ソフトウェア工学 50－2
（１９８６．９．２４）

ソフトウェア設計図式用グラフィックスシステム

則房 雅也、　萩原 規子、 小松 久美、 紫合 治
日本電気技術情報　日本電気（株）
システム開発（株）　ソフトウェア
第三開発部　生産技術研究所

本論文ではソフト開発で利用されている種々の図式の作成を行えるグラフィックスシステムについて述べる。このグラフィックスシステムはＰＣ９８００／ＰＣ－ＵＸ上に実現したもので、マウスやポップアップメニューやマルチウィンドを採用しユーザフレンドリなものになっている。種々の図式を扱うために、図式クラスという概念を導入しツールで扱えるようにした。この中には特にソフトウェア設計図の特徴を分析しその結果を反映させている。作成した図式は解析することが容易で、この例として、ＳＤＭＳモジュール設計図式クラスの図式解析を紹介する。この実例によって、図式作成が単なるドキュメント作成ではなく設計作業そのものになることを示す。

WGSE50-2
GRAPHICS FOR SOFTWARE DESIGN DIAGRAMS (in Japanese)

*Masaya NORIFUSA, **Noriko HAGIWARA,
**Hisami KOMATSU, **Osamu SHIGO

*Sofware Development Department 3
NEC Scientific Information System Development Company Ltd.
60, Mizonokuchi, Takatu-ku, Kawasaki, Kanagawa 213, Japan

**Software Product Engineering Laboratory, NEC Corporation

This paper describes the graphics software system implemented on the NEC personal computer PC9800/PC-UX in order to construct various kinds of structured diagrams used in software development. A diagram class concept, abstracting the used diagrams, is introduced so that the system can handle many kinds of diagrams. The system facilitates a mouse locator, pop-up menus and multiple windows for sophisticated user interfaces. Constructed diagrams are analyzable. As an example, analysis of the diagrams in SDMS module design diagram class is explained. This shows that diagram construction can be positioned as a designing work rather than as a conventional documentation work.

1．はじめに

ソフトウェアシステムの開発では、文章やプログラムコードだけでなく、多種多様の図式を種々の目的で作成し活用している［1－13］。これらの図式は、直感的なイメージを表現したり全体の流れや構造を包括してとらえるとき有効であり、この点に関して文章よりもまさっている［20］。従来の作図はテンプレート等を使い手書きにより行なっていたが、近年ハードウェアの進歩と低価格化が進み、パソコンレベルの計算機でグラフィックスツールが提供されるようになり、これらを利用することができる［18］。現在提供されているツールは、限られた図を対象にした専用ツールと、何でも書けるお絵書き的な汎用ツールに分類することができる［19］。このように個人レベルで手軽にツールを利用できる環境は、潜在ユーザの多いソフト開発部門にとって重要である。

一方ソフト開発では、同じ作業者が多種の図式－システム構成図、モジュール構成図、流れ図等－を扱う必要がある。また作図作業はそれだけに閉じているわけではなく、作図結果を他のツールで使ったり、データベース等でテキストデータと一緒に一元的に扱いたい［1－6、8、9］。この場合、専用ツールでは多種の図式作成が行なえず、汎用ツールでは他のツールとの統合化を行ないにくい。このような観点から、これまでのツールをソフト開発作業の流れの中に効果的に組み込むことは難かしい。

このような背景から、ソフト開発作業中作成する図式を効果的に利用するために、種々の図式が扱えかつ作図結果を他のツールでも簡単に利用できるグラフィックスを、ＰＣ９８００／ＰＣ－ＵＸ上に実現した。ここでは、このグラフィックスの特徴的な機能と他のツールと組み合わせソフト開発に適用している一実例について報告する。

2．ソフトウェア設計図式の特徴

ソフト開発で利用する図式は構成や流れを表現するために用いられることが多い。例えば、モジュール構成図［13］、データフロー図［7］、状態遷移図［3］、ＳＡＤＴ図［4］、構造化プログラム図［11］等である。このような図式では、図式中の構成要素や構成要素間の関連およびそれらの説明文が意味を持ち、こういったものの組み合わせで作成者の意図を伝えている。

