ソフトウェア工学 32－4
（1983. 9. 28）

対話型設計システムの作成支援ツール

岸 知二　　紫合 治
日本電気(株) ソフトウェア生産技術研究所

1. はじめに

標準の端末において、高度な対話画面を容易に作成・修正するためのツールを紹介する。これは、現在開発中のソフトウェア設計支援システムのユーザインタフェースを実現するために使われる。

ソフトウェア設計支援システムでは、現在、図形による設計情報の入出力[1]と、設計記述言語による入出力が可能である。図形による入出力は分かりやすく簡単であるが、現状では端末の制約があるのが問題点といえる。一方、言語による入出力は端末を選ばず、またバッチ的処理には適しているが、ユーザにとっての分かりやすさ、使いやすさの点で問題がある。

こうした事から、標準の端末で用いることができ、かつユーザにとって分かりやすいインタフェースを提供することを目的として、本ツールを開発することになった。本ツールによって作成される設計支援システムのインタフェースは、入力された情報を見ながら新たな情報を入力したり、過去の情報を修正したりするインタラクティブな使用に適している。一方、大量の情報の一括入力などには、従来の図形や言語によるインタフェースが優れていると考える。

対話画面を作成するツールとしては、OAやソフトウェア向け生産システムの分野で、いくつかのものが作られている[2]。また、ソフトウェアのプロトタイピングの目的で作られたものもある[3]。

設計支援システムにおいては、入力情報が多様なため、一般のデータエントリ的画面では不十分であり、また内部での処理も特殊で標準化がしにくい。さらにユーザにとっての分かりやすさを考えると、標準の端末（キャラクタディスプレイ）でもある程度高度な画面制御が必要と思われる。

本ツールは、プログラマの定義に基いて画面データを生成する部分と、そのデータを参照して実際に画面入出力を行なうルーチン群からなる。プログラマはデータ生成後、自分のシステムからこのルーチン群をコールすることにより、画面入出力を行なうことができる。

2. 設計支援システム

我々は従来、ソフトウェア開発保守システム（SDMS）[4]を開発・適用してきたが、その経験を踏まえて、新たな設計支援システムを開発中である。これは、システムをトップダウン的に詳細化していく過程を中心に支援を行なうシステムである。

この設計支援システムでは、システムをユニットという単位で表現する。システムはいくつかの機能によって実現されるが、この機能も各々ユニットとして表現する。この場合、システムを表わすユニットと、それを実現するための機能を表わすユニットの間には親子関係があるという。この機能を表わすユニットも、さらにいくつかの機

能によって実現されるので、いくつかの子ユニットを持つことになる。このようにシステム全体からプログラムモジュールレベルのユニットまで、親子関係による階層構造によって詳細化していく。

さらにユニット間には、callやreadなどのコントロールやデータの流れに相当する機能間関係を持たせることができる。（図1）

本設計支援システムでは、上述の親子関係や機能関係を骨組として、様々な設計情報を表現することができる[5]。

3. 対話画面のイメージ

設計情報を入力する際に、どのような画面を提供すれば作業がやり易いかを考えてみる。

図2(a)は、機能間関係を登録するときの画面の例である。

機能間関係は、あるユニットの子ユニット間に定義でき、主語、関係型、目的語といくつかの詳細情報からなる。この画面では表の形で主語、関係型、目的語を入力させている。この際、主語、目的語となれるユニット名を右上に、関係型名のリストを右下にメニューとして示し、番号を選ぶことにより、キーインを容易にしている。詳細情報は、指示を与えると左下のエリアで入力できる。

こうした表やメニューは、そのエリアだけで自由にページ送りができ、たくさんの情報を表示できるようにする。

図2(b)は同様のレイアウトでユニットの子ユニットを登録するときの画面の例である。ユニット名、ユニット型などを表で入力し、詳細情報を左下のエリアで入力できる。ここでは右上のエリアに親ユニットの情報を検索して、表示している。

