多言語に対応した解析・理解支援のフレームワークに関する研究

伊藤　徹弥 (0710008)

北陸先端科学技術大学院大学 情報科学研究科

2009 年 2 月 5 日

キーワード: 理解支援, ナビゲーション木, 言語依存性, 多言語.

1　背景と目的

現在のソフトウェア開発において、機能の修正等の変更要求に対応するためには特定の機能を実現している箇所を見つける必要がある。しかしながら，ソースコードの構造が複雑であるため特定の機能がどこで実現されているかは開発者が正しく理解することが困難な場合がある。

本研究では変更を実現するために必要な情報をソースコード上から効率的に抽出する手法を提案する．また、複数の言語に対応させるために言語固有の特性について検証し、先行研究で提案されたフィルタの機能を補うために新たなフィルタを提案する。その結果からフレームワークを作成することを目的としている。

2　情報量の制御

情報量を制御する方法として本研究で用いたのは先行研究で新倉が提案した細粒度フィルタによる情報抽出である．関心とはユーザが注目している箇所 (範囲) で、関心の拡大とはユーザが関心を元に、注目している範囲を変更することである。本研究ではフィルタの合成によって関心の拡大を実現している。新倉が提案したフィルタと，機能不足や多言語対応のために本研究で新たに提案した計 10 種類のフィルタを以下に示す.

- 関心を決定するフィルタ
  - 変数と型の宣言部を抽出 (DECL)
  - 入力した文字列を含むクラス，関数，変数の宣言部を抽出 (EXT_NAME)
- 関心を広げるフィルタ

  - 制御構造の実行ブロックを抽出 (CTRL)
  - 注目行の近傍を表示 (NEIGHBOUR)
  - 代入文の依存を抽出 (TRACE)
  - 関数の宣言部を抽出 (F_DECL)
  - 関数が呼び出されている箇所を抽出 (REF)
  - 抽象クラスやインタフェースの実装部を抽出 (IMPLEMENTED)
- 関心を絞るフィルタ
  - 範囲を宣言部と実行部に分離 (SEP)
  - 変数の出現範囲を抽出 (RANGE)

フィルタは、ソースコード全体を現す AST と関心を表す情報 (ファイル名、行の範囲) の集合を共通パラメタとして、ソースコードから情報を抽出する。各フィルタ固有のパラメタとして、注目行、注目する関数、注目する変数などが挙げられる。例えば、ソースコードを解読している時にある関数に注目しているとする。そのとき注目している関数を呼び出している箇所を知る必要があるとする。その際のパラメタとして、関数の名前とその行番号、引数の数と型、そして戻り値の型をフィルタに与える。その結果その関数が呼び出されている箇所が行番号と関数呼び出し文が抽出される。

先行研究のフィルタは C 言語のみを対象としていたが、これを Java 言語に対応させるために、言語固有の特性を分類した。具体的にはオーバーロードがその例として挙げられる。

3 ナビゲーション

情報を抽出するために、フィルタを用いた情報抽出法について検討した。その手順を以下に示す。

1. 機能の実現部に関連がある可能性が高い名前を検索
2. 検索した結果に含まれる関数呼び出しや引数から制御フローを追う

また、効率を上げる手段の一つとしてナビゲーション木を導入した。ナビゲーション木とはフィルタの適用履歴を表す木である。ノードは注目箇所の情報を保持している。エッジは適用したフィルタとパラメタの情報を保持している。生成された木を見れば注目していた箇所と適用したフィルタの情報を容易に得ることができる。抽出した結果が、変更に関係がない場合には、生成された木をユーザに提示することで手戻りをすることが可能となる。

4　フィルタの適用実験

提案したフィルタと手法の有効性を確認するために、2 つの実験を行った。1 つめの実験は、機能変更を必要とする箇所を抽出する実験である。2 つめの実験は、変更すべき箇所を全て抽出可能かどうかという実験である。実験に用いたのはオープンソースソフトウェアの 2 つのバージョンである。まず新バージョンのリリースノートから、旧バージョンと何が違うかという情報を得る。そしてこれを旧バージョンにおける変更要求とする。旧バージョンに対して提案手法を用いて変更すべき箇所の候補を抽出する。そして旧バージョンと新バージョンの差分を取得し、その差分と抽出した結果を比較することによって変更すべき箇所を抽出できたか確認する。

1 つめの実験の結果、機能変更を必要とする箇所を抽出することができた。

2 つめの実験の結果、ソフトウェアの構造が変更されていたが、変更すべき箇所の候補については全て修正されていた。

この結果から、提案手法は有効であるといえる。

5　まとめ

細粒度フィルタの利用法による効率的な情報抽出方法を提案した。さらにナビゲーション木を導入し、情報抽出プロセスが停止した際の手戻りを可能とした。また、フィルタの改良を行った。そしてフレームワークを作成するために言語依存性と多言語対応させる際に考慮に入れる点について示した。