これら図式に共通した特徴を以下のようにとらえることができる。

・ノード（箱や円のような図形シンボル）とアーク（矢印のように図形シンボルを結ぶ関係線）とこれらの意味を補足的に説明するテキストで構成する。（図－1）

・ノードはその形状で、機能や処理の存在およびその種類を明示する。

・アークは必ずノードにつなぎ、ノード間の関連を明示する。

・一枚の図式では全てを書ききれないので、階層的に分割して表現する。

そして、これら共通した特徴を持つ個々の図式は、以下の特徴によって区別することができる。

・ノード、アーク、テキストといった図式構成要素の種類

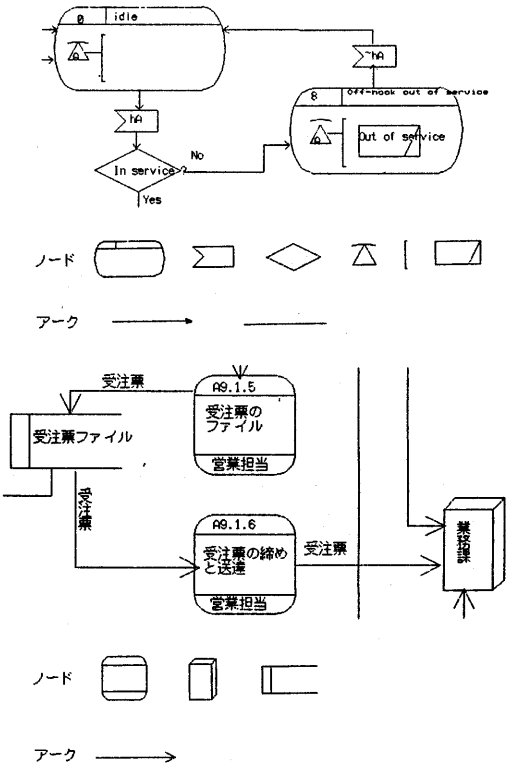
・図式構成要素の種類間の関連付け規則

このように、図式をある特徴によって分類することができ、分類された集まりを図式クラスとよび、以下のように定義する。

1）図式クラスは、ノード型の集合、アーク型の集合、属性型の集合および、これらの間の関連付け規則の集合で表わす。

2）ノード型、アーク型は形状の性質によって決められる。例えば、線種、大きさ等である。

3）属性型は、テキストの性質によって決められる。例えば、日本語文章、テキストファイル名等である。



図－1　ノードとアークで構成する図式

4）関連付け規則には、ノード間、ノードとアーク間、ノードと属性間、アークと属性間に対するものがあり、ここで与えた関連だけを図形要素間につけることができる。

この図式クラスは、データフロー図式クラス、状態遷移図式クラス、プログラム図式クラスというよりは、ＳＡデータフロー図式クラス［７］、ＣＣＩＴＴ／ＳＤＬ状態遷移図式クラス［１７］、ＳＰＤチャート図式クラス［１２］というものである。実際に作成した図式は、ある図式クラスのインスタンスと考えることができる。

３．図式作成グラフィックスシステム

本グラフィックスは、図式クラスの集合を扱うことができる。システム構成図を図－２に示す。図式クラス定義ツールを使って一つの図式クラスを定義する。この結果はクラス定義ファイルにおとされ、作図ツールはこの定義を参照して図式作成をガイドする。作図ツールを使って作成した全ての図式は、その時参照した図式クラスのインスタンスとなる。作図結果は、簡単な構文のテキストデータに変換され図式データファイルにおちるので、他の工程のために解析することは容易である。

３．１　図式クラスの定義

図式クラスの定義は図－３に示すようなツールを使い、以下に示す項目について行なう。

（１）ノード型について
- ａ．ノード型名
- ｂ．形状（形、線種、大きさ）
- ｃ．接続可能なアーク型
- ｄ．付加可能な属性型
- ｅ．子ノード

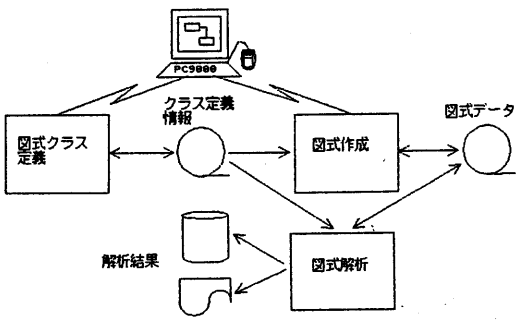
（２）アーク型について
- ａ．アーク型名
- ｂ．始点、終点の形状
- ｃ．付加可能な属性型

