することができる．

4 おわりに

本稿では，逆エンジニアリングツールを作成するときに考えた OSS 再利用のための一例を紹介した．これは，OSS 再利用のための一般的な方法を追求する活動のはじめの一歩にすぎない．OSS の良さは，無料で使用できることだけではない．ソースコードを無料公開することで，OSS に関与する人の数を増やし，OSS を土台にしたソリューションを増やすところにその良さがある．OSS の活動が活発になってきているアジアに住んでいるからには，OSS に対して何らかの貢献をしていきたいと考えている．

参考文献

[1] 情報処理振興事業協会, オープンソースソフトウェアのセキュリティ確保に関する調査報告書 (2003).
[2] 経済産業省, オープンソースソフトウエアの利用状況調査／導入検討ガイドラインの公表について (2003).
[3] Asia Open Source Software Symposium, *http://www.asia-oss.org/* .
[4] Startup Story of Cobalt, *http://cobaltqube.org/story/* .
[5] Akmal B. Chaudhri, and others, XML Data Management, Addison Wesley (2003).
[6] Ronald Bourret, XML and Databases, *http://www.rpbourret.com/xml/XMLAndDatabases.htm* .
[7] GNU プロジェクトの思想, *http://www.gnu.org/philosophy/philosophy.ja.html* .
[8] Main Page - Mono, *http://www.mono-project.com/Main_Page* .
[9] jay Homepage, *http://www.informatik.uni-osnabrueck.de/alumni/bernd/jay/* .