CX-Checker: 柔軟なカスタマイズが可能なC言語コーディングルールチェッカー

小林　隆志[†1]　大須賀　俊憲[†1]　上原　伸介[†1]　蛭牟田　英治[†1]　林　英志[†2]
間瀬　順一[†3]　山本　晋一郎[†2]　渥美　紀寿[†4]　川口　直弘[†5]　鈴村　延保[†6]
阿草　清滋[†1]

†1 名古屋大学大学院情報科学研究科　†2 愛知県立大学大学院情報科学研究科
†3 アイシンコムクルーズ株式会社　†4 南山大学情報理工学部　†5 株式会社サンテック
†6 アイシン精機株式会社

1 はじめに

本デモンストレーションでは，我々が開発したソフトウェアの保守性・再利用性の向上を目的としたカスタマイズ性の高いコーディングチェッカCX-Checker[1]を紹介する．CX-Checkerはルールの複雑さに合わせて，XPathを用いたルール，ルール記述用APIを用いたルール，DOMを用いたルールの3つのルールの実装方法を持つ．

ソフトウェアの保守性・再利用性の低下を避け，バグの混入を事前に防ぐために，コーディング規約が広く用いられている．コーディング規約とは，ソースコードを記述する際に守るべきルールのことであり，命名規約といった簡単なものだけではなく，ソフトウェアの信頼性向上のためのガイドラインとなるルールまでを指す．代表的なコーディング規約としてGNUコーディングスタンダードやMISRA-C[2]が挙げられる．

ソフトウェアの大規模化・複雑化に伴い，レビューやインスペクションといった手動での規則確認が困難になってきており，コーディング規約に違反する記述を自動的に検出するツールであるコーディングチェッカが提案されてきた．商用ツールでは，QACやSQMlint, RainCode Checkerが，オープンソースではCheckStyle等がその代表である．これらのツールはソースコードを静的解析し，事前に定められた規約に対する違反を確認する．

コーディングチェッカを用いることでコーディング規約のチェックのコストを大幅に下げることができるが，多くのプロジェクトが独自に定めるコーディング規約は既存のコーディングチェッカを用いてチェックすることが困難である．そのため，現在でもコードレビューなどによる人手のチェックがなされており，作業コストが問題となっている．

本研究では，利用者が柔軟にルール記述をカスタマイズ可能なC言語コーディングチェッカCX-Checkerを開発した．CX-Checkerは3種類のルール記述方法を持ち，単純なコーディング規約から複雑なものまで利用者がカスタマイズできる点が最大の特徴である．更に，XPathを用いたルール記述を容易にする補助インタフェースを持つ．

2 CX-Checker

CX-Checkerの概要を図1に示す．CX-CheckerはJava言語とXMLで書かれており，規模は約10,000行である．CUIのインタフェースとEclipseプラグインのインタフェースの2つを持ち，検査したいCソースコード群とルール群を入力とし，検査を実行する．Cソースコードは内部でCASEツールプラットフォームSapid[3]のCX-model[4]に変換される．CX-modelはC言語の抽象構文木をXMLで表現したものである．

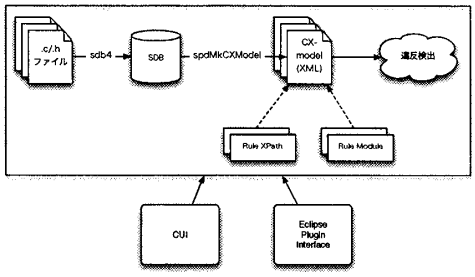

図1: CX-Checkerの概要

CX-Checker: A Customizable Coding Checker for C

Takashi Kobayashi, Toshinori Osuka, Nobuyuki Uehara, Eiji Hirumuta[†1], Eiji Hayashi[†2], Junichi Mase[†3], Shinichiro Yamamoto[†2], Noritoshi Atsumi[†4], Naohiro Kawaguchi[†5], Noboyasu Suzumura[†6], Kiyoshi Agusa[†1]

† 1 Graduate School of Information Science, Nagoya University, † 2 Graduate School of Information Science, Aichi Prefectural University, † 3 Aisin Comcruise Co., Ltd. † 4 Faculty of Information Sciences and Engineering, Nanzan University, † 5 SUNTEC Corporation, † 6 Aisin Seiki Co.,. Ltd.

