ライブラリ利用パターンを用いた ソフトウェア開発支援に関する研究

情報工学専攻　渥美 紀寿

ソフトウェア開発における再利用技術は，繰り返し利用されるノウハウをパターンとして蓄積し，それを利用してソフトウェアを開発する事によって生産性や信頼性を向上させる技術である．再利用技術はソフトウェア開発の要求分析，設計，コーディングの各行程において取り入れられているが，本研究では，最もコストのかかるコーディングにおける再利用を行う．コードの再利用は以下の手順で行われる．

1. 再利用可能なコード断片を見つける
2. 1 で見つけたコード断片をデータベースに蓄積し，管理する
3. データベースから検索し，必要なコード断片を取り出す

本研究ではライブラリに着目し，ライブラリを利用するためのノウハウを既存のソフトウェアから抽出し，再利用可能なコード断片として蓄積する手法を提案する．さらに，蓄積されたコード断片から効率的に検索するために，類似したコード断片を分類する手法を提案する．

本研究ではライブラリの利用パターンを関数呼出依存グラフとして抽出する手法を提案する．高級言語ではライブラリが用意されており，そのライブラリを用いてプログラムが作成される．操作は代入と基本的な算術・論理演算に限られており，入出力をはじめ文字列操作，整列操作等は全てライブラリとして提供されている．また，個々のドメインにおいて，汎用的な操作はライブラリとして作成され，そのドメイン内で利用される．そのため，プログラムを作成する際，ライブラリに関するコーディングノウハウが必要となる．

ライブラリに付随するマニュアルには，引数に与える値についての説明および返り値についての説明が記述されている．しかし，実際にライブラリを利用する際，引数には他のライブラリの返り値を与え，ライブラリを組み合わせて利用する．例えば，C 言語によるサーバからのメッセージを受け取るプログラムでは `socket` により通信するための *socket* を作成し，`connect` により，*socket* 上で接続を開始する．その後，`fdopen` により，*socket* を開き，`fgetc` などで，メッセージを読み込む．このようなライブラリの組み合わせは，マニュアルを参照しただけではわからない．そのため，開発者はライブラリを利用する際，マニュアルを参照するだけではなく，既存のソフトウェアから利用箇所を検索し，実際の利用例を参照する．

既存のソフトウェアからライブラリの利用例を参照する最も低レベルな手法は `grep` などの文字列検索ツールなどにより，ライブラリの名前を検索語として検索する手法である．

この検索では，変数や関数，文字列，コメントなどのコード中の各要素を区別する事ができず，ライブラリに特定して検索する事ができない．また，コードを構文解析して構文要素を指定して検索を行う手法も存在するが，膨大な検索結果が得られ，その中から求めている情報を取得する事が困難である．その検索結果には同じような利用例がたくさん含まれているため，何度も類似した利用例を参照する事となる．このような事を防ぐため，本研究では検索結果を類似度により分類する．

ライブラリの利用パターンを関数呼出依存グラフとして既存のソフトウェアから抽出する．関数呼出依存グラフでは，関数を節点，関数間のデータ依存関係および制御依存関係を辺として表現する．本研究では，特にライブラリ関数を中心として関数呼出依存グラフを構成する．ライブラリはソフトウェアの基本機能を表しているため，ライブラリ関数を中心とした関数呼出依存グラフはソフトウェアの機能を反映している．これにより開発者にライブラリの利用例として提示する事で，利用する際に必要となる情報を無駄な情報無く提示できる．

さらに，本研究では，検索結果として得られる関数呼出依存グラフを類似度により分類する．自然言語において，類似した文書を分類するために，TF・IDF 法が用いられる．これはある 1 つの文書に頻出する単語は，その文書において重要な単語と考えられるが，そのような単語でもあらゆる文書に出現する単語は重要ではないとし，重要な単語が共通して含まれる割合により分類している．この考えを参考に，関数呼出依存グラフの依存関係に重み付けをし，関数呼出依存グラフを分類する．これにより，類似したライブラリの組み合わせをまとめる事ができる．

Web ページの検索として良く知られている Google では各 Web ページの質をリンクされている数によって評価し，多数のページからリンクされているページを上位にランク付けする．これは膨大な検索結果全てを参照する事が困難であり，一般的には上位数件から数十件しか見ないため，このようなランク付けが必要となつ．この考えはソフトウェアにも必要である．本研究では，あらゆる関数呼出依存グラフに出現する依存関係はそのライブラリを利用する上で必要な依存関係であるとし，そのような依存関係を多く含む関数呼出依存グラフを上位にランク付けする．

実際に `FreeBSD 4.5-RELEASE` の `/usr/src/usr.sbin` にある平均 1,500 行のソフトウェア 152 個を対象に関数呼出依存グラフを抽出した．さらにその関数呼出依存グラフから効率的にライブラリの利用例を検索するシステムを実装した．この検索システムを利用し，利用頻度の高いライブラリに対して検索した結果，平均約 40 個の関数呼出依存グラフを平均 5 個のカテゴリに分類する事ができた．さらに個々のカテゴリにおいて，ライブラリの利用例をランク付けする事によって，返り値検査などのライブラリが必要とする処理を行なっている例を上位にランク付けする事ができた．