ソフトウェア工学 103－7
（1995．3．9）

建築物設計を事例としたドメイン特化概念モデルとその実装

佐藤 俊孝　奥園 高基　出口 孝一　廣田 豊彦　橋本 正明　片峯 恵一

九州工業大学 情報工学部

建築物設計のような専門家の知的な作業を支援するためのソフトウェアを開発する場合，生産性や信頼性を向上するため，特に要求分析工程を確実に実施することが重要な課題となっている．そこで筆者らは，建築物設計を事例として，ドメインの知識を表すための概念モデルをドメインに特化し，そのモデルに基づいて仕様記述言語を規定する研究を行なった．さらに，その言語で記述された仕様から設計支援システムを生成するための研究を行なった．ドメインに依存した仕様記述言語を規定し，ドメインの専門家に直接，要求仕様を記述させることは，システムの要求分析に有効な手段といえる．

A Domain Specific Conceptual Model and Its Implementation in Architectural Design

Toshitaka SATO　Takaki OKUZONO　Kouichi DEGUCHI
Toyohiko HIROTA　Masaaki HASHIMOTO　Keiichi KATAMINE

Department of Artificial Intelligence
Kyushu Institute of Technology
Iizuka 820 JAPAN
Tel:+81-948-29-7639 E-mail : sato@minnie.ai.kyutech.ac.jp

In developing a software which supports experts work such as architectural design, requirements analysis is essential to improve its productivity and reliability. We have proposed a domain specific conceptual model and description language in architectural design. And we have developed a compiler which generates an architectural CAD system from specifications described in this language. Using this description language, a designer can easily modify and improve the design knowledge. Thus system requirements analysis will become efficient by using a domain specific conceptual model and description language.

1 はじめに

建築物設計のような専門家の知的な作業を支援するためのソフトウェアを開発する場合，その生産性や信頼性を向上するため，特に要求分析工程を確実に実施することが重要な課題となっている．そこで筆者らは，建築物の設計者が要求仕様を直接記述することができるように，建築物設計のドメインに依存した概念モデルを作成し，そのモデルに基づいて仕様記述言語を規定する研究を行なった．さらに，その言語で記述された仕様から設計支援システムを生成する研究も行なった [1][2][3].

建築物設計は意匠設計と構造設計，設備設計，積算の4種に分業化されている．それぞれを担当している専門の設計者は，互いに意思疎通を図りながら協調して作業を進めていかなくてはならない．しかし，現在は分業を徹底させ，流れ作業で設計を行なっているので，それぞれの専門の設計者は，他の専門の設計者と意思疎通を図るのが困難となっている．

また，従来の建築物設計支援システムは，分業化された専門の設計をそれぞれ独立して支援している．独立した支援システム間の設計情報の交換は手作業で行っているので，支援システムの間で設計情報を整合させるのが容易でない．このため，前述の意思疎通の問題を解決するのをさらに困難にしている．

この問題を設計支援システムの側面から改善するには，それぞれの設計作業に必要な設計情報を全て一元化し，設計作業の流れ全体を一貫して支援することが必要である．そのような設計支援システムを開発するには，まず，システムの要求仕様を決めなければならない．そこで筆者らは，新しい試みとして，建築物の設計者が要求仕様を直接記述することができるように，建築物設計のドメインに依存した概念モデルを作成し，そのモデルに基づいて仕様記述言語を規定する研究を行なった．

2 建築物設計の現状と問題点

建築物設計は，意匠設計，構造設計，設備設計，積算の4種に分業化されており，それぞれを専門の設計者が担当している．ところで，各設計者は，相互に他の設計者の援助を必要としている．たとえば，意匠設計者は，構造設計者や設備設計者と意思疎通を図りながら共同で作業を進めていかなくてはならない．

しかし，現在は分業を徹底させ，流れ作業で設計を行なっているので，各設計作業の間で意思疎通を図るのが容易ではない．このことから，それぞれの専門の設計者は，他の専門の設計者の判断を求めることが困難となっている．このため，設計者は過去の経験や見込みで設計を進めることが多い．

