B-010

XML によるプログラム表現とそれに基づく統合ソフトウェアテスト支援環境の構築
Proposal of the Integrated Software Testing Support Environment based on the XML-based source code representation

佐々木 亮太† 末広 暁久† 芳賀 博英†
Sasaki, Ryota Suehiro, Akihisa Haga, Hirohide

1. はじめに

近年の IT 社会において，ソフトウェアの品質が保証されていることが厳しく求められている．そのため，品質管理は企業にとって至上命題である．しかし，開発費用と期間は有限であるため，ソフトウェアテストを十分に行い，ソフトウェアの品質を保証できているプロジェクトは少ない．そこで，本研究室では，上記の問題を解決する統合ソフトウェアテスト支援環境 IST（Integrated Software Testing Support Environment）[1]の開発を進めている．

IST を構築するための第一歩として，様々な言語のソースコードを，汎用性の高い形式に変換する必要がある．そこで，汎用性の高い形式のひとつとして XML（eXtensive Markup Language）[2]に着目し，様々な言語のソースコードを XML に変換するための手法を開発し，Ansi C で書かれたソースコードを XML 化するコンバータ SC2ML（Sorce code of C to Markup Language）を実装した．

本稿では，パーサジェネレータのひとつである SableCC[3]を用いて，様々な言語に対応することができ，オープンソースとして二次利用可能な，プログラム XML 化ツールの構築を目的とする．

2. 統合ソフトウェアテスト支援環境 IST

本研究室で開発を進めている IST の概念図を図 1 に示す．以下の①から③の手順でソフトウェアテストの効率化を図る．

① プレーンテキストで記述されたソースコードの構文情報を XML で記述する．
② XML 化されたソースコードを XML-DB に保存する．
③ XML ソースコードを対象としたソフトウェアテスト支援アプリケーションに適用する．

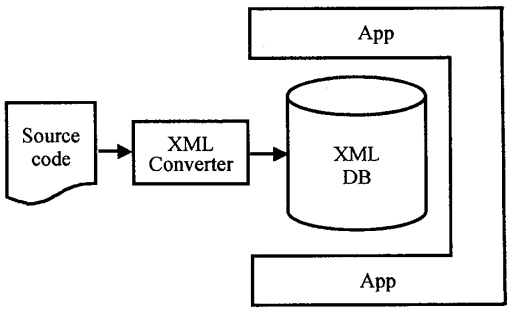


図 1 IST の概念図

† 同志社大学大学院工学研究科
GraduateSchool of Engineering Doshisha University

3. プレーンテキストソースコードの XML 化

XML Converter により，プレーンテキストで記述されたソースコードを XML 化するにあたって，本稿では導出木（derivation tree）に着目する．これは，ソースコードの解析情報における出力の主な形式であるため，ソースコードを XML 化する際に，ソースコードの解析情報を無駄なくシンプルに表現することが可能である．XML Converter について，以降，システムの概要を述べた後，本ツールの開発に利用したソフトウェア SableCC について述べる．

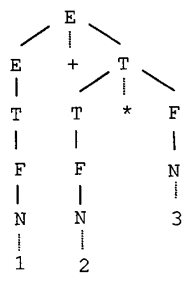
3.1 システムの概要

XML Converter の実行の流れは以下の①と②の手順である．四則演算の文法（図 2）に基づいた数式 1+2*3 の導出木を XML で表現したものを図 3 に示す．

① ソースコードから，生成規則に基づき，導出木を生成する．
② 生成された導出木を Top-down で辿る毎に，XML において，各ノードに対応したタグを追加していくことにより，導出の構造を XML で表現する．

```
E → E + T | E - T | T
T → T * F | T / F | F
F → N | (E)
N → 0|1|2|3|4|5|6|7|8|9
```

図 2 BNF 記法を用いた四則演算の定義




```
<E>
 <E>
  <T>
   <F>
    <N>1</N>
   </F>
  </T>
 </E>
 <orerator>+</operator>
 <T>
  ...
 </T>
 <operator>*</operator>
 <F>
  ...
 </F>
</E>
```

図 3 数式 1+2*3 の XML 表現

3.2 パーサジェネレータ SableCC の概要

SableCC は Java ベースのパーサジェネレータである．これは，BNF に基づいた文法ファイルを読み込み，その文法

で書かれたデータを解析するための Java または C#のソースを生成する．本稿では C#を使用し，利用するクラスを実装した．SableCC で生成される主なクラスは，字句解析器 Lexer と，構文解析器 Parser，Parser によって生成される Node の導出木，それを辿るためのインタフェース Analysis である．SableCC によるデータの解析手順は，まず，文法定義ファイルを読み込み，その文法で書かれたデータを解析するための Java または C#のプログラムソースを生成する．次に，それらの生成されたプログラムソースと，自分で作成した利用クラスを class ファイルにコンパイルする．最後に，実行時にそれらのオブジェクトに解析データを与え，解析する．以上の解析手順を図 4 に示す．

