25. 会話型データ処理システムのための ソフトウェア開発支援ツール：SOFT

慶応義塾大学 工学部 数理工学科　田辺裕久 (Hirohisa Tanabe)　青木隆 (Takashi Aoki)　川口明 (Akira Kawaguchi)　浦昭二 (Shoji Ura)
慶応義塾大学 情報科学研究所　近藤頌子 (Shoko Kondo)

1. はじめに

計算機の発達にともない、システムの開発・保守の過程に、システムの対象とする分野の専門家、すなわち計算機に対する知識の深くない人が関与する機会が非常に増えている。こうした状況においては、計算機の利用者側に立った、非専門家にとっても容易に扱うことができる支援ツールが求められている。

1978年度から東京本郷にある水野調剤薬局と協同で設計・開発を進めてきた薬歴情報システムAPO-S（Advanced Patient Oriented Pharmacy-System）において、これは強く意識された事である。

APO-Sは投薬に際して発生が予想される種々の薬禍から患者を守る事を目的とし、調剤薬局を訪れる各患者ごとにその個人の体質及び薬歴といった情報を集めて保持し、これをもとにしてより安全な調剤活動を行なおうとするシステムである。またAPO-SではCRTキャラクタディスプレイ端末を使用して、患者の情報（性別・職業・特異体質・保険等）・処方内容の入力・入力された処方に対するチェック（薬自体・薬対薬・薬対人間）結果の表示・レセプトの発行といったすべての処理を画面上で行なう。

ここにおいてAPO-Sを使用するのは一般の薬剤師であり、計算機の専門知識は持ち合わせていない。つまりAPO-Sの開発は、システムの利用者の立場に立って行なわれた開発の典型的な例である。本論文においては、APO-S開発の過程で得られた種々の経験にもとづいて考案した　会話型データ処理システムのためのシステム開発支援ツールSOFT（Screen Oriented Flexible fabrication Tool）について述べる。

2. SOFTの基本概念

SOFTは前述のように、計算機の使用者を中心とするシステム稼動環境に計算機を適応させ、計算機の知識をあまり持たない人々でも容易に対象とするシステムに対する要求の定義を行なう事ができるような基本概念を提供する事を目的としている。すなわち、システムが稼動する環境の構造を変化する事なしにモデル化する事ができ、そうした環境構造で扱われる情報の収集・維持に対する責任範囲等を変化する事なくシステムを実現する事ができる。さらにそうしたシステムに対する要求の定義を行ない、システムの稼動を制御する操作しやすいToolを提供する。

また、SOFTが支援の対象とするのはキャラクタディスプレイを端末とした会話型データ処理システムである。特に1つの処理が終了した後、次に行なう事が可能な処理を示し、利用者にそのうちの1つを選択させる事で次の処理に移行する、いわゆるメニュー形式のシステムを仮定している。

2.1 システムのモジュール分割

近年のソフトウェアの複雑化・大規模化は開発・保守時のコスト・労力・時間

を増大させた。このような大規模化したシステムに発生する問題に対処するための手段として、モジュール化の技法が提案されている。システムのモジュール化は目的とするシステムの機能を明確化し、それはまたシステム開発者がシステムをよりよく理解するための手段ともなる。また、完成後に必要となったシステムの拡張・変更に対しても、変更を必要とする部分が限定されるため比較的容易にそれらの要求に対処する事ができる。

ところでモジュール識別の基準としては

1)機能分割　　—モジュールの果たす機能の違いによる分類
2)データ分割　　—モジュールの基本としてデータをとらえた分類
3)トランザクション分割　—処理される側の立場からの分類

などをあげる事ができる。またPSL/PDL, SADT等の支援システムもモジュールの記述・整理には非常に有効である。

しかしながら、これらの基準や支援システムは、モジュール分割を行なってくれるのではなく、モジュール分割は依然としてシステム設計者が最も知恵を搾らなければならない作業である事にかわりはない。従って計算機の知識があまりない人々にとっては何の手助けともならず、さらにこのような計算機内部の機能による分割は望ましいものではない。

