B-009

# XML に基づく C 言語用のマークアップコンバータの設計と開発手法
# Design and Development Methodology of C Language Markup Converter Based on XML

砥板 卓矢† 吉田 万輝‡ 芳賀 博英†
Takuya Toita Kazuki Yoshida Hirohide Haga

## 1. はじめに

近年，ソフトウェアの用途は多岐にわたり，高い品質が求められている．そのため，ソフトウェアテストの重要性が高まってきている．そこで，我々は IST(Integrated Software Testing Support Environment)と呼ばれる，ソフトウェアテスト支援環境を開発してきた[1]．IST ではプログラミング言語のソースコードを XML 形式で表現し，データベースに格納している．このデータベースの構築を容易にするために，ソースコードを XML 形式に変換するための変換系が必要である．Java 言語についてはすでに JavaML[2]と JJmlt[3]という DTD とソフトウェアが存在しているが，C 言語については我々が必要としている粒度のものは存在しない．そこで，C 言語用のマークアップ言語 CML(C Markup Language)の設計と処理系の開発を行った．開発は構文解析生成系である Yacc を用いた．

## 2. IST

IST(Integrated Software Testing Support Environment)は，ソフトウェアテスト支援環境である．図 1 に IST の概要を示す．

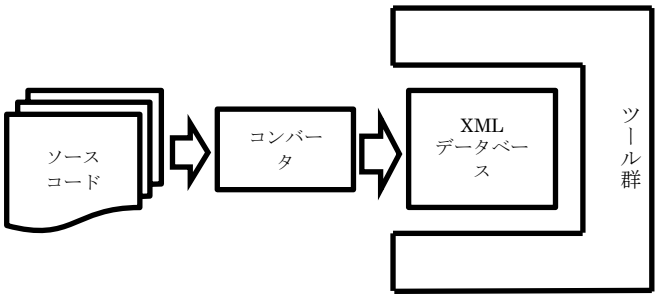


図 1 IST 概要

まず，ソースコードを XML 形式に変換し，データベースに格納する．各種のツールは構文情報を必要としており，ツールごとに毎回構文解析を行うのでは非効率なため，構文解析を済ませた状態で保存しておくためである．また，XML 形式は人間が読めることと，ソースコードに復元可能であるという利点もある．格納された XML 形式のプログラムに対して，開発されたツールを用いてソフトウェアテストを行う．なお，現在ツール群にはミューテーション解析ツールやコードクローンツールが含まれ，将来コードリーディングサポートツールやコードスライシングツールなどを我が研究室で開発していく予定である．

† 同志社大学大学院 Doshisha University
‡ 現在(株)クックパッド COOKPAD

## 3. CML の基本的な実装手法

変換方法は次の通りに設計を行った．構文規則は C 言語の規格として ISO 及び ANSI が定めた ANSI-C[4]を利用した．この構文規則に基づいて，構文解析を yacc(Yet Another Compiler Compiler)，字句解析を lex(Lexical Analyser Generator) で行う．この構文規則から抽象構文木を作成し,抽象構文木を走査することにより，C 言語のソースコードを XML 形式に変換した．

yacc のアクション・ルーチンに構文情報を格納する構造体を生成する関数を埋め込むことで, 構文規則から抽象構文木を作成する処理を実現している．図 2 に構文情報を格納する構造体の宣言を示す.

```
typedef struct Node{
  char name[100];
  char attr[100]; int type;
  int children;
  struct Node *child1, *child2,
              *child3, *child4;
}Node, *NP;
```

図 2 Node 構造体

構文情報を格納する Node 構造体はタグ名と属性情報の他に最大で 4 つの子要素を保持できる仕組みをとっている．CML は ANSI-C の構文規則に基づいて設計されており，Node 構造体も ANSI-C の文法規則に対応できる形に設計されている．Node 構造体を生成する Make 系の関数を作成し，yacc の文法にしたがって図 3 のように関数を埋め込む.

```
INITIALIZER:
  ASSIGNMENT EXPRESSION
    { $$ = MakeN1("INITIALIZER", $1); }
  | '{' INITIALIZER LIST '}'
    { $$ = MakeN1("INITIALIZER", $2); }
  | '{' INITIALIZER LIST ',' '}'
    { $$ = MakeN1("INITIALIZER", $2); }
```

図 3 yacc の文法例

生成された構造体は$$という特殊な関数に格納される．$$変数は還元した結果の意味値を格納する変数であり，$$に値を格納することで次の還元処理に値を引き渡すことができる．このような値の受け渡しによって構文情報が木構造の構造体に変換される．

## 4. XML のタグの生成

作成した抽象構文木を走査することにより，タグを生成する．走査は pre-order 順で行う．図 4 に基本的な二分木のタグ生成プログラムを示す．

```
void genTag(NP np){
  if(np==NULL)return;
  if(np->children==0)leafTag(np);
  else{
    preTag(np);
    genTag(np->child1);/*左のノード*/
    genTag(np->child2);/*右のノード*/
    postTag(np);}}
```

