7J-06

抽象構文木の双方向変換によるコードクローン管理に向けて

相原 崇弘 †
法政大学大学院 情報科学研究科 †

日高 宗一郎 ‡
法政大学 情報科学部 ‡

1 はじめに

ソフトウェアの大規模化に伴い，ソフトウェア保守は困難となっている．ソフトウェア保守の重要性は高く，システム総費用の中で占める割合は大きい．ソフトウェア保守が困難な理由の一つとして，コードクローンが存在する．コードクローンが発生する主な理由として，コピーアンドペーストが存在する．コピーアンドペーストによって発生したコードクローンは，拡張や修正がなされる．あるコードクローンへの拡張や修正は関係するコードクローンに変更を伝搬する必要性がある．コードクローンに対する変更を伝搬する仕組みとして，例えば，Wit ら [2] や Cheng ら [3] の手法がある．これらの手法では伝搬がどのような性質を持っているかは明らかではない．本研究では，コードクローンの編集の伝搬に一定の望ましい性質を持たせることによって，伝搬規則を明らかにすることにより，一貫性を保持したコードクローンに対する系統的な編集を柔軟に行う機構の提案を目指している [7]．編集の伝搬には，日高らによって提案，実装された双方向変換言語の一つである UnQL+ [4] を用いることによって，UnQL+ が持っている性質を満たす伝搬の仕組みを提供する．本稿では，特に，編集操作の選択的伝搬手法について提案する．

2 提案手法の概要

コードクローンの編集は，関わりのあるコードクローン集合全体に対し適切に行う必要がある．適切な修正はコードクローン毎のプログラム構造を考慮にいれたコードの挿入，削除，移動，更新，複製を双方向変換言語によって記述し，修正を双方向変換によって伝搬することによって実現出来る．提案手法では，抽象構文木を通してプログラムの構造を分析し，コードクローン毎への異なる変更を UnQL+ により記述することで，双方向変換による規則性をもった修正の伝搬を行う．

提案手法では，コードクローンの編集を Higo らが提案する編集スクリプト [1] によって記述し，編集スクリプトを UnQL+ へと変換することによってコードクローン間の同期を行う．提案手法の概要を図 1 に示す．STEP の詳細は以下の通りである．

STEP1(構文解析) コードクローンを含むソースファイルの構文解析を行い，抽象構文木を生成する．

STEP2(コードクローンへの双方向変換) ソースファイルから抽象構文木形式でコードクローン部分へと双方向変換する．

STEP3(編集スクリプトの動的な生成) コードクローン毎に特有な編集を編集スクリプトによって表現する．

STEP4(UnQL+ への変換) 編集スクリプトを UnQL+ へと変換する．

STEP1 では，抽象構文木の全ての要素を一意に特定可能なようにラベルを付ける．STEP2 では，STEP1 で生成した抽象構文木のコードクローンにあたる部分をコードクローンの木へと変換する．STEP1 で生成した抽象構文木の要素は一意に特定可能なため，コードクローンの木の要素と対応付けが可能である．すなわち，well-behavedness(ラウンドトリップ性) を満たすような変換がソースファイルの抽象構文木とコードクローンの木の間で記述可能である．STEP3 では，Higo らが提案する手法 [1] に従って，コードクローンの木毎の特有な変更を記述し，プログラミング動作の追跡を行う．ただし，Higo らの手法は抽象構文木に対してのみ適用可能な点に注意が必要である．コードクローンの木は，ソースコードの抽象構文木の部分木なため，抽象構文木であるとは限らない．しかし，抽象構文木に類似した木であ

Toward Code Clone Management through Bidirectional Transformation of Abstract Syntax Trees
†TAKAHIRO AIHARA, Graduate School of computer and Information Sciences, Hosei University
‡HIDAKA SOICHIRO, Faculty School of Computer and Information Sciences, Hosei University

