24M－05

「車載ソフトウェアの品質向上のための自動検査システムの開発」

猪股俊光（岩手県立大学ソフトウェア情報学部　教授），
福原和哉（岩手県立大学i-MOSプロジェクト研究員），
高橋耶真人（岩手県立大学大学院ソフトウェア情報学研究科　博士前期課程）

＜要旨＞
組込みソフトウェア開発では、製品出荷前のソフトウェア検査が品質向上の観点から非常に重要である。本研究課題では、主に人手によって行われていた検査作業の一部を自動化する検査システムを開発した。検査システムはC言語のソースコードを対象とし、MISRA-C（C言語のコーディング標準規約の一つ）の検査項目が適用可能（一部制限有り）、検査項目の追加が可能、GUIによる操作が可能、といった特徴をもつ。

1　研究の概要

組込みソフトウェア開発では、製品出荷前のソフトウェア検査が品質向上の観点から非常に重要である。特に、車載ソフトウェアの場合、安全面から品質検査には多大な労力が必要となる。製品出荷前の検査としては、ソースコードがあらかじめ決められたコーディング規約に従っているかどうか、既知の不具合と同様なコードが含まれているかどうか、不具合の発生を防ぐための対策が施されているかどうか、などを人間の目視、あるいは自動検査ツールの利用によって行うコードレビューが広く行われている。人間の目視による場合には、検査精度は実施者の経験や技術力に依存する。自動検査ツールを利用する場合には、検査実施者の能力差の影響をなくすることはできるものの、検査できる項目が使用ツールに依存することになる。

筆者らは、これまでに、主に組込みソフトウェアを対象とした自動検査方法ならびにそれに基づいて動作するプロトタイプの開発を試みた[1]。その結果、次の成果が得られた：

(a)　C言語で記述されたソフトウェアを対象として検査できる。
(b)　検査担当者やソフトウェア開発者が行っている検査内容を形式的に表現（エラーパターンとよぶ）することができる。
(c)　検査担当者やソフトウェア開発者が自身で検査内容を追加修正できる。

さらに、開発したプロトタイプを、H23年度高度技術者養成講習会「第３回ＳＷ検証Ｉ」（いわてものづくり・ソフトウェア融合テクノロジーセンター、H23年７月23日、講師　福原和哉）の中で参加者（県内企業社員）に使用してもらったところ、次のことがらの指摘を受けた：

(1)　エラーパターンの記述が難しい（正しく記述できたかどうかが確かめにくい）。
(2)　プロトタイプの操作手順が複雑で、検査に手間と時間がかかる。

このように、検査作業の効率向上が課題とされた。

そこで、本研究課題では、これらの問題点を含む、以下の課題の解決を試みた。

(i)　自動検査の結果表示の読解性が困難であることの解消
(ii)　検査の操作性の向上
(iii)　自動検査の対象とすることができる検査項目の拡大

これらの課題が解決されれば、複雑化する組込みソフトウェアの検査を精度よく短期間で行うことで、開発期間とコストの削減を可能にし、品質向上による競争力強化に貢献できると期待される。

2　研究の内容

2.1 検査の対象とする項目

本研究では、C言語で記述されたソースコード(構文エラーを含んでいないもの)を対象とし、

ソースコードの作成途中または作成後に行われるコードレビューにおけて、検査の対象となる以下の項目を対象とする。

・一般的なコーディング規約に準拠しているかどうか。
　－例えば、MISRA-C：2004[2]
・開発現場独自のコーディング規約に準拠しているかどうか。
　－自作ライブラリの利用方法
　－独自の命名規則　など
・過去の不具合の要因となったコードが含まれているかどうか。

2.2 パターン照合による検査方法

コードレビューでは、コーディング規約のルール通りにソースコードが記述されているかどうかを、ルールとコードを目視によって照らし合わせることで検査が行われる。例えば、「goto文を用いてはならない」というルールの場合、ソースコード中に“goto”が含まれているかどうかを検索する。これらは、文字列照合によって自動化が可能であることから多くの検査ツールで、パターン照合による検査方法が実用化されている。本研究においても、人手により行われているコードレビューの自動化という観点から、図１にしめすように、パターン照合による検査方法を採用することとした。

