

7G-4

HCPチャートデザイナー
—設計図を良構造にするための支援法—忠海 均, 米田 実男, 浅見 秀雄
NTT通研

はじめに

プログラムの設計にあたって、その論理構造を良くすることは重要である。従来、筆者らはHCPチャート記法によってプログラムの構造化を進める研究を行ってきた^[1]。本稿では、更にプログラムを良構造にする支援手法と、HCPチャートデザイナー（HCPチャートに基づく設計支援システム）^[2]での実現法について報告する。

1. プログラム構造化の問題点

構造化プログラミングの提案以来、設計法（設計上の規則）によってプログラムの構造を良くする試みが成されてきた^[3]。HCPチャート記法においては、WHAT（何のために）とHOW（どうやって）の階層概念に基づいてプログラムの構造化設計をすることを勧めている^[1]。しかし、HCPチャート記法を用いても、以下の問題点がある。

- ・プログラム構造の決定にあたっては設計者の意志に依存する部分が大きく^[4]、それを一意に定めることが困難である。
- ・プログラム構造化上のノウハウを記法上の規則として詳細に定めると、記法の適用域が狭まったり、設計者の記憶上の負担を増したりして好ましくない。
- ・複数の設計者がプログラムモジュールを分担して設計する開発形態では、設計の自由度が大き過ぎると、設計者のスキルによって設計書の品質にばらつきが出て好ましくない。

2. HCPチャート記法における構造化の原理

筆者らは、前記の問題点を解決して設計図を良構造にすることを、以下の考えに基づいて支援することとした。

形式的な構造チェックを行う

つまり、プログラムの構造を形式的なHCPチャート記号の並びとして捉えて、処理の意味には立ち入らずに構造化を阻害する並びをチェックすることにより、良構造化を支援する。ここでチェックにあたっての規則は、HCPチャート記法の基本となっている「プログラム構造化上の原理」から論理的に導

出できるHCPチャート記号の並びに関するものを用いる。この方法には以下の利点がある。

- ・処理の意味を解釈しなくても構造チェックができる。
- ・記法上の規則を詳細に定める必要がない。

「プログラム構造化上の原理」については、M. A. Jacksonをはじめとする^[5]多くの提案がある。しかし、これらは人間が処理の意味を理解して構造化を進めていく上では効果的であるが、それに基づいて機械支援できるほどには形式化されていない。

そこで、「プログラム構造化上の原理」を反映しているHCPチャート記法に対し、論理的な構造化の原理を以下に定義する。これらは、HCPチャート記法に限らずプログラムの構造化上妥当な形式化と考えられる。

① 等価性原理

ある処理記号の下位の処理記号の並びは、その記号と形式的に等価である〔図1〕。即ち、図1(a)の例では、処理Aは処理BとCの接続と形式的に等価である。これより、(b)や(c)のような並びは、等価な変換を行うとHCPチャートの構造を破壊するため、禁止される。

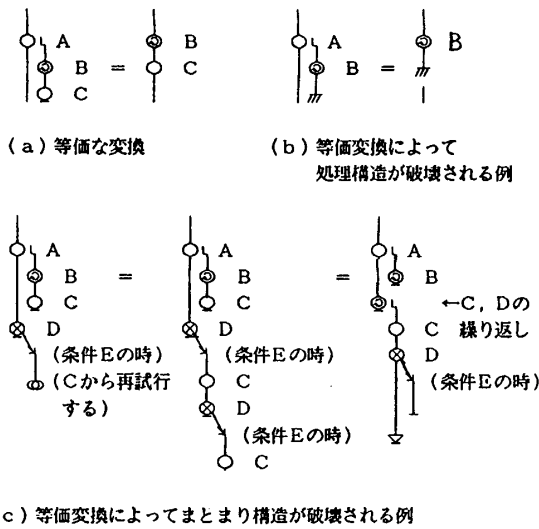
② 焦点の原理

時系列な手順である処理の並びを作る上で、主たる処理部分に焦点を置くと、手順は、主たる処理部分及びその準備と後始末の三つに分割される。即ち、処理記号の接続の個数は“3”が基本である。例えば、処理記号の個数が4個以上あってそれらが独立なデータを扱っていないような手順は、焦点が定まらず好ましくない。

③ 階層化原理

ある処理記号の実現手段が2つ以上の処理に分割されないような詳細化は、原則として高々2レベルしか続かない〔図2〕。即ち、このような詳細化は、他の処理との区分上必要な階層(a)の詳細化がたまたま1つの処理記号(b)からなり、それがたまたま1つの処理記号で表せる命令(c)の抽象化になっている場合のみである。