（３）属性型について
- ａ．属性型名
- ｂ．基本型

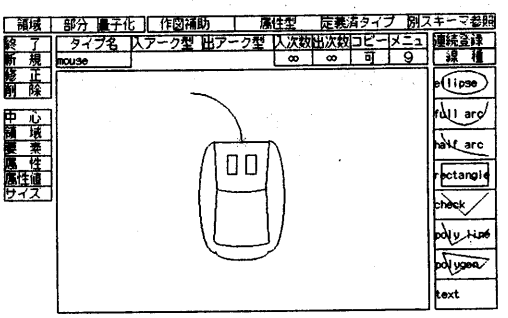
この定義によって一つの図式クラスが定まり、このクラス定義を参照して作成した全ての図式は、この図式クラスのインスタンスである。上記項目中、型名以外の全ての入力は、メニューとマウスを使って定義することができる。形状は基本形状（四角形、楕円、線分等図－３内右端のメニュー）を組み合わせて定義する。型名と形状以外はデフォルト値があり、接続可能なアーク型が指定されなかった時は、全てのアークが接続可能と解釈し、付加可能な属性型が指定されなかった時は、付加すべき属性はなしと解釈する。ノードに従属させて一塊のものとして扱いたいノードが子ノードであり、親ノード内の相対位置と共に指定する。子ノードもノードとしての定義を持つので、子ノードであることを考慮しないように解釈することも可能である。属性に対しその内容を規定する基本型があり、それらは〝文字列〟、〝文書〟、〝図式〟という型であり、それぞれ名前を持つ。任意の名前に基本型を対応させ基本型名として追加することができる。これによって、より強力な図式の解析が可能となる。

３．２　図式作成編集操作

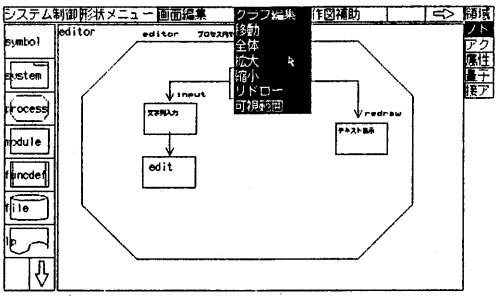
定義を参照し、図－４に示すツールを使い図式を作成する。使い易いユーザインタフェース［２１］を実現するために、本ツールでも定義ツール同様メニューとマウスでほとんどの図式作成編集操作が可能である。画面上段に出ているバーメニューをポイントすると、全体の編集に関するメニューが開く（図－４）。個々のノードやアークをポイントすると、個々の編集に関するメニューが開く。これらのメニューは階層的に構成されており、ポイントした項目にサブメニューがあればその横に続いて開く（図－５）。定義したノードの一覧もメニューとして開くことができる（図－４の左端）。このメニュー



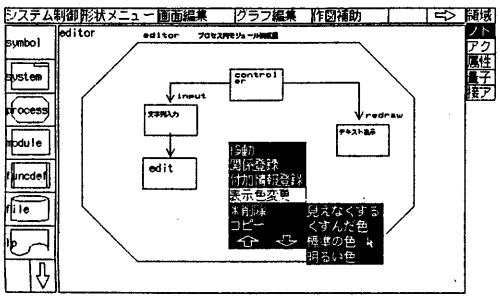
図－２　システム構成



図－３　ノード定義例



図－4　図式作成例



図－5　階層的メニュー表示

から登録したいノードを選び、作図領域内に展開する。アークは有方向で、始点となるノードを選び終点となるノードまでその間を結ぶように張る。この時途中に中継点を設け、折れ線で表現してもよい。このようにして登録したノードやアークに属性を与えて図式を完成させていく。表－1に編集コマンドの一覧を示す。