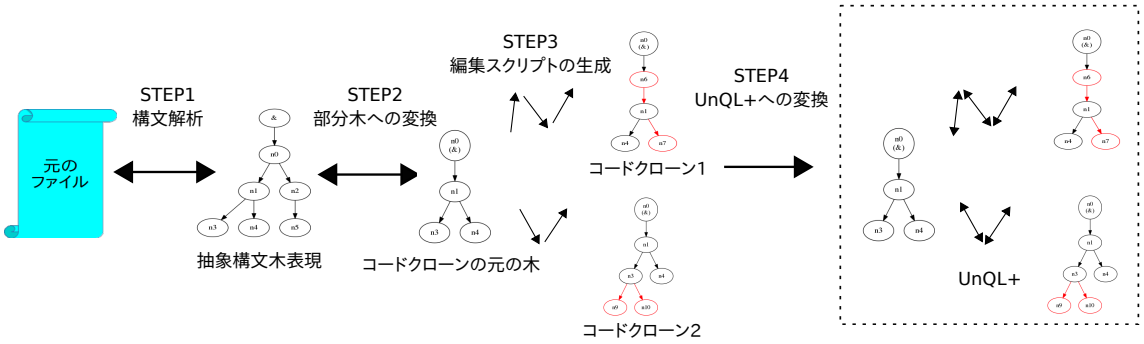

図 1　提案手法の概要

るため，Higo らの手法が適用可能であると推測される．STEP4 では，編集スクリプトを UnQL+ へと変換する．

3　編集操作の選択的伝搬

コードクローンの編集には，伝搬すべきものとすべきでないものがある．例えば，Java ファイルをコピーしたとする．コピーしたファイルに対し，ソースコードのリファクタリングとして変数名の変更，補助関数の導入を行い, 新しい機能の追加を行うためにメソッドの追加，変数の追加を行ったとする．このような場合に，リファクタリングした内容に関しては，コピー元のファイルにも変更を伝搬させたいが，それ以外の変更，今回の場合には新しい機能については伝搬させたくないという要請が発生する．提案手法では，伝搬を行わない変更については双方向変換言語によって記述し，伝搬を行う変更については，双方向変換がもっている伝搬規則に従って伝搬を行う．双方向変換言語によって変更を記述すため，双方向変換言語は変換を組み合わせ可能である必要がある．本研究では，双方向変換言語 UnQL+ によって双方向変換を記述し， UnQL+ が持つ伝搬規則に従って，伝搬を行う．

関連研究としては，編集操作を双方向変換で表した研究 [5] やモデルからソースコードの変換を行い，変換の範囲を選択することによりモデルとソースコード間の一貫性を保つ研究 [6] がある．本研究では，自動的にコードの編集を追跡し，編集操作の伝搬の有無の選択が可能である．

4　まとめ

本稿では，コードクローンの変更を伝搬する提案している仕組みの概要を述べ，編集操作の選択的伝搬方法を提案した．双方向変換言語を使用したコードクローンの管理手法は我々が知る限りない．双方向変換言語を使用することで，コードクローン編集の伝搬をより柔軟に扱えることが期待される．

今後の課題としては，提案手法に沿って実装を行い，コードクローン編集の伝搬手法として有用性を確かめることである．

参考文献

[1] 肥後他，”編集スクリプトへのコピーアンドペースト操作の導入によるコード差分の理解向上の試み,” 情報処理学会論文誌, 2017, pp. 833-844.

[2] M. de Wit et al, “Managing Code Clones Using Dynamic Change Tracking and Resolution,” *25th ICSM*, 2009, pp. 169-178.

[3] X. Cheng et al, “Rule-Directed Code Clone Synchronization,” *24th ICPC*, 2016, pp. 1-10.

[4] S. Hidaka et al, ”Towards a Compositional Approach to Model Transformation for Software Development,” *SAC’09*, 2009, pp. 164-173.

[5] Z. Hu et al, ”A programmable editor for developing structured documents based on bidirectional transformations,” *PEPM’04*, 2004, pp. 178-189.

[6] Y. Yu et al, ”Maintaining invariant traceability through bidirectional transformations,” *34th ICSE*, 2012, pp. 540-550.

[7] 相原他，”双方向変換言語を用いたコードクローン管理に向けて,” 第 59 回プログラミングシンポジウム, 2017(発表予定)