Title	制約指向に基づく基本配置設計支援システムの研究 (第2報 ： オブジェクト指向によるシステム構築)
Author(s)	赤木, 新介; 藤田, 喜久雄; 仲戸川, 哲人; 井上, 具大
Citation	日本機械学会論文集 C編. 56(528) P.2294-P.2301
Issue Date	1990-08
Text Version	publisher
URL	http://hdl.handle.net/11094/3216
DOI	
rights	

日本機械学会論文集(C編)
56巻528号(1990-8)

論文 No. 89-0996 A

制約指向に基づく基本配置設計支援システムの研究*
(第2報：オブジェクト指向によるシステム構築)

赤木新介*1, 藤田喜久雄*2
仲戸川哲人*3, 井上具大*3

A Layout Design System based on Constraint-directed Reasoning
(2nd Report: Building the System with Object-oriented Programming)

Shinsuke AKAGI, Kikuo FUJITA,
Tetsundo NAKATOGAWA and Motohiro INOUE

A layout design of a system such as a power plant is a time-consuming and expertise-based task and needs to satisfy the spatial constraints. In the first report, we have proposed the general approach composed of the general procedures and the individual specifications on a layout. In this paper, we construct the system based on the proposed approach with object-oriented programming technique. On the system, the general procedures such as constraint propagation are represented as methods on the class objects with the hierarchical structure for the representation of the specifications, and the individual specifications of a design object such as a layout space, components and spatial constraints are represented as instance objects and associations among them. Moreover, we provide the useful interface for interaction with a user and the representation of a layout result. The system structure has the generality and enables to apply it to various kinds of layout design.

Finally, we apply it to the layout design of a nuclear power plant and ascertain its validity and effectiveness.

Key Words: Design Engineering, Computer Aided Design, Layout Design, Object-oriented Programming, Plant Design.

1. 緒言

各種の発電設備や化学プラントなどの大型プラントの設計においては，プラントを構成する各種構成機器の性能を発揮させ，かつ空間的な制約を満たすように，それらの位置関係を定めるいわゆる配置設計，なかでも基本配置設計が高い比率を占めている．前報[1]では，このような配置設計は各種の制約条件を満足するように配置を行っていく点にその特質があることを示すとともに，知識情報処理技術のひとつである「制約指向プログラミング[2][3]」に基礎を置く手法を提案した．この手法では，空間的な制約などの設計条件とそれを満足する設計解を得るための設計機構とを明確に分離し，さらに，配置設計における複雑な制約関係を取り扱うために，値域の概念[3]といわゆる生成検査法とを組み合わせて，前者の宣言的な記述に従って後者の一般的な機能により配置設計を実行する．本手法は汎用的で拡張性に優れており，各種の配置設計問題に対する支援システムの基礎となるものである．一方，そのようなアルゴリズムに対して，具体的なシステムを構築するためには，その問題解決方法に合致した対象モデルを計算機上で構成する必要があり，その構成がシステムの最終的な汎用性や拡張性を左右することになる[4]．

本報では，前報で示したアルゴリズムに対して，問題を構成する設計対象物の構成要素や配置制約などについて実際の配置設計問題を調査して，それらをネットワーク状の関係として整理し，それをもとに具体的なシステム構築の方法として「オブジェクト指向プログラミング[5]」を採用する．これにより，各問題構成要素をオブジェクトとして表現して，各種の関係をオブジェクト間の協調関係として効率的に扱い，設計処理を実行するようにして，システムを構築する．さらに，全般的なシステム環境を考え，設計条件の設定，ユーザによる設計処理や結果の把握におけるユーザインターフェースの機能の向上をはかる．ここで用いた設計支援システムの構築方法は配置設計のみならず各種の設計問題に対して有効である．

* 原稿受付 平成元年8月1日.
*1 正員，大阪大学工学部 (〒565 吹田市山田丘2-1).
*2 正員，大阪大学大学院.
*3 正員，三菱原子力工業(株) (〒105 東京都港区芝公園2-4-1).

2．配置設計の過程と制約指向による基本アプローチ

2･1　配置設計の過程と取り扱う情報　配置設計は，プラントなどの設計対象物を構成する各種の要素の位置関係を定める過程であり，その過程は，処理の対象となる情報のレベルに応じて，設計対象物の構成要素のトポロジカルな位置関係を取り扱う基本配置設計の過程と，その結果に基づいてそれらに厳密な位置寸法を定める詳細配置設計の過程の２つに分割することができる[1]．本研究では，このうち前者の基本配置設計の過程について考える．このとき，配置しようとする対象空間はその問題の性質に従って適切なサイズの「単位格子」の集合として取り扱うことが有効であり，その場合，基本配置設計の過程で取り扱うものは以下のようにして整理することができる[1]．

(1) 配置要素（component）：配置すべき設計対象物の構成要素であり，それぞれの形状は後出(2c)の単位格子の個数とそれらの組合せ関係により把握できる．

(2) 配置空間（space）：(1)の要素を配置していく対象であり，(2a)配置空間の全体・(2b)配置空間の部分・(2c)配置空間の単位格子（compartment）に分けられる．

