細粒度マークアップに基づく カスタマイズ可能なコーディング規約検査器

金子 伸幸† 今井 敬吾† 山本 晋一郎‡ 阿草 清滋†
名古屋大学大学院情報科学研究科† 愛知県立大学情報科学部‡

1．はじめに

コーディング規約とそれに基づくソースコード検査はソフトウェアの保守性や信頼性の向上に有益であり[1]、これまでに多くのコーディング規約検査器が開発されている。また、近年では、利用者がコーディング規約を追加できる拡張可能なコーディング規約検査器[2][3]も開発されている。実際のソフトウェア開発では、標準的なコーディング規約に加え、開発プロジェクト独自の規約を設けることが一般的であるため、コーディング規約検査器において独自の規約が追加できることが重要である。

従来の拡張可能なコーディング規約検査器に対して独自の規約を追加する際、利用者はコーディング規約検査器の実装ルールに従い、独自の規約を検査するモジュールを開発し、コーディング規約検査器へ組み込む方法が一般的である。しかし、この方法は、コーディング規約検査器が採用するソースコードモデルに対する理解や検査モジュールの実装に対するコストが必要なため、利用者にとって独自規約の追加は容易ではない。また、手続き的に記述された規約は、その再利用が難しい。

本研究の目的は、利用者にとって独自規約の追加が容易なコーディング規約検査器の実現である。我々は以下の機能を有するコーディング規約検査器を開発した。

- 宣言的なコーディング規約の記述による検査規約の追加
- 記述例の導出と試行によるコーディング規約の記述支援

検査規約の追加を宣言的なコーディング規約の記述により実現することで利用者の実装コストを軽減し、記述例の導出と試行によるコーディング規約の記述支援によりコーディング規約検査器が採用するソースコードモデルに対する理解コストを軽減する。

A Customizable Coding Checker based on Fine Grained Source Code Markup
† Nobuyuki KANEKO, Keigo IMAI, Kiyoshi AGUSA
Grad. School of Info. Sci., Nagoya Univ.
‡ Shinichiro YAMAMOTO
Faculty of Info. Sci. and Tech., Aichi Prefectural Univ.

2．コーディング規約の宣言的記述

ソースコードファイル 1 行あたりの文字数指定などの規約を除き、コーディング規約の大半は、構文情報を利用したソースコードパターンの指定とパターンに一致するコードに対する解釈により定義される。このため、我々は、ソースコードに対して、その構文情報を細粒度でマークアップした XML をコーディング規約検査器内部のソースコードモデルとして採用し、Xpath 式によるパターンの指定と Xpath 式の評価結果に対する解釈の指定により、コーディング規約を定義する。

Xpath 式の評価結果（パターンに一致するコード）に対する解釈は以下の３種類である。

- ***prohibit***: 違反パターンを表現する。Xpath 式の評価結果として要素が取得される場合、そのソースコードはコーディング規則に違反している。
- ***require***: 必須パターンを表現する。Xpath 式の評価結果として要素が取得されない場合、そのソースコードはコーディング規約に違反している。
- ***prerequisite***: 前提条件となるパターンを表現する。***prohibit*** や ***require*** による検査が行われるためには、***prerequisite*** で表現されるパターンに適合しなければならない。

3．記述例の導出と試行によるコーディング規約の記述支援

構文情報がマークアップされた XML に対する Xpath 式によるパターンの指定は、利用者にコーディング規約検査器が採用するソースコードモデルのスキーマの理解を強いることになる。この理解コストを軽減するため、我々は利用者がソースコードの位置を特定することで、その位置に存在する要素を取得する Xpath 式を生成する機能と利用者が記述した Xpath 式により取得される要素をソースコード上に表示する機能（Xpath 式試行機能）を開発した（図 1）。

図 1のツールにおいて、ソースコード上で特定の場所を選択（ドラッグ）し、生成ボタンを押下することにより、condition テキストボックスに生成された Xpath 式が表示される。また、

condition および precondition テキストボックスに Xpath 式を記述し、チェックボタンを押下することにより、Xpath 式で記述されたパターンに適合する箇所がソースコード上に表示される。

