5L-4

XML形式のソースコード構造モデル変換を用いたC言語用静的解析ツールの開発

堀江 佑太 † 福原 和哉 †† 今井 信太郎 † 新井 義和 † 猪股 俊光 †
† 岩手県立大学ソフトウェア情報学部 †† 岩手県立大学大学院ソフトウェア情報学研究科

1 はじめに

ソフトウェア製品の品質を確保する手法の一つとして，ソースコードの体系的な検査（査読）を行うコードレビューがある．コードレビューの作業は目視で行われることが多いため，担当者の能力によって品質に差が生じたり，レビュー作業に多くの人員や日数が費やされたりすることなどが，問題として挙げられている [1][2].

そのため，一定の品質を保つために特定の不具合になる可能性のあるソースコードを検索するための定型化された静的解析ツールや，解析を自動で行いレビュー作業の人員や日数の削減を目的とした静的解析ツールの研究・開発が行われている．しかし，市販されている静的解析ツールの多くは，決まった形式の不具合箇所を検出することを目的としているため，利用者がもとめる不具合が検出できるとは限らない．また，利用者自身が静的解析ツールに新規の不具合項目を追加することができないため，拡張性が低いという問題がある [3].

そこで，本講座では，不具合とされるコードをパターンで表す（これをエラーパターンとよぶ）こととし，エラーパターンからなる「パターンファイル」と「ソースファイル」を用意することで不具合を自動検出する静的解析ツールを開発した．しかし，独自の中間表現を用いているため第三者の管理・応用が難しいといった課題があった [4].

そこで本研究では，中間表現として汎用性の高いXML形式を用いたC言語用静的解析ツールを考案し，実装・評価を行う．

2 解析ツール

2.1 概 要

本研究で開発する静的解析ツールの構成を図 1 に示す．静的解析ツールは，入力されたソースコードの構文解析を行い XML データを出力する解析器，XML データを SXML データに変換する変換器，SXML データを用いて検査を行う検査器の 3 つから構成される．

エラーパターンは，パターン記号（2.2 参照）を組合せることで記述される．変換器によって出力された SXML 形式のデータとエラーパターンをもとに検査器でパターンマッチングが行われ，不具合とされる箇所が検出結果として出力される．

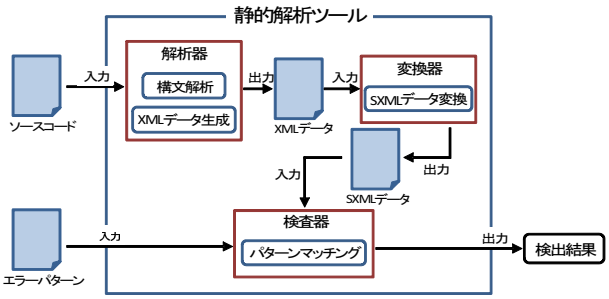

図 1: 静的解析ツールの構成図

- 解析器　図 2 のようにソースコードを解析して XML データを出力する解析器は，Clang を用いて実装した．Clang は，C 言語をターゲットとした新しいコンパイラである [5].
- 変換器　変換器では，図 2 のように解析器で出力した XML 形式の構文解析データを検査器で使用するために SXML 形式へ変換したのち，SXML ファイルを出力する．
- 検査器　検査器は、変換器が出力した SXML 形式のデータと利用者によって記述されたエラーパターンとのパターンマッチングを行う．

2.2 パターンの記述

エラーパターンは、パターン記号を用いて記述を行う．パターン記号には表 1 の 8 種類があり，これらの記号を組合せてエラーパターンが記述される．

表 1 の複数表現やラベル付与は，例えば，パターン記号の複数表現`$*d`は「`int a,b;`」のような連続した宣言を表す．また，パターン記号に「`$v_value`」のようにラベルを付与することで，パターン記号に情報の付加が可能になる．

Development of a Static Analysis Tool for C Language that Based on Conversion of the Source Code Structured Model of the XML Form
† Yuta HORIE,Shintarou IMAI,Yoshikazu ARAI, Toshimitsu INOMATA
†† Kazuya FUKUHARA
Faculty of Software and Information Science, Iwate Prefectural University (†)
Graduated School of Software and Infomation Science, Iwate Prefectural University (††)