図 4　yacc の文法例

この genTag()は調べるノードがない場合は return，ある場合はタグを生成する．preTag()では開きタグを生成し，postTag()は閉じタグを生成している．このタグ生成プログラムは再帰的な構造である．調べるノードに子ノードがあれば，一つ目の genTag()で左のノードのタグを生成し，二つ目では右のノードのタグを生成する．

## 5. 冗長な構文木の修正

構文木作成に使用した ANSI-C の構文規則は一般的な C 言語の構文に対応するために，文法が詳細に定義されているので，文法構造が深く，このままタグを生成すると非常に膨大な量となる．これを避けるために，冗長と思われる構文部分を判別しまとめる作業を行う．図 5 にサンプルプログラムである return 文の構文解析木を示す．

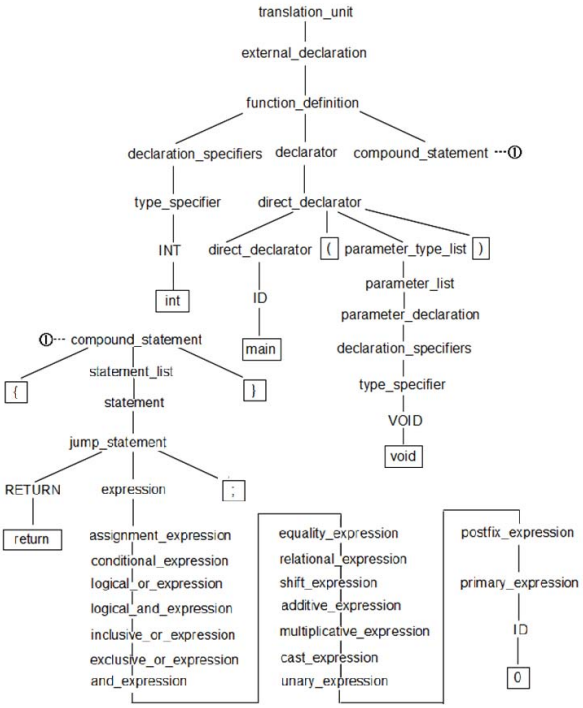


図 5　return 文の構文解析木

本論文で提案する手法では，ANSI-C の文法規則から作成した構文解析木の冗長な部分を削除していき CML の構文解析木を作成していく．図 6 に ANSI-C の構文規則に忠実に従って“expression”から文字“x”を出す場合の構文解析木を示す．

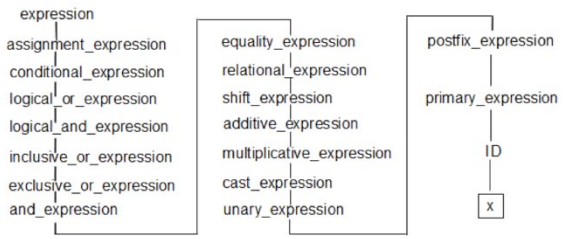


図 6　“x”の構文解析木

例えば，“return 0;”という命令では，”0”のような数字や文字等を構文解析木で表現する場合，構文解析木の“expression”から“primary_expression”までの構文規則は同じになっている．したがってこの構文規則は冗長であると判断し，省略することで修正を加える．このようにいくつかのプログラム構文解析木の比較を行い，冗長と判断した部分に修正を加えている．

## 6. おわりに

本論文ではソフトウェアテスト支援環境 IST のフロントエンドにあたる C 言語用のマークアップ言語 CML(C Markup Language)の定義と，変換系(converter)の設計と開発を行った．今回開発した CML の処理系は ANSI-C(C90)の構文規則に基づき設計を行い，構文解析に構文解析生成器 yacc，及び字句解析生成器 lex を用いた．ANSI-C の構文規則を我々の必要とする粒度まで簡略化した後，構文解析で得られた構文情報を一度構造体の木構造への変換を行い，トップダウンで木構造の走査を行うことにより木構造のタグを表示できるよう設計をし，コンバータを開発した．C 言語のほぼすべての文法を XML 形式で表現することができた一方，define によるソースコードの置換処理や include による外部ファイルの読み込みなどその他いくつかのプリプロセッサ処理への対応が実装されていない．大規模なソフトウェアはファイルが複数に分割されていることが一般的であるので，より大きなソフトウェアを解析するためにプリプロセッサ処理の対応について早急に検討する必要がある．

### 参考文献

[1] 末広 暁久，佐々木 亮太，芳賀 博英，”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提案”，電子情報通信学会技術研究報告.KBSE,知能ソフトウェア工学，110(61)，33-38(2010).

[2] Greg J. Badros，”JavaML: A Markup Language for Java Source Code”, http://www.badros.com/greg/papers/badros-javaml-www9.pdf，参照日時(2014/05/11).

[3] H.Aman ，”JJmlt”，http://se.cite.ehime-u.ac.jp/tool/JJmlt/ ，参照日時(2014/05/11).

[4] 一般社団法人情報処理学会，プログラム言語 C，http://www.jisc.go.jp/app/pager?id=478754，参照日時(2014/05/11).