IBDS: 建築物の統合化設計支援システム†

長澤 勲†† 手越義昭††† 牧野 稔††††

近年建築設計の分野では，パーソナルコンピュータ上に実現されたCADシステムが普及し，設計作業の合理化に貢献している．しかし，これらのCADシステムは，建築設計の幾つかの側面である設計図書の編集，構造計算，積算等の作業を個別的に支援しているのが現状であり，次のような問題点がある．建築設計では，設計が完了した後で発注者の要求や施工上の問題点を解決するために頻繁に再設計が行われる．設計条件が変化した場合の再設計は，従来から行われている手作業と，個別作業の支援のシステムを併用する方法では，図面の変更や構造計算の基礎データを設計者自身が修正しなければならず十分な効果を上げることが困難である．本研究では，設計の個別的な支援による問題点を解決するため，次の特徴をもつCADシステムを開発した．(1)建築物の設計情報を一元的に表現した建築物モデルを中心として，意匠設計，構造設計，積算などの設計作業を支援するモジュールを配置した．このことによって設計者は，設計データの変換にわずらわされることなく一貫して設計作業を進めることができる．(2)建築物モデルの一貫性を管理する建築物モデル管理機構を設け，システムの保守を容易にした．(3)設計者が行う標準的な設計作業の流れを想定し，これを一貫して支援できるように配慮した．実際の設計例に適用した結果，従来の方法に比べて設計期間を約 1/20 に短縮できた．

1. まえがき

近年建築設計の分野では，パーソナルコンピュータ上に実現されたCAD (Computer Aided Design) システム[1)~5)]が普及し，設計作業の合理化に貢献している．しかし，これらのCADシステムは，設計図や設計書（以下，図書）の編集，構造計算，積算等の作業を設計作業の流れに乗って進めるものではなく，流れの一部を取り出して個別的に支援するシステムであり，次のような問題がある．建築設計では，設計が完了した後で発注者の要求や施工上の問題点を解決するため，頻繁に再設計が行われる．設計条件が変化した場合の再設計において，従来から行われている手作業と個別作業の支援システムを併用する方法では，設計者自身が図書の変更や構造計算の基礎データを修正しなければならず，設計ミスの発生や設計変更に迅速に対応できないという問題がある．この問題の解決には，設計作業の流れ全体を一貫して支援する必要がある．このような試みは，住宅設計においては既に行われており，実用システムも開発されている[6)]．しかし，ビル設計を対象とした研究・開発は，その複雑性のためにまだ十分な成果を上げているとは言えない[7),8)]．本研究では，設計の個別的な支援による問題を解決するため，次の特徴をもつCADシステムIBDS (Integrated Building Design System) を開発した．(1)建築物の設計情報を一元的に表現した建築物モデルを中心として，意匠設計，構造設計，積算などの設計作業を支援するモジュールを配置した．このことによって設計者は，設計データの変換にわずらわされることなく一貫して設計作業を進めることができる．(2)建築物モデルの一貫性を管理する建築物モデル管理機構を設け，システムの保守を容易にした．(3)設計者が行う標準的な設計作業の流れを想定し，これを一貫して支援できるように配慮した．

2. 建築物実設計の現状と研究のねらい

ここでは，建築物の概要を決める設計作業を基本設計，建築物を建設するための詳細な設計作業を実設計とよぶことにする．本研究は，後者を対象とする．現状の建築物の実設計は，異なる専門の設計者によって，大きく意匠設計，構造設計，設備設計，積算の4つに分業されている．意匠設計とは建築物の規模，用途，間取り，仕上げ等を決める設計作業，構造設計とは建築物の柱，梁，床，耐震壁等の配置や断面寸法等の，いわゆる，骨組を決める設計作業，設備設計とは給排水，空調，電気設備等の設備を設計する作業，積算とは意匠設計，構造設計，設備設計等で使われている部材の価格を基に建築物全体の価格を見積る作業で

† IBDS: An Integrated Building Design System by ISAO NAGASAWA (Department of Computer Science and System Engineering, Faculty of Computer Science and Mechanical System Engineering, Kyushu Institute of Technology), YOSHIAKI TEGOSHI (Department of Architecture, Faculty of Engineering, Hiroshima Institute of Technology) and MINORU MAKINO (Department of Architecture, Faculty of Engineering, Kyushu University).
†† 九州工業大学情報工学部機械システム工学科
††† 広島工業大学工学部建築学科
†††† 九州大学工学部建築学科

ある．次に，本研究で取り扱う実設計の支援について述べる．意匠設計者は設計作業を進めるに当り，構造設計者や設備設計者の援助が必要である．例えば，意匠設計を進める場合，建築物の構造体の材種や寸法，あるいは，空調設備の機種，給排水設備の配管スペースや高架水槽の位置がある程度決まっていなくてはならない．このために意匠設計者は，構造設計者や設計設備者等の，異なる専門の設計者と打ち合せながら共同で作業を進めねばならない．ところが現状では，分業化を徹底させ流れ作業で設計することが行われているため，互いに十分な意志の疎通ができないことや，各設計者が必要とした時点でただちに専門外の判断を求めることが困難である．このため，設計者はある過去の経験値から見込みで設計を進めることが多い．見込み違いが起これば設計変更が必要となるので余裕の多い設計から収斂させて行くことが行われている．設計変更や再設計を行うには，時間がかかるとともに，問題解決のためには複数の分野の設計者の協力が必要となり，さまざまな図書間の整合性の管理に手間取る．このことは，高品質の建築物を低コストに設計できない原因である．このような困難を解決するには，設計作業の流れを一貫して支援し，設計変更を容易にすることが必要である．設計変更が容易になれば，それを前提とした設計過程が可能となり，設計の可能性が拡大する．

設計作業の流れをCADシステムを用いて一貫して支援するには，設計の各段階における設計情報を建築物モデルに一元化すること，建築物モデルを修正の容易な柔軟な構造にすること，および，設計過程を標準化することが必要である．建築物の設計情報の一元化は，建築物モデル管理機構を用いて建築物モデルの編集，一貫性管理，構造設計モデルへの変換等を行うことにより，また，設計過程の標準化は，建築物モデルの中で使用される設計データの流れを管理することにより行った．この結果，設計変更に際して設計案を容易に修正できる機能，図書間の整合性を管理する機能，複数の分野にまたがる設計上の問題点を指摘する機能を実現できた．

3. 設計方針

本システムの設計方針は次のとおりである．

（1） 意匠設計の支援

意匠設計者は設計作業を進めるに際して，仕上げ材，構造材等の規格品のカタログを検索し，建築物の用途，設計条件，法規や施主の好みなどを考慮しながら，部材の種類，色，形等の意匠を決定する．このとき設計者は，平面図，立面図等，目的に応じてさまざまな視点から捉えた意匠図を用いる．この部材のカタログを検索する作業や，図書間の整合性を保つ作業は，設計者にとり大きな負担である．したがって，本システムでは，意匠図面の編集機能と部材等のカタログ検索機能を有機的に結合すること，さまざまな図書間の整合性を管理すること，および床面積や窓開口率を自動計算することを支援の対象とする．

（2） 構造設計の自動化

建築物の設計では，意匠の概要（建設地域，用途，規模，構造種別）が与えられると構造の概要（柱，梁の配置と断面の仮定）が設計される．次に，意匠設計者は，この結果を基に意匠の詳細を設計する．さらに，この結果を基に構造の詳細が設計（構造計算）される．構造の詳細設計の結果が不適当であれば，意匠設計者は構造に影響を及ぼす意匠の詳細を変更する．経済的，力学的に高品質の建築物を得るためには，この過程を反復しなければならないが，現状では設計コストがかさむことになる．この問題に対して，本システムでは構造の概要設計および詳細設計を自動化する．

（3） 積算の自動化

意匠設計者は，部材の単位当りの価格を参考にして，意匠の詳細設計を行う．意匠の詳細設計が終了すると詳細な積算を行う．この結果と実行予算との間に差があれば，まず意匠の詳細設計のうち価格に大きな影響を与える仕上げを変更する．それでもまだ差があれば構造部材，設備機器の容量等の見直しを行い，可能な変更を試みる．このような過程を反復することによって実行予算に近い仕様の建築物が設計される．ところが，現状ではこの反復作業は，設計コストの上昇を招いている．意匠設計者の選択した部材が，建築物全体の価格に及ぼす影響を，把握することは一般に困難である．したがって，本システムではこの積算作業を自動化する．

（4） 設計支援の範囲

本システムの対象とする建築物は，一般建築物の大半を占める15階建て，高さ60メートル以下の一般的な用途のラーメン構造建築物とする．構造種別は鉄筋コンクリート造（RC造），鉄骨造（S造）および鉄骨鉄筋コンクリート造（SRC造）とする．

（5） 操作性の良い安価なシステムとして実現

上記の建築物の設計は，小規模な設計事務所で行われる場合が多い．したがって，各設計者が占有できる安価なパソコン上に実現することを目標とする．

4. システムの概要

システムは，図1に示すように2つのデータベースと7つのプログラムモジュール（以下，モジュール）からなる．部材データベース①は，建物を構成する基本要素を格納したものである．この部材データベースの管理・編集を行うモジュールが，部材編集モジュール②である．建築物モデルデータベース③は，設計中の建築物モデルを格納する．④～⑦のモジュールは，意匠，構造等の設計者が行う個別の設計作業に対応した設計支援を行う．意匠編集モジュール④は建築物モデルの意匠の編集，構造設計モジュール⑤は建築物モデルの構造の検討，積算モジュール⑥は建築物の価格算出をそれぞれ支援する．その他，必要に応じて専用のモジュール⑦が追加できる．建築物モデル管理機構⑧は，建築物モデルの一貫性を管理するとともに，設計作業ごとに必要な情報を建築物モデルから取り出し操作するモジュールである．ユーザインタフェース⑨は利用者とシステムの接点である．このモジュールには，個々の設計作業において設計者が慣用的に用いている図書を通して設計情報を表示・編集できる工夫を行っている．

5. 建築物モデル

建築物モデルには，次の点に配慮しなくてはならない．

（1） 設計情報の一元管理

建築物の設計は，表1に示すように，意匠，構造等の設計作業に適した図書を用いて行われているが，これらの図面は相互に関連し合って建物を表現している．建築物モデルは，これらの図書の内容を整合性を保って管理し，設計作業ごとに必要な情報を提供できるものでなくてはならない．

（2） 再設計に対する修正の容易性

建築物の設計は，使い勝手，経済性，耐震性，施工の容易性など多面的な設計要求の調和を目的とした再設計の繰り返しである．このため建築物モデルでは，再設計に対して修正の容易な柔軟な構造でなくてはならない．

（3） カスタマイズの容易性

建築物に使用する部材や基準の取り方は，企業や設計グループごとに異なっている．このため建築物モデルは利用者の要求に合わせて容易にカスタマイズできなくてはならない．

従来のシステムでは，建築物の設計情報がファイルやデータベースとプログラムの形で取り扱われることが多く，これらの点について満足な解決を与えているとは言えない．本研究では，(1)建築物や建築物を構成する部材を対象指向言語のオブジェクトの概念を用いて表し，設計に必要な全情報を一元化する，(2)建築物モデルの編集，すなわち部材の配置，削除，属性の変更に際して，従属する部材の削除や属性の変更を建築物モデル自身に自動的に行わせる，(3)部材の配置や属性の計算を行わせるプログラムを部材の定義の形で宣言的に記述できるようにする，方針をとった．

5.1 建築物モデルの表現

建築物モデルを構成する基本要素をここでは部材とよぶ．これらの部材には他の部材との接続関係（2項関係として扱う）および各種の属性が与えられる（図2，図3参照）．

部材は，インスタンスとクラスによって表現する．インスタンスとは建築物を構成する個別部材の表現，

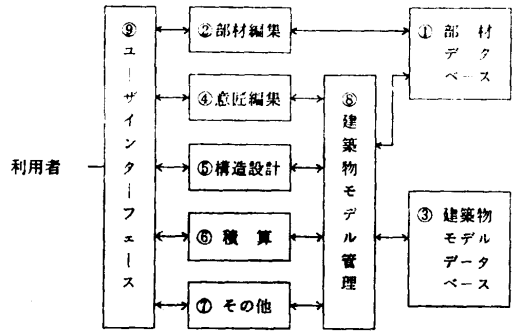


図1 システムの概要
Fig. 1 An overview of the system.