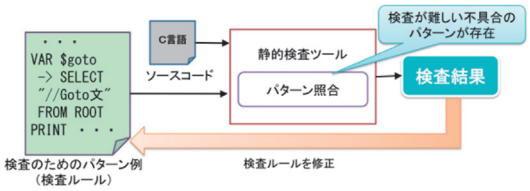

図１　パターン照合による自動検査

検査にあたっては、検査項目を次の検査ルールのXML形式[3]に表す必要がある。

＜ルール記述＞
　＜名前＞エラーパターンの名称＜／名前＞
　＜解説＞説明文＜／解説＞
　＜検査コード＞
　　検査のためのパターン群
　＜／検査コード＞
　＜／ルール記述＞

このうち、“検査のためのパターン群”がソースコードとパターン照合される。例えば、「goto文を用いてはならない」というルールは、図２のように記述される。

```
<ルール記述>
  <名前>gotoは禁止</名前>
  <解説>goto文を検出する</解説>
  <検査コード>
VAR $goto -> SELECT “//Goto文” FROM ROOT
PRINT_LINE $goto
PRINT $goto
  </検査コード>
</ルール記述>
```

図２　検査ルールの記述例

ソースコードは静的検査ツールによって、後述するCXML形式[4]（C言語の構文に対応したXML形式）に変換される。すなわち、検査対象のソースコードは木構造（XML木構造）として表される。木構造の各ノードや部分木（ノード群）には、変数宣言、for文、switch文などが対応づけられる。検査ツールは、＜検査コード＞で記載されたパターンと、各ノード（に含まれるコード）と照合し、照合したノード群を収集する。

＜検査コード＞では、“$”で始まる文字列を変数として、図３の関数などを用いながら検査したい項目を記述する。例えば、図２の＜検査コード＞の１行目“VAR $goto -＞SELECT …”は、変数$gotoにSELECT関数で求められたノード群が代入される。このように変数にはノード群を代入することができ、さらには、変数$Aと$Bに対して、次のような集合演算を適用することができる。

和集合	$A | $B	積集合 $A & $B
差集合	$A - $B	補集合 !$A
対称差集合	$A ^ $B	

関数名	引数	説明
SELECT	"XPath" FROM $var|ROOT	変数$varの各要素をコンテキストノードとして、"XPath"に属する要素をノードセットとして返す
REGEX_MATCH	"正規表現" FROM $var	変数$varの持つ各要素のうち、"正規表現"に照合した要素をノードセットとして返す
ATTRIBUTE_MATCH	"属性" IS "値" FROM $var	変数$varの各要素のうち、"属性"の属性の値が"値"であるものをノードセットとして返す
SELECT_FIRST_PARENT	"要素名" FROM $var	変数$varの中で、要素名が"名前"である最も直近の先祖をノードセットとして返す
MATCH_NODE_PATTERN	$pattern FROM $var	変数$varの各要素のうち、$patternと照合するものをノードセットとして返す
PULL_SELF	$var1 WITH "XPath" IS $var2\|$pattern	変数$var1の各要素のうち、"XPath"に属する要素が$var2の要素と一致、もしくは$patternと照合した、要素のノードセットを返す
MAKE_NODE_PATTERN	"ノードパターン"	"ノードパターン"で指定されたパターンの返す

図３　検査パターン記述のための関数

2.3 対話型静的検査ツール

本研究課題で開発した検査ツールは図４に示す対話型の静的検査ツールである[5]。

検査対象のソースコードは解析器CParserによって、中間表現であるCXMLに変換される。このCXMLと検査内容が表された検査ルールとの間でのパターン照合が、検査器

