応用ドメインに特化した概念モデル記述言語に関する一考察

廣田豊彦† 橋本正明† 長澤勲††

知的設計支援を行うためには，設計対象の知識を分析，整理し，記述することが不可欠である．この設計知識は設計の専門家が暗黙の形で持っているものであり，知識記述言語は，その知識を専門家が自ら整理して記述できることが望ましい．しかし，ドメインの専門家は言語設計の知識は持たず，知識工学者やソフトウェア工学者はドメインの知識を持っていない．著者らはこの3者が協調して知識ベースシステムを開発するプロセスをすでに提案した．本研究もそのプロセスの試行事例であり，建築設計を対象として，属性モデリングに基づく概念モデル記述言語 BDL を規定した．我々は建築物が部材の接続関係によって定義されることに着目し，その接続関係を直接的に表現するためにプラグとソケットという概念を BDL に用意した．さらに BDL では，部材の存在従属性や属性従属性などに関する設計知識を，部材の接続関係上で容易に表現できる．本論文では，BDL を仕様記述言語 PSDL やプログラミング言語 C++と比較するために記述実験を行った．この実験の結果，応用ドメインに特化した記述言語では，ドメインモデルの構造が言語に反映されており，制約記述や計算機構もドメインモデルに最適なものが選択されていることで，記述言語の最小性，記述性と理解性，拡張性，形式性などが高まっていることが明らかになった．

A Discussion on Conceptual Model Description Language Specific for an Application Domain

Toyohiko Hirota,† Masaaki Hashimoto † and Isao Nagasawa††

To realize intelligent CAD, it is essential to analyze, systematize, and describe the knowledge of design objects. Such design knowledge is owned in implicit form by design experts, and knowledge description language should be used to systematize and describe their own knowledge by themselves. However, domain experts do not have language design knowledge, and knowledge engineers and software engineers do not have domain knowledge. We have already proposed a development process in which these three parties collaborate to build a knowledge-based system. In this research, as an experimental case of our development process, we have proposed a conceptual model description language BDL specific to architectural design based on attribute-based modeling. We have perceived that a building is defined with conjunctive relation of its parts, and have prepared the concept of plug and socket in BDL. In BDL, design knowledge such as existence dependency and attribute dependency between parts can be easily described with respect to this conjunctive relation. In this paper, we have experimented to compare BDL with specification description language PSDL and programming language C++. Our experiment has shown that a description language specific to an application domain has the specific structure to the domain model and has the constraint description and computation mechanism optimized for the domain model. These specializations increase the minimality, constructibility and comprehensibility, extensibility, formality of the description language.

1. はじめに

知的設計支援システムを開発するためには，設計対象の知識を分析，整理し，記述することが不可欠である．これに対して従来のソフトウェア工学では，OMT (Object Modeling Technique)[1] などのオブジェクト指向方法論や，各種の CASE(Computer Aided Software Engineering) ツールなど，汎用性の高いものが提案され，使用されてきた[2]．

設計対象の知識は，そのドメインの専門家が直接記述することが望ましいが，ドメインの専門家が上述のような汎用性の高い方法論やツール，言語を使いこなすことは容易ではない．そこで，知識を記述するため

† 九州工業大学情報工学部知能情報工学科
Department of Artificial Intelligence, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology
†† 九州工業大学情報工学部機械システム工学科
Department of Mechanical Systems Engineering, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology

の概念モデル記述言語をドメインに特化し，ドメインの専門家に使いやすいものにする必要がある[3]．

しかし，ドメインの専門家は言語設計の知識は持たず，また知識工学者やソフトウェア工学者はドメインの知識を持っていない．このため，それぞれが単独ではドメインに特化した言語は設計できない．そこで，著者らはこの3者が協調して知識ベースシステムを開発するプロセスを，4フェーズ，すなわち協力体制確立，プロトタイプ開発，実用システム開発，最適化のフェーズに分けることを提案した[4]．その上で建築設計のドメインを事例としてCAD実験システムを開発中である．

汎用性の高い概念モデルを，建築専門家から獲得した知識を用いて，建築物の構造を表すための概念モデルに特化し，それをプログラミング言語C++で実現した第1フェーズはすでに終了した．現在，第2フェーズを実施中であり，その特化された概念モデルに基づいて，建築物の構成要素である部材や，その依存関係，部材の各属性間の制約を，建築設計の知識として言語BDL(Building part Description Language)を規定した．

