ソフトウェア工学 43－4
（１９８５．７．３１）

# 設計図を用いたプログラム試験方式

米田 實男　加藤 保夫　浅見 秀雄　忠海 均
（ＮＴＴ ソフトウェア生産技術研究所）

## １．はじめに

プログラム開発の試験工程では、プログラム設計書にもとづき実際に計算機を使用して機能確認試験が実施される。試験の一方法として、プログラムリストをもとにチェックポイントを設定し、試験データを投入してプログラムの実行状況を収集し、プログラムリスト／プログラム設計図と対応して収集した実行状況を解析する方法があり、重要な試験作業となっている。本作業はかなりの稼働が掛かるのみならず、高度な知識を必要とするため設計者の能力に依存しており、作業効率の向上には限界があった。

本問題に対処するため、データベース化されたプログラム設計図上にチェックポイントを設定し、設計図から自動生成されたプログラムの実行状況を設計図上に表示する試験方式を考案した。本試験方式は、①従来の文字でなく設計図という図形によるマンマシンインタフェース、②設計図作成から試験までの自動化、③画面表示を用いたデバグによるペーパレス化、等を実現することにより作業能率の改善を狙いとしている。

本稿では、設計図を用いたプログラム試験方式について、その考え方、構成法について報告する。

## ２．試験の現状と課題

プログラム開発における試験として、①製造工程ではプログラム設計図をもとにコーディングしたコードリストがプログラム設計図通りであるか否かの机上確認試験、及び②試験工程ではコードリストをコンパイルし計算機上で実際に走行させて確認するマシン確認試験が行われる。

マシン確認試験をする場合、プログラム設計書をもとに試験に先立って、試験機能項目、試験環境、試験データ、予想される結果等の試験仕様が作成される。試験仕様に従って計算機を使用して試験を実施し、出力されたプログラムの実行結果と予想した結果とを比較して、プログラムの機能を確認する。一般にプログラムにはバグが内在していることから、トラブルが発生し目的とする結果が得られない。プログラムのトラブル解析をするために、試験データからプログラムの動作を予測してプログラムリスト上にチェックポイントを設定し、チェックポイントの通過状況、内部データの変化等の実行状況を収集する。得られた実行状況からトラブル発生原因を解析し、ソースプログラムの修正、プログラム設計書の修正等が行われる。

### ２．１ チェックポイントの設定

チェックポイントは、試験項目に応じた必要十分な実行状況を収集できる様に設定しなければならない。必要とする実行状況を得るためのチェックポイントが設定されていない場合、同じ試験を再度行わなければならないため、試験工数の増加、試験期間の遅延等の問題が発生する。更に、計算機パワー、用紙等のムダ使い等の問題も発生する。また逆に当該試験と関係ないチェックポイントが設定されている場合（既に終了した試験のためのチェックポイント等）、目的とする実行状況の他に当該試験と関係ない実行状況が出力される。出力された実行状況の中に不必要な実行状況が混在すると、実行状況の抽出作業が必要になるため、被疑箇所の絞り込みに時間が掛かることになり、試験工数の増加、試験期間の遅延等の問題が発生する。更に、チェックポイント解析のための計算機パワー、情報出力するための多量な用紙を必要とする。現状ではチェックポイントの設定／解除は試験実施者が行っており、大きな負担になっている。また必ずしも適切なチェックポイントの設定ができていないため、効率的な試験が行われていないのが実状である。

今までにチェックポイントを自動的に設定するシステムの研究[1] [2]が行われているが、ソースプログラムリストをベースにしたチェックポイントの設定であり、①設計者の意図の反映、②設定の容易性等の問題がある。従って、プログラム設計者の意図を反映するにはプログラム設計図に基づいたチェックポイントの設定を容易に可能とすることが必要である。

## ２．２　実行状況の解析

実行状況として収集する情報には、実行位置を示す情報（メモリアドレス等）、内部データの格納位置を示す情報とそこに格納された内容そのものの情報がある。得られた実行状況をもとにトラブルを解析するためには、プログラム設計書（プログラム設計図）とプログラムロジックとの不一致箇所を明確にし、トラブル発生原因を解明しなければならない。不一致箇所の検出は具体的には以下の手順で行われる。