(3) 配置制約：上記の要素や空間の間で満たさせるべき空間的な位置関係に対する条件であり，それに基づいた処理の内容の違いにより，(3a)領域制約（region constraint）・(3b)検査制約（test constraint）・(3c)選択制約（priority constraint）・(3d)大域制約（global constraint）の４種類に類別することができる．

プラント設計の場合，(1)にはプラントを構成するポンプやタンクなどの各種機器や部屋が，(2a)にはプラントの建屋が，(2b)にはその部分である特定の階層などが対応する．また，(2c)については，事例として取り上げる原子力発電プラントの場合には建屋の"通り芯(補強壁の位置)"に対する考慮をもとにしてそのサイズを設定することができる．さらに，(3a)については「Ｖタンクは Ｆポンプよりも上階に設置する」，(3b)については「Ｒ冷却器は Ｒポンプの真上に設置する」，「Ｒポンプと ＰＳの長手方向は互いに異なっている」，(3c)については「Ｐ水冷却器およびポンプは Ｐに近い方が良い」，(3d)については「ＰＳは出入口に通じていなければならない」などが例として上げられる．以上の問題構成要素に対して，単位格子の概念を導入したことにより，基本配置設計の問題は「各配置要素に対してその形状に見合った配置空間の単位格子を配置制約に従って割当てていく」という問題となる．

2･2　制約指向による基本アプローチ[1]　上記の各種の配置制約は，満たされるべき条件であると同時に，それらを設計処理の点からみると，それぞれの要素を配置しようとしたときに，配置可能な単位格子を限定したり，単位格子を組み合わせることによって生成される配置候補を検査したり，さらに得られた複数の候補のなかでより良い候補を優先付けたりする働きをする．前報で示した手法[1]は，このうち後者の働きに基づいて，ある配置要素に着目しては，制約の記述から配置手続きを展開していくことにより，その要素の配置を行って，設計を進めていくものである．このような手法の利点は，具体的な配置処理の方法を個々の配置状況に応じて配置制約という設計条件から展開している点にあり，一般性に富んだシステムを構築する足掛りとなるものである．

3．システム構築方法とその構成

3･1　制約指向のオブジェクト指向による実現

前報[1]で示した手法に従ってシステムを構築するにあたり，各種の情報や知識を表現して配置処理を実現する方法として，以下の理由により，オブジェクト指向プログラミング[5]を用いることにする．

(1) 制約の伝播に係わる非決定論的な処理を，従来の手続き的なプログラミング手法で実現することは容易ではない．

(2) 制約系を構成する空間や制約をそれぞれオブジェクトとしてモジュール化して扱うことができ，それらの連携した処理をオブジェクト指向における各オブジェクト間のメッセージ送信による連携に対応させることができる．

(3) 制約などに対する一般的な処理方法はクラスにメソッドとして，具体的な対象の構成や状態についてはインスタンスに変数として表現することによって，汎用性と拡張性に優れたシステムを比較的容易に構成できることが期待できる．

なお，従来の制約指向プログラミングに基づいたいくつかのシステムもオブジェクト指向のもとで構築されている[6][7]が，配置問題においては，制約は複雑で，その性質や取扱いも従来のシステムが扱ったものとは異なるため，独特のシステム構築が必要となる．

3･2　システム構成　図１に本システムの構成を示す．システムは，Lisp をプログラミング言語として用い，Lisp により記述したオブジェクト指向プログラミング環境のもとで，①：オブジェクト指向による知識ベースを中心に，②：設計全体を管理する設計

マネージャー，③：設計の過程においてユーザとインターフェースをとる部分，④：設計条件をもとにして各種のオブジェクトを初期設定する部分，⑤：配置設計の結果を視覚的に表示したり，配置設計の結果をファイルに出力したりする部分から構成される．それぞれの詳細は以下の章で示すが，①には Lisp 関数として記述された各種のメソッドが含まれ，また，視覚的なインターフェースの実現には，ワークステーション上のウインドウシステムなどの機能を用いる．

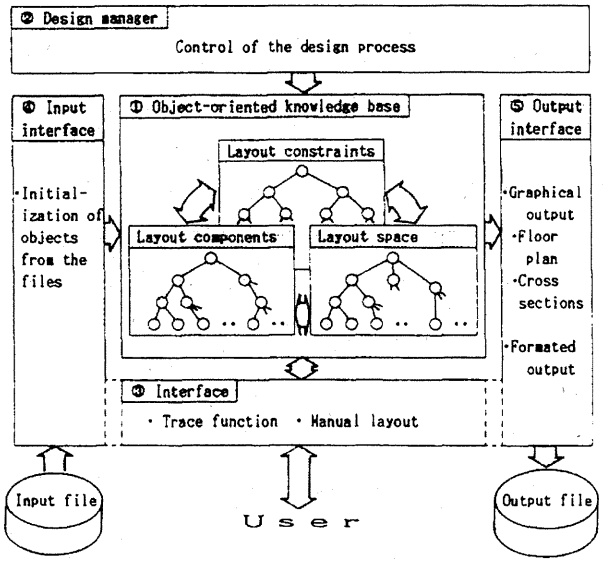


図１　配置システムの構成

なお，本研究では，EWS Sun-3 上で Common Lisp[8] によりシステムを記述し，ユーザインターフェース機能の一部についてはウインドウシステム SunView[9]，グラフィックインターフェース SunCore[10] を用いてＣ言語により記述する．また，オブジェクト指向環境は，Lisp上のハッシュ表，構造体，連想リストなど[8]を用いて構築した．

