プログラム理解のリバースエンジニアリングへの応用

7N－7　中島歩[*1]　佐藤徳幸[*2]　神保至[*2]　古宮誠一[*2]　上野晴樹[*1]
[*1]東京電機大学理工学部経営工学科
[*2]情報処理振興事業協会

1　はじめに

現状のソフトウェア開発の多くが，既存ソフトウェアの修正や機能拡張の作業に費やされている．しかし，対象とするソフトウェアが古い，設計ドキュメントがない，開発担当者がいない等により保守作業に支障を与えている場合が多い．そこで既存ソフトウェアの保守作業を支援することを目的としたリバースエンジニアリング技術が注目を集めている．リバースエンジニアリングを実現するためには，既存ソフトウェアを理解する技術が必要であると考えられる．

本稿では，著者らが開発したプログラム理解システムALPUSのリバースエンジニアリングへの応用の有用性について述べる．

2　リバースエンジニアリング

リバースエンジニアリングとは，「システムの構成要素とそれらの相互関係を同定すること，システムを別の形式あるいはより高いレベルの抽象概念で表現することで対象システムを分析する工程」である[Chikofsky 90]．

リバースエンジニアリング実現するためには，各種の技術が必要であると考えられるが中でも既存ソフトウェアを理解する技術が最も重要な役割を果たす．[Biggerstaff 89]では，デザインリカバリー（リバースエンジニアリングのサブセット）を実現するにあたり，プログラム，ドメイン，言語構造などの設計パターンを知識ベース化したドメインモデルを利用しプログラムを理解している．また[四野見92]では，プログラムから構造化分析/設計で用いられている各種ダイヤグラム生成することによりプログラム理解を支援している．しかし，これらの技術はプログラムの構造（構文）レベルでの理解（理解支援）に終っており，意味の理解[上野 88]までは行っていない．そのため，より深い設計情報（意味的概念）まで抽出することは不可能である．

著者らは今までにプログラム理解システムALPUSを構築してきた．ALPUSはアルゴリズムを中心としたプログラミング知識を利用し，プログラムの意味理解を行うシステムである．その技術を，リバースエンジニアリングに応用することにより，構文的概念は勿論，意味的概念まで抽出できると考えている．

本研究におけるリバースエンジニアリングとは，「ソフトウェアを理解することにより，意味的概念を抽出すること」である．なお，現在のところ仕様の抽出までは考えていない．

3　プログラム理解

本章では，現在までに開発してきたプログラム理解システムALPUSを簡単に紹介する．なお，詳細は[中島 92]を参照されたい．

3.1　プログラミング知識

我々は，認知科学的実験および教科書から抽出したプログラミングに使われる知識を階層的に体系化し，知識ベース化している．プログラミング知識は，問題をアルゴリズム化し設計するための言語独立なプログラミング技法知識（意味論的知識）と具体的に特定の言語表現にするためのプログラミング言語知識（統語論的知識）に分けられると考えている．このことは1つの技法を，複数の言語表現にできることからも裏付けされる．

プログラミング技法知識は，知識の抽象度に焦点を当てることにより以下のようにレベル分け（階層化）できる．

- 問題解決概念知識レベル
- アルゴリズム知識レベル
- データ処理技法知識レベル

Program Understanding Application to Reverse Engineering

*1 Ayumi NAKAJIMA Haruki UENO
Department of Systems Engineering,Tokyo Denki University
Ishizaka,Hatoyama-cho,Hiki-gun,Saitama 350-03,Japan
*2 Noriyuki SATOU Itaru JINBO Seiichi KOMIYA
Information-Technology Promotion Agency,Japan(IPA)
1-38,Shibakoen 3-chome,Minato-ku,Tokyo 105,Japan

- データ操作技法知識レベル

また，プログラミング言語知識は，ある言語表現のプログラムを他の言語表現へ変換可能であること等から，PASCAL,Cなど個別のプログラミング言語知識レベルの上位に，すべてのプログラミング言語共通の概念的知識レベルが存在すると考えられる．つまり，プログラミング言語知識は2段の階層構造をしている．

なお，現在体系化し利用しているのはアルゴリズム知識レベル以下の知識である．

3.2 理解の方法

プログラムは通常，同じ処理を行うにしてもプログラマによって様々な書き方が存在し，プログラムを見やすくするために冗長性を含んだ書き方をするのが普通である．このような状態のままでは，プログラムは読みづらく，また理解しづらいので，ALPUSでは，理解の前処理としてプログラムを特定の形になるようにコードの書き換えを行っている．理解はこのプログラムに基づいて行われる．

理解の処理は，ボトムアップとトップダウンの組み合せで行う．まず，プログラム内の変数がどのような役割を持っているか同定する．プログラムで使われるであろう変数を役割によってモデル化しているので，それを手がかりにしてこの処理を行う．続いて，プログラムセグメントと各プログラミング知識を対応づける．対応づけがついたときそのプログラムセグメントは理解できたと考える．

この方法により，プログラミング知識の各レベルが同定できたと考える．

4 リバースエンジニアリングへの応用

4.1 設計情報

[Biggerstaff 89]，[四野見 92]など今までの技術では，プログラムの構造（構文）レベルでの設計情報のみしか抽出できなかったが，本システムでは前述したプログラミング知識を利用することにより，構文的概念から意味的概念まで抽出できる．

4.2 応用の有用性

ALPUSは初心者のプログラムを対象にした論理チェックシステムとして開発されてきたので，正しいプログラムは勿論，稚拙なあるいはミスを含むプログラムも理解できる．また，前述したALPUSでのプログラム理解の方法を踏まえ，この技術をリバースエンジニアリングへ応用することにより以下のような有用性が考えられる．

- ミス，稚拙な表現を含むプログラムにおいても実現可能である．
- 稚拙な表現を正しい（特定の）表現に変換可能である．
- 複数の言語が混在しているプログラムにおいても実現可能である．
- 構文的概念から意味的概念まで抽出できる．

5 おわりに

本稿では，プログラム理解のリバースエンジニアリングへの応用について述べた．ALPUSを応用することにより，数々の有用性が考えられるが，現状ではより上位の設計情報（仕様）を抽出することは不可能であり，また対象ソフトウェアを限定しなければならない．この問題点を解消するためには問題解決概念知識レベルの体系化が急務であろう．

参考文献

[Biggerstaff 89] Biggerstaff,T.J.:Design Recovery for Maintenance and Reuse,IEEE Computer, pp.36-49,1989.7.

[Chikofsky 90] E.J.Chikofsky et al.:Reverse Engineering and Design Recovery,IEEE Software, pp.13-17,1990.1.

[上野 89] 上野晴樹：プログラムの意味と理解，信学技法，AI88-24,1988.

[四野見 92] 四野見秀明，ほか：構造化分析/設計の方法論に基づいたプログラム理解支援ツール，信学技報，SS92-8,1992.

[中島 92] 中島歩，上野晴樹：プログラミング知識の表現およびプログラム理解システムへの応用，信学技報，KBSE92-4,1992.