従来の設計支援システムは，分業化された専門の各設計を独立に支援している．これらの独立した支援システムの間の設計情報の交換は手作業で行なわれており，設計者自身が支援システムの設計図や設計書 (設計図書) を変更しなければならず，設計情報の整合性を保つのが困難になる．その結果，たとえば意匠設計が終了した時点で設計の手戻りが困難となり，結果として構造耐力の不足や非常に高額の見積り金額になってしまうことがある．

このような問題を解決するには，設計作業の流れ全体を一貫して支援することが必要である．そこで，筆者らは，建築物設計に依存した概念モデルを介して，それぞれの各設計支援システムを統合することを目指している．

3 概念モデル

設計者が設計対象を設計支援システム上で設計する際，設計支援システムはその設計作業を支援しなければならないが，設計支援システムが設計対象の概念を持っていないと充分な支援が出来ない．このことから，設計支援システムは設計対象の概念を取り入れるべきである．設計者は設計対象の絵のモデルを持っているのではなく抽象化された属性を持っているので，設計支援システムが設計対象の概念を取り入れ，設計者が設計対象の絵を編集するのではなく，抽象化された属性を編集できるようにしなくてはならない．

設計作業の流れ全体を一貫して支援するには，各設計作業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構造にすることが必要である．筆者らは，オブジェクト指向かつ属性指向の概念モデルを中心としたシステムを構築している (図 1).

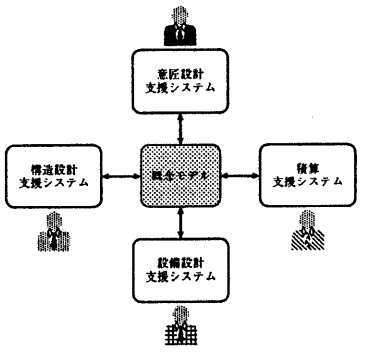

図 1: 概念モデルを中心とした統合設計支援システム

設計支援システムが設計対象の概念モデルを取り入れることで次のような利点がある．

- 設計支援システムが概念モデルを持つことで，設計支援システムの更新が容易 … 従来のシステムは，建築物モデルの変更が容易ではないが，概念モデルを持つことによって保守が容易となる．
- 一元的に管理する設計情報を概念モデルを用いて容易に設計できる … 各設計作業で必要な設計情報は，中心となる建築物モデルを概念モデルを用いて設計することによって一元化できる．

まず筆者らは，中心となる概念モデルの実装と，分業化された設計の一つである構造設計支援システムの試作を行なっている．

中心となる概念モデルは，設計対象である建築物の，それぞれの専門の設計で必要とされる属性や，属性の間の関係を含んでいる．さらに，それらの属性は，建築物の構成要素である部材の内部属性として保持され，関係に従って自動計算が行なわれる．また，概念モデルは，このような部材間の関係も管理する．

各専門の設計支援システムは，この建築物モデルから必要な情報を得たり，変更したりする．たとえば，構造設計の支援では，設計対象の構造上の問題の有無を調べるために属性情報を参照し，必要なら計算する．問題が生じた場合，柱の幅などの属性を変更したり，構造部材の組み換えを行なう．筆者らは中心となる概念モデルに続いて，構造設計における部材間の組合せなどを変更する構造操作のモデル化を行なった．これらの構造操作は，構造編集や問題解決時に用いられる．

3.1 構造

建築物の構造の概念モデルは，分業化された各専門の設計作業で必要な建築物そのものに関する情報を，建築物の属性の集合として表すための枠組である．また，これらの属性は，建築物を構成するそれぞれの部材に所属している．これらの属性には相互に依存関係があり，設計対象の編集によって一部の属性の値を変更すると，他の属性の値も再計算しなければならない．これを属性の従属性と呼ぶ．また，梁が二本の柱に支えられているように，部材間には存在の依存関係があり，これを存在の従属性と呼ぶ．