4．オブジェクトの連携による配置処理

4･1　配置設計におけるネットワークのオブジェクトによる表現　　配置設計の過程では，2･1節で列挙した様々の問題構成要素を取り扱う必要があり，これらの間には以下に示す種々の関係が存在する．

(1) 制約関係：ある制約がどの要素と要素あるいは配置空間との空間的な位置関係を規定しているかを表現する関係であり，制約自体が前述のように基本的に４種類に分類できることに対応して，この制約関係も同様に分類することができる．

(2) 配置関係：ある要素がどの単位格子に配置されているかを表現する関係．

(3) 空間部分関係：配置空間における全体と部分との関係，その部分と単位格子との所属関係を表現する関係．

(4) 単位格子間関係：ある隣接する２個の単位格子間の相対的な位置関係を表現する関係．

これら関係は，図２に示すようにそれぞれの種類に応じて，各問題構成要素をノード，関係をアークとするネットワークを形成している．配置設計における処理は，4･4節で後述するような方法により，配置関係を

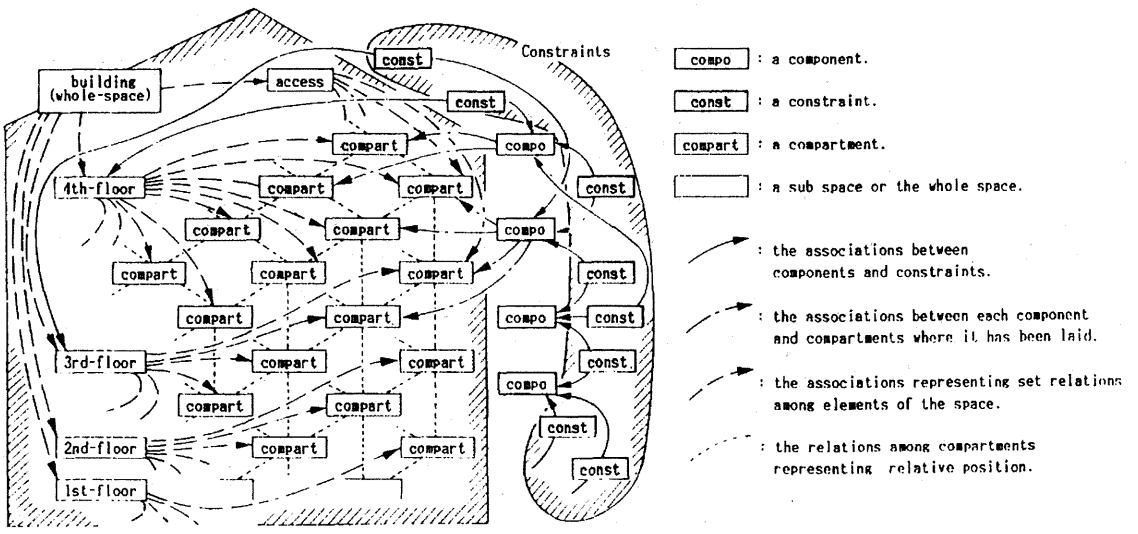


図２　問題構成要素間のネットワーク

生成しながらその上をノードからノードへと渡り歩いていくことにより行われるものとして理解することができる．本システムでは，このようなネットワークに対してオブジェクト指向プログラミングを導入して，図2の各ノードをモジュール化してインスタンスとして表現し，上記の各種の関係に対しては，新たに「オブジェクト間の関係（association，relation）」として扱うことにする．また，配置要素におけるその形状や配置候補群，配置空間におけるそれぞれの絶対的な位置などの各インスタンスに固有の情報は，それぞれのインスタンスに変数として保持するようにする．

なお，上記の「関係」による表現方法を加えることにより，オブジェクトそれぞれの内部状態の変数としての表現に加えて，オブジェクト間の帰属関係や支配関係などを容易に扱えるようになる．この表現は，関係が一度定義されると双方のインスタンスから参照できる，インスタンスが消去されるとそれに関連する関係はすべて消去されるなどの性質をもつ．

4･2　各種オブジェクトの階層的関係　　前節で示したネットワーク上の処理は，それぞれのインスタンスが固有の処理を行い，それらが連携することによって実行するようにする．オブジェクト指向においてはインスタンスにおける処理はそのクラスから継承されるメソッドにより行われるため，各種処理の種類や内容に応じて，クラスを用意してスーパークラス・サブクラスの階層構造を構成する必要がある．特に，配置制約においては，前述の領域制約・検査制約・選択制約などの区分に加えて，さらに詳細な区分があり，各制約において同じ抽象的な処理を行う時に，具体的な方法は共通の処理で対応できる場合もあれば，それぞれに個別の方法によって行う必要がある場合もある．このような処理内容の違いに対応してインスタンスを分類し，その分類に従ってクラスの階層を構成して，共通的な処理は上位のクラスに，個別の処理は下位のクラスに記述することにより，オブジェクト指向における差分プログラミングの性質を生かした効率的な処理の記述が可能になる．