表－1　編集コマンド一覧

対象	編集コマンド
作図領域	作成、削除、変更、拡大、縮小
図式	移動、拡大、縮小、全体表示、所在確認
ノード	移動、削除、コピー、属性編集、関係登録、ズームイン、表示色変更、大きさ変更
アーク	移動、削除、属性編集、中継点追加、表示色変更 始点確認、終点確認
属性	内容更新、表示変更、削除
その他	図式のセーブ、ロード、テキスト編集

作図の便宜を計る特徴的な機能を以下に示す。

1）マルチウィンド

一つのウィンドに一つの図式を割り付け、これを複数個作成することができる。この特別なものとして〝所在確認〟用のウィンド（図－6）がある。これは、現在編集中のウィンドに表示されている図式が、図式全体の中でどの辺りなのかを確認するものである。

2）省略的アーク登録

アークの登録はノード間に対して行うが、ノード、アークが多く複雑な場合、見やすく登録するのは難かしい。このとき、他のアークの中継点をノードとみなして接続することができる。論理的な接続先は、つないだアークを辿って到着するノードである。辿り方はアークの方向で決まる。（図－7）

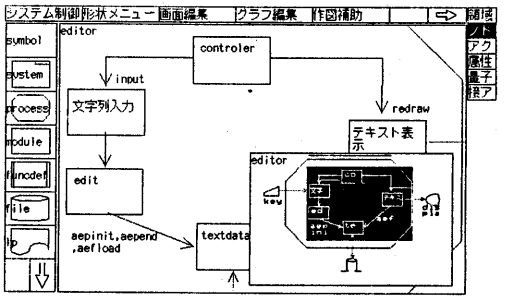
3）編集モード

ノードやアークを領域で選ぶ、ます目を表示しておく、抽出座標値をます目で量子化する、ポイントできる対象を限っておく、アークを登録するときの線分を水平、垂直方向に直す等、登録編集を行い易くするモードがある。

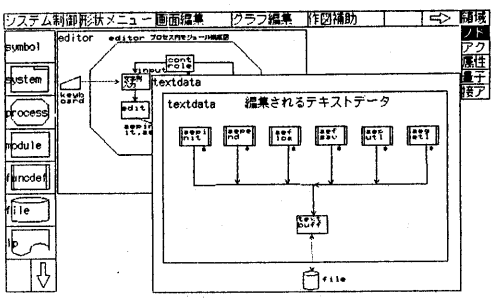
3．3　クラス定義のチェック

作図過程で使われる各編集コマンド処理を行うとき、以下の点につきクラス定義を守るようにチェックしてから実行する。

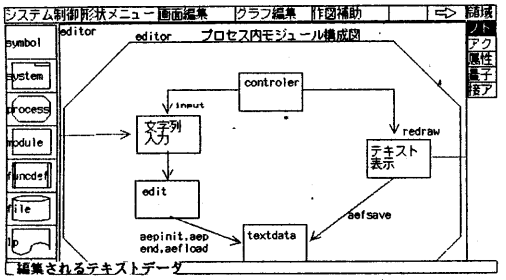
1）アークの登録は始点、終点各ノードで接続可能と定義された型の場合だけ可能である。（例えば、ｄａｔａ



図－6　所在確認コマンド

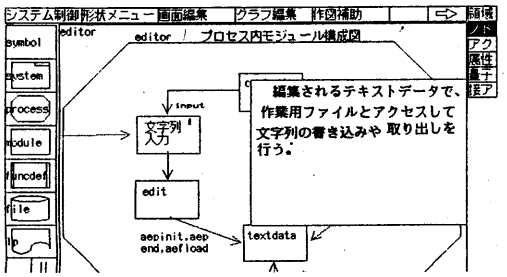


図－7　階層的図式作成



終了は＾D（CTRLキーを押したままDを押す）

図－８　ｔｅｘｔｄａｔａモジュールの見出し入力例



終了は＾D（CTRLキーを押したままDを押す）

図－９　ｔｅｘｔｄａｔａモジュールの機能概要入力例

型ノードにはｃａｌｌ型アークを張れない等。）中継点を利用してアークを登録する場合も同様である。

2）子ノードの登録は、登録先ノードで子ノードとして登録できると定義されている場合だけ可能である。

3）属性値の内容更新はその属性の基本型で解釈が異なる。＂文字列＂型の場合、新しい文字列でその値を置換える。＂文書＂型の場合、属性値をファイル名とする日本語テキストファイルの更新を行う。＂図式＂型の場合、属性値を図式名とする図式の編集を行う。