表1 建築図面の種類
Table 1 Kinds of architecture drawings.

意匠図	構造図	設備図
配置図	各階伏せ図	給排水設備図
各階平面図	基礎伏せ図	電気設備図
立面図	軸組図	空調設備図
断面図	架構図	仕様書
各種リスト	各種リスト	
各種詳細図	仕様書	
矩計図		
仕上げ表		
仕様書		

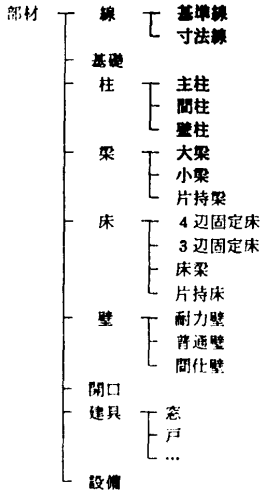


図 2 部材の分類
Fig. 2 Classification of building parts.

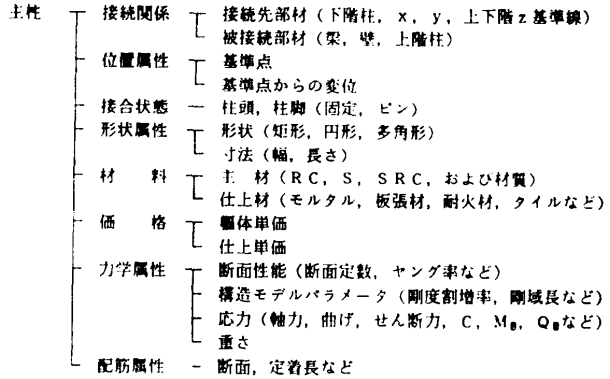


図 3 部材の接続関係と属性（主柱）
Fig. 3 Connecting relations and attributes of a building part (column).

instance i	class C
c : c	super C_s
p : $p_1, \cdots, p_m$	plugs $P_1, \cdots, P_m$
s : $s_1, \cdots, s_n$	sockets $S_1, \cdots, S_n$
g : $g_1, \cdots, g_u$	givens $G_1, \cdots, G_u$
d : $d_1, \cdots, d_v$	derives $D_1, \cdots, D_v$
	methods $M_1, \cdots, M_r$
	views $V_1, \cdots, V_t$
(a) インスタンス	(b) クラス

図 4 部材の記述形式
Fig. 4 Syntax of class and instance representing building part.

クラスとはインスタンスの性質を代表する表現である．

インスタンス 部材のインスタンスは，図 4 (a) の形式で与える．ここで i をインスタンス名，c を所属クラス名，$p_1, \cdots, p_m$ をプラグ，$s_1, \cdots, s_n$ をソケット，$g_1, \cdots, g_u$ を独立属性，$d_1, \cdots, d_v$ を従属属性とよぶ．

インスタンス名 i は指定部材の識別名，所属クラス名は c は指定部材の属するクラスの名称である．

プラグおよびソケットは指定部材の接続相手のインスタンス名を格納する．プラグ p_k $(k=1, \cdots, m)$ は，(V_p, I_s) のように与える．ここで V_p はプラグを表す変数，I_s は指定部材の接続先のインスタンス名である．ソケット s_k $(k=1, \cdots, n)$ は，(V_s, I_p) のように与える．ここで V_s はソケットを表す変数，I_p は指定部材に接続された部材のインスタンス名のリストである．

独立属性 g_k $(k=1, \cdots, u)$ は，指定部材の与件として与えられる属性を表し，(V_g, A_g) のように与える．ここで V_g は独立属性を表す変数，A_g は属性値である．従属属性は他の属性から一意に決定される属性を表し，(V_d, A_d) のように与える．ここで V_d は従属属性を表す変数，A_d は属性値である．

クラス 部材のクラスは，図 4 (b) の形式で与える．ここで C クラス名，C_s を上位クラス名，$P_1, \cdots, P_m$ をプラグ定義，$S_1, \cdots, S_n$ をソケット定義，$G_1, \cdots, G_u$ を独立属性定義，$D_1, \cdots, D_v$ を従属属性定義，$M_1, \cdots, M_r$ を操作定義，$V_1, \cdots, V_t$ をビュー定義とよぶ．

クラス名 C はクラスに与える一意な名称，上位クラス名 C_s は上位のクラス名称である．プラグ定義およびソケット定義は個別部材間の接続関係を代表する．

プラグ定義 P_k $(k=1, \cdots, m)$ は (V_p, C_s, V_s) のように与える．ここで V_p はプラグを表す変数，C_s および V_s は接続先の部材のクラス名とソケットを表す変数である．ソケット定義 S_k $(k=1, \cdots, n)$ は，(V_s, C_p) のように与える．ここで V_s はソケットを表す変数，C_p は接続される部材の属するクラス名である．

独立属性定義 G_k $(k=1, \cdots, u)$ は部材の独立属性を代表し，(V_g, C_g, D) のように与える．ここで V_g は独立属性を表す変数，C_g は属性値のクラス名，D はデフォルト値である．従属属性定義 D_k $(k=1, \cdots, v)$ は，部材の従属属性を代表し，(V_d, C_d, F_d) のように与える．ここで V_d は従属属性を表す変数，C_d は属性値のクラス名，F_d は属性値を計算する関数である．

操作定義 M_k $(k=1, \cdots, r)$ は部材の操作を定義し，

$$m(a_1, \cdots, a_p) : B_1, \cdots, B_q$$

のように与える．ここで m $(a_1, \cdots, a_p)$ を頭部，$B_1, \cdots, B_q$ を本体とよぶ．頭部の m は操作名，$a_1, \cdots, a_p$ は引数，本体の $B_1, \cdots, B_q$ は式である．式とは，定数（数値，クラス名，インスタンス名），変数（プラグ，ソケット，独立属性および従属属性を表す変数，局所

変数)，関数および後述のメッセージ式である．

操作定義はメッセージ式 [X $m(t_1, \cdots, t_p)$] によって使用される．ここでXはクラスあるいはインスタンス名，m は操作名，$t_1, \cdots, t_p$ は式である．メッセージ式の評価はXの属するクラスに定義された操作定義によって行われ，評価値として定義本体の B_q の値が返却される．

ビュー定義 V_k $(k=1, \cdots, t)$ は，設計作業ごとに使用する操作集合を定義し，v：$m_1, \cdots, m_w$ のように与える．ここでvはビュー名，$m_1, \cdots, m_w$ は操作名である．

5.2 建築物モデルの機能

建築物モデルの機能は，部材データベース，建築物モデルデータベースおよび建築物モデル管理機構の組合せによって実現される．部材データベースは部材のクラスを，建築物モデルデータベースは設計中の建築物モデルを格納する．建築物モデル管理機構は2つのデータベースを使用し，建築物モデルの一貫性の管理や各種の設計作業に必要な情報の提供を行う（表2参照）．

ここでは，建築物モデルの基本的な概念である部材の配置と従属性について述べる．これは部材は別の部材に依存して配置（接続）され，その属性の一部は別の属性から一意に決定されること（属性従属性），またある部材が削除されるとき，その部材に接続された部材も同時に削除されること（存在従属性），を意味する．

部材の接続 部材の接続関係は，クラスのプラグおよびソケット定義とインスタンスのプラグおよびソケットによって表す．今，クラス C_b に属する任意のインスタンスがクラス C_a に属する任意のインスタンスに接続できるものとすれば，C_a のソケット定義を $(V_s\ C_b)$，C_b のプラグ定義を $(V_p\ C_a\ V_s)$ のように与えれば良い．C_a のインスタンス I_a に C_b の2つのインスタンス I_{b1}, I_{b2} が接続された場合，I_a のソケットは $(V_s\ [I_{b1}, I_{b2}])$，I_{b1}, I_{b2} のプラグはいずれも $(V_p\ I_a)$ ように設定される．ここで $[I_{b1}, I_{b2}]$ は，I_{b1}, I_{b2} からなるリストである．

部材の生成 クラスCが与えられたとき，Cのインスタンスは，メッセージ式 [C new($I_1, \cdots, I_m$)] を評価することによって生成される．ここで $I_1, \cdots, I_m$ は生成されたインスタンスの接続先のインスタンスであり，Cのプラグ定義 $P_1, \cdots, P_m$ に対応している．生成されたインスタンスのプラグおよび，インスタンス $I_1, \cdots, I_m$ のソケットは前述のように設定される．また，生成されたインスタンスの独立属性はデフォルト値，従属属性は nil（未定義を示す）である．new 操作は，クラスCに対して**図5**(a)のように定義される．同図において，①はクラスCの定義から変数値が未定義なインスタンスを作成 (make) し，局所変数 I_{new} に代入する，②は I_{new} と各インスタンス $I_1, \cdots, I_m$ 間に接続関係を定義 (plugin) する，③は I_{new} をインスタンスの集合からなる建築物モデル M_{odel} に挿入

```
new(I1, ···, Im):
    Inew←make(C),                          ···①
   [Inew plugin(I1, ···, Im)],             ···②
    Model←cons(Inew, Model),               ···③
    forall(I, Model, [I reflesh]),         ···④
    Inew                                   ···⑤
```

(a) インスタンスの生成

```
delete( ):
  [Self kill],                             ···⑥
  forall(I, Model, [I set(Flag, nil)]),    ···⑦
  forall(I, Model, [I clear]),             ···⑧
  forall(I, Model, [I reflesh])            ···⑨
kill( ):
  [Self plugout],                          ···⑩
  Model←omit(Self, Model),                 ···⑪
  [Self set(Flag, die)]                    ···⑫
clear( ):
  case(Flag→Flag,                          ···⑬
    forsome(Vp, [Self plugs], [Vp clear]=die),  ···⑭
        → [Self kill],
    true → [Self set(Flag, live)])         ···⑮
```

(b) インスタンスの削除

図5 部材の生成と削除
Fig. 5 Creating and deleting building parts.

表2 建築物モデルの機能
Table 2 Functions of building models.