4･3　オブジェクト指向による知識表現　　前節までに示したオブジェクトの表現の実例を以下に示す．

図3は，6章で取り上げる原子力発電所の事例における各種インスタンスの表現例である．図中(a)は配置要素の記述例であり，その形状や配置候補群を変数として，関連する制約を関係により保持している．(b)は配置空間の中のある単位格子の記述例であり，その絶対的な位置を変数として，配置されている要素を関係により保持している．(c)は(b)のような単位格子を

```
(inst (name sfp-p) (class regular-area)
      (iv (width 1) (depth 1) (height 1)
          (candidates
            ((comp-5-14-2) (comp-5-15-2))) ···      )
      (as (laid-to comp-6-13-2))
      (as-re (limit-search-comparts vertical-56 ··· )
             ···  ···                                ) )
```

（a）　配置要素の記述例

```
(inst (name comp-6-13-2) (class compartment)
      (iv (column 6) (row 13) (floor 2))
      (as (composed-of floor-2))
      (as-re (laid-to sfp-p)) )
```

（b）　配置空間の単位格子の記述例

```
(inst (name grand-level) (class boundary-compartments-set)
      (as-re (connect-to comp-12-1-4 comp-12-2-4 ··· )) )
```

（c）　配置空間の部分集合の記述例

```
(inst (name vertical-56) (class vertical-const)
      (iv (lower sfp-p) (upper sfp-hx) (inter-floors -1))
      (as (limit-search-comparts sfp-hx sfp-p)) )
```

（d）　配置制約の記述例

図3　インスタンスの記述例

```
(class (name search-comparts-const)
  (super region-constraint)
  (im (can-evaluate       (call can-eval-search-comparts-const) )
      (get-opposite-side
            (call get-opposite-side-of-search-comparts-const) )) )
```

（a）　ある領域制約のクラスの記述

```
(class (name contain-comparts-const)
  (super test-constraint)
  (im (can-evaluate       (call can-eval-contain-comparts-const) )
      (get-opposite-side
            (call get-opposite-side-of-contain-comparts-const) )) )
```

（b）　ある検査制約のクラスの記述

```
(class (name just-close-const)
  (super search-comparts-const)
  (cm (create (call create-just-close-const          ) ))
  (im (reduce-compartments
            (call reduce-comparts-by-just-close-const) )) )
```

（c）　ある隣接関係に関するクラスの記述

図4　クラスの記述例

元とする配置空間の部分集合の記述例であり，それに属する単位格子を関係により保持している．(d)は配置制約の記述例であり，その制約に関連している要素や領域を関係により保持している．なお，このようなインスタンスは，個々の問題に応じて4･5節に示すような記述から自動的に生成するようにする．

図4は，クラスの記述例であり，(a)はある領域制約のクラス，(b)はある検査制約のクラスであり，両者のスーパークラスには共通のものが存在し，同名のメソッドを持っているが，その処理の実態であるLisp関数はそれぞれ異なってたものを保持している．例え

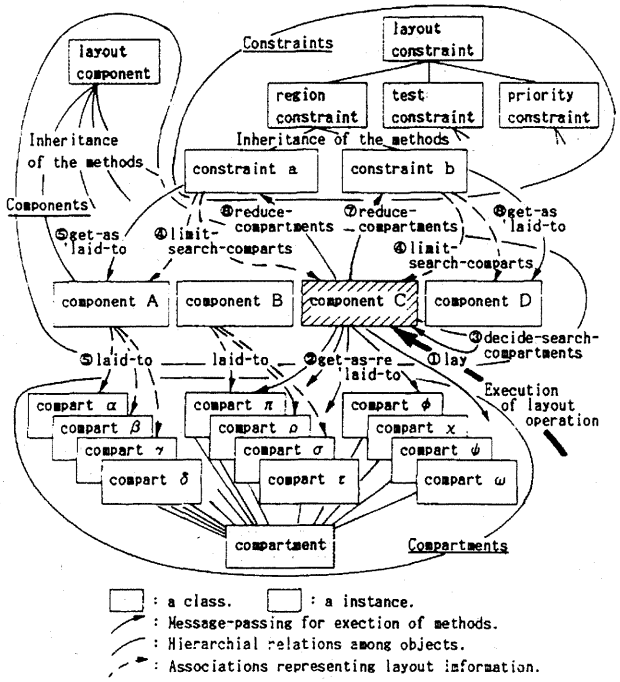


図５　メッセージパッシングによる配置処理の一例

```
(area charge-pa    (3 1 1) controlled   connect-to-access)  ……①
(area boric-ac-ta  (1 1 2) controlled   connect-to-access)
                    …   …   …
(area n-relay      (4 3 1) uncontrolled connect-to-access)
                    …   …   …
```

（ａ）　配置要素の記述例

```
(height            4 )
(north-west-length 16 )
(east-west-length  12 )                                    ……②
(grand-direction (4 north))
(grand-direction (4 west ))
     …   …   …
```