こうした画面を実現するためには、以下のような機能が必要となる。

①画面をいくつかのエリアに分割し、各々のエリアで必要な情報の入出力ができる。

②決められたシーケンスでの入力、出力の繰り返しだけでなく、表中では自由にカーソルを動かして入力が行なえる。

③画面上に表示する各エリアの内容を動的に制御できる。

④表などは、自由にページ送りすることを可能にし、画面の大きさに制約されない表を表現できる。

本ツールは、こうした画面を容易に作成、修正するための手段を提供することを目的としたものである。

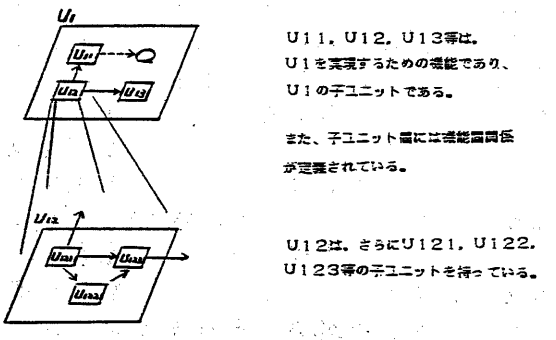


図1　設計支援システムでの、ユニットの親子関係と機能間関係

4. 機能概要

以下に、本ツールによって作成できる画面の機能について述べる。

4.1 フィールドとテーブル

画面での入出力の基本単位は、"フィールド"と"テーブル"である。フィールドは"メッセージ"と"入力エリア"からな

る。メッセージは画面に出力される文字であるが、常時同じ内容を出力する"コンスタント"の部分と、実行時に値の変わり得る"変数エリア"の部分からなる。入力エリアは、エンドユーザが入力を行なうエリアである。

このフィールドは、次のような使い方ができる。

① メッセージ表示後、入力エリアを示し、入力されるまで待つ。

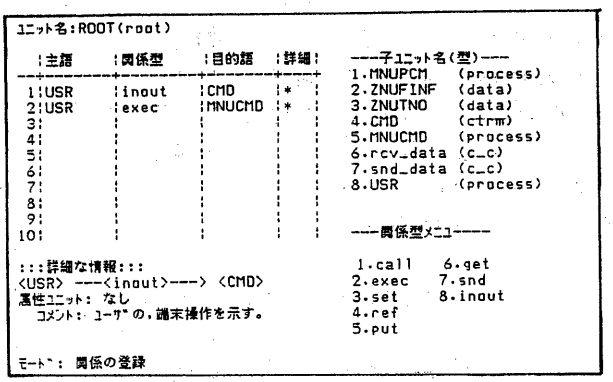
② メッセージ表示後、入力エリア中にそのフィールドに対する最新の入力値を表示し、入力を待つ。この場合、そのままリターンキーを押すと前回と同じ値が入力され、修正後リターンキーを押すと修正された値が入力される。

③ メッセージ表示後、前回の入力値を入力エリアの位置に表示する。(入力はさせない。)

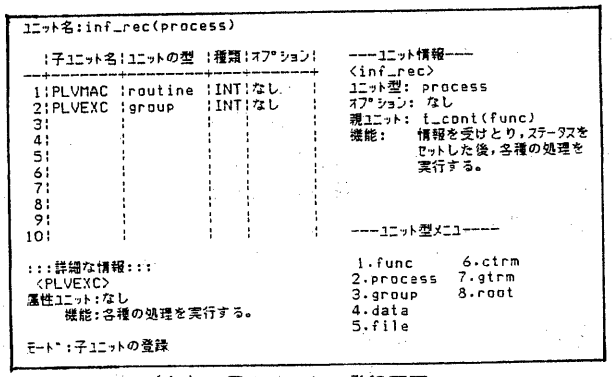
④ そのフィールドの、メッセージや入力エリアを消す。

またテーブルとは表形式の入出力を行なうものであり、メッセージだけからなるヘッダと、メッセージといくつかの入力エリアからなる任意個の行とで構成される。

テーブルの入力エリアは、行と列によって指定することができる。最大行数は実行時に指定できるが、一時に表示できる行数は画面定義時に決まってしまう。そのため実行時には、入力エリアの指定に応じて必要部分を表示する。テーブルは以下のような使い方ができる。