4）ノードの削除で、このノードの全ての接続アーク、子ノード、属性も削除する。

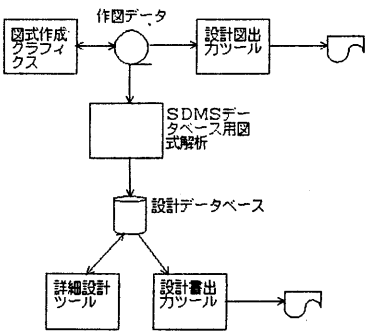
5）アークの削除で、このアークの全ての属性を削除する。

6）ノードの移動、コピーで、このノードの全ての子ノード、属性も同じ状態で移動、コピーする。また移動したノードに接続している全てのアークは、ノード上の接続位置を保つように端点が移動する。

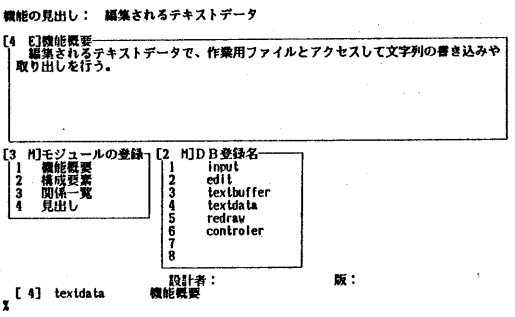
3．4　ソフトウェア設計図の作成

実際の作業の中で使われているソフトウェア設計図の特徴は、一度に大きな図式を作成することが不可能であるため、幾つかの小さな図式として階層的にあるいは分割的に作成されていることと、適切な量のコメント、特に日本語が説明的に付け加えられていることである。（図－７）

本グラフィックスでは、属性を活用してこれを行うことができる。＂ズームイン＂コマンドは、ノードやアークのより詳細な説明を図式で与えるためのもので、＂ｎａｍｅ＂という図式名を与えるための属性型が定義されていると、その属性値（文字列）を図式名とみなし、新しいウィンドを作成してその中で編集できるようにする。この機能で階層的な図式の作成を行える。一方、分割的な図式作成には、＂図式＂という基本型を持った属性を使う。ノードやアークにこのような属性を複数個定義しておき、それぞれに分割した中の一つの図式を与える。操作は＂ズームイン＂コマンドと同じように行うことができる。勿論、仮想座標系なので上記の方法をとらずに一枚の大きな図式として作成することもできる。この選択は作業プロセスに合わせて行えばよい。



図－１０　ＳＤＭＳでの図式解析フロー



図－１１　詳細設計ツールでの設計入力

ノードやアークに対する短い日本語の説明は、図式と一緒に表示することができる。これには"文字列"という基本型を持った属性を使う。大量の説明文書を与えるときには"文書"という基本型を使う。共に入力のために日本語エディタ用のウィンドを開くが、"文字列"の場合一行だけ（図－８）、"文書"の場合は制限なく入力できる。（図－９）"文書"の場合、図式中では文書が入力されたファイルの名前を管理することになる。

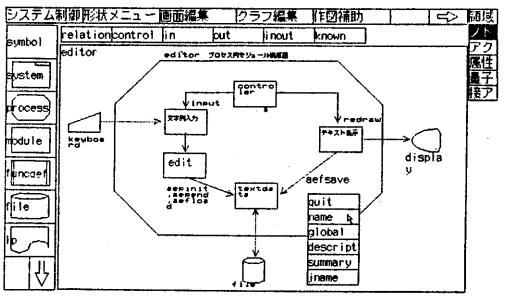
このようにして複数の図式や文書との関連をつけながら、ソフトウェア設計図の作成つまりソフト設計作業を進めてゆくことができる。出来上った図式は、プロッタやプリンタに清書したり、他のツールで様々な角度から図式データを解析する。４ではこの実例について述べる。

