

7S-5

COBOL-AP開発の標準化
- HCPチャートベースのプログラミングシステム CODA -

小林 勝哉 笠松 治雄 塚田 清治 林口 一雄
NTT 関西社内情報システム開発センタ

1・はじめに

本稿では、日本語HCPチャートからCOBOLソースコードを生成するチャートコンパイラ「CODA」の考えを紹介する。CODAはコーディング作業の標準化と自動化を実現し、COBOLソースよりも上位の設計情報であるHCPチャートそれ自体をソースとして扱えることから、論理エラーの減少、保守性の向上をはかることが出来る。標準化の作業には、5つのプロジェクトからなるワーキンググループを結成し、半年にわたり現状分析、表記法統一、記述実験までを行った。現在プロトタイプの開発を終え、平成2年度サービス開始を予定している。

CODA開発の背景としては、我々が開発する大規模統合情報システムのAP(アプリケーション・プログラム)がCOBOLで製造されており、現在保守段階のシステムも多く、最近のCASEに関する研究成果の実用化が重視されてきたことがあげられる。本年、我々は「3C」というCASEツールを順次開発し、社内で適用を開始しつつある。3Cとは、画面プロトタイプ支援ツール「CADDIE」、HCPチャートベースのプログラミングシステム「CODA」、COBOLシンボリック・クロスデバッグ「COLT」の3つのソフトウェアから構成されている。上記CASEツールを既存ツール群と統合し、上流にはワークステーション上の構造化分析ツール、中流にはパソコン上の「3C」、下流にはメインフレーム上のリモートVMを適用することにより、一貫した開発環境を整備中である。

2・開発思想

CODAは、標準化された日本語記述の木構造チャートをコンパイルし、COBOLソースを生成するシステムである。我々はCODAの木構造チャートに、段階的詳細化記述の可能なHCPチャート^[1]を記法として採用した。

CODAの目指すところは、次の諸点である。

- | |
|---|
| <p>1・標準化を推進
大胆に命令セットを制限
日本語表現のテンプレート化</p> <p>2・誰でも使えるCASEツール
マウスによるミニマム・オペレーション
パソコン上で実現</p> <p>3・ドキュメンテーションの機械化
インタフェース仕様書のDB化
日本語HCPチャート
保守性の向上</p> |
|---|

Standardization of COBOL Application Development Method
- CODA : A Programming System based on HCP Chart -
K. KOBAYASHI, H. KASAMATSU, S. TSUKADA, K. HAYASHIGUTI
Kansai Internal Information System Development Center,
Nippon Telegraph And Telephone Corporation

同様の木構造チャートコンパイラの中でも日本語表現からソース生成を行うものには、繁田ら^[3]、米川ら^[4]の例があるが、CODAは、次の特徴を有する。

- ①標準化を行ったこと、
- ②日本語からソースコードを完全生成すること、
- ③インタフェースデータをDB化して活用すること、
- ④適用性が確認されている(記述実験レベル)

3・処理概要

CODAは、図1に示すように、詳細設計工程で作成されたインタフェースデータ仕様(テーブル/ファイル構造)をDB化したドキュメントDBと、プログラム設計工程で作成された日本語HCPチャートファイルを入力として、構造化COBOLのソースコードを生成する。HCPコンパイル時に人の介入は一切必要としない。

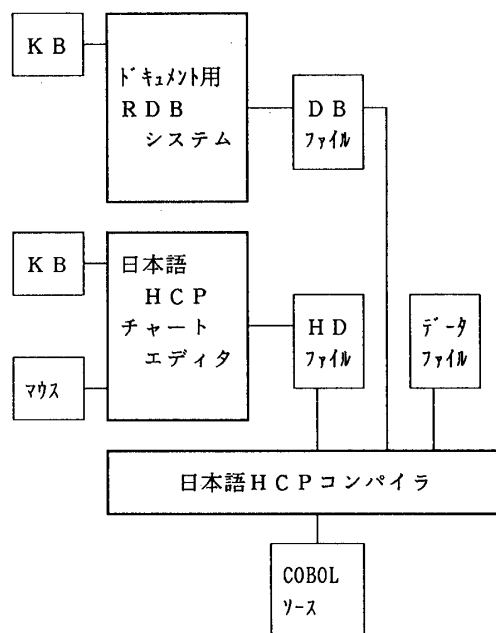


図1 CODAの構成

3・1 ドキュメントDBシステム

ドキュメントDBは、ファイルサーバー上におかれるRDBであり、詳細設計工程完了時に作成するものである。このDBには、テーブル構造、ファイル構造、共通ルーチン仕様などが管理されており、すべてこのDBを原本として印刷、修正が行われる。

3・2 日本語HCPチャートエディタ

日本語HCPチャートを作成・修正するために、専用のHCPチャートエディタがあり、標準化された日本語表現と、インタフェースデータの日本語名票の参照・引用を可能としている。操作はマウスによるポップアップメニュー選択により行われ、キーボード入力ほとんど必要としない。作成されるHCPファイルは、HCP情報交換用標準情報形式「HCPF」^[2]を採用しており、他のHCPエディタとファイル互換を有する。

