CASEツールによるアセンブリ言語の支援技術*

2M－3

岡山 敬　松本 憲幸　金子 博　森本 順子†

(株) 東芝‡

1　はじめに

ソフトウェア開発の生産性向上・効率化のために開発作業を一貫支援するさまざまなCASEツールが開発・提供されている。構造化分析手法を実現したCASEツールは、通常、システムの要求分析や仕様決定などの設計フェーズで適用される。この構造化分析・設計段階で作成された仕様書をもとにプログラム設計を行なっていく場合、高度な構造化記述が可能な高級言語で実装することにより、上流工程の構造化記述が保存される。

しかし、細かいビット制御やフラグ制御が必要なローエンドのマイコン制御用ソフトウェア開発の分野で利用されるアセンブリ言語の場合、上流工程における構造化記述がプログラム設計段階で保存されず、設計手法の高度化との間に大きなギャップが生じる。

本稿では、アセンブリ言語でのソフトウェア開発をCASEツールにより、支援する際の概略・詳細フローからプログラム変換、リバースエンジニアリングの問題や対策およびその効果についてCASEツールFORMULATORの実現方式をもとに述べる。

2　チャート・ソース双方向変換

2.1　変換モデル

本稿で考えるモデルは、概略フローや詳細フローを記述したチャートとアセンブリ言語プログラムから構成される(図1)。チャートの編集は、チャートエディタで行う。ソースの編集は、構文エディタで行う。チャートからソースへの変換は、ソース生成ツールで行う。ソースからチャートへの変換は、チャート逆生成ツールで行う。

構造化設計段階で作成するチャートは、記述内容のレベルにより次の通り分類できる。

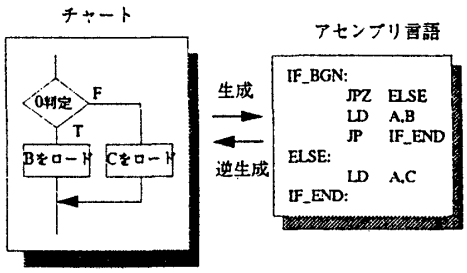

図1: チャート←→ソース変換構成図

- 概略フロー
- プログラムフロー
- 高レベル制御構造

この記述レベルとアセンブリ言語プログラムとの間の双方向変換について考える。

2.2　概略フロー

概略フローを作成してアセンブリ言語プログラムを生成する場合は、特に問題は生じない。概略フローから生成できるソースは、スケルトンであるから、実際のコード部分を構文エディタにより追加・編集して　完成させていく必要がある。

逆に、完成したソースから概略フローを生成する場合にも、特に問題は生じない。

2.3　プログラムフロー

通常、アセンブリ言語命令は、条件判定命令に分岐ラベルを記述するので、アセンブリ言語プログラムをチャート上に記述する場合には、条件判定やwhileなどのシンボルにラベルを記述することが必要になる。

しかし、チャート上にラベルまで記述するなら、チャートは、プログラム作成時に使用する構文エディタの代替としての意味しか持たなくなる。

---

*Support technique of assembly language by CASE tools

†Takashi Okayama, Noriyoshi Matsumoto, Hiroshi Kaneko, Junko Morimoto

‡TOSHIBA Corporation

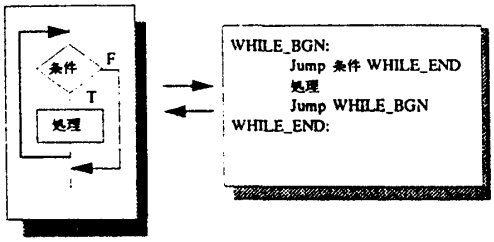

図 2: while 構造の例

よって、このプログラム設計方法を支援するには、効率的なチャート編集作業のための機能の改良や制御構造を表現するコードの一部の記述から構造全体のコードを自動生成するなど生成機能の強化が必要であろう。

一方、既存のアセンブリ言語プログラムからプログラムフローを逆生成することは、変換する記述間の記述レベルが同じであることから、既存アセンブリ言語プログラムの理解の容易性、成果物としてのドキュメント作成の自動化を支援する意味で非常に効果がある。

また、逆生成したプログラムフローをチャート上で編集するのではなく、表示することにより、チャートエディタをプログラムビューアとして利用することもプログラム開発において有効である。

2.4 高レベル制御構造

高レベル制御構造からソースを生成する場合も特に問題は生じない。ただし、チャート上に記述した高レベルの制御構造がプリミティブな制御構造にブレークダウンされる。

チャート逆生成により、プリミティブな制御構造から高レベル制御構造を取り出すことができるかが問題になる。制御構造の入り口ラベルを共有している記述が含まれている場合や複数の解釈が可能な構造が存在する場合など高レベルの制御構造を抽出することが難しくなる。

これらの記述が含まれている場合でも、パターンマッチングを行えば、高レベル制御構造の抽出が可能だが、別解もありうる。そのため、チャート逆生成時に、パターンマッチングのプライオリティ制御ができないと正しい制御構造を持ったチャートを得ることができない可能性がある。もし、プライオリティ制御が可能だとしても、プログラム中のすべての構造に対して、同一のプライオリティを適用すると正しいチャートが得られない可能性もありうる。

```
; while 条件
_L1:
;
jump cc, _L2
; 処理
;
; end_while
jump _L1
_L2:
```

図 3: コメントを生成したソースの例

この問題の解決方法として、ソース生成時に特有のコメントを出力しておく方法が考えられる (図 3)。

この方法を使用すれば、生成したコメントを削除・修正しない限りは、チャート上の制御構造が保存されるので、生成・逆生成による可逆性が保証される。

特有のコメントを出力する際に、単純に制御構造に 1 対 1 対応させると、冗長なコメントを生成する場合が存在する。一方、冗長コメントの生成を制御すると、チャート逆生成時に制御構造が復元できなくなる場合がある。したがって、ソース生成時に冗長コメントの抑制を行い、かつ、チャート逆生成時に制御構造の復元が可能なコメント生成規則が必要となる。

3 おわりに

概略・詳細フローとアセンブリ言語との間の双方向変換を CASE ツールで支援する際の問題や対策および効果について述べた。現在、本稿での対応方法を実現した CASE ツールを作成して評価を行なっている。

アセンブリ言語の開発環境は、まだ、整備されていない部分が多く、今後もさらにさまざまな支援ツールの作成および改良を行なっていく必要がある。

参考文献

[1] Hatley et.al. "リアルタイムシステムの構造化分析", 日経 BP 社,(1989).
[2] Yourdon. "構造化手法によるソフトウェア開発", 日経 BP 社,(1987).
[3] Martin et.al. "ソフトウェア構造化技法", 近代科学社,(1986).