本論文の目的は，この新しく規定したBDLを，汎用性の高い概念モデルを適用した仕様記述言語PSDL[5]およびプログラミング言語C++と比較・評価し，その特徴を考察することである．現在あらゆる分野で知的設計支援システムに対する開発要求がある．そのようなシステム開発の生産性や信頼性を向上させるためには，応用ドメインに特化した概念モデル記述言語が重要性を増してくると考えられるが，従来，その特徴を考察した論文は見当たらない．以下，2章では本研究の関連研究について述べる．3章では建築設計の特性と，概念モデルについて述べ，4章では概念モデル記述言語を説明する．5章では評価のための記述実験の方法を述べ，6章で考察を行う．

2. 関連研究

2.1 概念モデル

設計知識を記述するための枠組として，形式モデルと，非形式的な概念モデルとを考えることができる[6]．ドメイン分析の結果として形式モデルに基づく記述が得られれば，その意味論に基づいて自動的にプログラムを生成することが可能であるが，そこまで厳密な記述を作成するのは容易ではない．そこで通常は概念モデルが用いられる．

概念モデルとして従来からよく利用されたのは，**データフローモデル**[7]である．ビジネス分野の応用では，対象となるお金，伝票，ものの流れが比較的はっきりしているので，それらをデータフローとして捉えることができる．しかし，CADを始めとして，多くのドメインでは，データの流れ自体がそれほど明確ではない．そこでドメインの概念やものに着目したモデルが，**ERモデル**[8]や**オブジェクト指向モデル**[1),9]である．これらのモデルではエンティティ(またはオブジェクト)とそれらの相互関係を中心として対象を記述する．データフローについては，1) データフローモデルを別に記述する[1]，2) 属性間の制約を記述する[10]，などのアプローチがある．

2.2 CAD

従来CADは**形状モデリング**を中心に開発がすすめられてきた．しかし，形状モデリングでは設計者を支援することができないことが明らかになり[11]，設計者にとってのドメインモデルが研究されるようになってきた[12),13]．

吉川は，必ずしもCADに限定されない形で設計を理論的に扱うための根拠として**一般設計学**[14),15]を提唱した．一般設計学では，設計対象を属性の集合として捉え，それが実体集合上で位相空間を構成するものとした．そして，設計を位相空間における操作として定式化した．

設計の分野においては，従来から定型設計や改良設計として，過去の設計の再利用が行われてきた．しかし，従来のCADはこのような再利用を支援することができなかった．そこで，Chandrasekaranら[16]は，定型設計を記述するために**汎化タスク**(generic task)の概念を導入した．

一方，長澤は様々なドメインの設計を調査し，設計者のドメインモデルとして，**属性モデリング**[11]を提唱した．長澤らのグループはこの属性モデリングの考え方に基づいて，生成検証を導入にした設計支援システムDSP[17]，拘束条件解法を用いた公差解析システム[18]，3次元モデラを導入した熱交換器設計支援システム[19]などの開発をすすめている．

このほかにも類似の様々なモデルが提唱されている．一例として，伊藤は認知科学的視点を導入して，**メンタルモデル**を提唱し，機械設計を支援するCONMOTOシステム[20]を開発している．

2.3 建築用CAD

建築用CADについても，オブジェクト指向の考え方を取り入れたモデリングが研究されているが，まだ実用的なものはない．Sriramらはコンカレント・エンジニアリングを支援する環境DICE[21]を開発しており，プロトタイプとして意匠設計と構造設計のモジュ

表 1 知識ベースシステムの開発サイクル
Table 1 Development cycle of knowledge-based system.