①　ソースプログラムリスト上で実行状況をもとにプログラムの動作を確認する。

②　上記確認内容とプログラム設計書（プログラム設計図）と対比する。

ここでトラブル解析をスムーズに行うためには、トラブル解析者がプログラム設計図の内容を十分に把握していることが必要であり、トラブル解析者への負担またはトラブル解析の遅れ（試験の遅れ）という問題が発生することになる。また、実行状況の情報には目的としない情報も含まれていることから、さらにトラブル解析を困難にしている。

実行状況の情報はプログラムが実行した時系列に従って、１６進表現、キャラクタ表現等で表示されることが多い。これに対してマンマシンインタフェースの向上の観点から、実行位置のプログラムリスト上への表示、データをその意味に合った図形で表示、データの変化を動画で表示等の研究[3]が行われている。しかし、設計者の意図を反映したプログラム設計図とは関連付けられていないことから、トラブル解析としてのマンマシンインタフェースの向上が十分に図られていない。

## ３．設計図を用いた試験方式の位置付け

２章で述べた問題点に対処するため、プログラム設計図を用いた試験方式を考案し、設計図試験システムとして現在試作研究を進めている。[4]

設計図試験システムの位置づけを図１に示す。本試験システムは、プログラムの設計から試験（プログラムの実行）の各過程で生成される設計図面、ソースコード、オブジェクトの各種情報を入力し、実行状況を設計図面上に表示する。

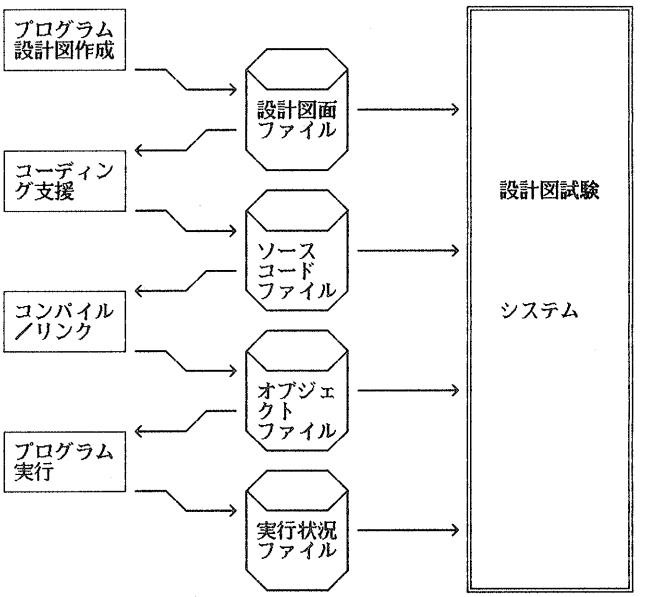


図１　設計図試験システムの位置付け

プログラム設計図は、ＮＴＴで開発した設計図法であるＨＣＰチャート記法[5] [6]を採用した。なおＨＣＰチャート記法にもとづく設計図の作成機能及びコーディング支援機能として、ＨＣＰチャートエディタを開発している。[7] [8] [9]

## ３．１　ＨＣＰチャートの採用理由

プログラミング用のドキュメントとして、従来のフローチャートに替り記述レベルの高水準化を図った、ＮＳチャート[10]、ＰＡＤ[11]、ＨＣＰチャート[5] [6]等の技法が提唱されている。これらについては文献[12]に解説されている。本検討においてＨＣＰチャートを採用した観点を以下に示す。なお、ＨＣＰチャートの特徴については付録に示す。

**(1) チェックポイント設定の容易性**

ＮＳチャート、ＰＡＤ等はプログラムのコーディングイメージの階層をそのまま表現しているが、ＨＣＰチャートは処理目的（ＷＨＡＴ）と実現手段（ＨＯＷ）とを対にして階層化して表現している。従って、特定の階層についてチェックポイントを設定することにより、処理の区切りを総て、かつ論理に沿った均質な設定を行うことができる。即ち階層を利用することによりチェックポイントを容易に設定／解除することが可能であり、試験の進捗に適合した設定を容易に行うことができる。