ところが、ここで対象としているメニュー形式の会話型データ処理システムにおいては、通常1つの画面が1つの仕事に対応している事が多い。すなわちある処理画面において1連の入出力を行ない、その後次に行なう仕事(画面)を選択する事によって対応する画面に移っていく。

従ってこうしたメニュー形式のデータ処理システムにとっては『画面』という概念は、モジュール化の基準として妥当なものであり、また広義のトランザクション分割と考える事もできるが、明確なモジュール分割の手段となり、さらに計算機の非専門家にとっても理解しやすく、納得のいく基準であると思われる。

そこで我々は画面を中心概念として、以下のようなモジュール分割を行なった。まず、画面を1つのState(状態)と定義し、システムは利用者からの何らかのSelection(選択)によってTransition(遷移)を起こし新たなStateに推移して仕事を行ない、これを繰り返す事により処理が進行する、と考える事にした。

このような処理の流れを計算機の内部処理と、人間の関係する外部処理とに分け、それぞれT-Process, S-Processと呼ぶ事にする。すなわち1つのStateで行なわれる利用者と画面を介しての入出力とそれに伴なう処理をS-Process, StateからStateへの遷移の過程で必要となるファイル入出力等の人間の作業を要さない部分の処理をT-Processと定義する。従ってシステムの処理は、S-ProcessとT-Processを交互に繰り返す事によって進行すると考える事ができる。(図1)

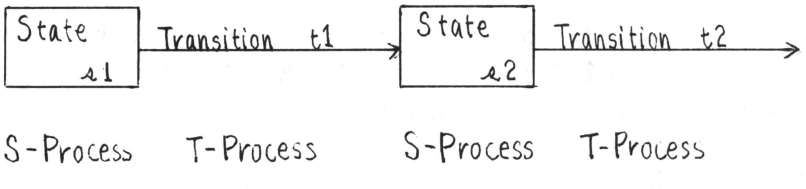

図1　システムの進行

2．2　S-Processのモジュール分割

次にS-Process内の処理について考える．一般に会話型データ処理システムにおいては、レコード内容の表示に際して次のような手順で実行がなされる．

(1) 基本的な画面の枠組みの出力
　　見出し等のその画面での固定されたものの出力
(2) データの入出力
　　検索されたレコードの表示、あるいは使用者からのデータの入力等
　　画面を介したシステムと使用者の会話部分
(3) 次のStateの選択
　　使用者が次に行なおうとする処理（次のStateとそこへ遷移する過程で行なわれるT-Process）の選択

我々は以上の3手順をそれぞれ (1) Frame (2) Conv (Conversation) (3) Selection と名づけた．

さらにS-Processでの処理の中心となるConvについて考える．Convは上述の定義どうり計算機と人間の画面を介してのデータの入出力を行なう段階である．ここでは画面の入出力フィールドに対して、計算機による出力、あるいは人間からの入力が順次行なわれる事で処理が進む。またフィールド間では入力データのチェック等の内部処理が行なわれる．従って『画面』を基本概念としたStateとTransitionの分割と同様に『フィールド』を基本概念として Sub State と Sub Transitionの分割を行なう事ができる．すなわち各フィールドを1つのSub State と定義し、Convにおける処理は使用者からの何らかのSub SelectionによってSub Transitionを起こし、新しいSub Stateに移り、これを繰り返す事によって進行すると考える事ができる．ここでも同様に計算機内部のデータチェック等の処理をSub T-Process，画面を介して実際に行なわれる処理をSub S-Processと定義する。（図2）

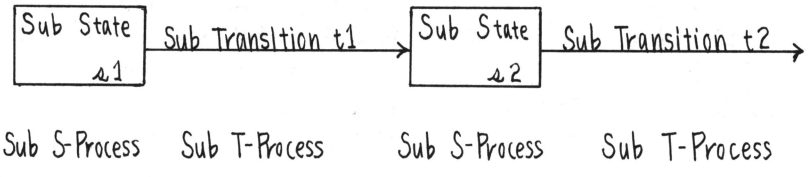

図2　Convの進行

ここでSub S-Processでの処理も、S-Processの処理と同じように分割する事ができる。すなわち各Sub S-Processは次の処理を順に実行する．