エラーパターンとSXML形式のデータが一致した場合は，不具合箇所と判断され，検出結果が出力される．検出結果には，不具合箇所や不具合の詳細が含まれる．不具合箇所は，SXML形式のデータから取得されている．

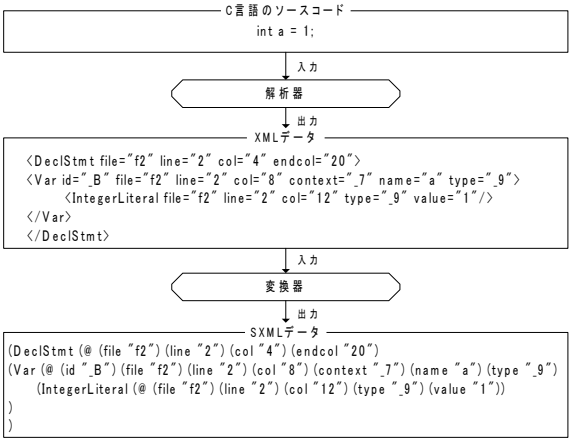

図 2: 解析器と変換器の出力例

表 1: パターン記号

構文要素	単体	複数	ラベル付与
宣言	$d	$*d	×
型	$t		○
変数	$v	$*v	○
関数	$f		○
式	#e	#*e	×
文	@	@*	×
ブロック開始	{{...{	{*	×
ブロック終了	}...}}	*}	×

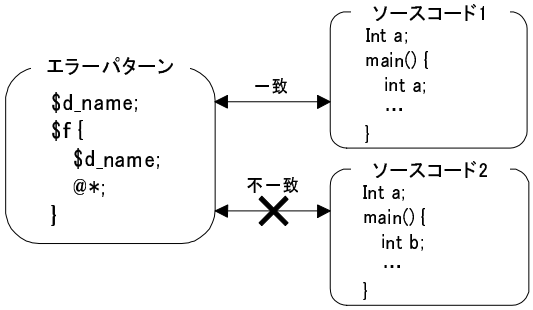

図 3: エラーパターンの記述例とソースコード例

3 評価

3.1 評価方法

本研究で開発したツールで，正しく不具合を検出することが可能であるかどうかを確認するために，車載用ソフトウェアのソースファイル（10MB）に対して図3のエラーパターンの検査を行った．図3のエラーパターンとして，ソフトウェア開発標準規格であるMISRA-Cの中の「外部スコープの識別子が内部スコープの識別子と同じ名前を用いてはならない。」を記述した．

3.2 結果と考察

図3のエラーパターンと一致するソースコードを3箇所検出することができた．今回の実験では1つのパターンについての検査を行ったが，様々なエラーパターンについて検査する必要がある．エラーパターンの追加は，パターン記号を用いてエラーパターンを記述することで行える．

今回実装したツールは，全てのエラーパターンを同等の不具合と判断している．そのため，重要な不具合情報が隠れてしまう可能性がある．そこで，エラーパターンに重みをつけ，不具合の種類を細かく分けたいと考えている．

4 おわりに

本研究では，XML形式の構文解析データを使用した不具合検出ツールの考案と実装を行った．車載用ソフトウェアのソースファイル対して実験を行い，不具合の原因となるソースコードを検出できることが確かめられた．今後の課題として，エラーパターンの重み付けが挙げられる．

参考文献

[1] トム・デマルコ/ティモン・リスター:ピープルウェア-ヤル気こそプロジェクト成功の鍵-, 日経BP社 (2001)

[2] ワッツ・S・ハンフリー: TSPiガイドブック (IT Architects’Archive ソフトウェア開発の課題10), 翔泳社 (2008).

[3] 経済産業省: 2010年版組込みソフトウェア産業実態調査報告書の公表について, http://www.meti.go.jp/policy/mono_info_service/joho/downloadfiles/2010software_research/index.htm.

[4] 福原和哉，猪股俊光，新井義和，曽我正和:ソフトウェア製品の不具合原因となるコードパターンを用いた検証手法，第4回システム検証科学技術シンポジウム，2007.

[5] ”clang” C Language Family Frontend for LLVM: http://clang.llvm.org/.