次に，属性の従属性と存在の従属性について説明する．

属性の従属性　設計対象物の属性間には依存関係があり，ある属性の値が変更された時は，依存関係に従って他の属性の値を再計算しなくてはならない．すなわち，ある属性Ａの値が属性ＢとＣの値に依存しているならば，属性ＢまたはＣの値が変更されたとき，属性Ａの値を変更しなければならない．

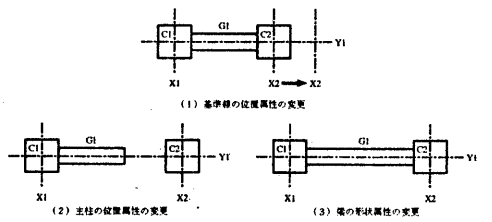

図 2: 属性の従属性の例

たとえば，図2において，主柱C1と主柱C2の間に張られている梁G1の両端の位置属性は，それぞれ二つの主柱の位置属性を参照して計算しており，これらの主柱の位置属性が変更されると再計算しなければならない．また，梁G1の両端の位置属性を参照して算出される梁の形状属性や，形状属性を参照する力学属性の応力や重量，価格などの属性も再計算しなければならない．

存在の従属性　建築物を構成する部材間には存在の従属性がある．たとえば，大梁は二つの主柱によって支えられており，この二つの主柱のどちらかが存在していなければ，この大梁は存在できない．したがって，設計中に大梁を支えている主柱が削除されると，その大梁も削除しなければならない．

また，ある部材を生成しようとしたとき，その部材が依存しようとする部材がすでに存在していなくてはならない．

3.2 構造操作

この項では，前項で述べた構造に対する操作について述べる．構造の操作とは，設計者の意図を実現するための建築物に対する個々の操作である．これらの操作は設計支援システム上で設計者が呼び出す一連の手続きであり，構造上の問題を解決するための定型的な手続きを表したり，構造の編集の効率を良くするために用いらる．構造の操作は，存在の従属性に基づき，設計対象に新しい部材を一つ加えたり，すでに存在する部材を一つ取り外したりするものや，他の操作を組み合わせて，部材のプラグやソケットをつなぎ換えたり，部材の持つ属性を変更したりして，複雑な操作を実現するものがある．

たとえば，ホテルのロビーのような広い空間を確保するのに不都合な主柱を取り除き，床をささえるために梁を伸ばして補強する操作がある．これを柱抜きという (図3)．この操作の手順として，主柱が削除された後に床を支える

ために二本の梁を一本につなぐ方法がある．具体的には，二本の梁のうち一方を削除し，もう一方を伸ばす．このとき，削除される梁に依存している小梁やスラブを伸ばす梁の方につなぎ換え，削除される主柱に依存する残りの梁も同様に伸ばす梁の方につなぎ換える．

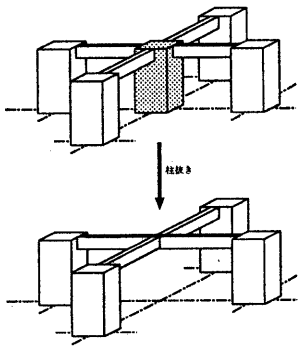

図 3: 柱抜き

4 仕様記述言語

筆者らは，前節で述べた概念モデルに基づいて建築物設計の知識を建築物設計者自身が整理し記述するための仕様記述言語を規定した．また，その言語で記述された仕様から，設計支援システムの生成を試みた．

この言語は，設計対象建築物の構造を記述するための構造記述言語と，構造の操作を記述するための構造操作記述言語からなる．また，設計支援システム上で部材を表示する方法などを記述するため，表示記述言語も用意した．次に，それぞれについて説明する．

4.1 構造記述言語

構造記述言語は，建築物の構成要素である部材を記述する．各部材の記述は，オブジェクト指向方法論のオブジェクトクラスの定義である．設計支援システム実行時に，このクラス定義から各部材のオブジェクトインスタンスが生成され，配置される．