**(2) 実行状況との対応の容易性**

ＨＣＰチャートはＮＳチャート、ＰＡＤ等に比べ処理方式、処理目的、及び実現手段が明快に表現されているため、設計者は処理の流れを一意に読むことが出来る。このため、デバグにおいてプログラムの動作を理解しやすい。

**(3) ソースコードとの対応の容易性**

ＨＣＰチャートを表現するためのデータ構造は、シンボルの順列で規定することが出来る。[9] すなわちソースと同様に直線上に並べることが可能である。従ってＨＣＰチャートからソースの構造及び一部コードを自動的に生成することが可能であり、ＨＣＰチャート上のチェックポイントをソース上でラベルを挿入することにより、容易に対応づけることができる。

## ３．２　ＨＣＰチャートエディタ[7] [8] [9]

ＨＣＰチャートエディタはパソコン上で動作するプログラムであり、現在以下の機能を提供している。

**(1) ＨＣＰチャートの作成／更新機能**

ＨＣＰチャート記法に基づいたワープロ的な操作により、ＨＣＰチャートの処理／制御の部分を容易に作成／更新できる。例えば、追加、挿入、削除、移動、複写等の各機能はシンボルを単位として操作可能である。また、ＨＣＰチャート記法に基づいたシンタックスチェック等が行われる。従って、試験に伴うプログラム設計図の修正を、容易に行うことができる。ＨＣＰチャートエディタの画面例を図２に示す。

**(2) プログラムコードの部分自動生成機能**

ＨＣＰチャートをもとにプログラムコードの一部を自動に生成する。例えば、Ａｄａ言語を使用する場合、現在は以下のコードを自動生成している。

① プログラムの開始／終了（PROCEDURE／END）
② 繰り返しの開始／終了（FOR LOOP／END LOOP）
③ 振分けの開始／終了（CASE IS／WHEN =>／WHEN OTHERS =>／END CASE）
④ 制御の移行（RETURN）
⑤ コメント（-- 処理説明 ）

当該機能は、チェックポイント情報を収集する際にプログラム設計図とプログラムコードとを対応づけるための重要な機能である。

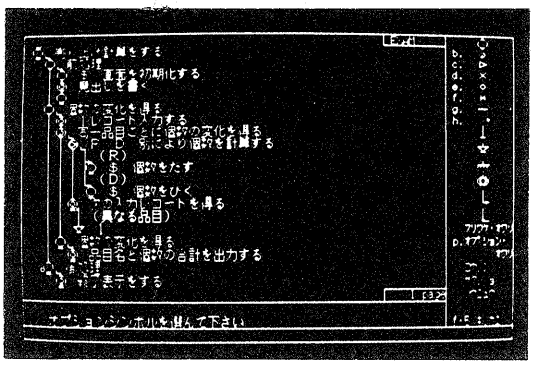

図2　ＨＣＰチャートエディタの画面例

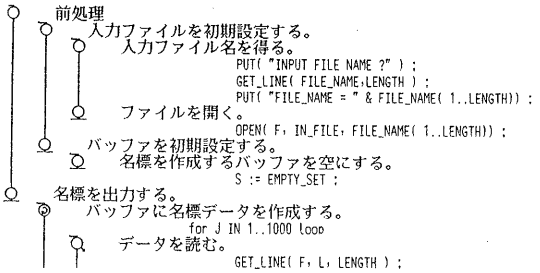


図3　ＨＣＰチャートとコードの同時印刷例

(3) コーディング支援機能

自動生成されたプログラムコードに対して、ＨＣＰチャートを参照しながらコーディングを行うことができる。本機能においても、ＨＣＰチャートの作成／更新と同様にワープロ的な操作が可能である。本機能における特徴として、自動生成された部分については更新不可能なようにプロテクトしている。自動生成された部分を更新するためには、先ずＨＣＰチャートを修正させることにより、設計図とコードとの不一致を生じにくくさせている。

