5D－5

リバースエンジニアリングのためのプログラム理解システムプロトタイプ「PRIPARE」(1) –概要–

佐藤徳幸†[1]　神保至[2]

情報処理振興事業協会(IPA)
技術センター

1　はじめに

昨今のダウンサイジング化の流れ及び景気後退にともなう新規ソフトウェア開発の絞り込み等によって、既存ソフトウェアの保守支援への期待が高まっている。

我々は1993年秋より、ソフトウェア・リバースエンジニアリングの支援を主目的として、認知科学に基づいたアプローチによりコンピュータによるプログラムの意味理解を行ない、C言語で書かれた事務処理プログラムのソースコードから詳細設計仕様書レベルの情報を抽出することを目標として研究開発を行なってきた。

本稿では、我々が試作を行なったプログラム理解システムプロトタイプ「PRIPARE」の概要を、我々が本テーマの研究を行なう事になった背景であるリバースエンジニアリングの動向の調査結果を踏まえながら報告をする。

2　ソフトウェア・リバースエンジニアリングの動向

実際の研究開発に先だって、我々の研究対象であるリバースエンジニアリング技術およびそれを取り巻く現状をあきらかにするために、調査検討を行なった。調査の手法は、インタビュー調査および文献調査である。インタビュー調査では、国内のリバースエンジニアリング関係の研究者や技術者の方々をその対象とさせて頂いた。その結果、現状のソフトウェア保守作業には次に挙げる問題点が存在する事がわかった。

2.1　ソフトウェア保守の問題点

保守の問題点は、「保守の対象に起因する問題」・「保守作業そのものに起因する問題」・「保守を行なう人間に起因する問題」にわけられる。

「保守の対象に起因する問題」を次に挙げる。

大規模プログラム　保守が深刻な問題となっているのは、銀行のオンラインシステムなどの規模の大きなものである。この規模のシステムでは、プログラム自体を読むことはいうまでもなく、プログラムの一覧表を見ることさえ人手では困難である。

非構造化プログラム　構造化分析やオブジェクト指向分析などの手法に則って開発を行なったプログラムは少数である等の理由のため、プログラムが構造化されていない場合が少なくない。その結果、それらプログラムは非常に理解しにくくなっている。

「保守作業そのものに起因する問題」を次に挙げる。

保守環境構築の困難さ　保守のためにはシステムが稼働している環境と同一の環境を用意しなければならない。しかし、これは多くの場合において費用の点で現実的ではない。

保守に必要な情報の欠如　ソフトウェアの開発についての方法論は色々と研究がなされてきたが、保守に焦点をあてた研究はほとんどなかった。一般的にソフトウェアのドキュメントの中では保守のためのものが少ないといわれる。しかし、保守に焦点をあてた研究がほとんどなかったということは、根本的に保守のためにどの様なドキュメントが必要かどうかが検討されてこなかったことを示している。

「保守を行なう人間に起因する問題」を次に挙げる。

モラルの問題　技術者は一般に新規開発を好み、保守を軽視する(または嫌う)傾向が強い。

プログラムの理解の困難さ　保守を行なう場合、そのプログラムの機能や構造や仕組みなどを理解しなければならない。しかし、一般的に開発担当者と保守担当者は異なることが多い。ゆえに、保守作業において最初に行なわなければならない作業はプログラムを理解することである。

これら問題点のうち、「プログラムの理解の困難さ」は最も早期に解決されなければならない。なぜならば、保守作業による修正箇所の特定も影響範囲の特定も、保守作業を行なう対象プログラムの理解が前提であるからである。さらに、保守作業に占める工数が、プログラムの理解と修正・拡張の実現ではほぼ半々の割合であるとの報告もされている[1]。この事からも、プログラム理解支援の重要さがわかる。

3　“PRIPARE”の概要

先に挙げた調査結果を踏まえ、我々はC言語で書かれた事務処理プログラムのソースコードからその詳細設計書レベルの

"PRIPARE": Program Understanding System Prototype for Software Reverse Engineering(1) –outline–

†SATOU Noriyuki (e-mail: noriyuki@stc.ipa.go.jp)
Software Technology Center,
Information-technology Promotion Agency, Japan
3-1-38 Shiba-kouen, Minato-ku, TOKYO 105 ,JAPAN
[1](株)日本コンピュータ研究所より出向中
[2](株)三菱総合研究所より出向中