建築物記述言語で記述するものは以下のようなものがあげられる．

- 部材名
- 存在の従属性のための依存関係
- 属性 (名前，型，範囲など)
- 属性の従属性のための属性間の制約 (関係，条件など)

次にそれぞれについて詳しく述べる．

部材の記述　この記述を次のように規定する．

```
PART < 部材名 >
        :
ENDPART
```

< 部材名 > にその部材の種類の名前を記述し，PART から ENDPART の間に，その部材の接続関係や部材が持つ属性名，属性間の制約等を記述する．

存在の従属性のための依存関係の記述　ある部材が存在するために必要な部材を指すものをプラグと呼び，それと対となるものをソケットと呼ぶ．これらは，存在の従属性のための依存関係を記述するために用いられるが，他の部材の属性を参照するためにも用いられる．その記述方法はそれぞれ次のようになる．

```
PLUG
   < プラグ名 > INTO < 依存部材名 > :: < ソケット名 > ;
           :
END
```

```
SOCKET
   < ソケット名 > TAKE < 被依存部材名 > :: < プラグ名 > ;
           :
END
```

設計支援システムの実行時に，部材オブジェクトが存在するためには，ここで記述したすべてのプラグがソケットにつながっていなくてはならない．逆に，ソケットはプラグに必ずしもつながっていなくてもよい．

属性の記述　部材の持つ属性は次の二つに分けられる．

- **固有属性** … 建築物設計のときに設計者が設定する属性．初期値を持つ．
- **従属属性** … 固有属性や，すでに値が決まった従属属性によって，値が導き出される．

固有属性の例として，部材の色や材質，基準点からの変移などがあげられる．これは，設計支援システムの動作時に，設計者が直接設定したり，構造の操作によって変更されたりする．

属性は次のように記述する．

```
ATTRIBUTE
   < 属性名 > < 型 > DEFAULT < デフォルト値 > ;
           :
   < 属性名 > < 型 > ;
           :
END
```

デフォルト値の記述がある場合は，固有属性とみなされる．デフォルト値が一定で不変である場合は，DEFAULT の代わりに CONSTANT とする．

属性間の関係　この記述は次のようになる．

```
RELATION
    ＜属性間の関係を表す式＞
                :
END
```

＜属性間の関係を表す式＞は代入式である．また，一意に値が決まらないのは仕様記述ミスとする．これは，＜属性間の関係を表す式＞の集合からデータフローを構成するためである．このデータフローから，属性の従属性の自動計算を行なえるような，属性参照メソッドを生成する．

4.2 構造操作記述言語

構造操作記述言語は，構造記述言語で記述された部材の操作手続きを記述するものである．また，GUIと深く関わっているのでGUI用の記述も含んでいる．前節の構造記述言語では，部材クラスの静的記述を行なったが，この構造操作記述言語は，部材の生成や，削除方法などの操作の手続きといった動的記述を行なう．筆者らは，構造設計での手続きを記述するための言語の規定を行なった．

構造操作記述言語は，MENUとOPERATIONの2種類の手続き記述からなる．

- MENU … GUIから呼び出され，実行すべき一連の手続きを記述したもの．
- OPERATION … MENUもしくはOPERATIONから呼び出され，前提条件と実行すべき一連の手続きを記述したもの．

MENU及びOPERATIONは次のように記述する．

```
MENU ＜MENU名＞
    ＜手続き記述＞
ENDMENU
```

```
OPERATION ＜OPERATION名＞
    ＜前提条件記述＞
    ＜手続き記述＞
ENDOPERATION
```

OPERATIONの＜前提条件記述＞は省略できる．次に，前提条件記述及び手続き記述について説明する．

前提条件記述　一連の手続きが呼び出すのに前提となる条件を記述する．また，手続きで用いられる変数の定義及び代入もここに含まれる．前提条件記述は，順不同に記述することができ，同一の変数への代入は一回限りとし，定義された時に代入を行ない，変更できない．条件文が失敗した場合，もしくは変数代入すべきところで失敗した場合は，手続きを実行しない．前提条件が失敗する時の例として，構造操作で主柱を指定すべきところを，他の部材を指定した時や，梁を張るために二本の主柱を指定するが，それぞれの主柱の階が異なるときなどがあげられる．