4．図式解析の実例

作図ツールで作成した図式は、以下に示す簡単な構文を持ったテキストとしてファイルにおとされる。大きく分けて、ヘッダ部、ノード情報部、アーク情報部があり、それぞれ識別記号で始まる。

```
S  <クラス名>    <図式名>
N  <ノードｉｄ>  <ノード型名>
   <属性>…
.
.
A  <アークｉｄ>  <アーク型名>
   <始点ノードｉｄ>  <終点ノードｉｄ>
   <属性>…
.
.
```

ソフトウェア開発保守支援システムＳＤＭＳ［2、3］では、上記書式の図式データを解析し、他のソフト開発



図－１３　ノード型、アーク型、属性型の例

工程に活用している。その流れを図－１０に示す。本グラフィックスを使って作成した図式は、ＳＤＭＳ設計データベース用解析ツールで解析され、構成要素間関係やその説明および図式間の階層関係がデータベースに登録される。この結果は、次の工程の詳細設計ツール［１４］に渡され、ここでより詳細かつ具体化される（図－１１）。最終的に詳細設計書生成ツール［１５］によって詳細設計書として出力される（図－１２）。この他にもＣプログラムスケルトン生成ツール等がデータベース中の情報を利用する。

このために、ＳＤＭＳモジュール設計図式クラスを定義した。ノード、アークは図－１３のメニューに示すような型を用意した。ノード、アーク間関係では、ｐｒｏｃｅｓｓというノード型にｃｏｎｔｒｏｌ，ｉｎ，ｏｕｔ，ｉｎｏｕｔというアーク型が接続可能といった定義が与えられている。ｐｒｏｃｅｓｓ（実行プログラムに相当）、ｍｏｄｕｌｅ（Ｃのソースファイルに相当）、ｆｕｎｃｄｅｆ（Ｃの関数に相当）、といったモジュー

6.　編集されるテキストデータ　（ textdata ）

6.1　概要

編集されるテキストデータで、作業用ファイルとアクセスして文字列の書き込みや取り出しを行う。

6.2　関数設計

6.2.1　関数一覧

（1）　外部関数

NO.	関数名（パラメータ）	意味
1	int　aepinit();	アクティブなエディタで使用するページテーブル、バッファテーブルとそれぞれのテーブルに付随する変数を初期化し、作業用ファイルをオープンする。
2	int　aepend();	作業用ファイルをクローズする。
3	int　aefload(fn);	指定されたファイルの内容を作業用ファイルへコピーし、同時にページテーブル、行情報テーブルへ必要な情報をセットする。
4	int　aefsave(fn);	ページテーブル、行情報テーブルとバッファテーブルとを参照して、最新の情報を指定されたファイルへ順次書き出す。
5	int　aeputl(ln,str);	１行分の入力文字列を、指定された行番号と共にバッファテーブルへ書き込む。
6	char　*aegetl(ln);	指定された行番号の文字列を取り出しバッファテーブル上のその文字列のアドレスという形で返す。