(a) 機能間関係の登録画面



(b) 子ユニットの登録画面

図2 設計情報の登録画面の例

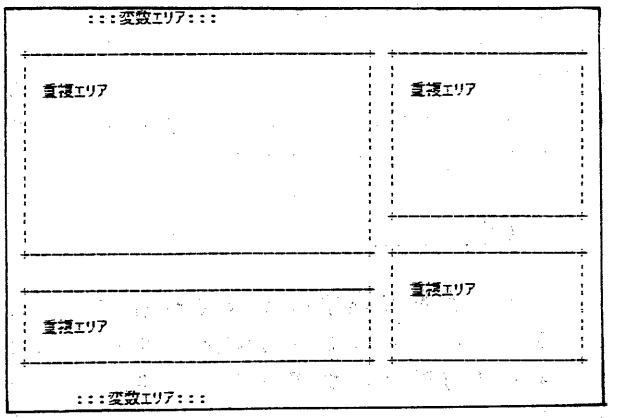


図3 図2の画面の画面レイアウト

①指定した行・列の入力エリアを表示し、入力されるまで待つ。
②ユーザが自由にカーソルを動かし、任意の入力エリアに値を入力する。この際ユーザは、入力完了を^D（コントロール－D）によって示す。
③出力するのみ
④テーブルを消す。

図2(a)の例では、関係登録の表をテーブルで、メニューをフィールドによって（この場合③の使い方）実現できる。

4.2 画面レイアウト

一枚の画面を"フォーム"という。フォームは、4.1で述べたフィールドとテーブルによって構成される。

同じ場所に2つ以上のフィールドやテーブルを定義するときは、"重複エリア"を用いる。

重複エリアは長方形の単位で定義する。重複エリアが重なり合ったり、入れ子になることは認めていない。重複エリア内にはいくつものフィールドやテーブルを定義できる。

以上述べた、様々な画面の情報のうち、入力エリア、変数エリア、重複エリアの位置や大きさなど、コンスタントを除く全情報を"画面レイアウト"と呼ぶ。

図2の(a)，(b)は、いずれも図3の画面レイアウト上で実現できる。

4.3 漢字データ

メッセージ等は漢字化した方が分かりやすい。本ツールでは2バイトコードによって漢字の入出力をすることができる。

しかし使用環境によっては漢字を扱えない端末を用いる場合もあり、漢字

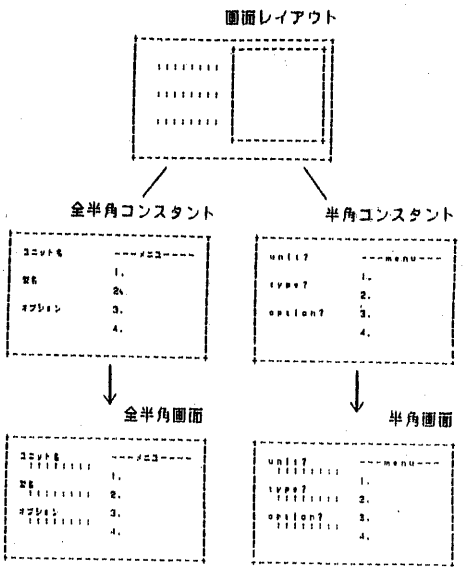


図4　画面レイアウトとコンスタントの関係

しか用いることができないことが不都合な場合もある。従って本ツールではひとつの画面レイアウトに対して、英文（半角文字）によるコンスタントと、漢字（全角文字）を使ったコンスタントとを別々に定義することができる。2種類のコンスタントを用意しておけば、実行時に端末に応じて出力するコンスタントを切り換えられる。もちろんユーザの必要に応じ、半角だけ、あるいは全角・半角の混ざった画面の一方だけを作ることもできる。（図4）

4.4 その他

本ツールではいくつかのコントロールキャラクタをユーザに開放している。これらのコントロールキャラクタがキーインされたときにどういう処理をするかを宣言することにより、画面の状態にかかわらずその処理を行なうことができる。この機能を用いることによ

