マイクロコンピュータとワークステーション　69－4
（１９９１．１０．１１）

ソフトウェア開発の問題とＴＳチャートの導入

石島敏夫　小林正道
アライド情報システム株式会社

ソフトウェア開発の生産性向上について様々な提案がなされている。
ドキュメントの重要性についてもいろいろと言われているが、実際には本当に役に立つドキュメントを残していない事もしばしばある。
当社においてはプログラム設計のドキュメントとしてフローチャートや構造化チャートを使用しているが、このたびＴＳチャートという手法を導入してみた。
ＴＳチャート導入の１事例として紹介する。

PROBLEM OF SOFTWARE DEVELOPMENT AND INTRODUCTION OF TS CHART

Toshio Ishijima　Masamichi Kobayashi
Allied Information Systems Co.,Ltd.

These days improvement of productivity of software development have been proposed.
The importance of software documentation has been discussed often, but few useful documentation have been designed. We have used a flowchart or a structured chart as a documentation of program's logic. We use TS chart to document the program.
In this paper, we introduce TS chart.

1　はじめに

ソフトウェア開発においてドキュメントは非常に重要であるにもかかわらず、かならずしも有効なものが作成されているわけではない。
中小システムハウスやソフトウェアハウスにおいてプログラム開発におけるドキュメントの問題として次のような点があげられる。

（1）一人または数人でおこなえるような比較的小さなシステムが多いため、作業の分業化がされずに設計、製造、保守まで同一の担当者が行ってしまう。このためＢＵＧが発生した場合ソースプログラム上でいきなり修正してしまい、ドキュメントの更新はあとで一括して行おうとするが更新忘れや修正ミスがあっても気がつかず、この結果ドキュメントとソースプログラムの内容の不一致がおこる。
分業化されていれば製造時に見つかった設計ミスは設計担当者にフィードバックされ、更新された設計書が戻って処理するのでドキュメントは常に最新のソースプログラムと同じ内容を保持できる。

（2）短納期、低価格の仕事をする事が多く、とりあえず動くものを作るという事に重点が置かれ、ドキュメントは納品物件として指定されているので作成しているにすぎない。このように営業力の弱いところではドキュメント作成工程を手抜きする事でかろうじて赤字にならないようにしている。

（3）ドキュメントがあっても最終的にはソースプログラムを読む事になるため、あまり有効に利用されない。

2　生産性の向上

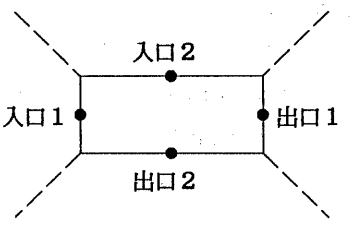
生産性の向上という点から次のような事が考えられる。

（1）モジュール化された分かりやすいプログラムを作る。
（2）一度作成したモジュールはライブラリ化して再利用する。
（3）ＣＡＳＥツール等の導入により自動化を進める。
（4）プログラムは属人的要素が強いため、技術教育により技術者のレベルをあげる。
（5）業種を特化しノウハウを蓄える。
（6）ＩＣＥ、デバッガをはじめとする開発環境を整える。
（7）プログラムの内容を他の人にうまく伝えられるようなドキュメントの作成を考える。

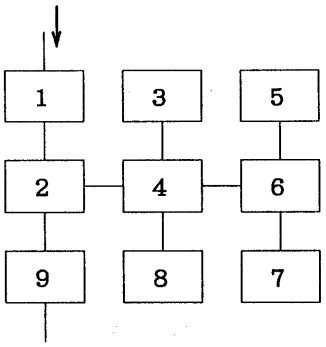
3　ＴＳチャートの特徴

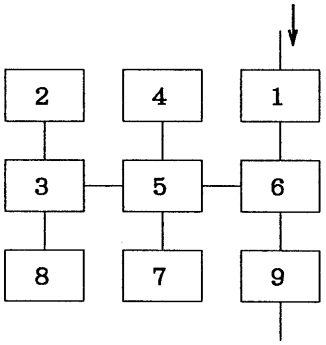
当社においても生産性の向上というテーマで検討し、ＴＳチャートというおもしろい設計技法に出会った。
設計時においてはトップダウン／ボトムアップのアプローチがとれたり、不確定な部分を記述できること。またレビューや保守においては処理の主体と従属、初期化処理と後処理が構造的にわかるようになっているので効率よく検討、解析が出来ると考えた。これらの特徴について紹介する。

