大規模パターンマイニングを用いた高品質ソースコードの検索

石尾　隆[†1]　伊達浩典[†1]
市井　誠[†1]　井上克郎[†1]

複数のプロジェクトに出現し，長期間に渡って変更されていない定型的なソースコード片は，安定し，かつ十分にテストされていることから，動作実績のある高品質なソースコードであると考えられる．本研究では，パターンマイニングを多数のソフトウェアの複数のバージョンに対して適用することで，そのような高品質なコードを抽出し，再利用する手法を提案する．

A Pattern Mining Approach for Retrieving Reliable Source Code

Takashi Ishio,[†1] Hironori Date,[†1] Makoto Ichii[†1]
and Katsuro Inoue[†1]

Source code fragments appear in various projects, locations and versions are stable and well tested, therefore, considered reliable. In this research, we apply a pattern mining approach to various software repositories in order to effectively reuse reliable source code fragments.

1. はじめに

近年，インターネット上に公開されているソースコードを検索するための手法が活発に開発されている[1)]．ソースコード検索手法は，入力として，開発者が必要とする機能を示すキーワードなどを受け取り，該当するソースコード片の集合を出力する．しかし，従来のソースコード検索手法は，開発者が求めている機能に対応したソースコードを提示するが，そのソースコードの品質は保証しない．品質の低いソースコードの再利用は，欠陥の発生につながり，経済的あるいは社会的な損失をもたらす可能性がある．

そこで，本研究では，ソースコードの利用実績に基づいて，品質の高いソースコードを特定する手法を提案する．複数のソフトウェアプロジェクトに出現し，長期間に渡って変更されたことがないソースコード片は，安定し，かつ十分にテストされていることから，動作実績のある高品質なソースコードであると考えられる．複数のプロジェクトで長期間生存したコードを特定するために，複数のソフトウェアリポジトリに対してコーディングパターンマイニング手法[3)] を適用する．

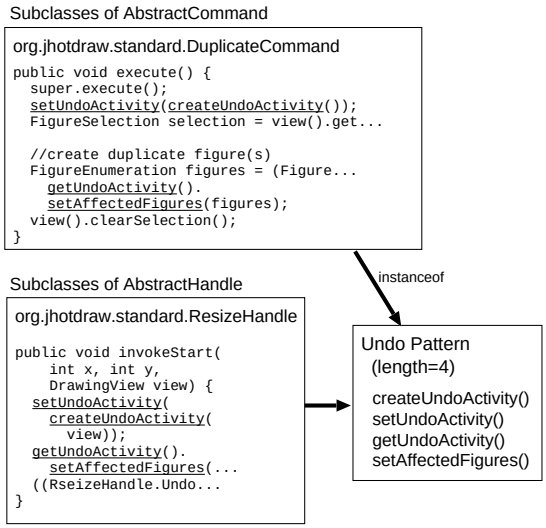

図 1 JHotDraw 5.4b1 における画像の編集内容を「元に戻す」機能の実装パターン
Fig. 1 Undo command pattern in JHotDraw 5.4b1

2. コーディングパターンマイニング

コーディングパターンとは，複数のモジュールに分散した定型的なコードである．我々は，コーディングパターンを，メソッド呼び出し要素とそれに付随する制御構造要素（条件分岐と繰り返し文）の定型的な列と捉えたパターンマイニング手法を提案している[3)]．

†1 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

図 1 に示すコーディングパターンの例は，画像編集ソフトウェアである JHotDraw 5.4b1 から抽出されたもので，各種編集操作の結果を「元に戻す」ためのコードの一部である．

本手法では，Java ソースコードをメソッド呼び出しおよび制御構造の列として正規化し，マイニング処理を行う．要素列の順序さえ保存されていれば，他の無関係な要素が挿入されても影響を受けないため，ソースコードの追加や削除に対応できる．従来の研究では，単独ソフトウェアの特定バージョンを解析していたが，本研究では，複数のプロジェクト，複数のバージョンのソースコードを同時に解析する．