当該機能ではＨＣＰチャートとプログラムコードを対応つけて管理していることから、ＨＣＰチャートとコードを一緒に表示／印刷（図３）することが可能とあり、机上確認試験を省力化できる。

(4) その他の機能

作成されたＨＣＰチャート／コード等の印刷機能、ＦＤ上のファイル管理機能、データ通信センタとの通信機能等がある。

## ４．試験システムの構成

設計図試験システムは、チェックポイント設定部、合成表示部から構成される。図４にシステム構成を示す。なお、本稿では試験システムとしてソースプログラムリスト上への表示について特に述べていないが、実用に供するシステムを構築する場合には、ＨＣＰチャート上への表示とソースプログラムリスト上への表示の両機能が必要になると考えられる。

### ４．１　構成方式について

(1) 実行状況の収集方法

プログラム実行時に実行状況を収集する方法として、①ＨＣＰチャート上の全部のシンボルについて収集する全体収集方法と、②指定されたシンボルについてのみ収集する部分収集方法がある。全体収集方法は部分収集方法に比べ、①実行時間が長い、②必要ファイル量が大きい等の問題があるが、以下の観点から全体収集方法を採用した。

(a) 多様な情報が収集されていることから、利用者の要求に応じた多様な合成表示が可能となる。

(b) 言語プロセッサ等が提供している既存のトレーサ等のデバグ機能が有効に利用可能となる。

(2) チェックポイントファイル

チェックポイントファイルの作成については、合成表示時の処理速度の向上の観点から、利用者が画面に表示しようとするチェックポイントと、画面に表示するために必要なマッピングの情報を統合して作成する方法

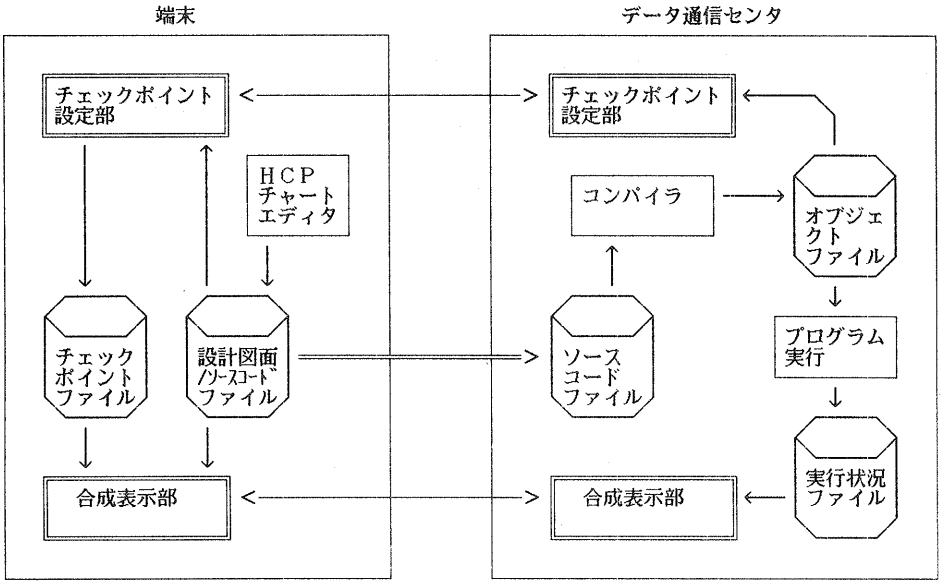


図4　システム構成

を採用した。本方式を採用することによりチェックポイント設定時各種ファイルを参照しなければならないため、チェックポイント設定の容易性が低下することが考えられる。この対処として、チェックポイント設定に必要な中間情報を収集する前段の処理を設定する方法がある。

## 4．2　チェックポイント設定部

ＨＣＰチャートの各シンボルに対応して、チェックポイントの指示を可能にするとともに、合成表示のためのマッピング情報を収集する。具体的には、チェックポイントとして指定された設計図面ファイル上のシンボルのレコード番号と、実行状況として出力される実行位置情報とのマッピング情報を生成する。内部データの格納位置情報についても同様に収集する。生成されたマッピング情報は、位置情報による検索を可能とする。また、内部データを論理的に表示するために必要な属性情報については、格納位置情報対応に生成する。