(1) 入出力項目の位置決め
(2) データの入出力
(3) 次のSub Stateの選択

これらも同様に (1) Sub Frame (2) Sub Conv (3) Sub Selection と名づける．

以上のように、S-Processにおける処理は同一の概念を利用して階層的に分割する事ができる．（次頁図3）　また、S-Processにおける処理は個々の入出力の積み重ねであり、各項目とそれらの間の関連を示す事で定義できる事がわかる．

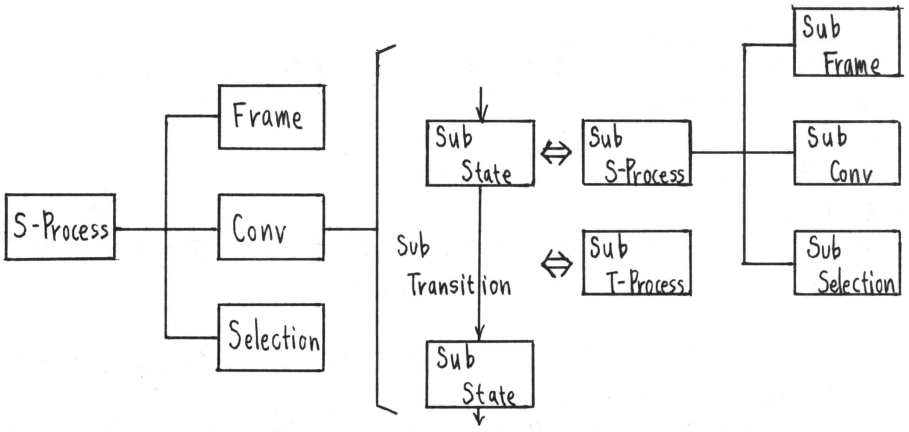

図3　S-Processの階層的分割

2. 3　T-Process・Sub T-Processのモジュール分割

T-Processは画面を介した入出力を含まない（人間の作業を必要としない）、計算機内部の処理からなるプロセスである。具体的には直前のStateで入力されたデータをファイルに書き込むとか、次のStateで必要となるデータをファイルから読み込むといったファイル入出力とそれに伴なう値の変換がなされる事となる。

同様にSub T-Processでは直前のSub Stateで入力されたデータの妥当性のチェック及び内部形への変換、次のSub Stateで出力されるデータの変換等が行なわれる事となる。

従ってT-Process、Sub T-Processにおける処理は対象とする分野によって千差万別であり、S-Process、Sub S-Processにおける分割のように『画面』あるいは『フィールド』という概念を用いた統一的なモジュール分割手法を提供する事は非常にむずかしい。しかし、上述のようにT-Process, Sub T-Processには、データ処理システムが本質的に必要とする処理と、計算機を使用するために必要となる、ファイル操作・データチェック等の処理がある。従って、T-Process, Sub T-Processの分割は、まずこの2つに対して行なわれる。

データチェックは後述の変数定義機能と合わせてSOFTによってサポートされ、ファイル操作はデータベースの発達に伴ないDBMSによってサポートされる事を考えれば、以上の分割は妥当なものと思われる。さらに本質的な処理の部分は、全てのT-Processから抽出される、独立かつ単一な機能（例えば前述のAPO-Sにおける患者対薬品チェックのモジュール）に分割される。

つまり、T-Process, Sub T-Processは、機能を基準として分割された小さなモジュールを順次実行する事により処理が進行する。

3. SOFTの構成

上述のように、会話型データ処理システムは画面を中心概念としてモジュールに分割する事ができる。これに基づいてSOFTは各入出力項目（Sub State）に必要な情報と、各State間のつながり等を記憶し、その情報を解析しながら処理を制御する事で、プログラミングに依らずに画面を媒介としたシステムを生成する事を可能にし、システム開発者の負担を軽減するためのツールである。

