コーディングパターンとキーワードを用いて生成したコードスニペットの推薦

関山 太朗† 伊達 浩典‡ 石尾 隆‡ 井上 克郎‡

†大阪大学基礎工学部情報科学科 ‡大阪大学大学院情報科学研究科

1 はじめに

ソフトウェア開発においては，ソフトウェアそれぞれに独自のコーディングルールが存在する．例えば，図1のように，テキストエディタjEditでは編集中のテキストデータを格納するjEditBufferオブジェクトに対して変更を加える際に，テキストが編集可能でなければ音を鳴らすという処理を共通のルールとして記述している．このようなコーディングルールを開発者はソースコードを調査することで理解するが，巨大なプログラム上で手作業で調査したり，ルールに従ったコードを正しく記述することは困難である．

我々の研究グループでは，過去にこのようなルールをパターンマイニング技術によりコーディングパターンとして抽出する手法を提案している[1]．コーディングパターン(以下単にパターン)とは，複数のソースコードに共通して現れるコード片で，メソッド呼び出しと制御構造要素の列で表現される．図1で得られたパターンは，jEditにおいて編集の可否によって行われる処理が異なることを示すパターンである．

本研究では，コーディングルールに従ったプログラムの作成を支援するために，開発者から入力された目的の機能に関するキーワードを用いて目的の機能に関連したコーディングパターンを抽出し，抽出されたコーディングパターンを基に，ソースコードの編集状況にあわせたコードスニペットを含んだコード例を開発者へ自動的に推薦する開発環境を試作した．

2 提案手法

本手法は，開発者から入力されたキーワードを用いて，既存ソフトウェアのソースコード群から得られたパターンのうち，入力キーワードに適合するものを抽出し，開発者が編集しているソースコードの状況にあわせたコードスニペットを生成する．そして，生成されたコードスニペットを編集中のソースコードに挿入したコード例として開発者に提示する．

図2は本手法をプログラミング言語Javaに適用した場合の例である．この例では，jEdit4.3pre11から抽出されたパターン群から，開発者が与えたキーワード{buffer,editable}を用いてパターンの抽出を行っている．図2下部のコード例は，図1のパターンから生成されたコードスニペットを図2上部のソースコードへ挿入したもので，開発者へはこのコード例が推薦される．

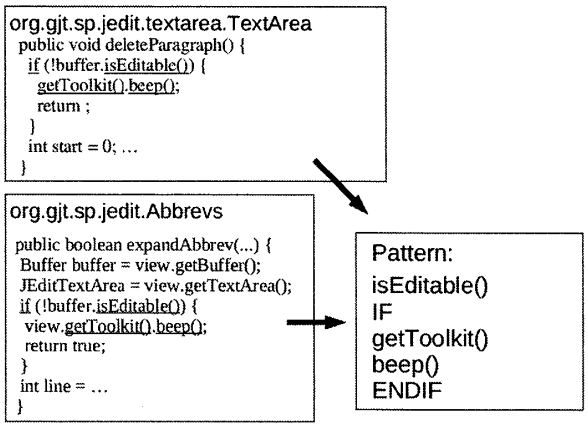

図1: jEditにおけるコーディングパターンの例

2.1 コーディングパターンの抽出

一般にプログラムには複数のパターンが存在する．本手法では開発者にとって有用なコード例を提示するために，開発者から入力されたキーワード集合IKを用いてパターンの評価値を求め，最も評価値の高いk個のパターンを選ぶ．選ばれたパターンはコードスニペットの生成に使用する．

パターンpの評価値$PS(p)$は，pとIKとが合致する度合いを表す．$PS(p)$は以下の手順で求める．

まず全てのパターンおよびIKに出現する単語の重みを計算する．単語の重み付けでは，出現回数の少ない単語ほどパターンやIKの特徴を表現する重要な単語であるとみなす．全てのパターン中に登場する全ての単語を個別に数えた場合の総数をNとし，また全てのパターンの中で単語wが登場する数をn_wとすると，単語wの重み$W(w)$は次の式で求められる．

$$W(w) = \begin{cases} \log(N/n_w) & (n_w > 0) \\ 0 & (n_w = 0) \end{cases}$$

続いて，各パターンに出現する全てのメソッド呼び出しmに対して，評価値$MS(m)$を求める．メソッド呼び出しmから得られる単語集合を$MW(m)$とすると，評価値$MS(m)$は次の式で求められる．

$$MS(m) = \sum_{w \in IK \cap MW(m)} W(w)$$

単語集合は$MW(w)$はメソッド呼び出しmのシグネチャをキャメルケースで分割して得られる．例え

Recommendation of Code Snippets Generated from Coding Patterns and Keywords

†Taro SEKIYAMA ‡Hironori DATE, ‡Takashi ISHIO, ‡Katsuro INOUE

†Department of Information and Computer Sciences, School of Engineering Science, Osaka University