[C new ($I_1, \cdots, I_n$)]	クラスCの部材を生成し，部材 $I_1, \cdots, I_n$ に接続する．
[M search (C, C_d)]	モデルMから条件 C_d を満たすクラスCの部材を検索する．
[M pickup (C, V_w, X, Y)]	ビュー V_w で座標値 X, Y にあるクラスCの部材を選択する．
[I delete]	部材Iを削除する．
[I V]	部材Iのプラグ／ソケット変数Vで指定した部材を参照する．
[I V]	部材Iの属性変数Vで指定した属性値を参照する．
[I change_V (A)]	部材Iの属性変数Vで指定した属性値をAに変更する．

(cons) する，④は M_{odel} の各インスタンスIについてその従属属性を再計算 (reflesh) する，⑤は生成されたインスタンスをメッセージ式の値として返却する，である．

部材の削除 インスタンスIを建築物モデルから削除するには，メッセージ式 [I delete] を評価する．delete 操作は，インスタンスIに直接あるいは間接に接続されているインスタンスも同時に削除する．delete 操作は図5(b)のように定義される．まず，全インスタンスにそれが削除されているかどうかを示す標識 Flag を与え，使用中，削除済，未検査に対してそれぞれ live, die, nil とする．delete 操作の意味は，⑥インスタンス Self を削除 (kill) する，⑦建築物モデルの全インスタンスの Flag を nil に設定 (set) する，⑧全インスタンスに対してそれが削除されるべきものであるかどうかを検査し，もしそうであれば削除 (clear) する，⑨全インスタンスの従属属性を再計算 (reflesh) する，である．kill 操作の意味は，⑩接続先との接続関係を消去 (plugout) する，⑪建築物モデルから除去 (omit) する，⑫Flag を die に設定する，である．また，clear 操作の意味は，⑬Flag の値が live または die であれば，そのまま値として返却する，⑭接続先の全インスタンス (plugs) を検査 (clear) し，いずれかが die であれば，インスタンス Self を削除 (kill) する，⑮いずれでもなければ，Flag を live に設定し値として返却する，である．

独立属性の操作 インスタンスIが与えられたとき，その独立属性 V_θ を参照するには，メッセージ式 [I V_θ] を評価する．また，独立属性 V_θ を属性値 A_θ に変更するにはメッセージ式 [I change_V_θ(A_θ)] を評価する．操作の定義を図6(a)に示す．参照操作の意味は①変数値を参照する，である．変更操作の意味は②属性値 A_θ を変数 V_θ に設定し，③建築物モデルの各インスタンスIについて，その従属属性を再計算 (reflesh) する，である．

従属属性の操作 インスタンスIが与えられたとき，その従属属性 V_δ を参照するには，メッセージ式 [I V_δ] を評価する．また，建築物モデルの変更に際して従属属性を再計算するには，メッセージ式 [I reflesh] を評価する．図6(b)に操作の定義を示す．参照操作の意味は，④もし変数値が，既に計算されていればそれを値として返却する，⑤もし変数値が nil であれば従属属性定義に与えられた関数 F_δ を用いて変数値を計算し，値として返却する，である．再計算

```
Vθ( ): Vθ                                          …①
change_Vθ(Aθ):
    [Self set(Vθ, Aθ)],                            …②
    forall(I, Model, [I reflesh])                  …③
```

(a) 独立属性の参照と変更

```
Vδ( ):
    case(Vδ→Vδ,                                    …④
        true→[Self set(Vδ, Fδ)])                   …⑤
reflesh( ):
    forall(Vδ, [Self derives], [Self set(Vδ, nil)]) …⑥
```

(b) 従属属性の参照と初期化

図6 部材属性の操作

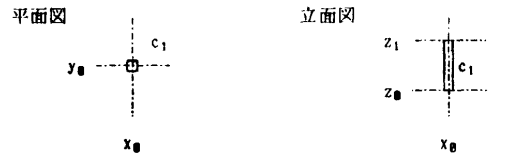
Fig. 6 Changing and computing the attributes of building parts.

操作の意味は，⑥すべての従属属性 (derives) の値を nil (未定義) に設定する，である．

5.3 建築物モデルの例

ここでは**図7**に示す簡単な例を用いて建築物モデルの機能を説明する．図7(a)の建築物モデルは，x基準線 x_0, y 基準線 y_0, z 基準線 z_0, z_1, および主柱 c_1 からなり，基準線 z_1 は基準線 z_0 に，主柱 c_1 は基準線 x_0, y_0, z_0, z_1 にそれぞれ接続 (依存して配置) されている．また，基準線 x_0, y_0, z_0 は他の部材には接続されていない．

図7(b)はz基準線のクラス (x, y 基準線のクラスもほぼ同じ) である．図中①～③は基準線のプラグとソケット定義を示す．Pred は先行 (接続先) 基準線を，Succ は後続 (被接続) 基準線を，Colm は被接続主柱のリストを表す．④～⑥は基準線の属性を示す．Span は先行基準線からの距離を，No は基準線に割り当てた一連番号を，Z は基準線の z 座標を表す．Span の値は change 操作によって変更できる．また，No および Z は従属属性であり，属性値の参照に際して次のように計算される．No の計算⑤は，もし先行する基準線がなければ0を，そうでなければ先行基準線の一連番号＋1を値として返却する．Z の計算⑥は，もし先行する基準線がなければ Span を，そうでなければ先行基準線の Z 座標＋Span を値として返却する．基準線 z_0 は前者の場合に，z_1 は後者の場合に対応する．⑦～⑨は基準線の操作定義を示す．fig_xz は立面図に表示する形状 (水平な破線) を，visible_xy は平面図からの可視性 (非表示を示す nil) を，visible_xz は立面図からの可視性 (表示を示す true) を値として返却する．⑩～⑫はビュー定義を示す．xy_plane は平面図の編集作業，xz_plane は立面図の編集作業，demo は例題のためのものである．xy_plane では，



(a) 部材の配置例

```
class  z基準線
  super 基準線
  plugs   (Pred  z基準線 Succ)                          …①
  sockets (Succ  z基準線),                              …②
          (Colm 主柱)                                   …③
  givens  (Span  整数 700)                              …④
  derives (No    整数 case(null(Pred)→0,                …⑤
                          true→[Pred No]+1)),
          (Z     整数 case(null(Pred)→Span,             …⑥
                          true→[Pred Z]+Span))
  methods  fig_xz() : line(Z, 50, Z, 10000)             …⑦
           visible_xy(Floor) : nil                      …⑧
           visible_xz(Frame) : true                     …⑨
  views    xy_plane : visible_xy                        …⑩
           xz_plane : visible_xz, new, delete,          …⑪
                      change_Span, fig_xz
           demo :     new                               …⑫
```

(b) z基準線のクラス

```
class  主柱
  super 柱
  plugs   (Xline  x基準線 Colm),                        …①
          (Yline  y基準線 Colm),                        …②
          (Zfline  z基準線 Colm),                       …③
          (Zhline  z基準線 Colm)                        …④
  givens  (D     整数 60),                              …⑤
          (Dx    整数  0),                              …⑥
          (Dy    整数  0)                               …⑦
  derives (Flno  整数 [Zhline No]),                     …⑧
          (Frno  整数 [Zline No]),                      …⑨
          (X     整数 [Xline X]+Dx),                    …⑩
          (Y     整数 [Yline Y]+Dy),                    …⑪
          (Zf    整数 [Zfline Z]),                      …⑫
          (Zh    整数 [Zhline Z])                       …⑬
  methods  fig_xy() : rectangle(X−D/2, Y−D/2,           …⑭
                      X+D/2, Y+D/2)
           fig_xz() : rectangle(X−D/2, Zf, X+D/2, Zh)
                                                        …⑮
           visible_xy(Floor) : equal(Floor, Flno)       …⑯
           visible_xz(Frame) : equal(Frame, Frno)       …⑰
  views    xy_plane : visible_xy, new, delete,          …⑱
                      change_D, change_Dx,
                      change_Dy, fig_xy
           xz_plane : visible_xz, fig_xz                …⑲
           demo :     new, change_D                     …⑳
```

(c) 主柱のクラス

```
class  xy_plane
  super window
  vars  (Floor 表示階)
  methods new(Flr) :                                    …①
              [Self set(Floor, Flr)],
              [Self reflesh]
          reflesh() :                                   …②
              forall(I, Model,
                case([I visible_xy(Floor)]
                     → [Self draw([I fig_xy])]))
```

(d) 平面図のクラスの一部

```
class demo
  super window
  methods test() : x0←[xline new(nil)],                 …①
                   y0←[yline new(nil)],                 …②
                   z0←[zline new(nil)],                 …③
                   z1←[zline new(z0)],                  …④
                   c1←[colm new(x0, y0, z0, z1)],       …⑤
                   [z1 change_Span(400)],               …⑥
                   xy1←[xy_plane new(1)],               …⑦
                   xz1←[xz_plane new(0)]                …⑧
```

(e) 操作例のクラス

```
instance x0
  c : x基準線
  p : (Pred nil)
  s : (Succ nil),
      (Colm [c1])
  g : (Span 100)
  d : (No    0),
      (Z   100)
instance y0
  c : y基準線
  p : (Pred nil)
  s : (Succ nil),
      (Colm [c1])
  g : (Span 100)
  d : (No    0),
      (Z   100)
instance z0
  c : z基準線
  p : (Pred nil)
  s : (Succ z1),
      (Colm [c1])
  g : (Span 100)
  d : (No    0),
      (Z   100)
instance c1
  c : 主柱
  p : (Xline  x0),
      (Yline  y0),
      (Zfline z0),
      (Zhline z1)
  g : (D     60),
      (Dx     0),
      (Dy     0)
  d : (Flno   1),
      (Frno   0),
      (X    100),
      (Y    100),
      (Zf   100),
      (Zh   500)
instance z1
  c : z基準線
  p : (Pred z0)
  s : (Succ nil),
      (Colm [c1])
  g : (Span 400)
  d : (No    1),
      (Z   500)
```

(f) インスタンス

図 7 建築物モデルの例
Fig. 7 A building model.

z基準線は表示も編集もされないので visible_xy だけが必要である．xz_plane では，z 基準線の生成 (new)，削除 (delete)，移動 (change_Span)，表示 (fig_xz) が行われる．demo では，z 基準線の生成が行われる．

図7(c)は主柱のクラスである．図中①～④は主柱のプラグ定義を示す．Xline, Yline, Zhline, Zfline はそれぞれ接続先の x 基準線，y 基準線，上階 z 基準

線，下階 z 基準線を表す．⑤〜⑬は主柱の属性を示す．独立属性として柱幅(D)，基準線からの x 方向変位 (Dx)，y 方向変位 (Dy)，従属属性として階番号 (Flno)，通番号 (Frno)，柱心 x 座標(X)，柱心 y 座標 (Y)，柱頭 z 座標 (Zh)，柱脚 z 座標 (Zf) を与えている．従属属性の値は，独立属性と基準線の属性から計算される．⑭〜⑰は操作定義を示す．fig_xy は平面図に表示する形状 (方形) を，fig_xz は立面図に表示する形状 (矩形) を，visible_xy は平面図からの可視性を，visible_xz は立面図からの可視性を返却する．⑱〜⑳はビュー定義を示す．xy_plane では，主柱の生成 (new)，削除 (delete)，柱幅変更 (change_D)，x 変位変更 (change_Dx)，y 変位変更 (change_Dy)，表示 (fig_xy) が，xz_plane では表示 (fig_xz) が，demo では生成 (new) と柱幅変更 (change_D) が行われる．

建築物モデルを操作するプログラムの一例を図 7 (d)，(e)に示す．同図(d)は平面図を編集する操作をまとめたクラスである．操作の意味は，①平面図を生成 (new) する，②平面図を再生 (reflesh) する，である．また，同図(e)は例題のためのクラスである．操作の意味は，①x 基準線 x_0 を生成する，②y 基準線 y_0 を生成する，③z 基準線 z_0 を生成する，④z 基準線 z_1 を生成して z_0 に接続する，⑤主柱 c_1 を生成して x_0, y_0, z_0, z_1 に接続する，⑥z 基準線 z_1 のスパンを 400 に変更する，⑦平面図 xy_1 を生成し，全部材を表示する，⑧立面図 xz_1 を生成し，全部材を表示する，である．図 7 (a)に示した建築物モデルは，メッセージ式 $demo_1$←[demo new], [$demo_1$ test] を順に評価することにより生成される．生成後のインスタンスの状態を図 7 (f)に示す．