本研究における課題は，解析対象の規模の拡大と，抽出されるパターンの分析法である．過去の実験では，約 50 万行の Azureus から 5000 種類ほどのパターンを抽出している[3] が，Eclipse のような大規模プロジェクト（2008 年 6 月時点で 1800 万行）を複数同時に解析する場合，計算コストと出力パターン数のいずれもが，膨大なものとなる可能性がある．

解析対象の規模の拡大に対しては，使用しているアルゴリズムを分散計算に拡張したもの[4] を実装することで対応する．一方，パターンの分析については，有用なパターンのみを効果的に抽出する基準の設定が必要であり，複数のバージョンから抽出されるパターンの特徴を示す新たな基準を設定する必要がある．

現在，有用なパターンの抽出のために，以下の情報を用いることを検討している．

クラスの一般性（Universality）[2] Java の標準ライブラリなど，複数のソフトウェアで一般的に用いられているクラスのみを用いるパターンもまた一般性が高く，新たなソフトウェアでの再利用が容易であると推測される．クラスの一般性は，ソフトウェアの多数の場所で使用されているクラスほど，また多数のソフトウェアで使用されているクラスほど高いと考え，クラスの被参照数と，クラスの出現ソフトウェア数の積によって，その値を計測する．

パターンの安定性（Stability） 一定期間変更されることがなかったパターンは，欠陥による修正がなく，高品質であると考えられる．パターンに該当するコードに対する修正回数によって，安定性を計測する．

ソフトウェアごとに開発時期が異なる可能性があるため，本研究では，実時間を用いて，各時刻でのソフトウェア群の状態を計算し，一般性や安定性を求める．

3. ソースコード検索への適用

一般性が高く，かつ安定性の高いコーディングパターンは，再利用性が高く，品質も高いソースコード片であると考えられる．しかし，開発者に対してどのようにパターン情報を提示するかというインタフェース面も，１つの課題である．パターンそのものを直接検索，閲覧するシステムを構築する以外にも，既存のソースコード検索システムを拡張し，検索されたクラスあるいはメソッドの付加情報として，それらを利用するためのコーディングパターンを提示するという方法が考えられる．

既存のソースコード検索システムが抽出したソースコード片に対して，品質に基づくフィルタリングを行うためには，コーディングパターンの含有率などをソースコード片の品質指標として使用するという方法が考えられる．たとえば Java のクラス単位で，パターンに合致するソースコードがどれだけの割合で含まれているかを計測することで，定型的なコードのみからなるクラスと，独自のコードを多数含むクラスとを区別することができる．定型的なコードほど欠陥は少ないと考えられるが，たとえば複雑さなど，他の品質指標との関連性は自明ではないため，欠陥情報が提供されているソフトウェアを用いた実験を行い，妥当性の評価を行う必要がある．

謝辞 本研究は，日本学術振興会科研費基盤研究（A）（課題番号:17200001）の助成を得ている．

参 考 文 献

1) Hummel, O., Janjic, W. and Atkinson, C.: Code Conjurer: Pulling Reusable Software out of Thin Air, *IEEE Software*, Vol.25, No.5, pp.45–52 (2008).
2) Ichii, M., Ishio, T. and Inoue, K.: Cross-application Fan-in Analysis for Finding Application-specific Concern. *Proceedings of the 4th Asian Workshop on Aspect-Oriented Software Development*, `http://appsrv.cse.cuhk.edu.hk/~aoasia/workshop/APSEC08/` (2008).
3) Ishio, T., Date, H., Miyake, T. and Inoue, K.: Mining Coding Patterns to Detect Crosscutting Concerns in Java Programs, *Proceedings of the 15th IEEE Working Conference on Reverse Engineering*, pp.123–132 (2008).
4) Sutou, T., Tamura, K., Mori, Y. and Kitakami, H.: Design and Implementation of Parallel Modified PrefixSpan Method, *Proceedings of the 5th International Symposium on High Performance Computing*, pp.412–422 (2003).