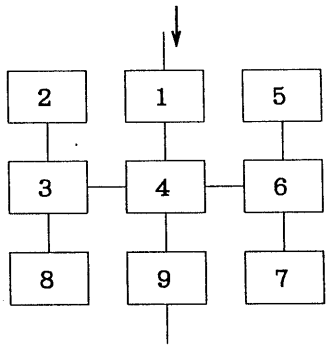
（1）　記述ルール
基本的にはＴＳチャート記号は２つの入り口と２つの出口を持ち、必ず入り口が左と上に位置するように置く。



読み方は左上から右下へ読み進む。
入り口１の側に記号があればその記号に進む。
入り口２の側に記号があればその記号に進み、なければＴＳチャート記号を読む。
ＴＳチャート記号を読み終わったときに出口１に記号がなく、出口２に記号があればその記号に進む。
ＴＳチャート記号を読み終わったときに出口１に記号があれば、出口２の記号の位置をスタックにいれ、出口１の記号に進む。
ＴＳチャート記号を読み終わったときに出口１、出口２のいずれにも記号がなく、スタックが空でなければスタックから位置を取り出しその記号に進む。
ＴＳチャート記号を読み終わったときに出口１、出口２のいずれにも記号がなく、かつスタックも空ならば終了する。

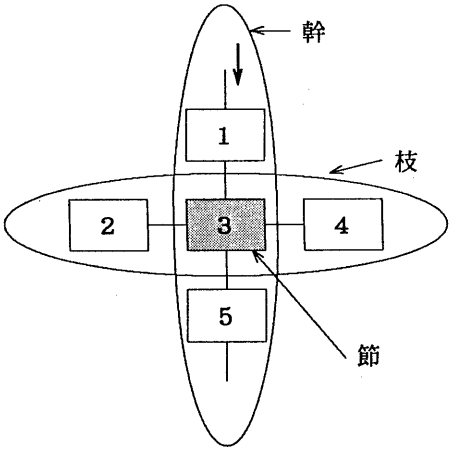






（２）主処理と付随処理

ＴＳチャートでは処理の主体を幹方向に置き、付随的な処理を枝方向に置く事により、幹方向に流れを追えばその大意をつかむ事が出来る。



数字は処理順番

１、３、５と読めば粗筋がわかる。
２、３、４は３つの処理が密接に関係し、１つにまとまっていると解釈する。
この解釈はＴＳチャートの構造が持つものではなく、ＴＳチャートの構造の上に意味をマッピングさせることで得られるものである。

（３）トップダウンとボトムアップ

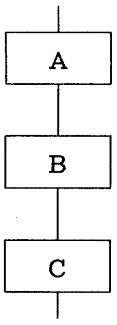
設計においてすべてトップダウンに作成するとは限らない。ライブラリを使用する場合など先にライブラリを決め、後からまわりを決めていく事もある。

回路設計などではチップのもつ機能で選択し、それらを結び付けていく事でシステムを実現しており、ボトムアップ的なアプローチと見る事が出来る。

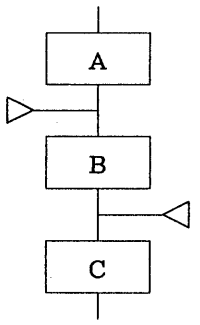
プログラム開発においても生産性を上げるためにライブラリを多用すれば、ボトムアップ的な記述が可能な設計手法が必要である。

