静的・動的ハイブリッドな解析によるコード・データのトレーサビリティリンクの抽出

津村 耕司[†1]　鷲崎 弘宜[†1]　深澤 良彰[†1]　土屋 良介[†1]　大島 敬志[†2]　三部 良太[†2]

あらまし ソフトウェアの保守・運用を行うためには，トレーサビリティが明確化されており，ソフトウェアが可視化されていることが望ましい．プログラムの静的・動的解析によるソースコード・データベース間のトレーサビリティリンク抽出は，それぞれに長所と短所が存在する．我々は，静的・動的解析を組み合わせることで，それぞれを単独で行うよりも効果的にソースコード・データベース間のトレーサビリティリンクを抽出する手法を提案する．

Extraction traceability links between source code and database schema by combining static and dynamic analysis

Koji Tsumura[†1]　Hironori Washizaki[†1]　Yoshiaki Fukazawa[†1]　Ryosuke Tsuchiya[†1]　Keishi Oshima[†2]
Ryota Mibe[†2]

Abstract In maintenance and operation of software, it is to be desired that traceability is clarified and software is visible. There are static and dynamic analyses to extract traceability links between source code and database schema, and each of them has advantages and disadvantages. We propose a technique extracting traceability links between source code and database schema by combining static and dynamic analyses more effective than using them individually.

1. はじめに

ソフトウェアの保守・運用を行うためには，トレーサビリティが明確化されており，ソフトウェアが可視化されていることが望ましい．特に，エンタープライズ系アプリケーションにおいては，機能追加やバグ修正，リファクタリング時にはソースコードとデータベーススキーマ(以下 DB スキーマとする)の両方を変更する必要が生じるので，ソースコードと DB 間の関連が可視化されていることで，変更箇所の特定が容易となり，工数の削減に繋がる．

ここで，ソースコードとDBの関連を表す方法として，図 1 のように，DB 中のレコードが，ソースコード中のどのクラスやメソッドで作成(Create)，参照(Rread)，更新(Update)，削除(Delete)されるかを，クラスやメソッドとDBスキーマのマトリックスで表現する CRUD 図が存在する．アプリケーションへ変更要求が発生した場合等には，このような CRUD 図を参照することで，ソースコードと DB スキーマの両方の変更箇所や変更規模の特定が容易となるため，工数の見積や削減が可能になる．

		テーブルα		
		カラム1	カラム2	カラム3
クラスA	メソッド1		R	
	メソッド2	R		R
	メソッド3	C	C	CU
	メソッド4	R	R	R

図 1 CRUD 図の例

ここで，ソースコードとデータベース間のトレーサビリティリンク抽出の手法としては，実際のプログラム実行を伴わない静的解析[1][2]と，実際にプログラムを実行した結果を用いる動的解析[3]が存在する．しかし，静的解析には「動的に SQL が組立てられる箇所は解析不可能である」という欠点が，動的解析には「実行されなかった箇所・パスは解析不可能である」という欠点が存在する．そこで，我々は静的解析・動的解析を組み合わせた，既存手法よりもより効果的にコード・データ間のトレーサビリティリンクを半自動的に抽出する手法を提案する．以下に，本研究の研究課題を定義する．

RQ1　エンタープライズ系ソフトウェアにおいて，静的・動的ハイブリットな解析によって，それぞれの解析を個別に実施した場合よりも多くの正解リンクが抽出できるか

†1 早稲田大学情報理工学科 Dept. Computer Science, Waseda University
†2 (株) 日立製作所 横浜研究所 Hitachi, Ltd., Yokohama Research Laboratory

RQ2　エンタープライズ系ソフトウェアにおける静的・動的ハイブリットな解析によるリンク抽出は，実用可能な時間内で実施可能か

以上の 2 つの研究課題が，提案手法により解決可能か検証を行っていく．

2. 提案手法

本手法は，Java[i]によって実装されたエンタープライズ系ソフトウェアを適用対象とし，メソッドと DB スキーマ(テーブルやカラム)との関連を抽出し，CRUD 図として出力する．最終的に出力された CRUD 図は，ソフトウェア保守・運用段階で変更箇所の特定や工数決定に用いられることを期待している．本手法は，大きく