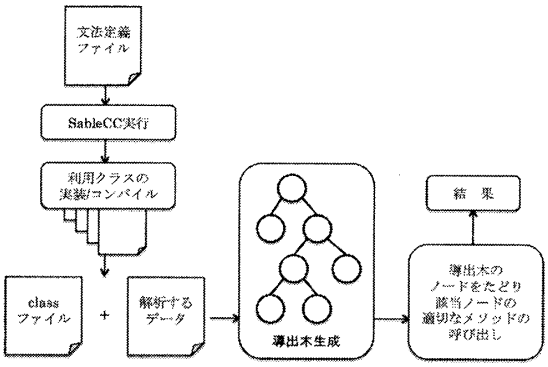


図 4　SableCC における解析手順

4. SCML の実装

Ansi C で書かれたプログラムソースの導出の構造を XML で表現する文書構造化言語 CML（C Markup Language）を考案した．本稿では，提案した手法を用いて，Ansi C で書かれたプロラグムソースを CML に変換する SC2ML を実装した．以降，SC2ML の概要を述べる．

4.1 SC2ML の概要

SC2ML は CML に変換するコンバータである．Ansi C のプログラムソースの例を図 5 に示し，それを SC2ML によって CML 化したソースを図 6 に示す．

```
int main(void){
  printf("FIT2010");
  return 0;
}
```

図 5　Ansi C プログラムソース

5. 考察

SableCC は自動的に生成された導出木をトップダウンで辿り，各ノードに対応した処理を行う．そのため，XML を扱う上で，子ノードに親ノードを追加できない問題に左右されない．さらに，導出木を自動的に生成するため効率的かつ短期間で XML Converter を実装できる．また，実装するあたって，SableCC は C#の利用するクラスを生成できるため，XML データ処理のためのクエリ言語である LINQ to XML などを活用できる利点もある．

6. おわりに

本稿では，導出木に着目し，ソースコードの XML Converter を構築した．本手法を用いることにより，様々な言語の XML Converter の実装が可能となるとともに，オープンソースとして二次利用が可能となった．よって，我々は IST 構築の第一歩を踏むことができた．今後は，様々な言語のプログラムソース XML Converter の実装はもちろんのこと，IST の構築を進めていく．

```
<?xml version="1.0" encoding="utf-8"?>
<Program>
 <AFile1File>
  <AFundefExternalDefinition>
   <ADeclspecFunctionDefinition>
    <AOnetypeSpecifierDeclarationSpecifiers>
     <AIntTypeSpecifier>int
     </AIntTypeSpecifier>
    </AOnetypeSpecifierDeclarationSpecifiers>
    <ANopointerDeclarator>
     <AParamtypelistDeclarator2>
      <AIdDeclarator2>main </AIdDeclarator2>
      <AOneParameterTypeList>
      …
       <AVoidTypeSpecifier>void
       </AVoidTypeSpecifier>
      …
      </AOneParameterTypeList>
     </AParamtypelistDeclarator2>
    </ANopointerDeclarator>
    <ANodeclFunctionBody>
     <AStlistCompoundStatement>
      <AMoreStatementList>
       <AOneStatementList>
       …
        <AIdPrimaryExpr>printf
        </AIdPrimaryExpr>
       …
        <AStringPrimaryExpr>"FIT2010"
        </AStringPrimaryExpr>
       …
       </AOneStatementList>
       <AJumpStatement>
       …
        <AConstPrimaryExpr>0
        </AConstPrimaryExpr>
       …
       </AJumpStatement>
      </AMoreStatementList>
     </AStlistCompoundStatement>
    </ANodeclFunctionBody>
   </ADeclspecFunctionDefinition>
  </AFundefExternalDefinition>
 </AFile1File>
</Program>
```

図 6　Ansic プログラムソースの CML 化

参考文献

[1] 末広暁久，”XML によるプログラムの表現とそれに基づく統合テスト支援環境の構築(1)ミューテーションテストのためのミュータントオペレータの実装”，電子情報通信学会（2010）
[2] “http://www.w3.org/TR/xml11/”
[3] ”http://sablecc.org/”