図（ａ）を基本型とすると図（ｂ）はＢの前後に始端記号と終端記号をつけモジュール化したものである。

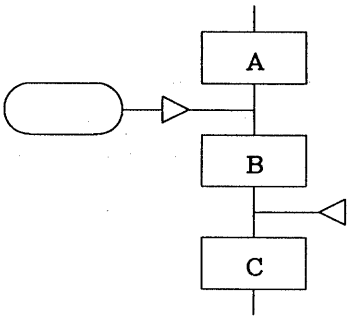
図（ｃ）、（ｄ）はモジュールＢにそれぞれループ記号、選択記号を寄生させる事でボトムアップ的に制御構造化したものである。



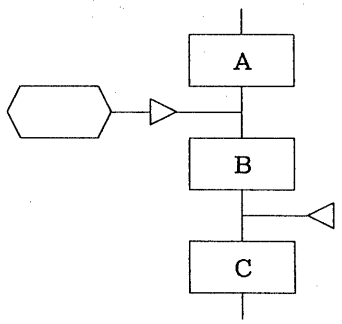
(a) 基本型



(b) モジュール化



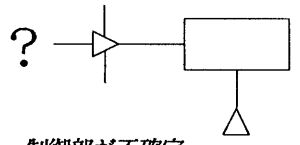
(c) 条件ループを寄生させ制御構造化



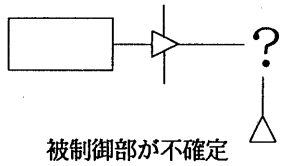
(d) 選択を寄生させ制御構造化

（４）不確定記号

考えがまだまとまらず処理内容が決まらないが、構造だけは先に書いておきたいときに使用する。

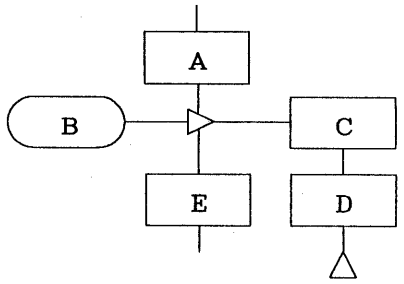


制御部が不確定



被制御部が不確定

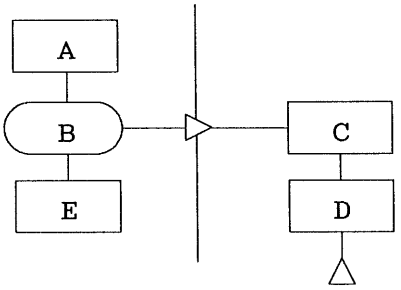
（５）初期化と後処理



(e)

Ｂは制御部で、モジュールＣＤは被制御部で

あり、モジュール部分は制御部側からは独立している。



(f)

Aはループのための初期化処理、Eはループ構造の後処理と意味づける。こうすることによりABEは1まとまりで制御部であると解釈できる。

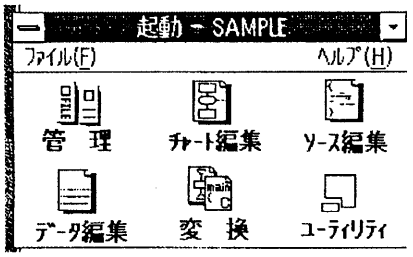
図（e）ではEおよびAが制御部に属するかどうかを構造的にとらえられない。

4　ツールの作成

TSチャートの特徴をいかした設計に慣れれば他の人に説明する場合でもモジュールのどこが主体処理で、何が付属的な内容なのか、初期化、後処理の影響する範囲はどこまでなのかを視覚的に示す事が出来る。