1. 静的解析の実施
2. 動的解析の実施
3. 両解析結果の総合

という 3 つのステップに分けられる．以下，それぞれのステップについて説明する．

2.1. 静的解析の実施

静的解析の手順について説明する．

まず，(1)クラスファイルを解析し，SQL クエリを発行し，DB にアクセスしているメソッドを抽出する．次に，(2)抽出されたメソッドを通じて，間接的に DB にアクセスしているメソッドを抽出する．また，(3)SQL クエリが動的に組み立てられるメソッドについては，そのメソッドに至るまでのメソッドコールグラフを解析・抽出する． 最後に，(1)(2)の結果から，CRUD 図を出力し，(3)の結果から，静的解析では解析不可能な DB アクセスメソッドのコールグラフを，通過パス情報として出力する．

2.2. 動的解析の実施

動的解析の手順について説明する．

まず，(1)メソッドの呼び出し情報，発行された SQL 文のログ取得コードを埋め込んだプログラムを，用意したテストケースに基づいて実行し，アプリケーションログを取得する．(2)次に，取得したログから，DB にアクセスしているメソッドを抽出し，CRUD 図としてする．ここで，(3)ログ解析の際には静的解析の手順(3)で得られた通過パス情報を参照し，ログ内における通過パスの出現情報を CRUD 図とともに出力する．出現情報をもとに，人手でテストケースを追加し，ログ取得・ログ解析を繰り返す．

2.3. 両解析結果の総合

両解析結果の CRUD 図を総合する．総合方法としては，より多くのリンクを抽出するという観点から，各メソッドについて，両解析結果のうち少なくとも片方の解析で抽出された関連を，結果として CRUD 図に出力する．

本手法の全体像を図 2 に示す．

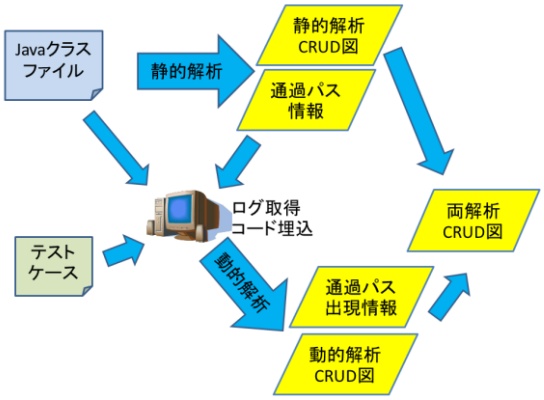

図 2 本手法の全体像

3. おわりに

我々は，静的解析と動的解析を組み合わせた，既存手法よりも効果的なコード・データ間のトレーサビリティリンク抽出手法を提案する．今後，提案手法の OSS や大規模実アプリケーションへの適用を目指す．また，本ワークショップの議論の中で，静的解析結果の動的解析への更なる効果的・効率的な利用方法について考案，改善していきたい．

参考文献

[1] Carlos Garcia-Alvarado, Carlos Ordonez and Veerabhadran, “Querying External Source Code Files of Programs Connecting to a Relational Database”, PIKM '12 Proceedings of the 5th Ph.D. workshop on Information and knowledge, pages 9-16, 2012

[2] DB-MAIN 9 Reference Manual, http://www-db.deis.unibo.it/courses/SIL-B/DOCS/DB-MAIN-Reference-Manual.pdf

[3] 風戸広史，林晋平，岡田敏，宮田俊介，星野隆，佐伯元司, “多層システムのための形式概念分析に基づく Feature Location 手法の提案”，電子情報通信学会技術研究報告. SS，ソフトウェアサイエンス 111(481), pages 139-144, 2012

[i] Java は，Oracle Corporation 及びその子会社，関連会社の米国及びその他の国における登録商標です．