そこでSOFTの持つ機能は大別すると次の2種となる。
(1) 各種の定義を行なう　Sub Language
(2) システムの実行を制御する　Monitor

3. 1　システムの生成

システムの開発者は、定義Sub Languageを用いて、変数・ファイル，State、Conv、Frameの定義を行なう。SOFTはこれらの情報から定義情報ファイルを生成する。以上の定義と共に、システムの開発者は、プログラミング言語により、T-Process、Sub T-Processで使用する単一機能モジュールを記述し、コンパイルしてオブジェクトプログラムを生成する。その後SOFTの実行制御モニタ、単一機能モジュールをリンカーにより結合し、目的システムを生成する。システムの処理は、モニタが定義情報ファイルを逐次参照しつつ、S-Process・T-Processを交互に繰り返し実行する事で進行する。

3. 2　定義 Sub Language

システム定義 Sub Languageは以下の5つの部分からなる。

(1) State 定義

ここではシステム内の各Stateが、どのFrame・Convを使用するかの定義、Selectionの際のスイッチとなる文字列と次に移るState，遷移の間で実行されるT-Processの指定を行なう。(図4)

さらに実際のシステムにおいては、T-Processの実行を行なわなければ移行するStateが決定できない事がある。(例えば必要とするレコードがファイルに存在しなかった場合のエラー処理等)そうした場合にそれらの遷移を定義する、dummy Stateを定義する事もできる。

```
STATE NO : 1

  FRAME  NO : 1
  CONV   NO : 3
  SELECTION
   SW  T-PROCESS  NEXT-STATE
   KG    TP1          4
   ND    TP2          11
```

図4　State定義例

```
FRAME NO : 1

   LINE   :2
   COLMN  :72
   STRING :ギョウム ガメン

          CONTINUE(Y/N)
```

図5　Frame 定義例

(2) Frame定義

ここではFrame中の各出力フィールドごとに、画面上の行位置・列位置・出力する文字列の定義を行なう。1つのFrameに複数のフィールドが存在する時には、必要回数定義を繰り返す。(図5)

(3) Conv定義

ここではConvの内容を各Sub S-Process単位に定義していく。実際には、Sub Frameに相当する行位置・列位置、Sub Convに相当する変数名とその使用タイプ(入力または出力)、Sub Selectionの際の遷移の起こる条件、次に移るSub State、遷移の間で実行されるSub T-Processの定義が行なわれる。

Stateと同じ理由からdummy Sub Stateの定義も行なう事ができる。(図6)

```
Conv NO : 3
  SUB STATE NO : 1
  LINE         : 10
  COLUMN       : #1*5
  VAR NAME     : NAME(#I)
  FOR USE      : 0
  SUB SELECTION
   CONDITION  SUB T-PROCESS  NEXT
  #I.GE.5       STP1         5
```

図6　Conv定義例

(4) 変数名・ファイル定義

ここではシステムで使用される変数・ファイルの定義を行なう。変数名定義は各Convで使用されるデータの変数名とタイプ・長さの定義を行ない、ファイル定義はファイル名と各レコード内のデータの変数名・タイプ・長さの定義を行なう。

(5) T-Process・Sub T-Process定義

ここでは各T-Process・Sub T-Processにおいて呼び出される単一機能モジュールのモジュール名とそれへのパラメータ、ファイル操作・データチェック機能の呼び出しと操作指示、及びそれらの実行手順が定義される。

3.3　実行制御モニタ

前述のとうり実行制御モニタは各定義Sub Languageによって生成された定義情報ファイルを逐次参照しつつ、S-Process・T-Processを交互に繰り返し実行する。さらにモニタ自身はSOFTの基本概念を忠実に反映したモジュール構造を持つ。