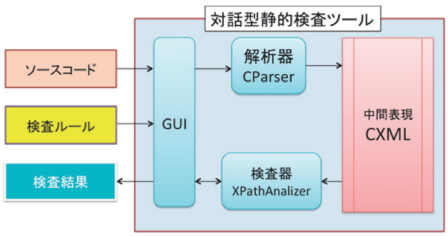

図４　開発した対話型静的検査ツールの構成

XPathAnalizerによって行われ、検査結果が出力される。CXMLは、C言語のソースコードをXMLに準拠した木構造で表すために本研究課題で考案した表現形式である。

GUIは図５のような画面を出力する。この画面を通じて、ソースコードの読み込み、検査ルールの入力、デバック情報の表示が行われる。

また、検査結果は、図６のように専用のウィンドウ上に検査対象のソースコードごとに照合した行番号が表示される。

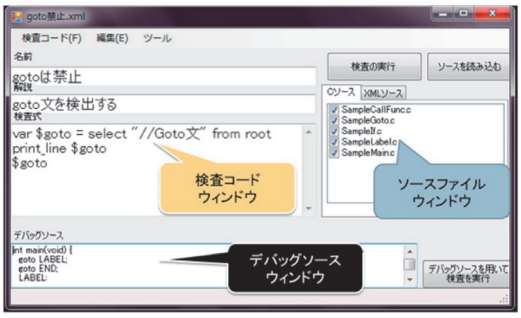

図５　対話型静的検査ツールのインタフェース

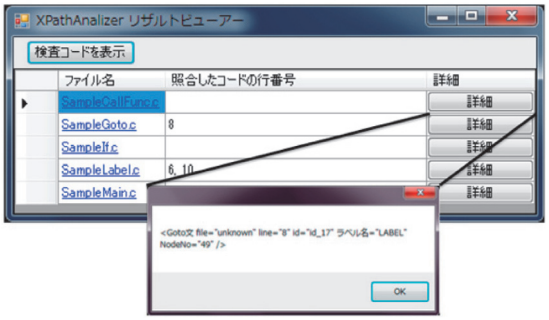

図６　検査結果の表示ウィンドウ

3　これまで得られた研究の成果

本研究課題では次の３項目を解決すべき課題とした。

(i)　自動検査の結果表示の読解性が困難であることの解消

2.3節で述べた対話型静的検査ツールのGUI機能により、検査結果をソースコードの検出箇所と対応づけて表示することが可能となった。これにより、可読性の向上が期待される。

(ii)　検査の操作性の向上

本研究課題で開発した対話型静的検査ツールでは、GUI形式によって、ソースコードの入力、検査ルールの入力、デバッグが可能であり、操作性が向上した。

(iii)　自動検査の対象とすることができる検査項目の拡大

2.2節で述べたパターン照合による検査方法では、例えば、MISRA-C：2004[2]の中の項目番号15。３「switch文の最後の節は、default節でなければならない」は、次のように検査コードとして記述できる。

```
VAR $switch -> SELECT "//Switch 文"
FROM ROOT
VAR $default -> SELECT "//DefaultLabel 文"
FROM ROOT
VAR $hasDefaultSwitch ->
  SELECT_FIRST_PARENT "Switch 文"
  FROM $default
VAR $noDefaultSwitch -> $switch -
$hasDefaultSwitch
PRINT_LINE $noDefaultSwitch
```

この他に、MISRA-Cのうち合計24項目を記述できる。

文字列のパターン照合によって検査内容を調べる方法では、パターン照合の表現能力（正規表現のように文脈に依存しない字面の照合のみに対応）のために、記述できる項目には限りがある。

そこで、筆者らは文脈に依存するような複雑な検査項目に対応するためにC#言語を用いて記述する方法も試みている[6]。この方法によれば、MISRA-Cの中のコーディング規約96項目のうち、65項目が記述可能である。

4　今後の具体的な展開

組込みソフトウェアの場合、製品の多様化に対応して、一つの製品の派生品が多種にわたっている。安価版から高級版へのシリーズ化、あるいは国内向けから海外向けへの拡張など、基となる製品からの派生品が数多く開発されるとき、開発期間の短縮のために、コードの一部を書き換えるだけの開発がしばしば行われている。