ＨＣＰチャート上の指定されたシンボルに対応するレコード番号から対応する実行位置情報／格納位置情報を得るには、以下の手順により可能である。

①　設計図面／ソースコードファイルから指定されたレコードに付与されたラベル名を取得

②　オブジェクトファイルからラベル名に対応する実行位置情報を取得

## 4．3　合成表示部

各種の情報をもとに、プログラムの実行状況をＨＣＰチャート上に表示する。

(1) 表示機能

(a) 実行位置の表示

設計図面ファイルをもとにＨＣＰチャート（処理／制御）を表示し、実行位置を強調表示する。ＨＣＰチャート上の実行位置は、チェックポイントファイルを参照して実行位置情報から変換する。

(b) データの表示

設計図面ファイルをもとにＨＣＰチャート（データ）を表示し、内部データ内容を表示する。ＨＣＰチャート上の内部データの位置は、チェックポイントファイルを参照して格納位置情報から変換する。また、内部データを論理的に表示するために必要な属性情報についても、チェックポイントファイルを参照する。

(2) 表示契機

合成表示の契機としては、以下の２方式が考えられる。

① オンライン表示＝＝＝プログラムの実行と同時に合成表示する。

② オフライン表示＝＝＝プログラムの実行時は一旦ファイルに出力し、後刻合成表示する。

これらの方式は表１に示す特徴がある。何れの方式を適用するかは、運用に合わせた選択が重要と考えられる。例えばパンコン上でのプログラム開発のように、制御の移行が比較的簡単な且つ走行環境が制約されている場合は、オンライン表示が有効と考えられる。逆に大型計算機上でのプログラム開発では、オフライン表示が有効と考えられる。

表１　合成表示の方式

| | オンライン表示 | オフライン表示 |
|---|---|---|
| 即時性 | 良い | 悪い |
| 機能 | 高度な機能は実現不可能<br>時系列順表示<br>表示速度の変更は困難 | 高度な機能を実現可能<br>逆転表示<br>表示速度可変 |
| 処理能力 | プログラム実行速度と関連 | 表示処理速度と関連 |
| 実行状況ファイル | 不要 | 実行状況の情報量に依存 |

## ５．実験システム

試作した合成表示部の実行位置表示のためのプログラムについて、図５に構成を、図６に表示中の画面例を示す。ここで、ＨＣＰチャートはＨＣＰチャートエディタにより、実行状況は実行状況作成ツールにより作成している。

実行位置表示プログラムは、以下の機能を具備している。

① 制御の移行関係がある複数のモジュールの表示が可能

② 表示モードとして自動／手動の切り替えが可能

③ 自動表示速度を切り替え可能

④ 実行状況にもとづいた逆順表示が可能

⑤ 表示するＨＣＰチャート上の詳細化レベルを指定可能

本試作プログラムで具備した実行時間軸を変更する機能は、被疑箇所の絞り込み作業の効率を高め得るという感触を得ている。またＨＣＰチャート上で実行位置を表示することにより、処理の流れを把握することが容易になった。