また実際には、T-Process・Sub T-Process内での処理の結果がシステムの進行に影響を与える場合、あるいはT-Process・Sub T-Process内でシステムの進行に関する情報を必要とする場合がある。前者の例としては、dummy Stateの所で述べたように、T-Processの実行によって初めて遷移すべきStateが決定する場合であり、後者の例としては、あるSub S-Processを繰り返し実行して同じようなデータを配列に格納する場合である。こうした要求のためにSOFTでは10個のシステム・パラメータを用意して、実行制御モニタとT-Processの通信に用いている。

さらにSOFTにはモード設定機能がある。1つはシステム開発過程でのテストをサポートするためのもので、このモードはT-Process・Sub T-Processを無視する。従って開発途上において各State間の遷移とState内の処理だけを実行し、システムの全体的な把握、画面の構成等をテストする事ができる。

この他には、遷移できるState・Sub Stateを制限するモードもある。こうしたモードは、例えばシステムに教育用の側面を持たせる場合に使用される。すなわち特定モードの時だけに実行されるState・Sub Stateに使用者に対する指示を定義しておけば、システムの立ち上がり時、あるいは初めてシステムを使用する人に対してそのモードで実行を行なうと、そうした指示を使用者に対して与える事ができる。

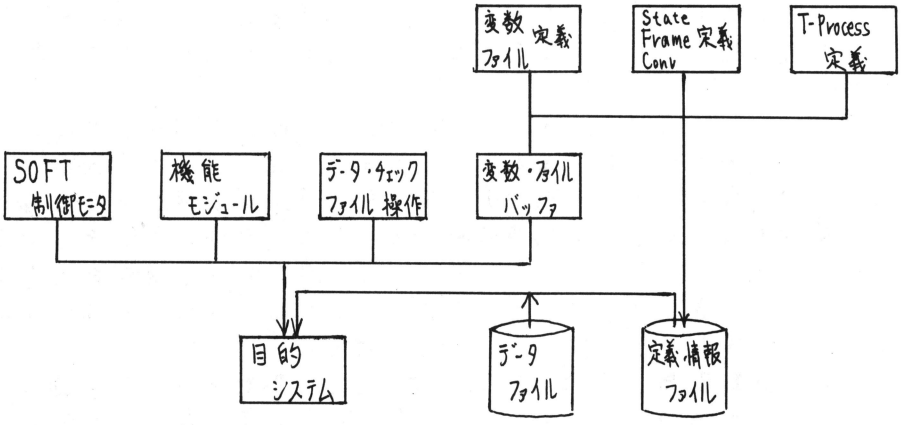

図7　システムの生成

4. SOFTの評価

PDP 11/34上にSOFTを実現し、前述のAPO-Sを記述する事を試みた。ただしこれはSOFTの初期バージョンであり、本論文中のファイル定義，ファイル操作・データチェック機能，T-Process・Sub T-Processの分割を除いた部分が実現されているものである。

こうした試みの結果APO-Sの通常オンライン業務は

- State　89
- Frame　54
- Conv　46
- T-Process　36
- Sub T-Process　63

によって記述する事ができた。

ここに見られるようにStateの数は予想以上に多い。従って使用者側に立ったシステム開発ツールを考える時、その上位概念としてStateを置く事は、SOFTの基本概念を開発初期段階に反映させる事ができない、という意味で好ましくない。さらにシステムで取り扱う情報の権限を定義に反映させていない。そこで我々は現在画面（State）を抽象化したモジュール基準の導入について検討を行なっている。例えばある会社組織における階層構造においては、部レベルにおいて各部が収集・維持すべき情報について定義し、各部の機能はその部に含まれる各課の機能を把握する事によって定義する、といった形式の機能定義を行なう事ができる。これと同様にSOFTにおいても各Stateの機能（会社における課の機能）を把握して、その上位の定義（部の機能）として情報を用いる事ができるのではないかと思われる。こうした考え方を用いると、Stateの定義を行なう前にそれらを抽象化した『情報』を基礎概念とした、モジュール分割基準を提供する事ができる。

