MVC モデルを考慮したソースコード補完手法の研究

橋浦研究室　　1135115　　猪飼　恒

１．はじめに

開発者が効率よくプログラムを開発するために，既存のソースコードの再利用する場合がある．統合開発環境にはコード補完機能が存在し，開発者に使用可能なソースコードを提示することが可能である．コード補完機能では，表示されたリストから使用するコードを探して選択する必要がある．リストには多くの選択肢が存在し，開発者が選択をする際に適切に選択できない場合がある．本研究では開発者が適切な選択肢を選べるようにリストのソートを行う．

２．研究目的

本研究では，開発者がコード補完を利用する際にソフトウェアのアーキテクチャから得られる情報をもとに開発者が適切な選択肢を見つけやすくなるように支援する． 本研究では選択肢の見つけやすさを適切な候補が補完リストの上位に出ることと定義する．

３．提案手法

開発者がコード補完を利用した際に効率よくプログラム開発を行うため山本[1]の研究が存在する．山本の研究では呼び出し文の前後に着目し，既存のソースコードから使用される可能性の高いソースコードを収集して補完リストの作成を行っている．そこで，本研究では MVC モデルで開発を行う際に領域によって使用されるソースコードの種類が異なる点に着目する．以下の手順が本研究の手法である．

1. MVC モデルの 3 つのデータベースを作成する．
2. MVC モデルの領域ごとに使用されたソースコードの種類と回数のデータを収集する．本研究ではソースコードごとの使用された回数をソースコードの出現頻度として定義する．
3. 作成したデータベースに出現頻度のデータを格納する．
4. データをもとに出現頻度の高い順にソートを行う．

例として，`getDispatcherType()`の出現頻度が高いデータが格納されていると仮定する．この場合，本来アルファベット順である図 1 のコード補完のリストが図 2 のようになる．

```
String id = request.getParameter("id");
IdProcessing ip = new IdProcessing();
UserBean bean = ip.getUserData(id);
RequestDispatcher rd;
if (bean != null) {
request.setAttr
rd =
```

```
getContextPath()
getCookies()
getDateHeader(String arg0)
getDispatcherType()
getHeader(String arg0)
getHeaderNames()
getHeaders(String arg0)
getInputStream()
getIntHeader(String arg0)
```

図 1　手法適用前補完候補例

```
String id = request.getParameter("id");
IdProcessing ip = new IdProcessing();
UserBean bean = ip.getUserData(id);
RequestDispatcher rd;
if (bean != null) {
request.setAttr
rd =
```

```
getDispatcherType()
getContextPath()
getCookies()
getDateHeader(String arg0)
getHeader(String arg0)
getHeaderNames()
getHeaders(String arg0)
getInputStream()
getIntHeader(String arg0)
```

図 2　手法適用後の補完候補例

４．アプリケーションの実現

本研究では，手法を実現するために統合開発環境でありコード補完機能の備わっている Eclipse のプラグインを開発する．プラグインとは，プログラムの機能の拡張を行うプログラムである．

プラグイン開発後，SQLite にデータベースを作成し，データの収集を行う．データベースの作成方法は以下の手順で行う．

1. SQLite を用いて領域ごとに使用された回

数とソースコードの種類を格納できるデータベースを作成する.

2. Github にて“MVC web application”で検索を行う.
3. パッケージ名かファイル名に MVC モデルの領域である model,view,controller のいずれかが含まれているプロジェクトを探す.
4. ファイルの中身を開き,使用されたメソッドと使用回数を記録する.
5. 使用された回数が 2 回以上であり,なおかつ 2 つ以上のプロジェクトで使用されている場合,作成したデータベースに記録したメソッドと回数を格納する.

5 の手順で 2 つ以上のプログラムで使用された場合と限定している理由は使用頻度が低い場合もデータとして記録してしまうと悪影響があるためである.

本研究ではこの手順で 30 プロジェクトからデータを収集した. 表 1 が収集したデータの一部である.

表 1　model に関する DB(一部)

使用回数	ソースコードの名前
4	valueOf(int i) : String - String
6	StringBuilder() - java.lang.StringBuilder
5	ArrayList() - java.util.ArrayList

５．実験

開発したアプリケーションの有効性を確認するために山本[2]の研究で行われた実験を参考にした. 本研究の実験では本研究の手法と従来手法のコード補完でどちらがコード補完を行う際に適切なソースコードが補完リストの上位にきているかの比較を行う.

以下に実験の具体的な手順を示す.

1. MVC モデルで開発されたソースコードを用意する.
2. ソースコードの中からランダムでメソッドが存在する部分を選択する.
3. メソッドの部分を空欄にする.
4. 空欄にした部分で従来のコード補完を行い補完リストの表示をする.
5. 空欄にする前のメソッドが補完リストの何番目に表示されているのか記録する.
6. 4 と 5 の手順を本研究の手法のコード補完で行う.
7. 2～6 の手順を 100 回行う.

６．結果と考察

実験の結果, 本研究の手法の平均順位が 9.16, 従来のコード補完の平均順位が 10.94 となった. 実験結果のｔ検定を行ったところ P=0.010030761 となったため有意水準(α<0.05)のとき有意であることが確かめられた.

７．結論と今後の課題

本研究では MVC モデルでのプログラミングは領域によって出現するソースコードの傾向が違うことに注目した. そして, 開発者がコード補完を使用する際にソースコードを適切な選択肢を選べるようにソートするアプリケーションを開発した. その後, 評価実験により本研究の手法と従来の補完の比較を行ったところ, 補完リストの上位に適切な候補がきていることの有意差が確認されたため, 候補を見つけやすくなったことが確かめられた.

課題として本研究の手法でコード補完を行う時に補完したいソースコードがデータベースに存在しなかった場合, その部分での補完リストに表示されるソースコードがデータベースに存在しているとその数だけ本研究の手法の順位が下がってしまうことがある. この課題はデータベースに格納されていないソースコードを補完する際に起こる現象のため, 格納するデータを増やすことで改善する可能性がある. しかし, 汎用性の高いソースコードを格納してしまうと, どこで補完を行っても汎用性の高いソースコードの順位が高くなってしまうため, 格納するソースコードを吟味する必要がある.

参考文献

[1] 山本　哲男,”ソースコードコーパスを利用したメソッド呼び出し文補完手法,”情報処理学会論文誌, Vol.54, No.2, pp.903-911, Feb. 2013.

[2] 山本　哲男,“制御構造を考慮したソースコードコーパスに基づくメソッド呼び出し文補完手法,”情報処理学会論文誌, Vol.56, No.2, pp.682-691, Feb. 2015.