6. 建築物の設計過程

6.1 設計作業の流れ

本システムを使用して行う標準的な設計作業の流れを図 8 に示す．図中破線は設計者がシステムを操作して行う作業，実線はシステムが自動的に行う作業である．①設計者は，設計の基本条件である建物所在地・建物用途・建物規模・スパン長・階高等の建物概要をシステムに入力する．②システムは，自動的に概略の構造計算を行い，柱・梁の構造材を配置する．③設計者は，配置された柱断面を考慮しながら，意匠設計を進める．つまり，意匠編集モジュールを用いて，意匠平面図上で柱・壁などの部材を配置する．④この部材

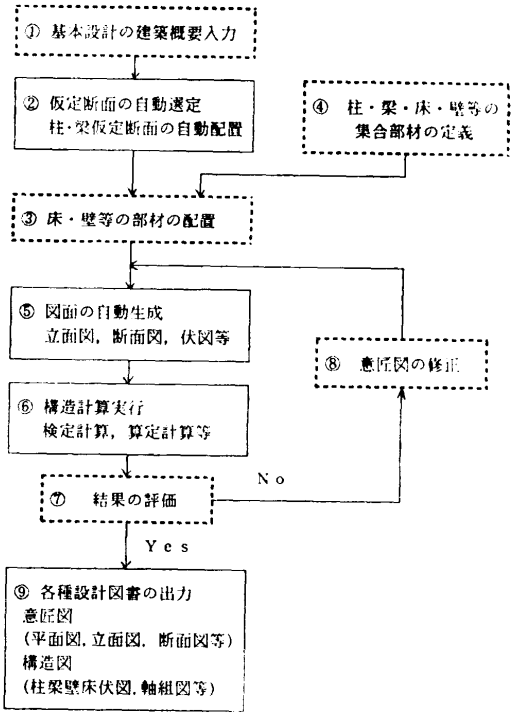


図 8 本システムを使った設計作業の流れ
Fig. 8 Design procedure by the system.

は③に先だって定義しておく．⑤システムは立面図，断面図等を自動生成する．⑥システムは意匠設計の結果を用いて，構造設計を行い伏図，軸組図等の構造設計図を自動生成する．⑦設計者は構造設計の結果を評価する．もし，意匠と構造の取り合いに不具合があれば，⑧設計者は意匠図の修正を行い，⑤，⑥を再度試みる．もし不具合がなければ設計を終了する．⑨システムは，意匠図，構造図，構造計算書等の設計結果を出力する．

6.2 意匠設計の支援

(1) 部材データベース

部材データベースには，素材と部材が格納される．素材とは建築物を構成する構造材や仕上材，部材とは複数の素材を組み合わせて建築物を構成する基本要素としてまとめた柱・梁・壁等である．素材は JIS 規格や各メーカの規格品情報を多量に収集して作成する．部材は建築物ごとの取り決めや各社の標準仕様を対象としており，素材に比べて少量である．これらはシステムの利用者が自由に定義できるように配慮されている．

(2) 意匠図の編集

意匠の編集には平面図，立面図，断面図を用いる．設計者は図 9 に示すように，部材データベースから部

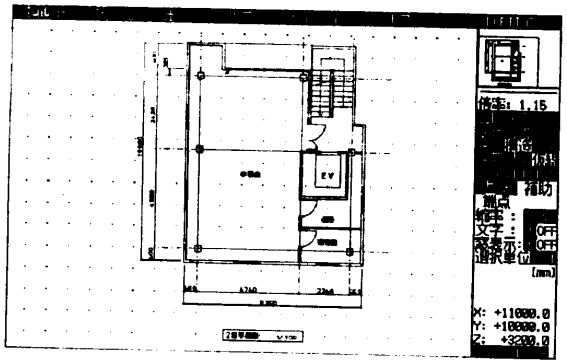


図 9 意匠図編集画面
Fig. 9 Screen for editing design diagram.

材をマウスで選択し，意匠平面図上に配置する．このとき，システムは一貫性を保ちながら建築物モデルを操作しているが，設計者からは見かけ上，平面図を描いているように見える．設計者が意匠図の柱寸法を変更した場合，これに依存する意匠図と構造図の柱や梁の寸法は，建築物モデルの機能より，自動的に変更される．同様に，各階の意匠平面図を作成することにより，立面図や任意の位置での断面図を自動的に作成できる．設計者はこの立面図と断面図を用いて，立面方向の取り合いを確認する．

（3） 利用者インタフェース

設計支援システムの利用者インタフェースは，設計者の思考を妨げないように，設計者が慣れ親しんでいる設計図を使用し，違和感なく操作できる必要がある．このため，意匠設計者が通常の業務で使用するのに適したウインドウシステムを整備した．本システムでは，座標値の選択やコマンドの入力等は，すべてマウスだけで行うことができ，また必要に応じて全操作をコマンドで行うこともできる．画面上のカーソルの色や形状は，利用者が識別しやすいように，自由に選択できる．補助線の色とファンクション・キーについても利用者が自由に定義できる．

6.3 構造設計の自動化

意匠の詳細設計が終了すると設計者は構造設計モジュールを用いて構造設計を行う．現時点では，構造設計モジュールが提案する設計解の品質は，構造設計の専門家が満足するほどではないが，意匠設計者の意志決定には実用上十分である．構造設計の専門家が操作すれば，さらに高品質な解を得ることができる．構造設計の詳細は別稿で述べる．構造設計の結果，構造材が耐力的に不具合な箇所は画面上に警告される．さらに，詳細に検討を行うには，構造計算の結果を出力して確認できる．システムが生成した構造図が，建築物として正しいかどうかを，意匠図に重ね合わせて表示することもできる．意匠設計者はこれらの結果を参考にして，必要があれば意匠の修正を行う．

表 3 設計効率の比較
Table 3 Comparison of design efficiency.

例題	構造種別	延べ床面積 (m^2)	個別支援による設計	本システムによる設計
①基準書	RC 造	約 2,300	1 週間	2 時間
②5 階建	S 造	約 3,000	1 か月	2 日
③12階建	SRC 造	約 7,000	3 か月	2 週間

6.4 積算の自動化

意匠設計を進めるに当って，設計者は，壁や床の部材を配置する際に，その単価や品質の等級を考慮しながら詳細設計を行う．この時，設計者にとって，使用した部材の単価が建物全体の価格に及ぼす影響を，詳細に把握することは困難である．本システムは，意匠図の編集時に設計者の要求に従って部材ごとの価格を積算する．また設計の終了時には，工事種別，部屋別，設計見積，業者見積等に分けて詳細な積算を行う．

7. システム構成

本システムは，日本電気製のパーソナルコンピュータ PC-9801 にC言語を用いて作成され，実行プログラムの大きさは約 4 MB である．高速化が必要なグラフィック制御モジュールや使用機種に依存するディジタイザ・ドライバはアセンブラで作成した．また，プリンタ・プロッタ等の周辺機器の構成は環境設定メニューの選択により，多機種に対応できるようにした．

8. システムの評価

本システムを評価するため，人手による設計との比較を行った．表 3 に示す 3 例は，①日本建築学会鉄筋コンクリート構造計算規準同解説[9] 掲載の 3 階建建築物，②実在する平面形状が変形した 5 階建事務所ビル，③実在する鉄骨鉄筋コンクリート造の 12 階建事務所ビルである．実験の結果，平面形状が単純な建物の場合には，専門家が従来から行っている個別支援の方法で設計して約 1 週間かかる設計作業を，実設計経験のない初心者が，本システムを使用して設計した場合，約 2 時間で終了した．設計結果に個人差はあるが，両者とも設計の品質は同等であった．なお，設計期間は，設計に用いた設計図の用紙サイズを基準とし，A1 は 1 人日，A2 は 0.5 人日として換算した．

本システムの効果は次のとおりである．(1)設計作業の効率は手作業に比べて 20 倍以上に改善された．(2)建物の設計品質については人手による場合と比較して遜色がない．(3)設計の変更は極めて容易である．(4)設計ミスがないこと，特に異なる図書間の不整合がない．(5)部材データベースを整備することにより，設計者の属する組織内での部材の標準化が行える．

9. むすび

本システムは，設計作業の効率化を図ること，設計の分業の弊害を防ぐこと，ならびに安価なシステムを供給することを目標に開発してきたが，この点に関してはおおむね初期の目的を達成したと考えている．

本システムの開発で明らかになった最大の問題点は，システムのソフトウェアを担当するソフトウェア技術者と建築の設計知識を整理する設計技術者の相互理解が容易でないことである．この点を改善するには，建築物モデルや設計過程の概念を陽に記述できる表現言語を開発する必要がある[10]．また，本システムの設計支援の範囲はまだ限定されている．例えば，意匠設計については建築法規規制，動線計画，避難計画，部屋配置，内装等の設計知識を整理することにより設計支援の範囲をさらに広げる必要がある．これらは，今後の課題である．

参考文献

1) 日本電信電話(株)：DEMOS BUILD-1 説明書 (1987).
2) (株)構造システム：BUS-2 一貫構造計算プログラム (1986).
3) ユニオンシステム(株)：Super Build/SSI 解説書 (1987).
4) オートデスク(株)：AUTO CAD ADE-3 EX 操作ガイド (1985).
5) (株)コミュニケーションシステム：STCAD 構造図化システム基本操作マニュアル (1987).
6) 篠田博水：3次元家モデルに基づく住宅設計一貫システム，UNIVAC TECHNOLOGY REVIEW 第8号，pp. 22-36 (1985).
7) 山田周平，栗原信一郎，藤田三男：建築設計支援システム DELTA (その1～その3)，第9回電子計算機利用シンポジウム，日本建築学会，pp. 199-216 (1987).
8) 吉川頌三，今井一延：建築，日本機械学会(編) CAD/CAM 事例集，第5章，pp. 103-154，技報堂出版 (1987).
9) (社)日本建築学会：鉄筋コンクリート構造計算規準・同解説 (1982).
10) (社)日本設計製図学会：高度技術化に対応する機械製図システムの標準化のための調査研究 (第三年度) 報告集 (1988).