すなわちシステム開発の初期段階として、システムのマクロなレベル（会社組織における部レベル）での分割を『情報』を用いて行なう。Stateはそうしたマクロなモジュールの機能を実際に実現する手段として提供され、『画面』を基本概

念として詳細なモジュール（課レベル）を実現する事となる。

このような現実社会を反映させたモジュール分割基準により、開発初期段階において多くのStateに悩まされる事もなくなり、また上述の情報に関する権限の定義にも役立つ。

またここで記述されたT-Process・Sub T-ProcessはFortranでプログラミングした別個のモジュールである。ここで書かれたモジュールは、ファイル入出力，ファイル内レコードから内部形への変換，入力データのチェック等、APO-Sの必要とする本質的な機能ではなく、計算機操作のためのものが全体の2/3をしめている。従ってこれも、ファイル定義、ファイル操作・データチェック機能の実現によりSOFT側が吸収してしまえば、使用者が記述を行なわなければならないのは、患者対薬品のチェック等たかだか20モジュールとなる。

ここで現在水野調剤薬局で実際にNCR I-8270上で稼動中の全てをCOBOLで書かれた元のAPO-Sと我々が作成したAPO-Sを比較したうえで、SOFTの特徴をまとめておく。

(1) 開発の容易性

SOFTが対象とする会話型データ処理システムのプログラミングでは、大部分が画面への入出力とそれに付随する命令から成り立っている。従ってプログラマはコーディングに際してそうした一連の作業を繰り返し行なう事になる。SOFTにおいては、これらの処理はConv定義においていくつかのパラメータを入力するだけで定義する事ができる。また、これらの入出力処理の処理フローについてもサポートを行ない、Conv定義に基づいてその実行を行なうため、プログラマは画面処理に関して、完全にプログラミングから開放されたと言ってよい。

同様に2次記憶装置へのI/Oに関するプログラミングでも、ファイルの入出力及びレコードと内部データの変換という一連の命令から成り立っている。SOFTにおいては、これらのファイルも、自己記述型ファイルの導入と、ファイル操作のサポートにより、サポート機能の呼び出しという形にまとめ、プログラマの負担を軽減している。

さらに、大規模システムの開発に際して常に問題となるモジュール化についても、SOFTは情報による抽象モジュール、画面によるState、入出力によるSub Stateという明確なモジュール分割の手段を提供している。これはまた、開発者に対して処理フローの理解を深めさせるという効果を持つ。

また、これらのモジュール間の実行制御は全てSOFTが管理している。そのために開発者に残されたプログラミングの負担は、対象システムに個有な、単一の機能を果たし、そのためにモジュール内の論理が明確に把握され容易に実現する事のできる各機能モジュールについてのみである。

また、前述のとうりSOFTにはT-Process・Sub T-Processを無視して実行を行なうモードがある。これによりState、Frame、Convの定義の正当性はたやすくテストする事ができる。加えてシステムパラメータを用いてT-Processの制御を行なえば、開発時の機能モジュールに対するデバックもSOFTを通して容易に行なえる。

(2) 変更の容易性

会話型システムの成否は利用者とのインターフェイスがいかに扱いやすいかに

かかっている。けれども、こうした側面は実際に使用してみて、利用者の意見をフィードバックしてみなければ、不備な点を発見できない事が多い。SOFTにおいては、画面処理はプログラムから明確に分離され、しかもStateの階層化によって修正箇所をたやすく発見できる。さらにこれらの定義は、項目単位のパラメータによって行なわれているため、変更はパラメータ値を修正するだけで行なう事ができる。

また、実際にプログラムされている部分についても、機能的な独立性が高い。従って必要な機能の追加・改良に際してもプログラムの変更が必要となるのは局所的であり、プログラム全体に影響を与える事はほとんどない。

さらにファイルのアクセスは単一の操作モジュールが行なっているため、ファイル構造との独立性も高く、ファイルの変更が影響を与えるのは操作モジュール内のファイル定義部のみである。