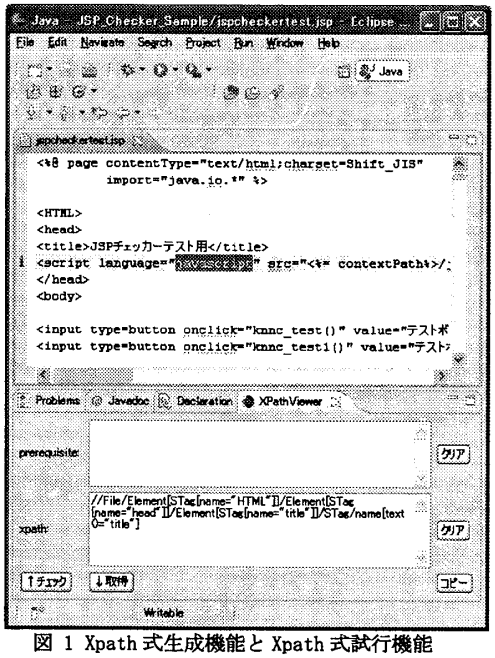

図 1 Xpath 式生成機能と Xpath 式試行機能

Xpath 式生成機能により、利用者はコーディング規約で特定したいパターンの 1 インスタンスを取得する Xpath 式を得る。また、Xpath 式試行機能により自身の Xpath 式をテストすることで、パターン記述の誤りを発見できる。生成された記述例を編集することでパターンを記述し、記述したパターンを試行することでパターンを洗練できるため、コーディング規約検査器が採用するソースコードモデルに対する理解が浅い段階からでも、コーディング規約が容易に記述可能である。

4．議論

本研究では、コーディング規約検査器のソースコードモデルとして、細粒度の構文情報を扱うマークアップ形式を採用した。コーディング規約では、「予約語 if の記述の後には半角スペースを挿入しなければならない」などのように空白も扱うケースが多く存在する。このため、コーディング規約検査器においては、空白も扱えるような粒度の細かいソースコードモデルは重要である。また、Xpath 式生成機能では、利用者による要素の指定において、位置情報を利用することでユニークな要素指定が可能であった。これらより、コーディング規約検査器のソースコードモデルとして細粒度のマークアップ方式を採用することは妥当である。

本研究で提案するコーディング規約記述の表現力を確認するため、我々は実際のプロジェクトで採用されている JSP プログラムに対するコーディング規約を提案方式により記述した。その結果、64 種類のコーディング規約のうち、50 種類のコーディング規約を宣言的に記述することができた。記述できなかった規約は、「ソースコードファイル中の 1 行あたりの文字数は半角 80 文字とする」などの構文情報を利用しないスタイル指定に関する規約や、「重複する処理は外部ファイルのインクルードとして分割する」などの設計の見直しを要求する規約であった。実際のプロジェクトで利用される規約の約 78%が表現できたことから、本研究におけるコーディング規約記述の表現力は実用的である。

5．おわりに

本研究では、宣言的なコーディング規約記述による検査規約の追加と記述例の導出と試行によるコーディング規約の記述支援を行うコーディング規約検査器を開発した。本ツールの利用者は、コーディング規約の記述により独自規約を追加可能であり、このことは、従来の機能拡張による検査規約の追加に比べて、規約追加のコストを軽減している。

宣言的なコーディング規約記述に関して、実際のプロジェクトで利用されているルールの記述による記述能力の評価は行ったが、標準的なコーディング規約[4]の記述による評価は行っていない。これは今後の課題である。また、記述例の導出と試行によるコーディング規約記述支援の効果に対する評価も今後の課題である。

謝辞

本研究の一部は文部科学省リーディングプロジェクト基盤ソフトウェアの統合開発 e-Society 高信頼 WebWare の生成技術および文部科学省科学技術研究費基盤研究(B)課題番号 17300006 の助成による。

参考文献

[1]P.Louridas,``*Static Code Analysis*'',IEEE Software, Vol.23, No.4, 2006, pp.58-61.

[2]O.Burn,``CheckStyle'',*http://checkstyle.sourceforge.net*

[3]InfoEther,``PMD'', *http://pmd.sourceforge.net*

[4]K.H.Fung and M.Roth, "*Code Conventions for the JavaServer Pages Technology Version 1.x Language*", http://java.sun.com/developer/technicalArticles/javaserverpages/code_convention/, 2003.