(平成元年2月27日受付)
(平成元年5月 9 日採録)

長澤 勲 (正会員)

昭和42年九州大学工学部電子工学科卒業，昭和44年同大学院工学研究科修士課程修了，昭和47年同大学院工学研究科博士課程単位取得退学，昭和47年九州大学中央計数施設講師．現在，九州工業大学情報工学部教授 (機械システム工学科)．工学博士．知識情報処理の立場からCAD，FA，ロボット，医療システム等の研究開発に従事．人工知能学会，日本建築学会，精密工学会，電子情報通信学会，日本機械学会，日本ロボット学会各会員．

手越 義昭 (正会員)

昭和49年広島工業大学工学部建築学科卒業，昭和52年九州産業大学大学院修士課程修了．工学修士．現在，広島工業大学助手 (建築学科)．建築情報システム技術の研究に従事，建築設計支援システムの構築法に興味を持つ．日本建築学会，人工知能学会，電子情報通信学会，精密工学会，CAI 学会各会員．

牧野 稔 (正会員)

昭和28年東京大学工学部建築学科卒業．工学博士．現在，九州大学工学部教授 (建築学科)．主として，建築構造学の研究に従事．日本建築学会，日本風工学会各会員．

IBDS: 建築物の統合化設計支援システム†

長澤 勲†† 手越義昭††† 牧野 稔††††

近年建築設計の分野では，パーソナルコンピュータ上に実現されたCADシステムが普及し，設計作業の合理化に貢献している．しかし，これらのCADシステムは，建築設計の幾つかの側面である設計図書の編集，構造計算，積算等の作業を個別的に支援しているのが現状であり，次のような問題点がある．建築設計では，設計が完了した後で発注者の要求や施工上の問題点を解決するために頻繁に再設計が行われる．設計条件が変化した場合の再設計は，従来から行われている手作業と，個別作業の支援のシステムを併用する方法では，図面の変更や構造計算の基礎データを設計者自身が修正しなければならず十分な効果を上げることが困難である．本研究では，設計の個別的な支援による問題点を解決するため，次の特徴をもつCADシステムを開発した．(1)建築物の設計情報を一元的に表現した建築物モデルを中心として，意匠設計，構造設計，積算などの設計作業を支援するモジュールを配置した．このことによって設計者は，設計データの変換にわずらわされることなく一貫して設計作業を進めることができる．(2)建築物モデルの一貫性を管理する建築物モデル管理機構を設け，システムの保守を容易にした．(3)設計者が行う標準的な設計作業の流れを想定し，これを一貫して支援できるように配慮した．実際の設計例に適用した結果，従来の方法に比べて設計期間を約 1/20 に短縮できた．

1. まえがき

近年建築設計の分野では，パーソナルコンピュータ上に実現されたCAD (Computer Aided Design) システム[1)~5)]が普及し，設計作業の合理化に貢献している．しかし，これらのCADシステムは，設計図や設計書（以下，図書）の編集，構造計算，積算等の作業を設計作業の流れに乗って進めるものではなく，流れの一部を取り出して個別的に支援するシステムであり，次のような問題がある．建築設計では，設計が完了した後で発注者の要求や施工上の問題点を解決するため，頻繁に再設計が行われる．設計条件が変化した場合の再設計において，従来から行われている手作業と個別作業の支援システムを併用する方法では，設計者自身が図書の変更や構造計算の基礎データを修正しなければならず，設計ミスの発生や設計変更に迅速に対応できないという問題がある．この問題の解決には，設計作業の流れ全体を一貫して支援する必要がある．このような試みは，住宅設計においては既に行われており，実用システムも開発されている[6)]．しかし，ビル設計を対象とした研究・開発は，その複雑性のためにまだ十分な成果を上げているとは言えない[7),8)]．本研究では，設計の個別的な支援による問題を解決するため，次の特徴をもつCADシステムIBDS (Integrated Building Design System) を開発した．(1)建築物の設計情報を一元的に表現した建築物モデルを中心として，意匠設計，構造設計，積算などの設計作業を支援するモジュールを配置した．このことによって設計者は，設計データの変換にわずらわされることなく一貫して設計作業を進めることができる．(2)建築物モデルの一貫性を管理する建築物モデル管理機構を設け，システムの保守を容易にした．(3)設計者が行う標準的な設計作業の流れを想定し，これを一貫して支援できるように配慮した．

2. 建築物実設計の現状と研究のねらい

ここでは，建築物の概要を決める設計作業を基本設計，建築物を建設するための詳細な設計作業を実設計とよぶことにする．本研究は，後者を対象とする．現状の建築物の実設計は，異なる専門の設計者によって，大きく意匠設計，構造設計，設備設計，積算の4つに分業されている．意匠設計とは建築物の規模，用途，間取り，仕上げ等を決める設計作業，構造設計とは建築物の柱，梁，床，耐震壁等の配置や断面寸法等の，いわゆる，骨組を決める設計作業，設備設計とは給排水，空調，電気設備等の設備を設計する作業，積算とは意匠設計，構造設計，設備設計等で使われている部材の価格を基に建築物全体の価格を見積る作業で

† IBDS: An Integrated Building Design System by ISAO NAGASAWA (Department of Computer Science and System Engineering, Faculty of Computer Science and Mechanical System Engineering, Kyushu Institute of Technology), YOSHIAKI TEGOSHI (Department of Architecture, Faculty of Engineering, Hiroshima Institute of Technology) and MINORU MAKINO (Department of Architecture, Faculty of Engineering, Kyushu University).
†† 九州工業大学情報工学部機械システム工学科
††† 広島工業大学工学部建築学科
†††† 九州大学工学部建築学科

ある．次に，本研究で取り扱う実設計の支援について述べる．意匠設計者は設計作業を進めるに当り，構造設計者や設備設計者の援助が必要である．例えば，意匠設計を進める場合，建築物の構造体の材種や寸法，あるいは，空調設備の機種，給排水設備の配管スペースや高架水槽の位置がある程度決まっていなくてはならない．このために意匠設計者は，構造設計者や設計設備者等の，異なる専門の設計者と打ち合せながら共同で作業を進めねばならない．ところが現状では，分業化を徹底させ流れ作業で設計することが行われているため，互いに十分な意志の疎通ができないことや，各設計者が必要とした時点でただちに専門外の判断を求めることが困難である．このため，設計者はある過去の経験値から見込みで設計を進めることが多い．見込み違いが起これば設計変更が必要となるので余裕の多い設計から収斂させて行くことが行われている．設計変更や再設計を行うには，時間がかかるとともに，問題解決のためには複数の分野の設計者の協力が必要となり，さまざまな図書間の整合性の管理に手間取る．このことは，高品質の建築物を低コストに設計できない原因である．このような困難を解決するには，設計作業の流れを一貫して支援し，設計変更を容易にすることが必要である．設計変更が容易になれば，それを前提とした設計過程が可能となり，設計の可能性が拡大する．

設計作業の流れをCADシステムを用いて一貫して支援するには，設計の各段階における設計情報を建築物モデルに一元化すること，建築物モデルを修正の容易な柔軟な構造にすること，および，設計過程を標準化することが必要である．建築物の設計情報の一元化は，建築物モデル管理機構を用いて建築物モデルの編集，一貫性管理，構造設計モデルへの変換等を行うことにより，また，設計過程の標準化は，建築物モデルの中で使用される設計データの流れを管理することにより行った．この結果，設計変更に際して設計案を容易に修正できる機能，図書間の整合性を管理する機能，複数の分野にまたがる設計上の問題点を指摘する機能を実現できた．

3. 設計方針

本システムの設計方針は次のとおりである．

（1） 意匠設計の支援

意匠設計者は設計作業を進めるに際して，仕上げ材，構造材等の規格品のカタログを検索し，建築物の用途，設計条件，法規や施主の好みなどを考慮しながら，部材の種類，色，形等の意匠を決定する．このとき設計者は，平面図，立面図等，目的に応じてさまざまな視点から捉えた意匠図を用いる．この部材のカタログを検索する作業や，図書間の整合性を保つ作業は，設計者にとり大きな負担である．したがって，本システムでは，意匠図面の編集機能と部材等のカタログ検索機能を有機的に結合すること，さまざまな図書間の整合性を管理すること，および床面積や窓開口率を自動計算することを支援の対象とする．

（2） 構造設計の自動化

建築物の設計では，意匠の概要（建設地域，用途，規模，構造種別）が与えられると構造の概要（柱，梁の配置と断面の仮定）が設計される．次に，意匠設計者は，この結果を基に意匠の詳細を設計する．さらに，この結果を基に構造の詳細が設計（構造計算）される．構造の詳細設計の結果が不適当であれば，意匠設計者は構造に影響を及ぼす意匠の詳細を変更する．経済的，力学的に高品質の建築物を得るためには，この過程を反復しなければならないが，現状では設計コストがかさむことになる．この問題に対して，本システムでは構造の概要設計および詳細設計を自動化する．

（3） 積算の自動化

意匠設計者は，部材の単位当りの価格を参考にして，意匠の詳細設計を行う．意匠の詳細設計が終了すると詳細な積算を行う．この結果と実行予算との間に差があれば，まず意匠の詳細設計のうち価格に大きな影響を与える仕上げを変更する．それでもまだ差があれば構造部材，設備機器の容量等の見直しを行い，可能な変更を試みる．このような過程を反復することによって実行予算に近い仕様の建築物が設計される．ところが，現状ではこの反復作業は，設計コストの上昇を招いている．意匠設計者の選択した部材が，建築物全体の価格に及ぼす影響を，把握することは一般に困難である．したがって，本システムではこの積算作業を自動化する．

（4） 設計支援の範囲

本システムの対象とする建築物は，一般建築物の大半を占める15階建て，高さ60メートル以下の一般的な用途のラーメン構造建築物とする．構造種別は鉄筋コンクリート造（RC造），鉄骨造（S造）および鉄骨鉄筋コンクリート造（SRC造）とする．

（5） 操作性の良い安価なシステムとして実現

上記の建築物の設計は，小規模な設計事務所で行われる場合が多い．したがって，各設計者が占有できる安価なパソコン上に実現することを目標とする．

4. システムの概要

システムは，図1に示すように2つのデータベースと7つのプログラムモジュール（以下，モジュール）からなる．部材データベース①は，建物を構成する基本要素を格納したものである．この部材データベースの管理・編集を行うモジュールが，部材編集モジュール②である．建築物モデルデータベース③は，設計中の建築物モデルを格納する．④～⑦のモジュールは，意匠，構造等の設計者が行う個別の設計作業に対応した設計支援を行う．意匠編集モジュール④は建築物モデルの意匠の編集，構造設計モジュール⑤は建築物モデルの構造の検討，積算モジュール⑥は建築物の価格算出をそれぞれ支援する．その他，必要に応じて専用のモジュール⑦が追加できる．建築物モデル管理機構⑧は，建築物モデルの一貫性を管理するとともに，設計作業ごとに必要な情報を建築物モデルから取り出し操作するモジュールである．ユーザインタフェース⑨は利用者とシステムの接点である．このモジュールには，個々の設計作業において設計者が慣用的に用いている図書を通して設計情報を表示・編集できる工夫を行っている．

5. 建築物モデル

建築物モデルには，次の点に配慮しなくてはならない．

（1） 設計情報の一元管理

建築物の設計は，表1に示すように，意匠，構造等の設計作業に適した図書を用いて行われているが，これらの図面は相互に関連し合って建物を表現している．建築物モデルは，これらの図書の内容を整合性を保って管理し，設計作業ごとに必要な情報を提供できるものでなくてはならない．

（2） 再設計に対する修正の容易性

建築物の設計は，使い勝手，経済性，耐震性，施工の容易性など多面的な設計要求の調和を目的とした再設計の繰り返しである．このため建築物モデルでは，再設計に対して修正の容易な柔軟な構造でなくてはならない．

（3） カスタマイズの容易性

建築物に使用する部材や基準の取り方は，企業や設計グループごとに異なっている．このため建築物モデルは利用者の要求に合わせて容易にカスタマイズできなくてはならない．

従来のシステムでは，建築物の設計情報がファイルやデータベースとプログラムの形で取り扱われることが多く，これらの点について満足な解決を与えているとは言えない．本研究では，(1)建築物や建築物を構成する部材を対象指向言語のオブジェクトの概念を用いて表し，設計に必要な全情報を一元化する，(2)建築物モデルの編集，すなわち部材の配置，削除，属性の変更に際して，従属する部材の削除や属性の変更を建築物モデル自身に自動的に行わせる，(3)部材の配置や属性の計算を行わせるプログラムを部材の定義の形で宣言的に記述できるようにする，方針をとった．

5.1 建築物モデルの表現

建築物モデルを構成する基本要素をここでは部材とよぶ．これらの部材には他の部材との接続関係（2項関係として扱う）および各種の属性が与えられる（図2，図3参照）．

部材は，インスタンスとクラスによって表現する．インスタンスとは建築物を構成する個別部材の表現，

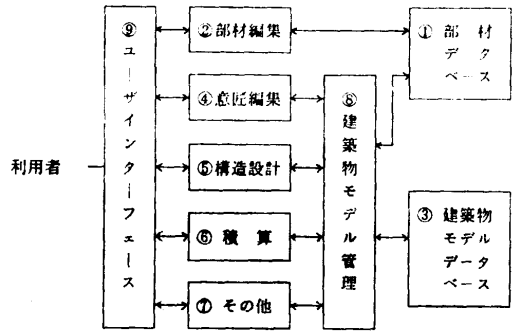


図1 システムの概要
Fig. 1 An overview of the system.