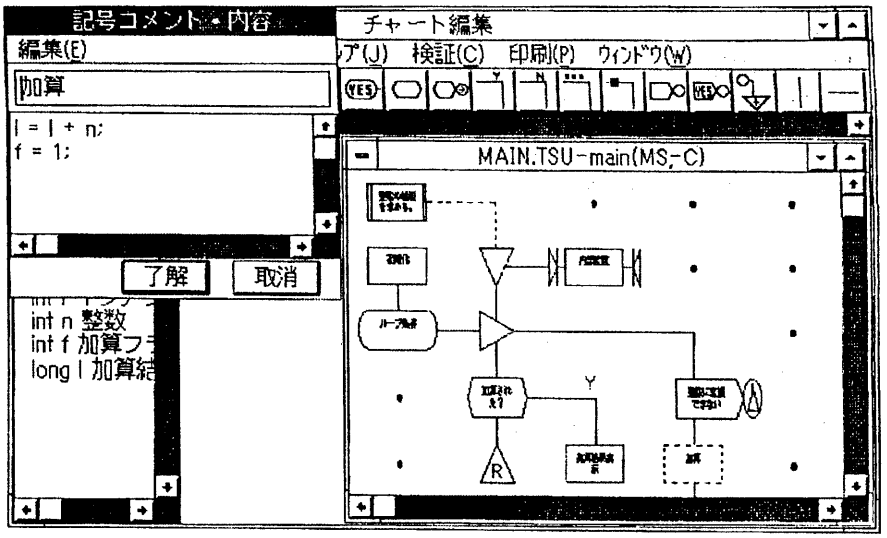
設計者の意図をチャート上に示す事が出来るので、新人教育などにおいてはベテラン技術者の考え方をそのまま学ぶ事が可能と考える。

そこで手続き型TSチャートと呼ばれる基本セットをサポートする開発ツールを企画した。



このツールは6つの部分から構成される。

チャートエディタ
データエディタ
ソースエディタ
ユーティリティ
ソース変換
管理プログラム



（１）チャートエディタ

ＴＳチャートを編集する。
チャート記号を使ってプログラムを設計する。記号に対してコメントと内容を入力する。チャート上の記号に表示されるのはコメントで、内容は変換においてソースプログラムとして生成される。
記号指定時に接続の可否がチェックされる。
ＴＳチャート空間は四方に広がるが記号はグリッドの位置にのみおける。

（２）データエディタ

グローバルデータを定義する。
ローカルデータはチャート内で定義し、グローバルデータおよび定数はここで定義する。
データファイルおよびＥＸＴＥＲＮファイルを作成する。

（３）ソースエディタ

ソースプログラムを編集するための専用エディタである。
自動生成されたソースプログラムには管理情報が入っている（コンパイルエラーにはならない）ため、これらを無視するようになっている。

（４）ユーティリティ

メイクファイルの作成、コンパイルオプション、アセンブルオプション、リンクオプションの指定、ファイル一覧、関数一覧、関数の参照しているグローバルデータの関連表、関数間の関連表を作成する。

（５）ソース変換

Ｃ言語またはアセンブラのソースプログラムを自動生成する。
チャート編集で入力した内容と制御構造の部分を自動生成する。現在、Ｃ言語はＭＳ－Ｃ　Ｖ５．１、アセンブラはＭＡＳＭ　Ｖ５．１に対応している。

（６）管理プログラム

ファイルの管理を行う。
このシステムではチャートとソースプログラムなど一対一に対応して処理するものがあるので、ファイルに対する操作は管理プログラムですべて行う。

５　おわりに

チャートを使って表現する事は直感が働き全体を大筋で把握できるなど文字のみによる表現に比べてはるかに有効である。ＴＳチャートの場合にはさらにＴＳチャートの構造の上に意味付けするという、その特徴を生かす事によって視覚的に判断できる内容をさらに広げる事が出来る。
このことはレビューや保守において非常に効果を発揮するはずである。
ここではＴＳチャートのごく一部しか紹介できなかったのでプログラム設計のレベルでの使い方になってしまったが、拡張ＴＳチャートまでおこなえばもっと上流工程での設計にも使用できると思う。

参考文献

［１］大原茂之
「ＴＳチャートによる並列表現について」
電子通信学会、ＣＡＳ８４－２０９

［２］大原茂之
「木構造化チャートによる階層構造的プログラミング」
東海大学紀要工学部、Ｖｏｌ．２７、Ｎｏ．１　１９８７

［３］大原茂之
「木構造化チャートによるプログラム開発技法」
日本システムハウス協会、ＴＳチャート研究会資料

［４］大原茂之
「ＴＳチャート入門」
オーム社