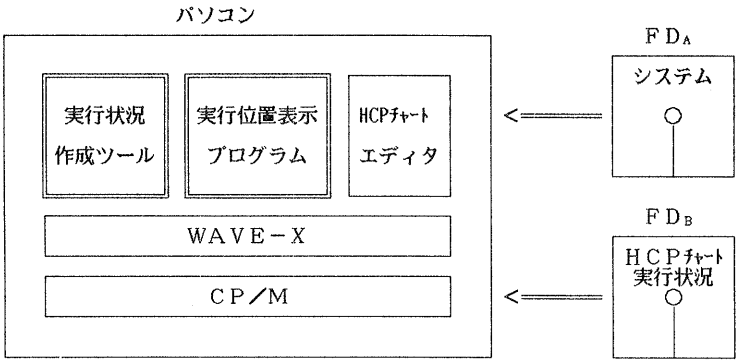


図５　実現システムの構成

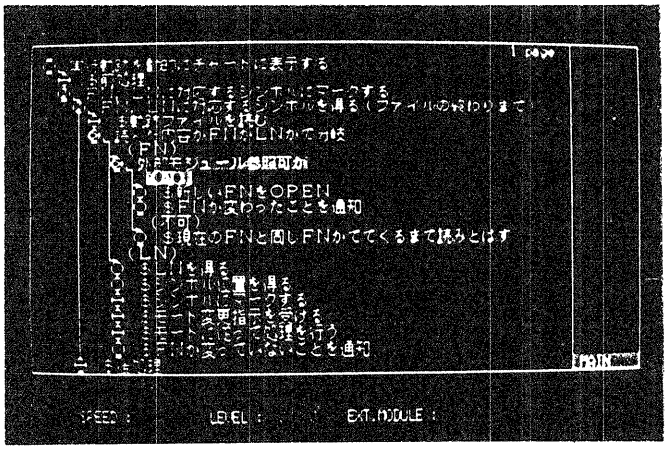


図６　実行位置表示の例

## ６．おわりに

設計者の意図が反映されたプログラム設計図、すなわちＨＣＰチャートをプログラムの机上確認試験だけでなくマシン確認試験に利用することにより、さらに作業能率の改善が可能であることを提案した。今後、チェックポイント設定部の試作、合成表示部のマンマシンインタフェースの改善等を実施し、総合化した試験系として評価を実施する予定である。

**【謝辞】**

日頃御指導いただくソフトウェア生産技術研究所の花田所長、伊東室長に感謝いたします。また実験システムの試作に御協力いただいた横田和隆氏（東大）にお礼申し上げます。

【付録　ＨＣＰチャートの特徴】

ＨＣＰチャートは以下の特徴がある。

① データの構造を階層的に記述できる。（データ構造）

② 処理の概要と詳細とを対応づけながら階層的に記述できる。（処理構造）

③ 制御の流れを記述できる。（制御構造）

④ データと処理との関係を明記できる。

⑤ プログラムの概略的機能から詳細な実現方法までを一つのドキュメント体系の中に一貫した記法で記述できる。

ＨＣＰチャートを用いることにより、プログラム論理をわかりやすく、コンパクトに表現することができ、プログラムの生産性、保守性が向上する。

【参考文献】

[1] 花田、岡、永瀬：フロー解析技法を応用したプログラムテスト法，情報処理ソフトウェア工学研究会，18-3(1981)

[2] 春原、大井、関本、中村：高位言語デバッギングシステム ＳＯＬＤＡ，情報処理，Vol.20，No.5，pp.405-411(1979)

[3] 磯田、下村、小野：データの動画表示機能を持つビジュアルデバッガシステム ＶＩＰＳ，情報処理学会第29回全国大会，4R-12，pp.667-668(1984)

[4] 米田、高木、忠海、横田：設計図を用いたプログラム試験方式，情報処理学会第30回全国大会，5S-1，pp.645-646(1985)

[5] 佐藤、浅見：フローチャート階層的表現のための一提案，情報処理学会第20回全国大会，2I-9，pp.285-286(1979)

[6] 花田：プログラム設計図法，企画センター(1983)

[7] 佐藤、浅見：ＨＣＰチャートを用いたプログラム作成支援システムの構想，情報処理学会第24回全国大会，5P-4，pp.423-424(1982)

[8] 米田、田野実：図を用いたプログラム設計における処理記述入力法，情報処理学会第28全国大会，3K-6，pp.603-604(1984).

[9] 忠海：ＨＣＰチャートエディタコマンド定式化の試み，情報処理学会第29回全国大会，4R-2，pp.647-648(1984)

[10]Nassi,I. and Shneiderman,B.：FLOWCHART TECHNIQUES FOR STRUCTURED PROGRAMMING，SIGPLAN Notices(Aug.1973)

[11]二村、川合、堤：問題分析図によるプログラムの設計と作成，情報処理学会第20回全国大会，2I-1，pp.269-270(1979)

[12]佐藤：プログラミング用ドキュメンテーション，情報処理，Vol.22，No.5，pp.383-389(1981)