このようにSOFTではモジュールの独立性が高く、システムの変更に対して、修正箇所の発見が容易であり、さらに修正もモジュール内に限られるため、たやすく行なう事ができる。

(3)実行速度

SOFTはシステムの実行に際して定義情報ファイルを解釈しながら実行を行なう一種のインタプリタである。従って全てがプログラミングされ、オブジェクトを実行するシステムと比較すると、実行速度は落ちる。しかし、SOFTがインタプリタとして動くのはConv部とT-Process・Sub T-Processの機能モジュールへの制御の引き渡し時のみである。ここでConvの実行速度を決定するのは、割り込み・転送等入出力に関する部分であり、さらに利用者側の応答速度も実行速度に影響を与える。従って多少の計算時間の増加は無視できる程度のものである。

また、T-Processについても、実際の処理は機能モジュールというオブジェクトによって行なわれるため、問題となる程の遅れはない。事実2つのAPO-Sを比較しても、人間が感じる程の差は生じていない。将来インテリジェント・ターミナル等を用いた処理の多重化が行なわれれば、その差はより小さくなると思われる。

(4)移植性

SOFTはほとんどがFortranによって書かれており、大きさもリースレベルで、8500行程度で、SOFTの基本概念を反映したモジュール構造を持っている。従ってFortranコンパイラを持つ計算機には比較的容易に移植する事ができる。ただし、以下の2モジュールはアセンブラで書かれており、書き直す必要がある。

- 入出力ルーチン
  一般にFortranはカナ文字を扱わない。また現在SOFTが実現されているOSの入出力ハンドラがカナ文字を考慮していないため、アセンブラで書かれた入出力ルーチンを使用している。
- データ転送ルーチン
  画面を介した入出力に際して、データの転送には実際のアドレスを用いている。アドレスによる参照はFortranには困難なため、アセンブラで書かれた転送ルーチンを使用している。

5. 結論

以上のようにSOFTは、要求仕様定義・設計・製作・テスト・運用・評価・改良と続く、通常のアプリケーションシステムのライフサイクルの全ての局面で使用する事ができるツールである。さらに計算機内部の機能によらず、画面を中心とした人間に把握しやすい概念を用いて、常に人間を中心としたシステム開発が可能となるように考慮されている。こうした概念はモジュール分割の手段ともなり、従って計算機の非専門家にとってもシステム開発が容易なものとなる。

このようにSOFTは設計時、プログラミング時の開発者の負荷を大幅に軽減しているが、逆に使用者はSOFTの定義Sub Languageと機能モジュールのための通常のプログラミング言語という2種類の言語を知っている必要がある。従って今後は機能モジュールの定義をいかにSOFT内にとりこんでいくかが問題となる。

6. 参考文献

1) 東 基衛
情報処理システム設計用ツールの現状と将来方向
情報処理 Vol. 20 No. 6 p.511~p.518 1979年6月

2) 前川 守
情報処理システムのモジュール化
情報処理 Vol. 21 No. 7 p.751~p.768 1980年7月

3) Parnas, D.L
A Technique for Software Module Specification with Example
CACM Vol.15 No. 5 p.330~p.336 1972年5月

4) Parnas, D.L
On the Criteria to be used in decomposing systems into modules
CACM Vol.15 No.12 p.1053~p.1058 1972年12月

5) Myers, G.J (国友義久, 伊藤武夫 訳)
ソフトウェアの複合/構造化設計
近代科学社 1979年6月

6) 浦, 青木, 小川, 川口, 近藤, 水野, 山本, 川俣, 野沢, 坪内
Advanced Patient Oriented Pharmacy System (APO-S)
MEDINFO'80 p.924~p.928 1980年10月

7) 川西 正章
薬事情報システムの設計
慶應義塾大学大学院 昭和53年度 修士論文

8) 青木 隆
システム開発支援ツール: SOFT
慶應義塾大学大学院 昭和54年度 修士論文