仕様書を抽出することを目標として、プログラム理解システム“PRIPARE”の研究開発を行なってきた。“PRIPARE”の実行例を図1に示す。

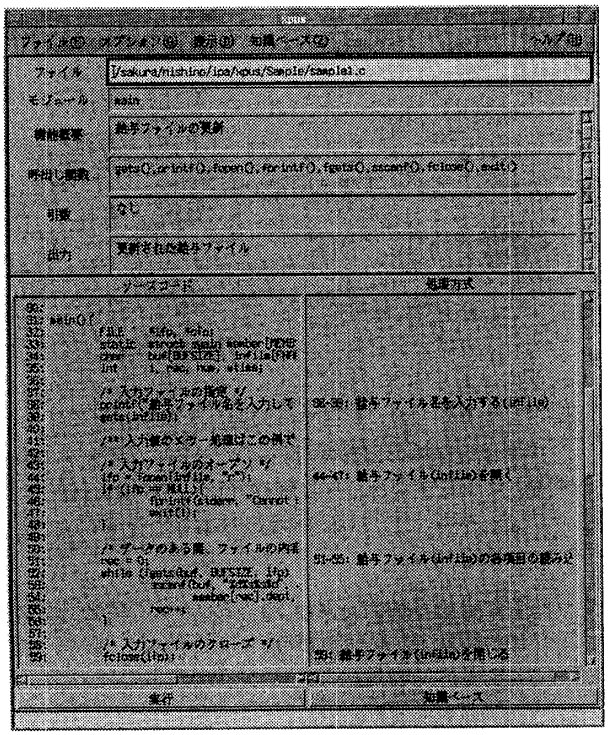

図 1: “PRIPARE”の実行例

3.1 システムの構成

本システムの構成は、正規化サブシステム、推論サブシステム、知識ベース(知識フレーム)、仕様生成サブシステム、ZEROサブシステム[2]、およびユーザインタフェース・サブシステムから成る。

3.2 システムへの入力

本システムへ入力すべき情報を次に示す。これら入力は、すべてユーザが行なう。

- ソースコード
  理解対象のプログラム(ファイル)を選択する。ファイルの形で保存されているものが理解対象のプログラムとなる。
- 業務領域(ドメイン)
  理解対象のプログラム(ファイル)を選択する時に、ユーザはそのプログラムがどの業務領域(ドメイン)に属するかを指定する。システムがドメインリストを表示するので、ユーザはこのリストの中から適切なドメインを選択する。
- 理解対象モジュール
  このシステムはモジュール単位(各関数毎)の推論を行なうため、推論開始前に理解対象プログラム中に含まれるモジュール名のリストを表示する。ユーザは、このリストの中から理解すべきモジュールを選択する。
- 入力値のデータ項目
  理解対象のプログラムに対して入力される値の意味、すなわちデータ項目を入力する。そのプログラム中で、ユーザからの値の入力を受け付けている部分やファイルから値を読み込んでいる部分があれば、システムがその度に入力値の意味の候補リストを表示するので、ユーザはこのリストの中から適切な意味を選択するか、あるいはリストの中に適切なものがない場合にはその意味をテキスト形式で入力する。

また、これらの情報以外に、ユーザは次の情報を必要に応じて与えることができる。

- 条件コンパイルオプションの設定
- 起動すべき推論エンジンの選択
- 推論のオプションの設定
- 内容を表示すべきインスタンスフレームの指定

3.3 システムからの出力

本システムから出力される主な情報を次に示す。

- モジュール全体の機能概要
- モジュール内部の各処理方式
- モジュールの引数および出力(戻り値あるいは処理結果)
- 呼出される関数リスト

また、これらの情報以外に、ユーザは次の情報を必要に応じて表示させることができる。

- 正規化処理によって生成された中間言語リスト
- 推論処理によって生成されたインスタンスフレームのツリー構造

4 おわりに

本稿では、我々が研究開発を行なってきたリバースエンジニアリングのためのプログラム理解システムプロトタイプ“PRIPARE”の研究開発の元となった背景であるソフトウェア保守の問題点とプロトタイプシステムの概要について述べた。

今後“PRIPARE”はPDSとしてanonymous ftpで公開する予定である。

参考文献

[1] 竹下亨 ソフトウェアの保守・再開発と再利用, 共立出版 (1992).

[2] 今井健志, 伊藤治之, 吉村貞徳, 上野晴樹 汎用フレーム・システムZERO−その概要とユーザ・インタフェースについて, 電子情報通信学会 人工知能と知識処理研究会, **AI-87-22** (1987), 35–42.