（ｂ）　配置空間の記述例

```
(close               spray-pb   safeaux-ps)   ……③
(different-direction spray-pb   safeaux-ps)
(just-under          spray-pb   spray-hxb )
(near                spray-addt spray-pb  )
          …   …   …
```

（ｃ）　配置制約の記述例

図６　設計条件の入力形式

ば，制約において対をなす要素の一方から他方を求めるための 'get-opposite-side' というメソッドは，共通的な処理をであるが，要素と制約を関係付けている関係の名称が異なるため，具体的な処理は異なったものとなる．また，(c)は，(a)のサブクラスの具体的な制約に対するクラスであり，インスタンスを生成して必要な関係や情報を保持される 'create' というメソッドや，そのインスタンスの内容に応じて単位格子を絞りこむ処理を行う 'reduce-compartments' というメソッドを保持している．

4･4　オブジェクトの連携による配置処理　　続いて，以上のような知識表現の下での配置処理の方法について示す．図５は，ある要素(C)の配置を行うために，何も配置されていない単位格子を領域制約によって絞りこんで，配置候補を生成する対象となる単位格子を集める処理の過程を示したものであり，以下の手順で制約伝播を行いながら処理が進められる．まず最初に，要素Ｃのインスタンスに対して配置処理を起動するメッセージ ①'lay' が送信されると，直ちに各単位格子のインスタンスに対する働きかけ②によって空の単位格子が集められる(③)．続いて，要素Ｃに関連する領域制約(ａとｂ)がインスタンス間の関係 ④ 'limit-search-comparts' から参照され，これらの制約によって単位格子群が絞りこまれていく．まず，制約ａによって要素Ａの位置が関係 ⑤'laid-to' によって参照され，制約の内容に従って一部の単位格子が除外される(⑧)．次に，得られた単位格子群がさらに制約ｂによって処理される(⑦)が，この場合は，要素Ｃと対をなすＤが配置されていない(⑧)ため，制約ｂによる単位格子の絞りこみは実際には行われない．このようにして目的とする単位格子の集合すなわち配置可能領域が得られる．ところで，上記の⑤と⑧の処理を行うメソッドはそれぞれの配置制約のクラスに共通的に記述されており，インスタンスに宣言的に保持された情報を参照しながら，対をなす反対側の要素の配置が行われているかどうかによって，その処理内容が配置過程において確定するようになっている．さらに，他の制約関係を参照しつつ処理を進めていき，配置すべき単位格子が定まると，この要素とそれらの単位格子の間で 'laid-to' という関係が生成されて，配置関係が表現される．

このほか各種の制約を用いた処理も同様の方法により，状況に応じた方向に制約を展開しながら行われる．

4･5　各種インスタンスの初期設定　　最後に，各インスタンスを配置条件に従って，初期設定する方法について示す．本システムでは，図６に示すような簡便な形式で設計条件をファイルに表現しておき，配置設計を行うにあたり，その内容に応じて自動的に図３のようなインスタンスを生成して必要な情報を保持させるようにする．図中(a)は配置要素とその形状などを定義した部分であり，例えば①の行は「charge-pa という要素が存在し，その形状が 3×1×1 個の単位格子に相当する」ことを示している．(b)は配置空間

に関する条件を与えている部分であり，②の部分は設計対象のプラントのサイズを単位格子の縦・横・高さ方向の個数として定義している．(c)は配置制約の記述であり，例えば③の行は「spray-pb と safeaux-ps が隣接する」という制約を表している．なお，このような制約の記述形式として，基本となるものを22個用意し，さらにそれに対して拡張子を付加することにより，数十種類を用意した．このような機能によって，ユーザは図３に示したようなシステム内部での表現を意識する必要がなくなる．なお，基本配置の結果についても同様の形式でファイルに出力し，続いて行われる詳細配置設計過程との連携が行えるようにする．