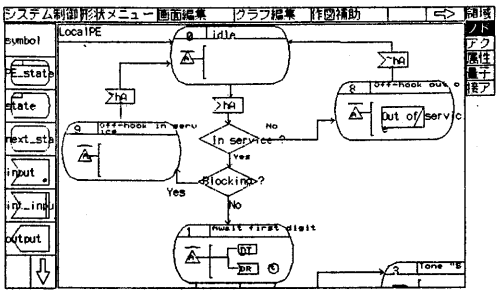
図－１２　設計書出力例

ル分割の対象になるノード型には、ｎａｍｅ（モジュール名を与えるためのもので、同時に詳細な機能を別の図式として作成するときの図式名となる。）、ｊｎａｍｅ（モジュールの日本語名を与えるためのもの）、ｓｕｍｍａｒｙ（モジュールの簡単な日本語説明文を与えるためのもの）、ｄｉｓｃｒｉｐｔｉｏｎ（大量の設計文書を与えるためのもの）、ｇｒｏｂａｌ（大域名か局所名かを与えるもの）という属性型を与える（図－１３）。ｊｎａｍｅとｓｕｍｍａｒｙは図式と共に表示して読み易さ、理解し易さを向上させる。これらの属性には図－８、図－９のように属性値（図中ウィンド内の日本語文）が与えられる。設計データベース用解析ツールは、データベース用の型変換情報を参照して図式を解析する。型変換情報は、図式クラスのインスタンスを解析し設計データベースに登録するための対応表のことで、例えば図式クラスでのｄｉｓｐｌａｙやｋｅｙｂｏｒａｄは、データベースではｆｉｌｅに統一され、ｐｒｏｃｅｓｓ間に張られたアーク型ｃｏｎｔｒｏｌはデータベース中の関係ｓｐａｗｎに、ｐｒｏｃｅｓｓとｍｏｄｕｌｅ，ｆｕｎｃｄｅｆ間の場合はｃａｌｌという関係に変換される。データベースに登録された解析結果は、詳細設計ツール（図－１１）や設計書生成（図－１２）に反映する。

ＳＤＭＳでの設計図式作成は、モジュールの完全な説明を図式作成時に与えてしまうことが目的ではなく、モジュール構成を決めているとき部分的にでも詳細なレベルが見えたときには、図式を作成しながらそれをメモしておくことにある。これらのメモは、次の詳細設計フェーズでの設計入力（処理概要やパラメータ）のガイドとして使われる。

5．おわりに

本論文では、ソフト開発で利用されている種々の図式作成を行えるようにした図式作成グラフィックスシステムについて報告した。そのために、図式クラスという概念を導入した。この図式クラスを本ツールを用いて定義できるようにし、従来特定の図式の解析には専用的なツールを必要としていたものを、ソフト開発の領域においてはこれを汎用的に扱えるようにした。本グラフィックスは、図式クラスの集合を扱えるという点、作図結果を目的に合わせて解析できるという点、手軽に使えるパソコンでしかもツール開発に適したｕｎｉｘ上に実現したという点で、本ツール自身の今後の拡張性だけでなく、他のツールの一部として組み込まれて使われることが容易となっている。ここでは、その一つの実現例としてソフトウェア開発保守支援システムＳＤＭＳでの利用方法について述べた。ＳＤＭＳではこの他にも状態遷移図作成およびその解析［3、１６］に、本グラフィックスを活用している（図－１４）。現システムでは、作図系か




```
                                <<0 : idle>>
A ------------------------->¦
                ? In service ? : yes ?
                   ? Blocking? : yes ?
                                <<9 : Off hook>>
-A ------------------------->¦
                                <<0 : idle>>


                                <<0 : idle>>
A ------------------------->¦
                ? In service ? : yes ?
                   ? Blocking? : no ?
                                <<1 : Await 1st digit>>
-t0 ------------------------->¦
                                <<3 : Tone "S">>
-A ------------------------->¦
                                <<0 : idle>>


                                <<0 : idle>>
A ------------------------->¦
                ? In service ? : yes ?
                   ? Blocking? : no ?
                                <<1 : Await 1st digit>>
d ------------------------->¦
             ? digit analysis : vacant ?
                                <<3 : Tone "S">>
-A ------------------------->¦
                                <<0 : idle>>


                                <<0 : idle>>
A ------------------------->¦
                ? In service ? : yes ?
                   ? Blocking? : no ?
                                <<1 : Await 1st digit>>
d ------------------------->¦
             ? digit analysis : local ?
             ? B party free ? : no ?
                                <<3 : Tone "S">>
-A ------------------------->¦
                                <<0 : idle>>


                                <<0 : idle>>
A ------------------------->¦
                ? In service ? : yes ?
                   ? Blocking? : no ?
                                <<1 : Await 1st digit>>
```

図－１４　状態遷移図とパス解析

ら解析系への情報伝達にはファイル（作図データファイル）を使っている。しかし、作図中動的に解析をし解析結果を図式へフィードバックさせ、図上でシミュレーションやアニメーションを行えば、より効果的な設計図式の作成が可能である。これを実現するためにはメッセージ等でより密なやりとりを行う方法がある。これについては、現在新しい構成でのシステムを開発しているところである。