フェーズ	1	2	3	4
目標	協力体制の確立	プロトタイプの開発	実用システムの開発	最適化
ドメインの専門家の役割	知識表現の勉強	小規模例題の作成	知識ベースの開発	知識ベースの発展
知識工学者の役割	ドメインの勉強	知識ベースの調査	知識記述の確立	
ソフトウェア工学者の役割		要求分析と仕様化	知識記述言語と処理系の開発	品質，性能，保守性の改善

ールを開発している．平沢ら[22]は部品概念をPrologによるフレームとして実現している．渡辺ら[23]は建築要素が外延，属性，内包など，様々な視点から認識されることに着目し，それらを表現するためのモデルを提案し，それに基づくCADシステムを開発している．

これらの研究はいずれもフレームワークの提案であり，専門家が，1）どのような知識を，2）どのように記述するのか，についての具体的な議論は見当たらない．

2.4 エキスパート支援システム開発手法

CADシステムのようなドメインのエキスパートを支援するシステムを開発するには，ドメイン・エキスパートと知識工学者，ソフトウェア工学者の3者の密接な協力が不可欠である．著者らは，エキスパートを支援する知識ベースシステムの開発サイクルとして**表1**[4]を提案している．本論文は，この開発サイクルにおけるフェーズ2の要求分析と仕様化に関する研究を行ったものである．

3. 建築設計のためのドメインモデル

著者らは以前に建築設計の特性を分析し，建築設計を統合的に支援するCADシステムとして，IBDS (Integrated Building Design System)[24]を開発した．本研究で規定した言語BDLは，IBDSの建築部材モデルを採り入れ，ドメインの専門家でも容易に記述できるような言語として実現したものである．ここではIBDSの開発に際して，著者らが着目した建築設計の特性と，それに基づく建築部材モデルについて述べる．

3.1 建築設計の特性

建築設計は，構造設計や意匠設計などの専門の設計作業に分業化されているが，それら専門の設計作業は完全に独立に行えるわけではない．たとえば，意匠上の理由で柱が1本削除されたとすると，構造上は何らかの対策を講じる必要がでてくる．したがって，意匠の概要が決定した後に，構造の基本設計が行われるなど，適切な手順に従って設計がすすめられる必要がある．しかし，建築設計においては，設計変更が行われることもまれではなく，その場合に専門の設計作業相互の一貫性を保つことは容易ではない．

従来から建築設計の一部分を支援するCADは様々なものが開発されてきたが，上述のような一貫性を支援することはできない．そこで統合的な建築設計支援システムが必要とされているが，そのようなシステムでは，1）建築オブジェクトを一元管理する，2）それぞれの専門の設計者に適切なドメインモデルを提供する，ことが要求される．

建築物の形状は3次元であるが，設計者がいつでも3次元を意識して設計をすすめるわけではない．一般の建築物では，X, Y, Z, それぞれの方向の基準線ならびに補助基準線上に部材を配置し，そしてすでに配置されている部材に相対的にさらに部材が配置されて，設計がすすめられる．したがって，建築オブジェクトは，1）部材の集合として扱われる，2）部材は基準線や他の部材との相対的位置関係によって配置される，などの性質をもつ．

建築設計を知的に支援するためには，設計対象に関する知識や，設計作業に関する知識を整理し，記述しなければならない．ここでは，そのような知識の一例として，**存在従属性**と**属性従属性**を取り上げる．

存在従属性とは，ある部材が存在するためには，それに接続されている別の部材が存在しなければならないことを言う．逆にある部材が削除されると，それに依存する部材も削除されなければならない．たとえば，梁はその両端の柱に依存し，柱は下の階の柱に依存する．**図1**に示す例で，柱1が削除されると，梁a，梁b，柱2が削除され，さらに梁c，梁dが削除されることになる．

属性従属性とは，ある部材の属性が別の部材の属性

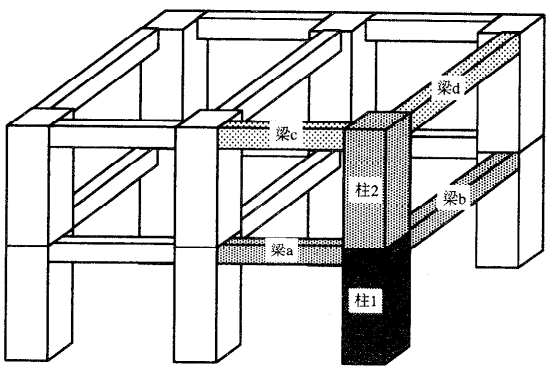
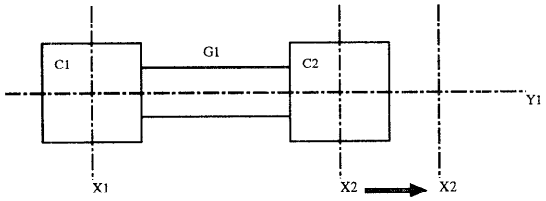
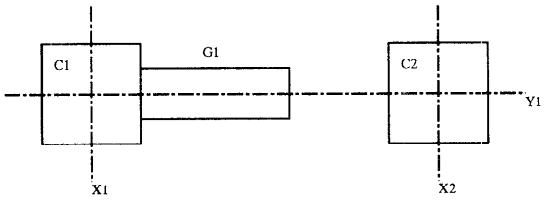