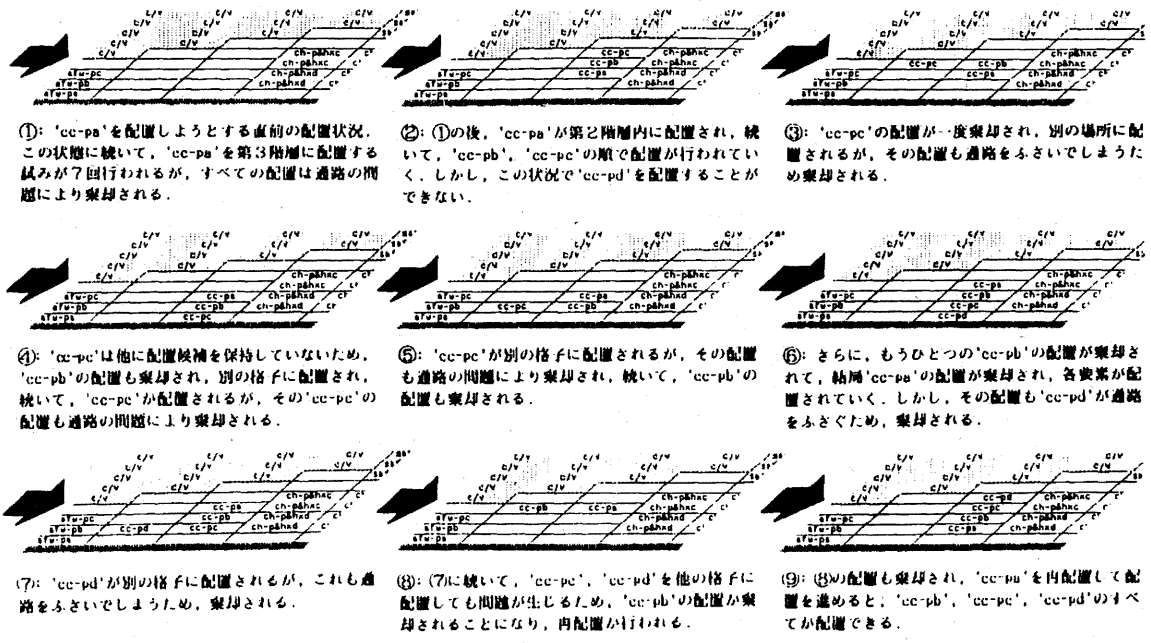
5. ユーザインターフェース

前章で示したようにして配置設計支援システムを構築することができるが，ユーザインターフェースは設計支援システムに欠かせない要素である．以下に本システムにおけるユーザインターフェースの機能を示す．

5･1　配置過程のトレース機能　一般に，設計支援システムにおいては，そのユーザが設計実務者であるため，設計処理の過程を示すことにより，解の背後に潜んだ個々の設計問題の性質を把握させ，その解の合理性を具体的に示すことが重要である．また，ここで扱っている配置設計の場合には，与えた配置制約に

```
>>> Is the current layout acceptable ?  : y
58                  CH-P&HXA's layout to ( COMP-2-1-2 ... ) is rejected. 100 c
58                  CH-P&HXA's layout to ( COMP-3-1-2 ... ) is rejected. 99 c
58                  CH-P&HXA is laid to ( COMP-4-2-2 ... ). 98 candidates are
59                   CH-P&HXB is laid to ( COMP-4-4-2 ... ). 1 candidate is r
60                    CH-P&HXC is laid to ( COMP-5-4-2 ... ).
61 CH-P&HXD is laid to ( COMP-5-2-2 ... ). 1 candidate is rest.
>>> Is the current layout acceptable ?  : y                          ------①
62  CC-PA's layout to COMP-5-3-3 is rejected. 94 candidates are rest.
62  CC-PA's layout to COMP-5-4-3 is rejected. 93 candidates are rest.
62  CC-PA's layout to COMP-5-2-3 is rejected. 92 candidates are rest.
62  CC-PA's layout to COMP-4-1-3 is rejected. 91 candidates are rest.
62  CC-PA's layout to COMP-5-1-3 is rejected. 90 candidates are rest.
62  CC-PA's layout to COMP-3-1-3 is rejected. 89 candidates are rest.
62  CC-PA's layout to COMP-2-1-3 is rejected. 88 candidates are rest.
62  CC-PA is laid to COMP-6-3-2. 87 candidates are rest.
63   CC-PB is laid to COMP-6-4-2. 2 candidates are rest.
64    CC-PC is laid to COMP-6-5-2. 1 candidate is rest.------------②
65     CC-PD cannot be laid.
64    CC-PC is removed.
64    CC-PC's layout to COMP-7-4-2 is rejected.------------------③
63   CC-PB is removed.
63   CC-PB is laid to COMP-6-2-2. 1 candidate is rest.
64    CC-PC's layout to COMP-6-1-2 is rejected. 1 candidate is rest.--④
64    CC-PC's layout to COMP-7-2-2 is rejected.------------------⑤
63   CC-PB is removed.
63   CC-PB's layout to COMP-7-3-2 is rejected.
61  CC-PA is removed.
62  CC-PA is laid to COMP-6-4-2. 86 candidates are rest.
63   CC-PB is laid to COMP-6-3-2. 2 candidates are rest.
64    CC-PC is laid to COMP-6-2-2. 1 candidate is rest.
65     CC-PD's layout to COMP-6-1-2 is rejected. 1 candidate is rest.--⑥
65     CC-PD's layout to COMP-7-2-2 is rejected.------------------⑦
64    CC-PC is removed.
64    CC-PC's layout to COMP-7-3-2 is rejected.
63   CC-PB is removed.
63   CC-PB is laid to COMP-6-5-2. 1 candidate is rest.
64    CC-PC cannot be laid.
63   CC-PB is removed.
63   CC-PB's layout to COMP-7-4-2 is rejected.------------------⑧
62  CC-PA is removed.
62  CC-PA's layout to COMP-5-1-2 is rejected. 85 candidates are rest.
62  CC-PA's layout to COMP-4-1-2 is rejected. 84 candidates are rest.
62  CC-PA's layout to COMP-3-1-2 is rejected. 83 candidates are rest.
62  CC-PA's layout to COMP-2-1-2 is rejected. 82 candidates are rest.
62  CC-PA's layout to COMP-7-1-2 is rejected. 81 candidates are rest.
62  CC-PA is laid to COMP-6-2-2. 80 candidates are rest.
63   CC-PB is laid to COMP-6-3-2. 2 candidates are rest.
64    CC-PC is laid to COMP-6-4-2. 1 candidate is rest.
65     CC-PD is laid to COMP-6-5-2. 1 candidate is rest.------------⑨
>>> Is the current layout acceptable ?  : y
66      SFP-HX is laid to ( COMP-7-15-2 ... ). 88 candidates are rest.
67       SFP-F is laid to COMP-7-14-2.
>>> Is the current layout acceptable ?  : y
```