り、例えば入力エリアに対する入力の途中で、メニューの表示を指示するなど、柔軟な処理が可能となる。

5. 画面定義の方法

5.1 概要

プログラマは本ツールを用いて画面を定義し、画面データとチェックルーチンを生成する。プログラマには、いくつかのルーチンが提供される。これらのルーチンは生成された画面データを見て、実際の画面入出力を行なう。プログラマはこれらのルーチンを自分のシステムからコールすることにより、画面制御を実現できる。（図5）

本ツールは、以下のような特徴を持っている。

①画面の定義は、完成時のイメージを見ながら行なうことを原則としている。このために、画面定義に適した機能を持つ専用画面エディタを提供する。

②定義した画面情報はデータとして蓄えられるため、画面の変更や修正は、データの修正のみで手軽に行なえる。

③画面定義の過程で指示した情報は、ファイルに蓄えられるので、一部の修正のために同じ指示を繰り返すことなく、簡単に修正ができる。

④画面情報はデータとして蓄えられる反面、それをどう用いるか（4.1参照）は、ルーチンのパラメータによって決定できるため、柔軟なシステムを作ることができる。

5.2 画面定義の手順

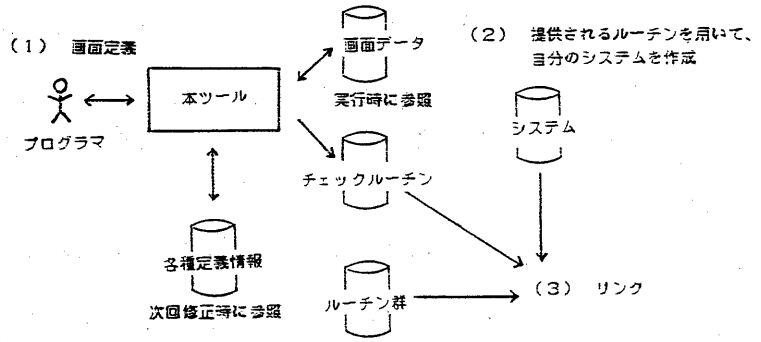


図5 作業手順の概要

エリア	最大サイズ	カーソル位置	指示する情報
入力エリア 変数エリア	80字*1行	左端	エリア名 エリア長
重複エリア	80字*22行	左上端	エリア名 エリアサイズ

表1 エリアの定義方法

ステップ1（画面イメージ作成）

専用の画面エディタを用いて、作りたい画面イメージを作成していく。図4で示したように、本ツールではひとつの画面レイアウト上に、半角画面と全半角画面を定義できる。半角画面定義もしくは全半角画面定義のいずれかを指示し、画面レイアウトとコンスタントを同時に定義できる。

エディタの画面は、完成時のフォームと同じ大きさ（80字×22行）なので、実際の位置にカーソルを動かしコンスタントや各種エリアを書き込んでいけばよい。漢字はローマ字漢字変換によっ

て入力可能である。

各種エリアは、表1の要領で定義できる。

重複エリア内にフィールドやテーブルを定義しているときは、エリア内しかカーソルを動かせない。エリア内は次々とページ送りができ、複数のフィールドやテーブルを定義できる。

テーブルは重複エリアにしか定義できない。テーブルの定義は図6に示した要領による。

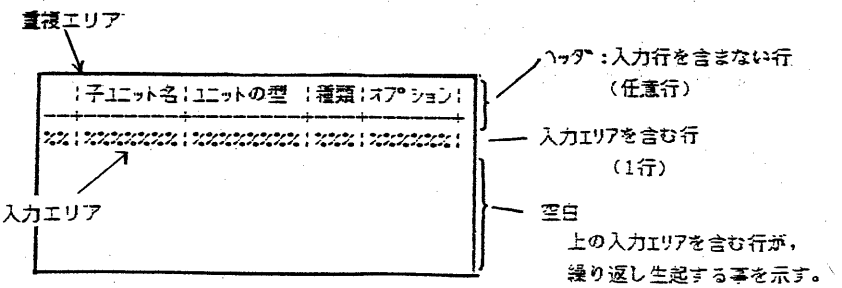


図6 テーブルの定義方法

ステップ2（フィールド分割）

1つのフォーム中には複数のフィールドが定義されている。ステップ1で作成した画面イメージでは、どのコンスタント，変数エリア、入力エリアが同じフィールドに属しているか、対応がはっきりしない場合がある。その場合には図7の要領でカーソルを動かして、画面をフィールド毎に分割する。