表1 建築図面の種類
Table 1 Kinds of architecture drawings.

意匠図	構造図	設備図
配置図	各階伏せ図	給排水設備図
各階平面図	基礎伏せ図	電気設備図
立面図	軸組図	空調設備図
断面図	架構図	仕様書
各種リスト	各種リスト	
各種詳細図	仕様書	
矩計図		
仕上げ表		
仕様書		

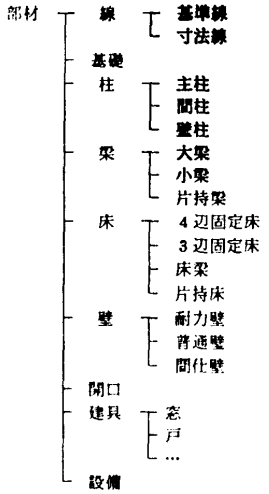


図 2 部材の分類
Fig. 2 Classification of building parts.

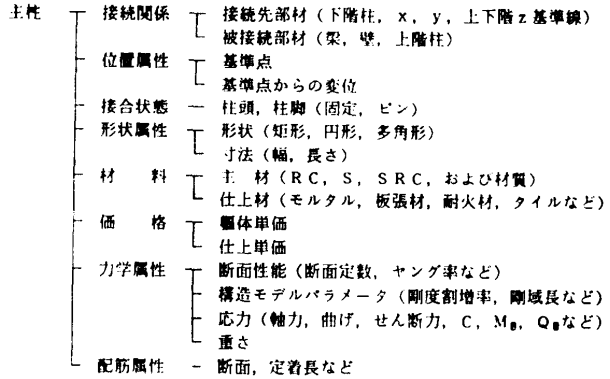


図 3 部材の接続関係と属性（主柱）
Fig. 3 Connecting relations and attributes of a building part (column).

instance i	class C
c : c	super C_s
p : $p_1, \cdots, p_m$	plugs $P_1, \cdots, P_m$
s : $s_1, \cdots, s_n$	sockets $S_1, \cdots, S_n$
g : $g_1, \cdots, g_u$	givens $G_1, \cdots, G_u$
d : $d_1, \cdots, d_v$	derives $D_1, \cdots, D_v$
	methods $M_1, \cdots, M_r$
	views $V_1, \cdots, V_t$
(a) インスタンス	(b) クラス

図 4 部材の記述形式
Fig. 4 Syntax of class and instance representing building part.

クラスとはインスタンスの性質を代表する表現である．

インスタンス 部材のインスタンスは，図 4 (a) の形式で与える．ここで i をインスタンス名，c を所属クラス名，$p_1, \cdots, p_m$ をプラグ，$s_1, \cdots, s_n$ をソケット，$g_1, \cdots, g_u$ を独立属性，$d_1, \cdots, d_v$ を従属属性とよぶ．

インスタンス名 i は指定部材の識別名，所属クラス名は c は指定部材の属するクラスの名称である．

プラグおよびソケットは指定部材の接続相手のインスタンス名を格納する．プラグ p_k $(k=1, \cdots, m)$ は，(V_p, I_s) のように与える．ここで V_p はプラグを表す変数，I_s は指定部材の接続先のインスタンス名である．ソケット s_k $(k=1, \cdots, n)$ は，(V_s, I_p) のように与える．ここで V_s はソケットを表す変数，I_p は指定部材に接続された部材のインスタンス名のリストである．

独立属性 g_k $(k=1, \cdots, u)$ は，指定部材の与件として与えられる属性を表し，(V_g, A_g) のように与える．ここで V_g は独立属性を表す変数，A_g は属性値である．従属属性は他の属性から一意に決定される属性を表し，(V_d, A_d) のように与える．ここで V_d は従属属性を表す変数，A_d は属性値である．

クラス 部材のクラスは，図 4 (b) の形式で与える．ここで C クラス名，C_s を上位クラス名，$P_1, \cdots, P_m$ をプラグ定義，$S_1, \cdots, S_n$ をソケット定義，$G_1, \cdots, G_u$ を独立属性定義，$D_1, \cdots, D_v$ を従属属性定義，$M_1, \cdots, M_r$ を操作定義，$V_1, \cdots, V_t$ をビュー定義とよぶ．

クラス名 C はクラスに与える一意な名称，上位クラス名 C_s は上位のクラス名称である．プラグ定義およびソケット定義は個別部材間の接続関係を代表する．

プラグ定義 P_k $(k=1, \cdots, m)$ は (V_p, C_s, V_s) のように与える．ここで V_p はプラグを表す変数，C_s および V_s は接続先の部材のクラス名とソケットを表す変数である．ソケット定義 S_k $(k=1, \cdots, n)$ は，(V_s, C_p) のように与える．ここで V_s はソケットを表す変数，C_p は接続される部材の属するクラス名である．

独立属性定義 G_k $(k=1, \cdots, u)$ は部材の独立属性を代表し，(V_g, C_g, D) のように与える．ここで V_g は独立属性を表す変数，C_g は属性値のクラス名，D はデフォルト値である．従属属性定義 D_k $(k=1, \cdots, v)$ は，部材の従属属性を代表し，(V_d, C_d, F_d) のように与える．ここで V_d は従属属性を表す変数，C_d は属性値のクラス名，F_d は属性値を計算する関数である．

操作定義 M_k $(k=1, \cdots, r)$ は部材の操作を定義し，

$$m(a_1, \cdots, a_p) : B_1, \cdots, B_q$$

のように与える．ここで m $(a_1, \cdots, a_p)$ を頭部，$B_1, \cdots, B_q$ を本体とよぶ．頭部の m は操作名，$a_1, \cdots, a_p$ は引数，本体の $B_1, \cdots, B_q$ は式である．式とは，定数（数値，クラス名，インスタンス名），変数（プラグ，ソケット，独立属性および従属属性を表す変数，局所

変数)，関数および後述のメッセージ式である．

操作定義はメッセージ式 [X $m(t_1, \cdots, t_p)$] によって使用される．ここでXはクラスあるいはインスタンス名，m は操作名，$t_1, \cdots, t_p$ は式である．メッセージ式の評価はXの属するクラスに定義された操作定義によって行われ，評価値として定義本体の B_q の値が返却される．

ビュー定義 V_k $(k=1, \cdots, t)$ は，設計作業ごとに使用する操作集合を定義し，v：$m_1, \cdots, m_w$ のように与える．ここでvはビュー名，$m_1, \cdots, m_w$ は操作名である．

5.2 建築物モデルの機能

建築物モデルの機能は，部材データベース，建築物モデルデータベースおよび建築物モデル管理機構の組合せによって実現される．部材データベースは部材のクラスを，建築物モデルデータベースは設計中の建築物モデルを格納する．建築物モデル管理機構は2つのデータベースを使用し，建築物モデルの一貫性の管理や各種の設計作業に必要な情報の提供を行う（表2参照）．

ここでは，建築物モデルの基本的な概念である部材の配置と従属性について述べる．これは部材は別の部材に依存して配置（接続）され，その属性の一部は別の属性から一意に決定されること（属性従属性），またある部材が削除されるとき，その部材に接続された部材も同時に削除されること（存在従属性），を意味する．

部材の接続 部材の接続関係は，クラスのプラグおよびソケット定義とインスタンスのプラグおよびソケットによって表す．今，クラス C_b に属する任意のインスタンスがクラス C_a に属する任意のインスタンスに接続できるものとすれば，C_a のソケット定義を $(V_s\ C_b)$，C_b のプラグ定義を $(V_p\ C_a\ V_s)$ のように与えれば良い．C_a のインスタンス I_a に C_b の2つのインスタンス I_{b1}, I_{b2} が接続された場合，I_a のソケットは $(V_s\ [I_{b1}, I_{b2}])$，I_{b1}, I_{b2} のプラグはいずれも $(V_p\ I_a)$ ように設定される．ここで $[I_{b1}, I_{b2}]$ は，I_{b1}, I_{b2} からなるリストである．

部材の生成 クラスCが与えられたとき，Cのインスタンスは，メッセージ式 [C new($I_1, \cdots, I_m$)] を評価することによって生成される．ここで $I_1, \cdots, I_m$ は生成されたインスタンスの接続先のインスタンスであり，Cのプラグ定義 $P_1, \cdots, P_m$ に対応している．生成されたインスタンスのプラグおよび，インスタンス $I_1, \cdots, I_m$ のソケットは前述のように設定される．また，生成されたインスタンスの独立属性はデフォルト値，従属属性は nil（未定義を示す）である．new 操作は，クラスCに対して**図5**(a)のように定義される．同図において，①はクラスCの定義から変数値が未定義なインスタンスを作成 (make) し，局所変数 I_{new} に代入する，②は I_{new} と各インスタンス $I_1, \cdots, I_m$ 間に接続関係を定義 (plugin) する，③は I_{new} をインスタンスの集合からなる建築物モデル M_{odel} に挿入

```
new(I1, ···, Im):
    Inew←make(C),                          ···①
   [Inew plugin(I1, ···, Im)],             ···②
    Model←cons(Inew, Model),               ···③
    forall(I, Model, [I reflesh]),         ···④
    Inew                                   ···⑤
```

(a) インスタンスの生成

```
delete( ):
  [Self kill],                             ···⑥
  forall(I, Model, [I set(Flag, nil)]),    ···⑦
  forall(I, Model, [I clear]),             ···⑧
  forall(I, Model, [I reflesh])            ···⑨
kill( ):
  [Self plugout],                          ···⑩
  Model←omit(Self, Model),                 ···⑪
  [Self set(Flag, die)]                    ···⑫
clear( ):
  case(Flag→Flag,                          ···⑬
    forsome(Vp, [Self plugs], [Vp clear]=die),  ···⑭
        → [Self kill],
    true → [Self set(Flag, live)])         ···⑮
```

(b) インスタンスの削除

図5 部材の生成と削除
Fig. 5 Creating and deleting building parts.

表2 建築物モデルの機能
Table 2 Functions of building models.