（a）　文字レベルのトレース機能



①: 'cc-pa'を配置しようとする直前の配置状況．この状態に続いて，'cc-pa'を第3階層に配置する試みが7回行われるが，すべての配置は通路の問題により棄却される．

②: ①の後，'cc-pa'が第2階層内に配置され，続いて，'cc-pb'，'cc-pc'の順で配置が行われていく．しかし，この状況で'cc-pd'を配置することができない．

③: 'cc-pc'の配置が一度棄却され，別の場所に配置されるが，その配置も通路をふさいでしまうため棄却される．

④: 'cc-pc'は他に配置候補を保持していないため，'cc-pb'の配置も棄却され，別の格子に配置され，続いて，'cc-pc'が配置されるが，その'cc-pc'の配置も通路の問題により棄却される．

⑤: 'cc-pc'が別の格子に配置されるが，その配置も通路の問題により棄却され，続いて，'cc-pb'の配置も棄却される．

⑥: さらに，もうひとつの'cc-pb'の配置が棄却されて，結局'cc-pa'の配置が棄却され，各要素が配置されていく．しかし，その配置も'cc-pd'が通路をふさぐため，棄却される．

(7): 'cc-pd'が別の格子に配置されるが，これも通路をふさいでしまうため，棄却される．

⑧: (7)に続いて，'cc-pc'，'cc-pd'を他の格子に配置しても問題が生じるため，'cc-pb'の配置が棄却されることになり，再配置が行われる．

⑨: ⑧の配置も棄却され，'cc-pa'を再配置して配置を進めると，'cc-pb'，'cc-pc'，'cc-pd'のすべてが配置できる．

（b）　図形レベルのトレース機能（第２階層の一部領域の配置状況の時間的推移）

図７　配置結果のトレース機能

不備があったり，必要な制約が欠如している場合にも対処できるように，個々の配置処理の進行状況をトレースしてユーザに提示することは特に重要となる．本システムにおけるこのようなトレース機能の実例を6章で取り上げる事例をもとに図7に示す．(a)は文字レベルの機能で，(b)は図形レベルの機能であり，図中の丸付きの番号で示した部分が互いに対応している．前者は，配置処理内容を適当な段付を行って表示するものであり，各処理が時間的にどのようにして行われてきたかを容易に知ることができる．後者は，その時点での配置状況を逐次視覚的に表示していくものであり，一目で状況を把握することができる．このような両者の機能が補完し合うことにより，個々の処理内容を容易に理解できるようになる．

なお，後者の機能については，Lisp プロセスの側から，ウインドウシステム上でC言語により記述された図形処理プロセスを起動し，Lisp 側のオブジェクトにおけるメソッドの処理の中から両者の間で配置情報を送信して，逐次配置状況を表示していくことにより実現する．

5･2 ユーザによるマニュアル配置の機能　本システムによる配置は，前報の配置ポテンシャルの概念によって大域的な配置が考慮されている[1]が，場合によっては問題のある配置状況に陥ったり，また，ユーザ自身が，システムが提示する以外の配置解を得ようとすることもあり得る．このような場合に対応するためには，ユーザが配置処理に積極的に介入できるようにする機能が必要となる．本システムにおいては，前節で示したトレース機能により配置処理過程を容易に把握することができるため，ユーザにとって真に介入が必要な状況を知ることは比較的容易である．そこで，システムがある一連の要素群（前報[1]における要素のグループ）の配置を終了した時点で，介入が必要である場合には任意の要素の配置を棄却して強制的に探索におけるバックトラックを行わせることができるようにする．これによって，配置制約を満足しないような配置結果に陥ることを避けつつ，ユーザがシステムの処理に介入できるようになる．