ステップ3（入力タイプ・条件指定）

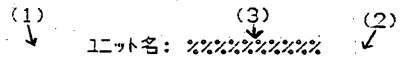
各入力エリアに対して、入力されるデータのタイプを指定する。タイプとしては、整数型，実数型，文字型、文字列型，単語型（先頭の空白を無視）及び全角型を指定できる。

これらは、実行時に入力された文字列がこのタイプに適しているかどうかをチェックする際に用いられる。

また入力条件についても、実行時にチェックされる。

タイプ、入力条件とも、入力時チェックに合わなければ、再入力を促す。

尚、全角型指定の入力エリアに入力させる時は、ローマ字漢字変換によって入力を行わせる。



(1) エリアの左上（1行のときは左）を指示。

(2) エリアの右下（1行のときは右）を指示。

(3) 入力エリアを指示。

図7 フィールド分割の方法

ステップ4（データ・プログラム生成）

上述のステップ終了後，画面データ及びチェックルーチンの生成を指示すると、それらが生成される。

5.3 システムの作成

表2に示すようなルーチンが提供される。プログラマは自分のシステムからこれらのルーチンをコールすることにより、5.2で定義した画面を実現できる。

これらのルーチンは、使われている端末を判断し、それに応じた制御シーケンスによって画面制御を行なう。き

た漢字の使える端末かどうかに応じて、半角画面、全半角画面のいずれかを表示する。

trminit()	端末に応じた初期化を行なう。
form(foname)	フォームの画面データ入力。
field(finame, mode, buf)	フィールド入出力。
varset(vname, fmt, args)	変数エリアへの値のセット。
tblinit(tname, maxline)	テーブルの最大行数の指示と初期化。
table(tname, l, c, mode)	テーブル入出力の指示。

表2　ルーチンの例

6. おわりに

本ツールは、標準の端末における高度な画面入出力の実現を支援するものである。

本ツールによって作成される画面は、以下のような特徴を持つ。

①画面レイアウトの定義により、画面をいくつかのエリアに分割して用いることができる。
②重複エリアを用いることにより、動的な画面制御が可能となる。
③ひとつのフォームが半角画面と全半角画面を持ち、漢字端末やそうでない端末の混在する環境でも柔軟な対応ができる。

また本ツールは以下のような特徴を持つ。

①完成時の画面イメージを見ながら作業ができる。
②画面のレイアウト等の変更はデータの修正のみで行なえる。
③フィールドやテーブルの使用法を実行時に決定したり、コントロールキャラクタを自由に用いることにより、きめ細かく柔軟なシステムを作ることができる。

なお本ツールは、我々が開発した対話型システム作製支援システム[6]の試行経験をもとに、機能改良と使いやすさの改善を図ったもので、現在Cを用いて開発中である。また本稿では触れなかったがスピードの改善もなされている。

既に述べたように、本ツールは現在開発中の、ソフトウェア設計支援システムのインタフェース作成に用いられる予定であり、今後さらに機能等の見直しを行なっていく予定である。

参考文献

[1]米田，秋原，岩元「ソフトウェア設計のための図形スキーマシステム」第27回情処大会，1983
[2]M.Matsumoto「SEA/I - Application Software Productivity System」，Soft Fair '83 Proceeding，IEEE cs No.478，1983
[3]A.I.Wasserman「User Software Engineering and the Design of Interactive Systems」5th ICSE 1981
[4]紫合，岩元，藤林「統一的設計方法論に基づくソフトウェア設計システム」情報処理 Vol.21 No.5 1980
[5]西谷，小山田，宮下，紫合「ソフトウェアの階層的設計表現モデル」第27回情処大会 1983
[6]岸，紫合「対話型システム作製支援システム」第26回情処大会 1983