前提条件は次のように記述する．

```
PREMISE
    ＜条件記述＞;
                :
THEN
```

＜条件記述＞は条件文と変数定義文とに分けられる．

条件文　これは，比較演算式，論理演算式及び以下の真偽を返す関数などから構成される．

```
EXIST( ＜変数名＞ )
LINKED( ＜変数名＞ -> ＜ソケット名＞ )
```

EXIST(＜変数名＞)は，変数名に代入されている部材が存在するなら真を返す．＜変数名＞が集合である場合は，空集合なら偽を返し，そうでないなら真を返す．また，LINKED(＜変数名＞ -> ＜ソケット名＞)は，部材変数名に代入された部材オブジェクトのソケットにプラグが接続されているなら真を返す．＜ソケット名＞が複数個のプラグを受ける場合は，それが一つもプラグと接続されていない場合に，偽を返す．

変数定義文　次のように記述する．

```
＜変数名＞: ＜型＞ := ＜式＞
```

＜型＞には，部材名や整数を表すINTなどが入る．このとき，部材名を複数列挙することができ，その時はリストの形で記述する．

手続き記述　前提条件部で定義された変数を用いて，構造操作の手続きを記述していく．手続き記述は，手続きの最小単位の関数を手続き的に記述していく．最小単位の関数は次のように分類することができる．

部材の生成，削除　部材の生成は次のように記述する．

```
NEW ＜変数名＞: ＜部材名＞;
```

この関数が呼び出されると，型が＜部材名＞のオブジェクトが新しく生成され，＜変数名＞に代入される．この関数以降，ここで定義された＜変数名＞を使うことができる．また，ここで生成された部材のプラグはまだソケットに接続されていないので，次の項で説明する関数を用いて接続しなくてはならない．構造操作の手続きが終了した時点で，このプラグが接続されていない場合は，エラーとなる．

部材の削除は次のように記述する．

```
DELETE ＜変数名＞;
```

この関数が呼び出されると，変数に代入されているオブジェクトが削除される．この関数を呼び出す前に，次の項で説明する関数を用いて，全てのソケットから接続されているプラグを外しておかなければならない．削除しようとしている部材のプラグは，自動的にすべて外される．また，この関数以降，＜変数名＞は他の関数で用いることはできない．

プラグ - ソケットの接続　部材の生成の後，部材削除の前，また，部材の接続関係の組換え時に，プラグ - ソケットの接続に関する関数が呼び出される．

プラグをソケットに接続する場合は次のように記述する．

```
PLUGIN( <変数1> -> <プラグ名>,
        <変数2> -> <ソケット名> );
```

<変数1> は，依存する部材が代入されている変数名，<変数2> は依存される側のる部材が代入されている変数名である．プラグに他のソケットが接続されている場合，または集合でないソケットに他のプラグが接続されている場合に，この関数を呼ぶとエラーとなる．

プラグをソケットから外す場合には次のように記述する．

```
PLUGOUT( <変数> -> <プラグ名> );
```

外そうとするプラグがすでに接続されていない状態でも，エラーにはならない．また，部材の全てのプラグを対応するソケットから外す場合には，次の関数を呼び出す．

```
ALLPLUGOUT( <変数> );
```

以上の関数で，変数にオブジェクトが代入されていない場合や，部材にないプラグ名やソケット名を指定した場合はエラーになる．

属性の変更　属性の値を変更する場合には，次のように記述する．

```
CHANGE_ATTRIBUTE( <変数> -> <属性名>,
                  <値> );
```

ただし，変更できるのは固有属性だけで，従属属性を変更することはできない．

手続き呼び出し　他の OPERATION を呼び出す場合には，次のように記述する．

```
CALL( <OPERATION名>, <引数リスト> );
```