6. 事例 ― 原子力発電所の配置設計への適用 ―

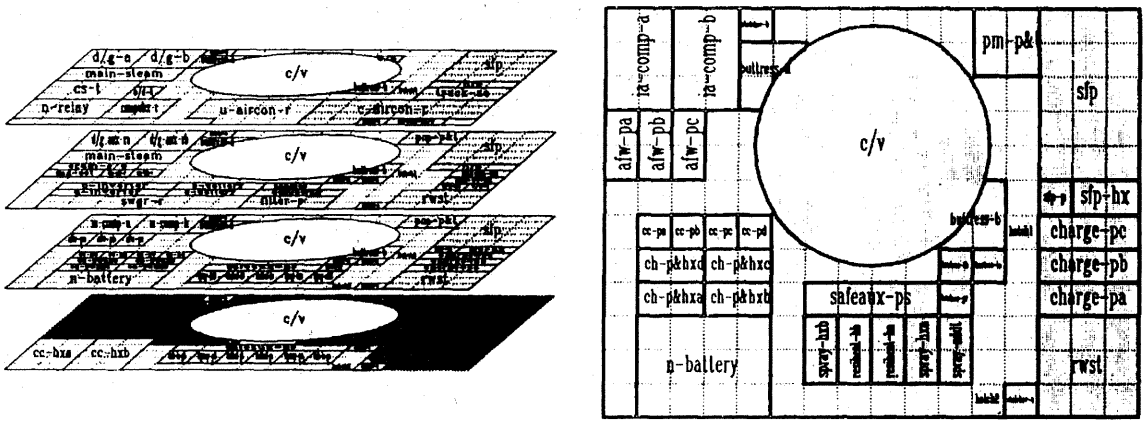
最後に，本システムを原子力発電所の基本配置設計に適用した事例を示す．この配置問題は，100 台前後の機器を3次元的に配置する問題であり，その設計においては経済性に加えて安全性や信頼性に対する要求が重要となる．しかるに，従来は熟練経験者の勘や経験によって設計が行われていたため，合理性に富む解を得るためには多大な設計時間を要していた．これに対して，本システムを適用することにより，有効に配置過程を支援しつつ，合理的な配置が容易に行えるようになる．以下にある発電所の配置設計を行った事例を示す．事例における設計対象の建屋は4階層から成り，その各階層は 12×16 の単位格子から構成されている．扱った配置要素には表1に示すようなものがあり，あらかじめその位置が決定されていた格納容器などの7個の要素を除いて計 73 個の要素の配置を行った．また，配置制約は熟練設計者による実設計の結果を参照するなどして作成し，およそ 200 個の制約を記述した．図8は配置結果を示したものであり，図中(a)は全体の透視図，(b)はある階層の平面図，(c)と(d)はそれぞれ東西・南北方向のある位置での断面図である．本結果を，制約条件の記述において参照した熟練設計者による配置結果と比較した結果，本システムによる配置が極めて満足すべきものであることを確認した．また，このほか，設計条件の変更に従って配置制約を変更した場合，それが配置結果に適切に反映されることや，型式の異なる他の発電所の設計事例においても本システムが有効であることを確認した．

表1　原子力発電所に配置される機器

機器／区域名称	区域区分	形状（単位格子の個数）	備考
格納容器	管理区域		既配置とする
充填ポンプA	管理区域	3×1×1	
充填ポンプB	管理区域	3×1×1	
充填ポンプC	管理区域	3×1×1	
ほう酸ポンプ	管理区域	1×1×1	
ほう酸タンクA	管理区域	1×1×2	
ほう酸タンクB	管理区域	1×1×2	
非再生冷却器	管理区域	2×1×1	
封水冷却器	管理区域	2×1×1	
・・・	・・	・・・	
・・・	・・	・・・	
燃料取替え用水ピット	管理区域	3×3×2	
トラックアクセス	管理区域	3×2×1	
・・・	・・	・・・	
・・・	・・	・・・	
主蒸気管室	非管理区域		既配置とする
復水タンク	非管理区域	3×3×1	
制御用空気圧縮器A	非管理区域	3×2×1	
制御用空気圧縮器B	非管理区域	3×2×1	
MGセット	非管理区域	2×1×1	
・・・	・・	・・・	
・・・	・・	・・・	

7. 結　言

本報では，前報で示したアルゴリズムに基づいて，基本配置設計に対する支援システムを構築した．本システムにおいては，配置設計の過程がその問題構成要素から形成されるネットワーク上の関係を扱っている



（a） 全体透視図　　（b） ある階層の平面図

（c） 東西方向の断面図　　（d） 南北方向の断面図

図8　原子力発電所の配置結果の一例

ものとの認識に基づいて，システム構築の具体的な手段としてオブジェクト指向プログラミングを用いて，上記のネットワークをインスタンスとそれらの間の関係によって表現し，また，汎用的な配置処理の機能をクラスのメソッドとして共通的に記述することにより，一般性と拡張性に優れたシステムとすることができた．さらに，各種のインターフェース機能を含めて，トータルなシステムを構築した．ここで取り扱った設計問題におけるネットワーク状の関係に基づいたオブジェクト指向による問題表現は各種の設計問題においても有効であり，設計支援システムの構築の一形態を与えるものである．

文献

(1) 赤木・藤田，本論文 第1報，機論 C (No. 89-995).
(2) Leler,W., Constraint Programming Languages – Their Specification and Generation, (1988), Addison-Wesley.
(3) 鈴木，(淵 監修)，知識プログラミング，(1988)，34，共立出版.
(4) 木村，情報処理，29-4 (1988)，368.
(5) Cox,B.J., Object-oriented Programming: An Evolutionary Approach, (1986), Addison-Wesley.
(6) Abelson,H. and Sussman,M.K., Structure and Interpretation of Computer Programs, (1985), 230, MIT Press.
(7) Winston,P.H. and Horn,B.K.P., LISP (3rd Edition), (1989), 335, Addison-Wesley.
(8) Steele,G.L.Jr., Common Lisp: The Language, (1984), Digital Press, (邦訳：後藤・井田，(1985)，共立出版).
(9) SunView 1 Programmer's Guide, (1988), Sun Microsystems.
(10) SunCore Reference Manual, (1988), Sun Microsystems.