[C new ($I_1, \cdots, I_n$)]	クラスCの部材を生成し，部材 $I_1, \cdots, I_n$ に接続する．
[M search (C, C_d)]	モデルMから条件 C_d を満たすクラスCの部材を検索する．
[M pickup (C, V_w, X, Y)]	ビュー V_w で座標値 X, Y にあるクラスCの部材を選択する．
[I delete]	部材Iを削除する．
[I V]	部材Iのプラグ／ソケット変数Vで指定した部材を参照する．
[I V]	部材Iの属性変数Vで指定した属性値を参照する．
[I change_V (A)]	部材Iの属性変数Vで指定した属性値をAに変更する．

(cons) する，④は M_{odel} の各インスタンスIについてその従属属性を再計算 (reflesh) する，⑤は生成されたインスタンスをメッセージ式の値として返却する，である．

部材の削除 インスタンスIを建築物モデルから削除するには，メッセージ式 [I delete] を評価する．delete 操作は，インスタンスIに直接あるいは間接に接続されているインスタンスも同時に削除する．delete 操作は図5(b)のように定義される．まず，全インスタンスにそれが削除されているかどうかを示す標識 Flag を与え，使用中，削除済，未検査に対してそれぞれ live, die, nil とする．delete 操作の意味は，⑥インスタンス Self を削除 (kill) する，⑦建築物モデルの全インスタンスの Flag を nil に設定 (set) する，⑧全インスタンスに対してそれが削除されるべきものであるかどうかを検査し，もしそうであれば削除 (clear) する，⑨全インスタンスの従属属性を再計算 (reflesh) する，である．kill 操作の意味は，⑩接続先との接続関係を消去 (plugout) する，⑪建築物モデルから除去 (omit) する，⑫Flag を die に設定する，である．また，clear 操作の意味は，⑬Flag の値が live または die であれば，そのまま値として返却する，⑭接続先の全インスタンス (plugs) を検査 (clear) し，いずれかが die であれば，インスタンス Self を削除 (kill) する，⑮いずれでもなければ，Flag を live に設定し値として返却する，である．

独立属性の操作 インスタンスIが与えられたとき，その独立属性 V_θ を参照するには，メッセージ式 [I V_θ] を評価する．また，独立属性 V_θ を属性値 A_θ に変更するにはメッセージ式 [I change_V_θ(A_θ)] を評価する．操作の定義を図6(a)に示す．参照操作の意味は①変数値を参照する，である．変更操作の意味は②属性値 A_θ を変数 V_θ に設定し，③建築物モデルの各インスタンスIについて，その従属属性を再計算 (reflesh) する，である．

従属属性の操作 インスタンスIが与えられたとき，その従属属性 V_δ を参照するには，メッセージ式 [I V_δ] を評価する．また，建築物モデルの変更に際して従属属性を再計算するには，メッセージ式 [I reflesh] を評価する．図6(b)に操作の定義を示す．参照操作の意味は，④もし変数値が，既に計算されていればそれを値として返却する，⑤もし変数値が nil であれば従属属性定義に与えられた関数 F_δ を用いて変数値を計算し，値として返却する，である．再計算

```
Vθ( ): Vθ                                          …①
change_Vθ(Aθ):
    [Self set(Vθ, Aθ)],                            …②
    forall(I, Model, [I reflesh])                  …③
```

(a) 独立属性の参照と変更

```
Vδ( ):
    case(Vδ→Vδ,                                    …④
        true→[Self set(Vδ, Fδ)])                   …⑤
reflesh( ):
    forall(Vδ, [Self derives], [Self set(Vδ, nil)]) …⑥
```

(b) 従属属性の参照と初期化

図6 部材属性の操作

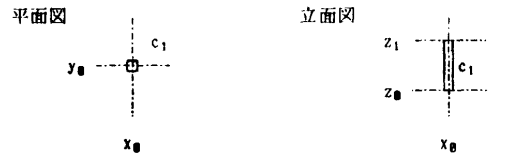
Fig. 6 Changing and computing the attributes of building parts.

操作の意味は，⑥すべての従属属性 (derives) の値を nil (未定義) に設定する，である．

5.3 建築物モデルの例

ここでは**図7**に示す簡単な例を用いて建築物モデルの機能を説明する．図7(a)の建築物モデルは，x基準線 x_0, y 基準線 y_0, z 基準線 z_0, z_1, および主柱 c_1 からなり，基準線 z_1 は基準線 z_0 に，主柱 c_1 は基準線 x_0, y_0, z_0, z_1 にそれぞれ接続 (依存して配置) されている．また，基準線 x_0, y_0, z_0 は他の部材には接続されていない．

図7(b)はz基準線のクラス (x, y 基準線のクラスもほぼ同じ) である．図中①～③は基準線のプラグとソケット定義を示す．Pred は先行 (接続先) 基準線を，Succ は後続 (被接続) 基準線を，Colm は被接続主柱のリストを表す．④～⑥は基準線の属性を示す．Span は先行基準線からの距離を，No は基準線に割り当てた一連番号を，Z は基準線の z 座標を表す．Span の値は change 操作によって変更できる．また，No および Z は従属属性であり，属性値の参照に際して次のように計算される．No の計算⑤は，もし先行する基準線がなければ0を，そうでなければ先行基準線の一連番号＋1を値として返却する．Z の計算⑥は，もし先行する基準線がなければ Span を，そうでなければ先行基準線の Z 座標＋Span を値として返却する．基準線 z_0 は前者の場合に，z_1 は後者の場合に対応する．⑦～⑨は基準線の操作定義を示す．fig_xz は立面図に表示する形状 (水平な破線) を，visible_xy は平面図からの可視性 (非表示を示す nil) を，visible_xz は立面図からの可視性 (表示を示す true) を値として返却する．⑩～⑫はビュー定義を示す．xy_plane は平面図の編集作業，xz_plane は立面図の編集作業，demo は例題のためのものである．xy_plane では，



(a) 部材の配置例

```
class  z基準線
  super 基準線
  plugs   (Pred  z基準線 Succ)                          …①
  sockets (Succ  z基準線),                              …②
          (Colm 主柱)                                   …③
  givens  (Span  整数 700)                              …④
  derives (No    整数 case(null(Pred)→0,                …⑤
                          true→[Pred No]+1)),
          (Z     整数 case(null(Pred)→Span,             …⑥
                          true→[Pred Z]+Span))
  methods  fig_xz() : line(Z, 50, Z, 10000)             …⑦
           visible_xy(Floor) : nil                      …⑧
           visible_xz(Frame) : true                     …⑨
  views    xy_plane : visible_xy                        …⑩
           xz_plane : visible_xz, new, delete,          …⑪
                      change_Span, fig_xz
           demo :     new                               …⑫
```

(b) z基準線のクラス

```
class  主柱
  super 柱
  plugs   (Xline  x基準線 Colm),                        …①
          (Yline  y基準線 Colm),                        …②
          (Zfline  z基準線 Colm),                       …③
          (Zhline  z基準線 Colm)                        …④
  givens  (D     整数 60),                              …⑤
          (Dx    整数  0),                              …⑥
          (Dy    整数  0)                               …⑦
  derives (Flno  整数 [Zhline No]),                     …⑧
          (Frno  整数 [Zline No]),                      …⑨
          (X     整数 [Xline X]+Dx),                    …⑩
          (Y     整数 [Yline Y]+Dy),                    …⑪
          (Zf    整数 [Zfline Z]),                      …⑫
          (Zh    整数 [Zhline Z])                       …⑬
  methods  fig_xy() : rectangle(X−D/2, Y−D/2,           …⑭
                      X+D/2, Y+D/2)
           fig_xz() : rectangle(X−D/2, Zf, X+D/2, Zh)
                                                        …⑮
           visible_xy(Floor) : equal(Floor, Flno)       …⑯
           visible_xz(Frame) : equal(Frame, Frno)       …⑰
  views    xy_plane : visible_xy, new, delete,          …⑱
                      change_D, change_Dx,
                      change_Dy, fig_xy
           xz_plane : visible_xz, fig_xz                …⑲
           demo :     new, change_D                     …⑳
```

(c) 主柱のクラス

```
class  xy_plane
  super window
  vars  (Floor 表示階)
  methods new(Flr) :                                    …①
              [Self set(Floor, Flr)],
              [Self reflesh]
          reflesh() :                                   …②
              forall(I, Model,
                case([I visible_xy(Floor)]
                     → [Self draw([I fig_xy])]))
```

(d) 平面図のクラスの一部

```
class demo
  super window
  methods test() : x0←[xline new(nil)],                 …①
                   y0←[yline new(nil)],                 …②
                   z0←[zline new(nil)],                 …③
                   z1←[zline new(z0)],                  …④
                   c1←[colm new(x0, y0, z0, z1)],       …⑤
                   [z1 change_Span(400)],               …⑥
                   xy1←[xy_plane new(1)],               …⑦
                   xz1←[xz_plane new(0)]                …⑧
```

(e) 操作例のクラス

```
instance x0
  c : x基準線
  p : (Pred nil)
  s : (Succ nil),
      (Colm [c1])
  g : (Span 100)
  d : (No    0),
      (Z   100)
instance y0
  c : y基準線
  p : (Pred nil)
  s : (Succ nil),
      (Colm [c1])
  g : (Span 100)
  d : (No    0),
      (Z   100)
instance z0
  c : z基準線
  p : (Pred nil)
  s : (Succ z1),
      (Colm [c1])
  g : (Span 100)
  d : (No    0),
      (Z   100)
instance c1
  c : 主柱
  p : (Xline  x0),
      (Yline  y0),
      (Zfline z0),
      (Zhline z1)
  g : (D     60),
      (Dx     0),
      (Dy     0)
  d : (Flno   1),
      (Frno   0),
      (X    100),
      (Y    100),
      (Zf   100),
      (Zh   500)
instance z1
  c : z基準線
  p : (Pred z0)
  s : (Succ nil),
      (Colm [c1])
  g : (Span 400)
  d : (No    1),
      (Z   500)
```

(f) インスタンス

図 7 建築物モデルの例
Fig. 7 A building model.

z基準線は表示も編集もされないので visible_xy だけが必要である．xz_plane では，z 基準線の生成 (new)，削除 (delete)，移動 (change_Span)，表示 (fig_xz) が行われる．demo では，z 基準線の生成が行われる．

図7(c)は主柱のクラスである．図中①～④は主柱のプラグ定義を示す．Xline, Yline, Zhline, Zfline はそれぞれ接続先の x 基準線，y 基準線，上階 z 基準

線，下階 z 基準線を表す．⑤〜⑬は主柱の属性を示す．独立属性として柱幅(D)，基準線からの x 方向変位 (Dx)，y 方向変位 (Dy)，従属属性として階番号 (Flno)，通番号 (Frno)，柱心 x 座標(X)，柱心 y 座標 (Y)，柱頭 z 座標 (Zh)，柱脚 z 座標 (Zf) を与えている．従属属性の値は，独立属性と基準線の属性から計算される．⑭〜⑰は操作定義を示す．fig_xy は平面図に表示する形状 (方形) を，fig_xz は立面図に表示する形状 (矩形) を，visible_xy は平面図からの可視性を，visible_xz は立面図からの可視性を返却する．⑱〜⑳はビュー定義を示す．xy_plane では，主柱の生成 (new)，削除 (delete)，柱幅変更 (change_D)，x 変位変更 (change_Dx)，y 変位変更 (change_Dy)，表示 (fig_xy) が，xz_plane では表示 (fig_xz) が，demo では生成 (new) と柱幅変更 (change_D) が行われる．

建築物モデルを操作するプログラムの一例を図 7 (d)，(e)に示す．同図(d)は平面図を編集する操作をまとめたクラスである．操作の意味は，①平面図を生成 (new) する，②平面図を再生 (reflesh) する，である．また，同図(e)は例題のためのクラスである．操作の意味は，①x 基準線 x_0 を生成する，②y 基準線 y_0 を生成する，③z 基準線 z_0 を生成する，④z 基準線 z_1 を生成して z_0 に接続する，⑤主柱 c_1 を生成して x_0, y_0, z_0, z_1 に接続する，⑥z 基準線 z_1 のスパンを 400 に変更する，⑦平面図 xy_1 を生成し，全部材を表示する，⑧立面図 xz_1 を生成し，全部材を表示する，である．図 7 (a)に示した建築物モデルは，メッセージ式 $demo_1$←[demo new], [$demo_1$ test] を順に評価することにより生成される．生成後のインスタンスの状態を図 7 (f)に示す．