その場合、実際の開発現場では、書き換えた箇所だけを検査するのではなく、（書き換え個所を含む）製品全体の検査が行われている。これに要する人手と時間を軽減するためには、書き換え個所が影響を及ぼすコードの範囲を機械的に特定し、検査をその範囲に限定することが有効である。そのために、書き換えによって更新される計算値がやりとりされる箇所（計算値が代入される変数の種類など）を特定することを試みる。

具体的には、次の３つの課題に取り組んでいく。

⑷ ソースコードの見える化のための中間表現（グラフ構造）の設計

従来、ソースコードの解析は、データフローグラフやコントロールフローグラフを作成することで行われている。この方法では、構造体などの複雑なデータを的確に表すことができない、ソースコードのサイズが大きくなるとグラフの複雑度（ノード数、アーク数）が急激に増加するなどの課題がある。そこで、本研究では、ソースコードの１行毎にグラフ化するのではなく、例えば、関数毎というようにソースコードをグループ化して、それらの間の関係に着目した中間表現（グラフ構造）を構成することとする。このために必要とされる各種ツール（プリプロセッサ、構文解析器など）は、これまでの研究で既に開発済みである。

⑻ 更新の影響範囲の特定するアルゴリズムの設計

ソースコードに現れている変数に注目し、変数の値が更新されるコード（例えば、代入文、変数を実引数とした関数呼び出し文など）、ならびに、その変数の値を参照（利用）しているコードを⑷で構築した中間表現を利用しながら抽出し、それらの関係（更新→参照）を明らかにする。このような関係を求めておけば「変数の更新」が影響を与える範囲を特定することができるようになる。

⑹ 影響範囲のグラフ構造およびソースコード上での表示（見える化）

⑻で特定された影響範囲を、(i)の中間表現を介して、ソースコードに対応づけ、それを画面表示する。このとき、開発者が、ある変数に対する代入文を書き直す、既存の変数の値を参照しながら新しい変数に値を代入する、などといった編集しているときに。そのコードの影響範囲がソースコード上で見てわかるように表示されれば効果的であることから、GUI 環境のもとで表示できるようにする。

5 論文・学会発表等の実績

・福原和哉、高橋耶真人、猪股俊光、新井義和、今井信太郎:「組込みソフトウェア向けコーディング規約チェッカのためのカスタマイズの一方式」、FIT2012、C-024 (2012)

・高橋耶真人、福原和哉、猪股俊光、新井義和、今井信太郎：「パターン照合を用いた対話型静的検査ツールの開発」、電子情報通信学会2012年ソサイエティ大会、A-9-1 (2012)

6 受賞・特許

な し

7 その他

[1] いわて産業振興センター研究開発支援事業委託研究「車載ソフトウェアの自動品質検査システムの開発」（研究代表者 猪股俊光、 H22年８月からH23年２月）

[2] MISRA-C研究会、“組込み開発者におくるMISRA-C：2004-C 言語利用の高信頼化ガイド-”（2006）

[3] W3C（World Wide Web Consortium）: Extensible Markup Language (XML) 1.0 (Fifth Edition)、http://www.w3.org/TR/REC-xml (W3C Recommendation 2008-11-26)

[4] 福原和哉、高橋耶真人：“CXML リファレンス”、岩手県立大学リアルタイムシステム 学講座（2012）

[5] 高橋耶真人、福原和哉、猪股俊光、新井義和、今井信太郎:「パターン照合を用いた対話型静的検査ツールの開発」、電子情報通信学会2012年ソサイエティ大会、A-9-1 (2012)

[6] 福原和哉、高橋耶真人、猪股俊光、新井義和、今井信太郎:「組込みソフトウェア向けコーディング規約チェッカのためのカスタマイズの一方式」、FIT2012、C-024 (2012)