データ入力支援インタフェースに関する一考察

3S－8

(株)東芝　総合研究所　亀山　研一

1．はじめに

一般的に設計支援システムでは、入力事項が多いため、全てのデータを逐一矛盾なく入力するのに多大な労力を要しており、データ入力を効率的に行えるインタフェースが強く望まれている。

本報では、筆者らが開発中のプラント機器配置設計支援システム[1]を例として、このような機能を持つインタフェースの実現方法を考える。なお、システムは主にAS3260で稼働するART3.1上で作成した。

2．構想

設計者は通常、概略から詳細へ、全体から細部へとトップダウン的に設計を進める。したがって、データ入力もこのような流れに沿って行われるのが最も自然でかつ効率的であると考えられる。

システム上には、入力データに内在する階層性や依存・制約の関係を利用して、この流れを実現する。例えば、ユーザが建屋の全長（概念的に上位の階層の項目－－全体をまとめるようなデータ）を入力すると、システムがそれに見合う各部の寸法（概念的に下位の階層の項目－－細かいデータ）を即座に算出して提示するようにする。各部の寸法に関しては、必要があれば随時ユーザがそのデータを修正する（図１）。

こうすれば、ユーザは下位の項目に関しては、必要箇所のみの入力で済むし、制約条件を特に意識することなく無矛盾なデータのセットを作成することができるため、大幅に入力工数を削減できる。さらに、必要があればシステムが求めたデータ（デフォルトデータと呼ぶ）を修正することで、ユーザの意図を制約条件の範囲内で設計に反映させられる。

ただし、デフォルトデータの修正など、最上位以外の階層の項目でのデータ入力を許すことで、場合によっては前に入力したデータと矛盾することがある（例えば、図２）。この場合、できるだけユーザの意図を生かしながら矛盾を解消する方法を検討する必要がある。

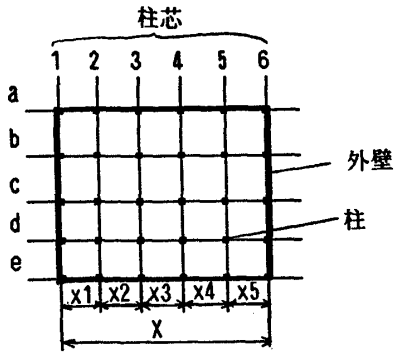
3．システムの実現

前章に述べた構想をシステム上に実現するため、

①入力事項の階層化
②デフォルトデータ導出の手続き化
③デフォルトデータ修正時の（矛盾解消の）手続き化
④上位項目から下位項目へのアクセスパスの設定

などを検討する。

機器配置設計支援システムの入力事項とその階層構造は、図３のようになる。



全長:X　柱芯間隔:xi,(i=1～5)　柱芯数:n=5

図１　建屋に関する入力事項

制約条件：Ｘ＝ｘ１＋ｘ２＋ｘ３

ユーザ入力：

stage 1:Ｘ＝１０

↕ 矛盾

stage 2:ｘ１＝４，ｘ２＝４，ｘ３＝４

図２　入力データの矛盾

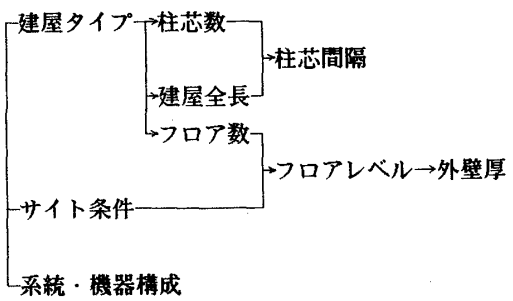


図３　入力事項とその階層構造

Consideration of Data Input Interface
Toshiba Corp.
K.Kameyama

図３の矢印で結ばれた項目同士の関係から、デフォルトデータを求める手続き（階層的に上位のデータから下位のデータを求める手続き。制約条件が満たされればよい）とユーザがデフォルトデータを修正したときの手続き（下位のデータを変更したときの他のデータへの影響。ユーザの意図をなるべく尊重するように全体を修正する）を定義する。

例えば、図１に示す建屋の柱芯間隔(x0,x1, …,xn)と全長(X)、柱芯数(n)の間には、

$$X=x1+x2+\cdots+xn \quad \text{———————} ⓪$$

という関係（制約）がある。これより、

1)上位のデータである柱芯数(n)または全長(X)から、柱芯間隔(x0,x1, …,xn)のデフォルトデータを求める手続き

a.デフォルトデータを初めて求める場合、または既デフォルトデータをユーザが全て修正している場合

$$xi=X/n \quad \text{———————} ①$$

b.デフォルトデータのうちのいくつかをユーザが既に修正している場合

ユーザが値を修正した部分（xm;m=j1,j2, …,jk)が影響を受けないようにするため、

$$xi=(X-\Sigma xm)/(n-k) \quad \text{———} ②$$

2)下位のデータである柱芯間隔(x0,x1, …,xn)を修正した場合の手続き

a.既デフォルトデータをユーザが全て修正しているまたはしたことになる場合

全長(X)と柱芯間隔の合計（$\Sigma xi, i=0,1, \cdots, n$）が一致しないなら、全長(X)を変更するかどうかをユーザに問い合わせる。全長を変更する場合は、Xの値のみ変更される（この場合、柱芯間隔のデータは全てユーザが手を加えているため、1)のa の手続きは実行されない）。変更しない場合は、ユーザが入力したデータ以外を、1)のb の手続きにより計算し直す。

b.デフォルトデータのうちのいくつかをユーザが既に修正している場合

1)のb と同様の手続きが実行される。

①、②はあくまでも制約条件⓪を満たすような値を求める式であり、他に条件があった場合は当然異なったものとなる。

これらの手続きはART のオブジェクト指向の機能のメソッド(ARTではaction) 用いて作成した（図４）。

また、データの入力は項目ごとにメニューを通して行われるようにしたが、このメニューのアクセス方法を上述のメソッド中に定義し、入力を行った項目以外の項目でデータが変更があった場合にその項目を参照できるようにした（図５）。

```
(defschema total          --オブジェクト名
 (value X)                --slot値
 (user-modify t))         --デフォルトデータを修正したかどうかのflag

(defschema t-method
 (is-a active-value))

(add-active-value
 'total 'value 't-method)

(defaction modify-after(t-method)
 (schema slot old-value new-value)
  ～手続きの実行プログラム～
```

slot値が変更されるとメソッドが自動的に起動される設定

図４　手続きのART 上での表現

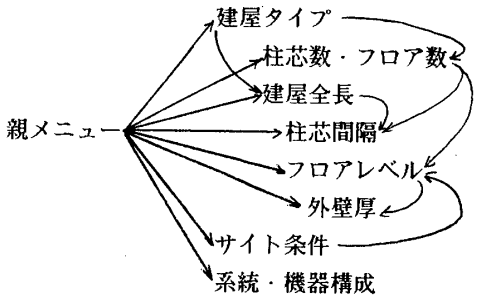


図５　メニューのアクセスパス

４．おわりに

入力工数を削減し、かつ無矛盾なデータセットの作成を支援するインタフェースについて、プラント機器配置設計支援システムを例にとって説明した。今後は、このようなインタフェースの作成手法を一般化し、他のシステムでの適用を図っていくのと同時に、システムの評価手法についても検討していく予定である。

参考文献

[1] 亀山他、原子力プラント機器配置設計支援エキスパートシステム、計測と制御、vol.27,no.10(1988)

[2] ART3.1 Reference Manual (1988)