ここで、処理の抽象化とは、形式的には、N(1以上の整数)個の記号の集まりを1個の記号として表現することである。1個の記号の抽象化は高々1



[図1] 階層構造の等価性

レベルであり、複数レベルの抽象化は冗長で好ましくない[図3]。

3. HCPチャートデザイナーによる支援法

上記の原理に基づいた設計支援は、HCPチャートデザイナー(以下、HD)上で実現できる。支援法は以下のとおりである。

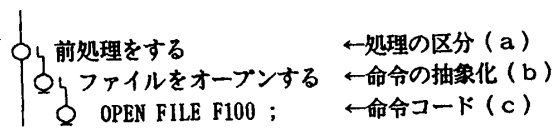
- (a) HCPチャート記法上、誤った記号の並びは投入不可能とする。
- (b) ①から③の3つの原理に照らして、構造上不適切な記号の並びは、HDがそれを検出して警告を発し、設計者に修正を促す。

ここで(a)は、HCPチャート記号の記法上の定義と、「等価性原理」から決まる記号列構文チャート[図4]から容易に実現できる。また(b)は、①から③の原理と記号の並びとを比較することにより検出できる。図5に(b)の実現イメージを示す。この例では、設計者のミスで、繰り返し記号とすべきところが一般記号に書き誤っていた。この例のように、処理説明の意味を解釈しないとミスが発見できないところでも、設計者に修正を促すことにより、バグを未然に防止することが可能となる。

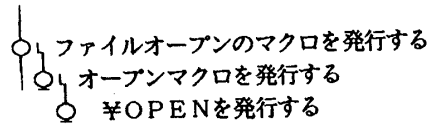
一般にプログラムの構造の悪いモジュール(分岐やジャンプが多いなど)にバグが集中することが多い^[6]。本機能は、モジュールの設計時に構造化を設計者に促すことにより、潜在バグを未然に防ぐことを狙っている。

おわりに

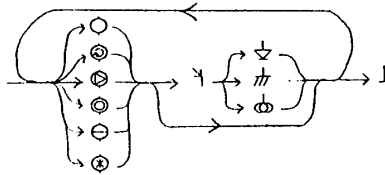
形式的な構造チェック機構により、HCPチャートの良構造化を支援できることを示した。しかし、2.に示した①から③の原理から導出できるルールを形式的に適用するアプローチには自ずと限界が予想される。今後は、形式的な記号の並びから、処理



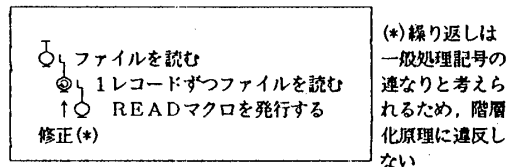
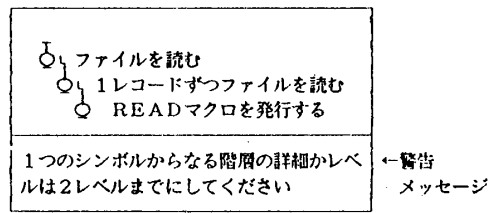
[図2] 階層化のレベル



[図3] 冗長な抽象化



[図4] HCPチャート記号の構文チャートの例



[図5] 構造のチェック例

の意味を含んだ構造上の性質を抽出する技術の確立が必要と考える。

参考文献

- [1] 花田, 佐藤ほか "DOCUMENTATION TECHNOLOGY FOR PACKING HIERACHICAL FUNCTION, DATA AND CONTROL STRUCTURES", PROCEEDING OF COMPCON FALL '81, PP. 284-290
- [2] 米田, 浅見, 忠海ほか 「設計部品の再利用を可能とするソフトウェアCADシステム」, 日科技連第6回ソフトウェアシンポジウム, 1986
- [3] O. J. DAHL et al. "STRUCTURED PROGRAMMING", ACADEMIC PRESS, 1972
- [4] 忠海ほか 「HCPチャートに基づくプログラム論理構造決定要因の一解析」, 情処第30回全国大会, 1985, pp. 699-700
- [5] M. A. Jackson "PRINCIPLES OF PROGRAM DESIGN", ACADEMIC PRESS, 1975
- [6] 花田ほか 「プログラム構造と信頼性に関する分析」, 情処論文誌, 第23巻第1号, JAN, 1982