6. 建築物の設計過程

6.1 設計作業の流れ

本システムを使用して行う標準的な設計作業の流れを図 8 に示す．図中破線は設計者がシステムを操作して行う作業，実線はシステムが自動的に行う作業である．①設計者は，設計の基本条件である建物所在地・建物用途・建物規模・スパン長・階高等の建物概要をシステムに入力する．②システムは，自動的に概略の構造計算を行い，柱・梁の構造材を配置する．③設計者は，配置された柱断面を考慮しながら，意匠設計を進める．つまり，意匠編集モジュールを用いて，意匠平面図上で柱・壁などの部材を配置する．④この部材

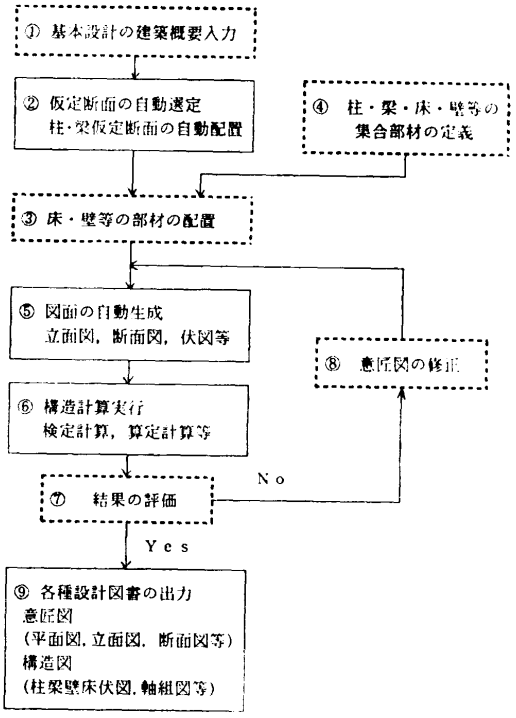


図 8 本システムを使った設計作業の流れ
Fig. 8 Design procedure by the system.

は③に先だって定義しておく．⑤システムは立面図，断面図等を自動生成する．⑥システムは意匠設計の結果を用いて，構造設計を行い伏図，軸組図等の構造設計図を自動生成する．⑦設計者は構造設計の結果を評価する．もし，意匠と構造の取り合いに不具合があれば，⑧設計者は意匠図の修正を行い，⑤，⑥を再度試みる．もし不具合がなければ設計を終了する．⑨システムは，意匠図，構造図，構造計算書等の設計結果を出力する．

6.2 意匠設計の支援

(1) 部材データベース

部材データベースには，素材と部材が格納される．素材とは建築物を構成する構造材や仕上材，部材とは複数の素材を組み合わせて建築物を構成する基本要素としてまとめた柱・梁・壁等である．素材は JIS 規格や各メーカの規格品情報を多量に収集して作成する．部材は建築物ごとの取り決めや各社の標準仕様を対象としており，素材に比べて少量である．これらはシステムの利用者が自由に定義できるように配慮されている．

(2) 意匠図の編集

意匠の編集には平面図，立面図，断面図を用いる．設計者は図 9 に示すように，部材データベースから部

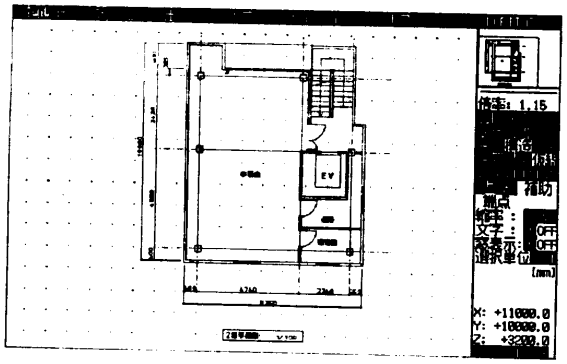


図 9 意匠図編集画面
Fig. 9 Screen for editing design diagram.

材をマウスで選択し，意匠平面図上に配置する．このとき，システムは一貫性を保ちながら建築物モデルを操作しているが，設計者からは見かけ上，平面図を描いているように見える．設計者が意匠図の柱寸法を変更した場合，これに依存する意匠図と構造図の柱や梁の寸法は，建築物モデルの機能より，自動的に変更される．同様に，各階の意匠平面図を作成することにより，立面図や任意の位置での断面図を自動的に作成できる．設計者はこの立面図と断面図を用いて，立面方向の取り合いを確認する．

（3） 利用者インタフェース

設計支援システムの利用者インタフェースは，設計者の思考を妨げないように，設計者が慣れ親しんでいる設計図を使用し，違和感なく操作できる必要がある．このため，意匠設計者が通常の業務で使用するのに適したウインドウシステムを整備した．本システムでは，座標値の選択やコマンドの入力等は，すべてマウスだけで行うことができ，また必要に応じて全操作をコマンドで行うこともできる．画面上のカーソルの色や形状は，利用者が識別しやすいように，自由に選択できる．補助線の色とファンクション・キーについても利用者が自由に定義できる．

6.3 構造設計の自動化

意匠の詳細設計が終了すると設計者は構造設計モジュールを用いて構造設計を行う．現時点では，構造設計モジュールが提案する設計解の品質は，構造設計の専門家が満足するほどではないが，意匠設計者の意志決定には実用上十分である．構造設計の専門家が操作すれば，さらに高品質な解を得ることができる．構造設計の詳細は別稿で述べる．構造設計の結果，構造材が耐力的に不具合な箇所は画面上に警告される．さらに，詳細に検討を行うには，構造計算の結果を出力して確認できる．システムが生成した構造図が，建築物として正しいかどうかを，意匠図に重ね合わせて表示することもできる．意匠設計者はこれらの結果を参考にして，必要があれば意匠の修正を行う．

表 3 設計効率の比較
Table 3 Comparison of design efficiency.

例題	構造種別	延べ床面積 (m^2)	個別支援による設計	本システムによる設計
①基準書	RC 造	約 2,300	1 週間	2 時間
②5 階建	S 造	約 3,000	1 か月	2 日
③12階建	SRC 造	約 7,000	3 か月	2 週間

6.4 積算の自動化

意匠設計を進めるに当って，設計者は，壁や床の部材を配置する際に，その単価や品質の等級を考慮しながら詳細設計を行う．この時，設計者にとって，使用した部材の単価が建物全体の価格に及ぼす影響を，詳細に把握することは困難である．本システムは，意匠図の編集時に設計者の要求に従って部材ごとの価格を積算する．また設計の終了時には，工事種別，部屋別，設計見積，業者見積等に分けて詳細な積算を行う．

7. システム構成

本システムは，日本電気製のパーソナルコンピュータ PC-9801 にC言語を用いて作成され，実行プログラムの大きさは約 4 MB である．高速化が必要なグラフィック制御モジュールや使用機種に依存するディジタイザ・ドライバはアセンブラで作成した．また，プリンタ・プロッタ等の周辺機器の構成は環境設定メニューの選択により，多機種に対応できるようにした．

8. システムの評価

本システムを評価するため，人手による設計との比較を行った．表 3 に示す 3 例は，①日本建築学会鉄筋コンクリート構造計算規準同解説[9] 掲載の 3 階建建築物，②実在する平面形状が変形した 5 階建事務所ビル，③実在する鉄骨鉄筋コンクリート造の 12 階建事務所ビルである．実験の結果，平面形状が単純な建物の場合には，専門家が従来から行っている個別支援の方法で設計して約 1 週間かかる設計作業を，実設計経験のない初心者が，本システムを使用して設計した場合，約 2 時間で終了した．設計結果に個人差はあるが，両者とも設計の品質は同等であった．なお，設計期間は，設計に用いた設計図の用紙サイズを基準とし，A1 は 1 人日，A2 は 0.5 人日として換算した．

本システムの効果は次のとおりである．(1)設計作業の効率は手作業に比べて 20 倍以上に改善された．(2)建物の設計品質については人手による場合と比較して遜色がない．(3)設計の変更は極めて容易である．(4)設計ミスがないこと，特に異なる図書間の不整合がない．(5)部材データベースを整備することにより，設計者の属する組織内での部材の標準化が行える．

9. むすび

本システムは，設計作業の効率化を図ること，設計の分業の弊害を防ぐこと，ならびに安価なシステムを供給することを目標に開発してきたが，この点に関してはおおむね初期の目的を達成したと考えている．

本システムの開発で明らかになった最大の問題点は，システムのソフトウェアを担当するソフトウェア技術者と建築の設計知識を整理する設計技術者の相互理解が容易でないことである．この点を改善するには，建築物モデルや設計過程の概念を陽に記述できる表現言語を開発する必要がある[10]．また，本システムの設計支援の範囲はまだ限定されている．例えば，意匠設計については建築法規規制，動線計画，避難計画，部屋配置，内装等の設計知識を整理することにより設計支援の範囲をさらに広げる必要がある．これらは，今後の課題である．

参考文献

1) 日本電信電話(株)：DEMOS BUILD-1 説明書 (1987).
2) (株)構造システム：BUS-2 一貫構造計算プログラム (1986).
3) ユニオンシステム(株)：Super Build/SSI 解説書 (1987).
4) オートデスク(株)：AUTO CAD ADE-3 EX 操作ガイド (1985).
5) (株)コミュニケーションシステム：STCAD 構造図化システム基本操作マニュアル (1987).
6) 篠田博水：3次元家モデルに基づく住宅設計一貫システム，UNIVAC TECHNOLOGY REVIEW 第8号，pp. 22-36 (1985).
7) 山田周平，栗原信一郎，藤田三男：建築設計支援システム DELTA (その1～その3)，第9回電子計算機利用シンポジウム，日本建築学会，pp. 199-216 (1987).
8) 吉川頌三，今井一延：建築，日本機械学会(編) CAD/CAM 事例集，第5章，pp. 103-154，技報堂出版 (1987).
9) (社)日本建築学会：鉄筋コンクリート構造計算規準・同解説 (1982).
10) (社)日本設計製図学会：高度技術化に対応する機械製図システムの標準化のための調査研究 (第三年度) 報告集 (1988).

(平成元年2月27日受付)
(平成元年5月 9 日採録)

長澤 勲 (正会員)

昭和42年九州大学工学部電子工学科卒業，昭和44年同大学院工学研究科修士課程修了，昭和47年同大学院工学研究科博士課程単位取得退学，昭和47年九州大学中央計数施設講師．現在，九州工業大学情報工学部教授 (機械システム工学科)．工学博士．知識情報処理の立場からCAD，FA，ロボット，医療システム等の研究開発に従事．人工知能学会，日本建築学会，精密工学会，電子情報通信学会，日本機械学会，日本ロボット学会各会員．

手越 義昭 (正会員)

昭和49年広島工業大学工学部建築学科卒業，昭和52年九州産業大学大学院修士課程修了．工学修士．現在，広島工業大学助手 (建築学科)．建築情報システム技術の研究に従事，建築設計支援システムの構築法に興味を持つ．日本建築学会，人工知能学会，電子情報通信学会，精密工学会，CAI 学会各会員．

牧野 稔 (正会員)

昭和28年東京大学工学部建築学科卒業．工学博士．現在，九州大学工学部教授 (建築学科)．主として，建築構造学の研究に従事．日本建築学会，日本風工学会各会員．