<引数リスト> は，OPERATION に引数を渡すために用いる．引数が必要ない OPERATION の場合は，空リスト [ ] を記述する．また，呼び出される OPERATION の前提条件部の引数参照用関数 INPUT(n) は，引数リストの先頭から n 番目をその値として返す関数である．

分岐，繰り返し　分岐のための関数の記述は，次のようなものがある．

```
BRANCH
    <分岐記述1>
    <分岐記述2>
        :
END
```

```
FORK
    <分岐記述1>
    <分岐記述2>
        :
END
```

その中の <分岐記述> は，

```
ACTION
    <前提条件記述>
    <手続き記述>
END
```

であり，前提記述は省略可能である．

BRANCH は，<分岐記述> の前提条件を記述した順に調べていき，最初に前提条件が真となる分岐を一つだけ実行する．一方，FORK は，全ての <分岐記述> の前提条件を順に調べていき，前提条件が真になる分岐を全て実行する．

繰り返しは OPERATION の再帰呼び出しで実現可能であるので，繰り返しのための関数は用意していないが，集合に対して同一の手続きを呼び出すための関数は用意してあり，次の様のように記述する．

```
FORALL( <集合要素変数名> : <集合変数>,
        BEGIN <関数1>; <関数2>; … END );
```

GUI　構造操作は，設計支援システムのユーザとの対話が基本となっており，手続き中にその対話が必要な時もある．現在用意しているものは，

```
GETPART( <変数名> : <型> );
GETPART( <変数名> : <型リスト> );
GETVALUE( <変数名> );
MESSAGE( <メッセージ> );
```

である．GETPART は，<型> もしくは <型リスト> のいずれかの部材のオブジェクトを一つ，ウィンドウ上でユーザに特定してもらい，<変数名> に代入するものである．GETVALUE は，ウィンドウ上でユーザに値を入力してもらい，<変数名> に代入するものである．GETPART もしくは GETVALUE が呼び出された以降，<変数名> を手続きで用いることができる．MESSAGE はウィンドウ上に <メッセージ> を出力するためのものである．

5　実装

筆者らは，建築物の設計支援システムを実際に動かすため，前章の言語で記述された仕様から，設計支援システムのプログラムを生成するためのコンパイラを開発している．現在はシステムの試作段階であり，プログラムの修正が容易であることから，目的言語は SICStus Prolog を用いている．また，オブジェクトの管理には SICStus Objects ライブラリを用いており、GUI 部には Graphics Manager (GM) パッケージを用いている．

本システムの構成は，構造管理部，構造編集部，表示管理部，GUI 部に分けられる．

構造管理部　SICStus Prolog のオブジェクトライブラリを用いており，OMT(Object Modeling Technology) に従ったモデルを prolog 上に実現している．

構造管理部は，一つの部材について，2つのオブジェクトを用意している．2つのオブジェクトのうちの1つは，その部材クラスにおいて不変の情報を持つもので，部材クラスのコントロール部と呼ぶ．もう一方は，その部材クラスがインスタンスを生成する際の原型となるオブジェクトであり，部材クラスのデータ部と呼ぶ．

また，プラグ-ソケットの関係にもクラスが定義されており，接続がなされた場合，関係オブジェクトを生成する．

部材の属性値を求める際は，世代管理を用いた要求駆動のメソッドを用いる．属性値を求める度に属性値を再計算していては時間がかかるため，このメソッドは，世代が変わっていなければ，各インスタンスが持っている一番最近求めた属性値をそのまま返す．

構造編集部　構造操作記述言語のMENUに対応する述語callBackと，構造操作記述言語のOPERATIONに対応する述語operationの2種類の述語群から構成される．

述語callBackのボディは，構造管理部のObjctID ::constructやObjctID ::getAttributeなどの述語呼びだしで構成される．述語operationのボディの前部は，前提条件に相当する述語呼び出しの組によって構成し，後部は，構造に対する操作の手順通りの述語呼び出しの組によって構成する．前提条件が満たされない場合は，次の操作の手順は実行されず，述語operationは失敗する．

