3N－1

アプリケーション再構築のための リバースエンジニアリング技術

長橋　賢児　上原　三八
株式会社富士通研究所

1. はじめに

近年保守コスト増大の解消、プラットフォーム移行などを目的としてソフトウェア再構築、さらにはリバースエンジニアリングへの要求が高まっている。しかし従来の技術では生成される仕様書の記述レベルの低さや処理速度、再構築への利用方法などが課題であった。筆者らはアプリケーション再構築の支援を目的として、実用性に主眼をおいた研究開発を進めている。本稿では最近実際に使用を開始した「BRSPEC リバースシステム」の実現技術について述べる。

2. 実用化のためのアプローチ

実際に使われるリバースシステムを構築するため、ユーザからの要求をもとに次のような方針で開発を行っている。

- CASE ツールの仕様書を生成することにより生成結果を直接再構築に利用できるようにする。
- 従来の仕様書生成研究で対象外とされがちな構造化の崩れた「汚い」プログラムを積極的に処理対象とする
- コーディング規約などを利用する部分はカスタマイズ可能にして適用範囲を広げる
- 自動生成だけでなく修正やチェックの支援ツールを用意することで人間の作業も含めた全体的な作業効率化をめざす

3. システムの概要

本システムは COBOL プログラムを解析し各種情報を出力するリバーサコアと、リバーサコアおよびその出力の利用を支援するツール群からなる。

3.1. リバーサコア

リバーサコアは COBOL プログラムを解析し、リバース処理を行い CUD(後述)等の情報を出力する。リバーサコアの大部分は C++で記述され、Sun ワークステーションで動作する。

3.2. 出力仕様書

リバーサコアの出力する仕様書は CUD(チェック更新定義書)と呼ばれるもので、データ中心開発技法に基づく統合 CASE AA/BRMODELLER がプログラム自動生成に使用する業務仕様書の一つである。CUD は{変数・代入値・実行条件}の 3 つ組でファイルレコードの更新を表現する更新記述と、{エラー処理・エラー条件}の組でエラーチェックを表現するチェック記述を中心とする表形式の仕様書である。リバーサコアは更新記述とチェック記述を生成する。

3.3. 支援ツール

リバース処理前の予備調査や、生成仕様書の印刷のためのツールが用意されている。また生成された仕様書を再構築に利用するには生成結果の検査・調整、再利用部分の抽出作業が必要になる。このために対話的なツールを提供している。

4. 制御フロー解析

リバース処理は厳密な制御フロー解析に基づいて行われる。

4.1. ＳＴＥＭ

COBOL 言語のプログラムは GOTO 文や特殊な実行論理を持つ実行文のために制御構造が複雑である。文の実行条件の抽出などを容易にするために、STEM と呼ぶ独自の制御フロー表現を使用している。STEM は図 1のような木構造を基本とするデータ構造で、プログラム全体の分岐構造を明確に表現しておりリバース処理のための様々な分析に利用できる。木構造は単純な再帰処理で走査可能

Reverse Engineering Technology for Application Reengineering
NAGAHASHI Kenji, UEHARA Sanya
Fujitsu Laboratories Ltd.

なのでリバース処理は著しく簡単になり高速な処理が可能になっている。

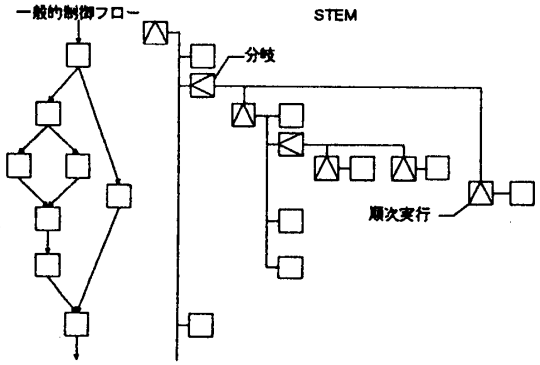

図 1: 一般的制御フローと STEM

4.2. 崩れた制御構造への対応

再構築の対象になるソフトウェアは古い開発技法や度重なる改変のため構造化の崩れが著しく、構造化されたプログラムだけを処理対象としていたのでは実用的になりえない。本システムは独自の手法によりGOTOループや形の崩れた分岐構造などを解析してSTEM上に表現し、リバース処理を行うことが可能である。別個の構造化ツールで前処理することも検討したが、1)生成仕様書と元のプログラムの対応が失われる 2)フラグ変数などが導入され生成仕様書の質が低下する、という理由からリバーサ内で処理を行う方法を選択した。

5. 更新仕様の生成

5.1. 表生成

更新仕様はSTEMを利用して実行文列に対応する末端のノードから上方に向かって実行条件を付加しながら表を併合していくことによって生成される。表に変換される範囲は変数への代入と条件分岐が連続する範囲としているが、この基準では人間の期待よりも小さい表になる傾向があり、範囲を拡大する方法を検討している。

5.2. 更新仕様への要求

更新仕様の生成には 1)CUD の仕様に由来する要求と 2) 人間から見た仕様書としての質に関する要求の2つがある。

更新仕様はまず仕様書としてプログラムの動作を忠実に、CUD の論理として表現していなければならない。この大前提の下で理解しやすい仕様書となるように、例えば同じ変数への代入は1箇所にまとめる、というような表の各行の配置の調整を行っている。これには文の相対位置関係の変化を伴うが、プログラムの論理を維持するためにデータの依存関係を解析して文の相対位置変化に制約を加えている。

6. チェック仕様の生成

チェック仕様はユーザが与えるヒントを元に、自動生成される。これは口語仕様書生成システム[1]で開発された手法をSTEMに適用したものである。ヒントはプログラム中のエラー処理の箇所を特定するために使用されるもので、エラーコード変数の名前やエラー処理セクション名などを与える。

7. 評価

7.1. 処理性能

実際のプログラムで試行した結果、大半のモジュールは2000行程度であり、これらに対して処理は40秒以内で終了する(SPARCstation-10での実行結果)という性能を達成した。これは十分実用的な速度であるといえるが、対象とするシステムには1Mステップを超えるものも多いのでさらに高速化が必要である。

7.2. 生産性

実際にはソースコードの予備調査、生成された仕様書の調整などが必要なため、リバース処理だけで作業が終わるわけではない。それを含めすべての作業量を人手の場合と比較したところ、初期の試行で3倍以上の生産性向上が報告された。

8. おわりに

本システムは第1段階の実用レベルを達成して再構築サービス業務への適用が始まっており、その中からリバースシステムに対する要求が多数獲得されている。今後それらの要求について検討・研究開発を行い、システムを拡張していく予定である。

参考文献

[1] "Reverse Engineering from COBOL to Narrative Specification," T. Yoshino, S. Uehara, T. Ookubo, S. Suguta, Y. Hotta, and M. Sonobe, Proc. Compsac '95 (to be published).