今後は導入が容易なパソコンやワークステーションが益々普及するにつれ、ソフト開発現場でもこれらを活用した開発体制がとられてゆくと考えられる。グラフィック的な手法を扱ったツールは、これまでのキャラクタ画面での限られたユーザインタフェースを越える手段として大きな可能性を含んでいる。本図式作成グラフィックスは、そのような領域の一アプリケーションである。図式と文書の融合は、プログラミング言語に代表されるようなテキスト中心のソフト開発分野で、考えを伝達する手段としてだけでなく思考そのものにも大きなインパクトを与えることになると考える。様々な図式に対し適用を試みることと、その適用を通して図式でどこまで何を表現できるのかという可能性を追及することが今後の課題である。

参考文献

[1] O.Shigo, et al., "A Software Design System Based on a Unified Design Method," Journal of Information Processing, Vol.3, NO.3 Sep.1980.

[2] N.Hagiwara, et al., "A Graphic Tool for Hierarchical Software Design," IEEE Workshop on Visual Language, Dec.1984, pp.42-46.

[3] M.Koyamada, et al., "Design support Facilities for Switching System Software Using Graphic Interface," ibid, pp.47-52.

[4] D.T.Ross, "Applications and Extensions of SADT," COMPUTER Apr.1985, pp.25-34.

[5] G.P.Brown, et al., "Program Visualization: Graphical Support for Software Development," COMPUTER Aug.1985, pp.27-35.

[6] M.Moriconi, et al., "Visualizing Program Designs Through PegaSys," COMPUTER Aug.1985, pp.72-85.

[7] C.Gane and T.Sarson, STRUCTURED SYSTEM ANALYSIS: tools and techniques , Prentice-Hall,1979.

[8] 小山田他、「ソフトウェア開発のためのＣＡＤシステムＣＡＤＡＰ」、ＰＩＸＥＬ No.28, pp.103-109

[9] 立田、「ソフトウェアの要求分析、仕様書作成を支援するＳＡツール解説」、ＣＯＭＰＵＴＥＲ ＤＥＳＩＧＮ 1985, pp.121-131.

[10] 大石他、「並列計算プログラムのデータフロー図による記述システム」、情報処理学会 第３１回全国大会、 pp.349-350

[11] 二村、「構造化プログラム図式」、コンピュータソフトウェア、vol1, No1,Apr.1984, pp.64-77.

[12] 東他、「構造化プログラム図ＳＰＤとその応用」、情報処理学会ソフトウェア工学研究会資料、85-SW-43-2 (1985)

[13] 米田他、「ＳＤＭＳ／設計サブシステムの図形出力機能」、情報処理学会 第２４回全国大会、 pp.419-420.

[14] 岸他、「ＳＤＭＳにおける設計情報画面入力機能について」、情報処理学会 第３０回全国大会、 pp.773-774.

[15] 寺嶋他、「ソフトウェア設計文書作成のための汎用レポータ」、情報処理学会ソフトウェア工学研究会資料、86-SW-46-22 (1986)

[16] O.Shigo, et al., "Designers Work Environment for Communications Software," GLOBECOM, 1985, pp.1301-1305.

[17] CCITT : FUNCTIONAL SPECIFICATION AND DESCRIPTION LANGUAGE (SDL),Red Book, Volume VI-Fascicle VI.10, Geneva 1985

[18] 石田他、「設計者１人に１台パソコンＣＡＤの時代」、日経コンピュータ、1985.2.18, pp.63-82.

[19] 大島、「パソコンＣＡＤシステムの評価」、事務と経営 臨時増刊、 1985, pp.44-51.

[20] S.B.Sheppard, et al., "The Effects of Symbology and Spatial Arrangement on the Comprehension of Software Specifications," Proc. 5th ICSE(1981), pp.207-214.

[21] B.Shneiderman, "Direct Manipulation: A Step Beyond Programming Languages," COMPUTER Aug.1983, pp.57-69.