表示管理部　平面図上に表示すべき図面を，構造管理部で用意している述語を用いて生成し，GUI部に渡す．また，GUI部でユーザが画面上で選択した座標から部材オブジェクトを特定する．

GUI部から来る上記の要求は，次の述語を介して処理する．述語getFigureは，基準線オブジェクトのIDを第一引数に持ち，そのウィンドウに表示する図形のリストを第二引数にユニファイする．述語getPartObjは，基準線オブジェクトのIDや，ユーザが指した座標を入力引数とし，選択された部材オブジェクトをユニファイして返す．

GUI部　XYZ各平面図の表示を管理するクラス(plane)と，GUI部全体を管理するオブジェクト(windowManager)からなる．

ユーザがplaneオブジェクトのウィンドウ上の構造操作のメニューを選択すると，windowManagerは，構造編集部の述語callBackを呼び出し，構造に対する操作を行なう．planeオブジェクトが図面を描画するときや，画面上でユーザが部材を選択するするときは，windowManagerを介して表示管理部の述語getFigureやgetPartObjを呼ぶ．図4は，編集時の表示例である．

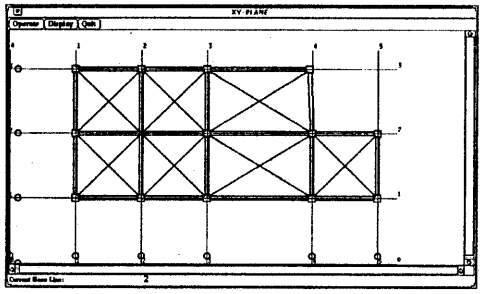

図 4: GUIの表示例

6 考察

6.1 概念モデルと仕様記述言語

ソフトウェアの生産性や信頼性を向上するために，ソフトウェア開発の要求分析工程を確実に実施することが重要な課題となっている．この要求分析工程を支援するため，OMTなどのオブジェクト指向方法論や，各種のCASEツールが開発されてきた．これらの方法論やツールは，ドメイン知識の表し方が汎用性の高いものとなっている．

筆者らは，OMTのオブジェクトモデルと機能モデルを建築物のドメインに特化して汎用性を狭めた概念モデルを作成した．さらに，そのモデルに基づいて仕様記述言語を規定した．この仕様記述言語を以下に考察する．

構造記述言語　この言語は，適用範囲を狭めることによって，少ない記述量で表せる最小性を重視している．また，思考に現れる概念と言語仕様が，おおよそ一対一に関係しているので，記述性，理解性が良いと考えられる．

形式性の例として，プログラムの自動生成や仕様の検証がある．この記述言語は信頼性と生産性を向上するためにソフトウェアの自動生成を前提としている．筆者らは，この言語で記述した仕様から目的言語であるprologのソースプログラムを生成した．このジェネレータが，プラグ-ソケットの関係や，属性間の関係など仕様記述に矛盾がないかをチェックする．

この記述言語は，適用領域を建築物設計の領域に特定することによって，適用性を狭め，最小性や記述性，拡張性を増している．記述性や理解性を増すことで，設計者自身が仕様を記述することが期待される．そのため，専門領域の知識を獲得することが容易になる．これらのことから，ドメインに依存した仕様記述言語を規定し，ドメインの専門家に知識を記述させることは，仕様の要求分析に有効な手段と考えられる．

構造操作記述言語　構造操作記述言語は，構造記述言語で記述した建築物モデルに対する操作を記述するためのも

のである．今回は，構造設計を行なうための，構造操作記述言語を規定した．この言語は，建築物モデルの部材の生成削除や，属性の一括変更など定型的な設計手法を記述することができる．プラグ-ソケットの接続などの最小単位の操作がPrologの一つの述語と対応しており，その呼び出す順序が重要となっているので，この記述言語は手続き型である．