‡Graduate School of Information Science and Technology, Osaka University

ば，m をクラス org.gjt.sp.jedit.buffer.JEditBuffer のメソッド boolean isEditable() とすると，$MW(m)$ = {org, gjt, sp, jedit, buffer, j, edit, buffer, boolean, is , editable} となる．また，IK={buffer,editable,log} とすると，$MS(m) = W(\text{buffer}) + W(\text{editble})$ となる．

最後に，各パターン p の評価値 $PS(p)$ を求める．$PS(p)$ は，p に出現するメソッド呼び出しの評価値の平均に対し，IK に含まれ p には含まれないキーワードの重みの総和をペナルティとして与えたものである．パターン p に含まれるメソッド呼び出しの列を $PM(p)$ とする．すなわち，あるパターン q で同じメソッドが複数回呼び出されているような場合，各メソッド呼び出しは個別に $PM(q)$ の要素となる．パターン p に出現する単語集合 $PW(p)$ は，p に出現するメソッド呼び出しから得られる単語集合の和集合である．$PW(p)$ を式で表すと次のようになる．

$$PW(p) = \bigcup_{m \in PM(p)} MW(m)$$

パターン p に対するペナルティ $PNL(p)$ を

$$PNL(p) = 1 + \sum_{w \in IK \setminus PW(p)} W(w)$$

とすると，パターン p の評価値 $PS(p)$ は次の式で求められる．

$$PS(p) = \begin{cases} 0 & (IK \cap PW(p) = \emptyset) \\ \frac{\sum_{m \in PM(p)} MS(m)}{|PM(p)|} \frac{1}{PNL(p)} & \end{cases}$$

以上の方法で全てのパターンの評価値を決定すると，評価値の大きい上位 k 個のパターンを選択し，コードスニペットの生成を行う．

2.2 コードスニペットの生成

本手法では，パターン1つに対し1つのコードスニペットを生成し，開発者へ提示する．コードスニペットの生成には，抽出されたパターンとコードスニペットが挿入されるソースコードのコンテキストを用いる．

パターン中のメソッド呼び出しによって生成された値は，その後のメソッド呼び出しや制御構造要素のパラメータ，もしくはコードスニペット以降のソースコードで利用されることがある．例えば図1のパターンでは，メソッド isEditable の戻り値が if 文の条件式で用いられている．そこで，本手法ではコードスニペット中で生成された値ごとに新しく変数を宣言し，生成された値をその変数へと格納する．そして，メソッド呼び出しや制御構造要素の条件式などに値が必要になると，適切な型をもつ変数のうち，必要な箇所の直前で宣言された変数を利用する．

図2下部は，図1のパターンから生成されたコードスニペットである．この例では，メソッド getToolkit の戻り値をメソッド beep の receiver object として利用しており，また if 文のパラメータとしてメソッド isEditable から得られた結果を利用していることがわかる．

以上の方法で生成されたコードスニペットは，その基となったパターンの評価の大きい順に，開発者が編集しているソースコードに挿入され，開発者に推薦される．

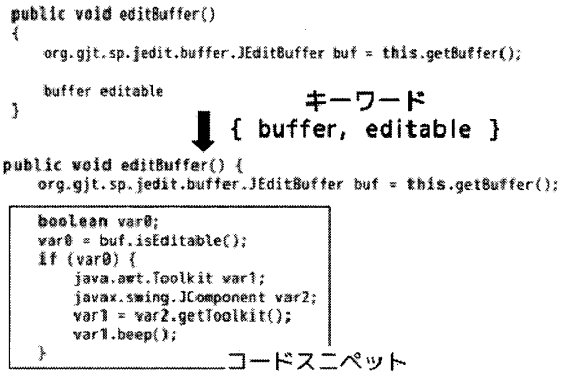

図2: 本手法の適用例

3 実装

本研究では，提案手法をプログラミング言語 Java に適用し，統合開発環境 Eclipse から利用できる環境を試作した．本ツールで用いるパターンは，シーケンシャルパターンマイニングによりパターンを抽出するツール Fung[1] により得られたものである．

図2下部は本ツールに対して図2上部のソースコードとキーワード buffer,editable を与えて推薦されたコード例のうち，3番目のパターンを基にして推薦されたコード例である．このコード例は，jEdit のテキスト編集処理を行う全てのメソッドに出現するコードと同じものである．なお，メソッドの戻り値や引数の型がソースコード等から得られなかった場合は，unknown という型として結果を出力する．

4 まとめ

本論文では，開発者がソフトウェアプログラムを記述する際に，ソフトウェアのコーディングルール調査にかかる負担を軽減させるための手法と，その実装について述べた．今後の課題としては，実験による本手法の評価，パターンの評価式の改善，ツールの利便性の向上などが挙げられる．

謝辞

本研究は，文部科学省科学研究費補助金若手研究（B）（課題番号:21700030）の助成を得た．

参考文献

[1] Ishio, T., Date, H., Miyake, T. and Inoue, K.: Mining Coding Patterns to Detect Crosscutting Concerns in Java Programs, *Inproceedings of 15th Working Conference on Reverse Engineering*, pp.123-132 (2008).