3・3 日本語HCPコンパイラ

HCPファイルを入力としてデータ部、命令部をコンパイルし、COBOLソースを生成する。

データ部コンパイラでは、補足票ファイルを参照し、プログラムID確認、パラメータ抽出、環境部・データ部のインクルード指定ファイルの結合、画面・帳票定義データの結合などを行う。

命令部コンパイラでは、HCPファイル解析、構造抽出、日本語解析、コード生成を行う。構造化COBOLにより、HCPチャートと非常によく対応する。

4・標準化の効果

4・1 標準化までのステップ

CODAの仕様検討に当たり、社内の大規模オンラインAPやバッチAPを開発している5プロジェクトの代表からなるWG(ワキガ・グループ)を結成し、6カ月をかけて現状分析、表記法の統一、記述実験を行った。

具体的には、ソース分析ユーティリティの作成、COBOL予約語のヒストグラム分析、日本語表記、テーブル/ファイル構造、フィールド属性表記の統一、生成するCOBOL言語の予約語の統一、実アプリケーションによる記述実験を経て言語仕様、生成仕様を決定した。

記述実験の結果、オンライン、バッチ、ユーティリティのいずれの分野でも充分な記述能力が確認され、チャートのドキュメントとしての分かりやすさ、メンテナンス性についても効果が確認された。

標準化された規約の例示をするために、現在のプロトタイプシステムで生成されたソースコードを図2に示す。

```
* 4ビット用HEX変換域
      03 NUM    PIC 9(4) COMP VALUE 0.
* 4ビット変換後文字
      05 MOJI   PIC X(1).
* 変換後データ(変換後インデックス)
      05 OUTB  PIC X(1) OCCURS 2 INDEXED DO.

* PRCOD3 CASE 英字変換の要否を判定する
* CASE1 CASE条件(4ビット用HEX変換域 > 9)
      EVALUATE TRUE WHEN (NUM > 9)
* PRCOD1 ○処理 4ビット用HEX変換域 =
* (4ビット用HEX変換域 + 55) を計算する
      COMPUTE NUM = (NUM + 55)
* CASE2 ELSE条件
      WHEN OTHER
* PRCOD1 ○処理 4ビット用HEX変換域 =
* (4ビット用HEX変換域 + 48) を計算する
      COMPUTE NUM = (NUM + 48)
      END-EVALUATE
* PRCOD1 ○処理 4ビット変換後文字を
* 変換後データ(変換後インデックス)に転送する
      MOVE MOJI TO OUTB(DO)
```

図2 CODA生成コード例

4・2 標準化のメリット

我々は、以下に示す3つの標準化を行った。

① COBOL予約語・構文の標準化

COBOLは非常にバリエーションが豊富な言語であり、同じ処理の記述に要する構文や予約語が一意に決まらない場合がある。CODAでは生成する構文、予約語を社内仕様として限定し、統一された構文を生成する。したがって限られた範囲のCOBOL言語仕様を教育すればよい。

② 日本語構文の標準化

日本語というと、自由文脈が必須と考えがちであるが、実はその必要性はほとんど無い。CODAでは、HCPチャートを採用しているため、WhatとHowを書き分けることができ、段階的詳細化により上位のWhat記述にて単なるコメントとして自由に処理内容を表すことが出来る。従って、How記述の日本語構文は単純化しても意図は十分に伝わる。

我々は、HCPの1シンボルをほぼ1命令に対応できるレベルの詳細チャートまで作成すること、および入力テンプレートのサポートを前提として、日本語表記法を「分かち書きされた限定語彙の単文」とした。

入力支援系の充実により、入力の煩雑さはほとんど問題とならず、かえって読みやすいとの評価を得た。

③ HCPレベルの構造表記の標準化

CODAでは、HCPの構造表記をも標準化している。これは、COBOLのSORT, SEARCHなどの高機能命令においては、パラメータ選択の前処理、処理結果編集の後処理までもが1命令としてサポートされており、HCPチャートの表記上は、1シンボルではなく標準化した構造表記を規定した。これにより、前処理、後処理が明確となり、分かりやすい。今後、チャートレベルの構造設計を支援するエキスパートシステムをバックエンドプロセッサとして開発していく予定である。

5・あとがき

日本語HCPチャートからCOBOLソースコードを生成するチャートコンパイラ「CODA」の考えを紹介した。本システムはコーディング作業の標準化と完全自動化を実現し、HCPチャートというCOBOLソースよりも上位の設計情報それ自体をソースとして扱えることから、コーディングの自動化、論理エラーの減少、保守性の向上を行うことが出来る。ただし、実際にHCPチャートだけをソースコードとしてメンテナンスしているかどうかは、ひとえにドキュメントDBのメンテナンスがプロジェクトとして責任を持って行われていくかどうかにかかっている。

今後、標準化の効果と上位レベルのチャート情報のソース化の効果を具体的に評価していきたいと考えている。

参考文献

- [1] 花田, 「プログラム設計図法」, 企画センタ, 1983
- [2] 加藤、神谷, 「HCPファイルの編成法」, 第35回情処全大 4Z-1 pp1163-4, 1988
- [3] 繁田、平井、青柳、蓮見, 「HCPチャートからのソースコード自動生成の一考察」, 第35回情処全大 5Z-1 pp1177-8, 1988
- [4] 米川、木原、林, 「パソコンにおけるYACIIプログラミングシステムの評価」, 第35回情処全大 5Z-5 pp1185-6, 1988