構造操作を行なう前の状態の概念と，行なった後の状態の概念を用意することによって，構造操作記述言語は宣言型にできる可能性がある．

6.2 実装

本システムの内部モデルは，将来の拡張に対応できるように，一般的なモデルであるOMTのオブジェクトモデルと機能モデルに基づいている．

性能においては，処理時間に問題がある．GUI上に部材を表示するときや，操作を行なうときの，構造の内部モデルに対する部材オブジェクトの参照方法が原因となっている．

例えば，二階の平面図を表示させる場合，表示のために内部の全部材オブジェクトにその平面上で表示可能かをチェックし，表示可能の際に図形を得て表示させる．表示画面が複数ある場合，このような動作を個々に行なう．また，構造に対する操作の際も，ある条件に見合う部材オブジェクトを得る時，内部の全部材オブジェクトに対して，条件をチェックしていくことがある．

このような問題を解決するために，幾つかの部材からなる複合部材クラスの概念を取り入れる方法が考えられる．例えば，各階を表す複合部材クラスや，ある基準線の断面図を表す複合部材クラスなどである．これによって，各表示平面図は，それぞれに対応する複合部材オブジェクトを構成する部材オブジェクトに対してのみ，表示用の図形を得ればよいようになる．

このような複合部材クラスは，構造計算の時にも用いることができる．例えば，建物が地震や強風などの水平の力に耐えられるかを調べる時，各階毎に計算する必要がある．また，部屋や，廊下といった概念を取り入れるために，このような複合部材クラスの概念を用いることができる．今後は，このような複合部材クラスの概念を取り入れることができるように，言語及び処理系を拡張しなければならない．

6.3 今後の課題

今後の課題としては，次の項目があげられる．

1. 仕様記述言語の適用実験
2. 意匠概念の導入
3. 他の専門の支援システムとの統合

1については，実際の建築物設計者と研究を行なわなければならない．2については，部屋や廊下といった意匠概念を取り入れ，概念の拡張や処理の高速化を図る．3については，各設計作業で用いられる設計情報を一元化するために構造モデルを拡張し，それに基づいて言語を拡張することが必要である．

7 まとめ

筆者らは，OMTのオブジェクトモデルと機能モデルを建築物のドメインに特化して汎用性を狭めた概念モデルを作成した．また，この概念モデルを記述する仕様記述言語を規定した．この言語は，思考に現れる概念と言語仕様が，おおよそ一対一に関係しているので，記述性，理解性が良いと考えられる．

この言語で記述した仕様から，設計支援システムを自動生成するコンパイラの試作も行なった．生成されたシステムは，OMTのオブジェクトモデルと機能モデルに基づいた内部モデルをprolog上に構築したものである．しかし，現在の内部モデルは部材一つずつを単体のオブジェクトとして取り扱うため，部材表示などにかなりの時間を要している．この問題を解決する一つの方法として，幾つかの部材からなる複合部材クラスの概念を取り入れる方法が考えられる．このような複合部材クラスは，構造計算の時にも用いることができる．

今後，概念モデルに部屋や廊下といった意匠概念を取り入れ，各設計作業で用いられる設計情報を一元化するために構造モデルを拡張しなければならない．また，仕様記述言語を建築物設計者が直接試用して評価しなければならない．

参考文献

[1] 廣田豊彦, 佐藤俊孝, 橋本正明, 長澤勲, 手越義昭. オブジェクトモデルに基づいた建築物設計支援システム. 第16回情報・システム・利用・技術シンポジウム, pp. 19–24, 12 1993.

[2] 廣田豊彦, 千原博司, 佐藤俊孝, 橋本正明. 要求分析のための概念モデル記述言語に関する一考察. 電子情報通信学会技術研究報告（知能ソフトウェア工学）, Vol. 94, No. 130, pp. 1–8, 7 1994.

[3] 佐藤俊孝, 千原博司, 廣田豊彦, 橋本正明. 建築物設計支援のための概念モデル. 電子情報通信学会技術研究報告（知能ソフトウェア工学）, Vol. 93, No. 407, pp. 25–32, 1 1994.