2.1 ルール記述方法

CX-Checkerでは，CX-modelを対象として(1)XPathを用いた法，(2)APIを用いた方法，(3)DOMを用いた方法，の3種類の方法でコーディング規約を記述することが可能である．

Xpathを利用する方法は，構造に対する条件をXpathで記述する．XPathを用いたルールを示す．

```
1 //sp[contains(text(),"&#x9;")]
```

この例はインデントにタブ文字を指定している場合に，違反を検出するルールである． XPath中の“//sp”はすべてのsp要素を取得するXPathである．CX-modelはspはホワイトスペースを表す．“[...]”は述語であり，中に書かれた条件を満たす要素を取得する．述語中ではcontains関数を用いて，テキストがタブ文字（“	”はタブ文字の実態参照）を含むものを取得している．このように，特定の条件の構造を見つけるという多くのコーディング規約の基本部分を簡潔に記述することが可能である．

APIを用いたルールでは，対象のXMLドキュメントをCX-Modelの各要素を表すオブジェクトとして表現し，各オブジェクトの論理操作としてルールを記述する．以下にサンプルコードを示す．

```
1 CFileElement cfile = new CFileElement(file.getDOM());
2 for (CFunctionElement function : cfile.getFunctions())
3 {    System.out.println(function.getName());  }
```

この例では，ファイルを表現するCFileElementクラスからCFunctionElementのインスタンスを取得した後，getNameメソッドを呼ぶことで関数名を取得している．このAPIを用いることにより，CX-modelやDOM APIに習熟していない開発者でもルールを記述することができる．

2.2 ルール作成支援機能

XPathによるルール記述を支援するために，CX-Checkerには，XPathViewerが実装されている．XPathViewerは，ルール記述の際に，XPath記述によってソースコード上で該当する箇所を容易に確認することができるほか，XPathの文法違反，CX-ModelのDTD違反を確認し警告表示を行う．また，高度は補完機能を有し，XPath文法，CX-Model DTDに基づく補完のみでなく，対象となるCファイル中の要素を動的に検索し，補完を行うことが可能である．この機能を用いることで，CX-Modelの全体を意識せず，漸進的にルールを記述/検証することが可能となり，ルール記述のコストを大幅に削減することができる．

3 記述性能

MISRA-C[2]は，欧州の自動車業界で発足されたMISRAによって標準化された，組込みソフトウェアの信頼性向上のためのガイドラインである．MISRA-Cは127個のルールからなり，環境，文字セット，コメント，識別子，型，定数，宣言と定義，初期化，演算子，変換，式，制御フロー，関数，前処理命令，ポインタと配列，構造体と共用体，標準ライブラリという17の観点からルールを定めている．

現在の実装では，型情報の取り扱いに関してCX-modelを拡張し，独自Xpath関数を導入することで，[1]の時点よりルール記述性能を向上させており，127ルールのうち102ルールが記述可能である．商用ツールであるQAC(118/127), SQMLint(86/127)でのカバー率と比較しても十分実用に耐える記述能力である．

4 まとめ

本論文では，我々が開発したC言語のためのカスタマイズ性の高いコーディングチェッカCX-Checker[1]とその機能を説明した．CX-Checkerは，対象のC言語を解析し，ユーザーの作成したコーディング規約に違反する箇所に警告を出すEclipseプラグインとして動作する．商用ツールと同等の記述能力をもち，さらにプロジェクト毎に定める独自ルールを利用者が容易に実装することが可能である．コーディング規約を実装する方法にXPathによる記述，専用のJava言語APIによる実装，DOM操作による実装の3種類の方法を持ち，XPath記述に関しての優れた入力支援機能を有する．

参考文献

[1] 大須賀, 小林, 間瀬, 渥美, 山本, 鈴村, 阿草: CX-Checker: C言語プログラムのためのカスタマイズ可能なコーディングチェッカ, ソフトウェアエンジニアリング最前線2009, 近代科学社, pp. 119–126 (2009).

[2] MISRA - C研究会: 組込み開発者におくるMISRA - C-組込みプログラミングの高信頼化ガイド, 日本規格協会 (2004).

[3] 福安, 山本, 阿草: 細粒度ソフトウェア・リポジトリに基づいたCASEツール・プラットフォームSapid, 情報処理学会論文誌, Vol. 39, No. 6, pp. 1990–1998 (1998).

[4] 渥美, 山本, 阿草: XML記述によるソフトウェアリポジトリを用いたコード検索, 情処研報, 2005-SE-149, pp.57–64 (2005).