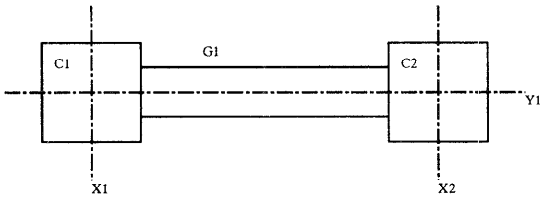
図 1 存在従属性
Fig. 1 Existence dependency.



（1） 基準線の移動



（2） 主柱の移動



（3） 梁の伸長

図 2 属性従属性
Fig. 2 Attribute dependency.

の値に依存することを言う．たとえば，**図2**において，まず基準線X2を移動すると，柱C2のX座標が再計算され，柱C2が移動される．そして今度は梁G1の長さが再計算され，梁G1が柱C1から柱C2に架かるように調整される．このように属性の従属性を維持するためには，ある属性が変更されたときに，それに依存する属性を再計算しなければならない．

3.2 建築部材のドメインモデル

IBDSは属性モデリング[11]の考え方に基づいており，各部材は属性の集合として扱われる．これはソフトウェア工学におけるオブジェクト指向に対応しており，各部材は一つのクラスとして記述され，部材の属性はそのままクラス属性となる．各部材に対しては，構造設計，意匠設計など，それぞれの専門の設計ごとに必要な属性があるが，それらをすべて一つのクラスの属性として統合的に扱う．一般にクラス属性は，外部からはクラスメソッドを通じてのみアクセスされる．そこでそれぞれの専門の設計に必要なメソッドを準備することによって，設計者は直接関係のない属性やメソッドを気にすることなく設計を行うことができる．すなわち，メソッドの集合によって，適切な概念モデルを提供することができる．

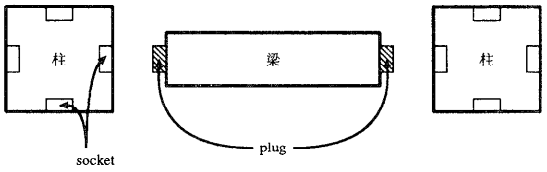


図 3 部材の接続関係
Fig. 3 Conjunctive relation of parts.

部材の接続関係を表すために，IBDSでは**プラグ**と**ソケット**の概念を導入している．一例として，**図3**に柱と梁の接続関係を示す．梁の両端に柱と接続するためのプラグがあり，柱には梁を接続するために4方向のソケットがある．このプラグとソケットを接続することによって，柱と梁が接続されることになる．また，柱などは基準線上に配置されるが，それを統一的に扱うために，基準線も部材とみなす．基準線は任意の数のソケットを持つことができ，必要な数だけの柱を配置することができる．

プラグとソケットは接続関係だけではなく，存在従属性をも表している．すなわち，ソケットを持つ部材が削除されると，そのソケットに接続されているプラグを持つ部材も削除されることになる．一方，プラグを持つ部材が削除された場合でも，ソケット側の部材は直接的な影響は受けない．

属性従属性も，プラグとソケットの接続関係に依存する．具体的にどのように属性が他の属性に依存するかはそれぞれの属性によって異なるので，属性の従属性は，クラスのメソッドあるいは従属性制約（4章参照）で記述する．

以上のような建築部材モデルの記述例として，基準線，柱，梁をOMT[1]のオブジェクト図* で表現したものを**図4**に示す．

4. 概念モデル記述言語

本論文の実験に用いた言語BDLとPSDLについて説明する．

4.1 建築部材記述言語 BDL

建築設計の専門の設計作業のうち，構造設計の分野について，部材の仕様を記述するために建築部材記述言語BDL(Building parts Description Language)を規定した．BDLは，建築物の構成要素である部材の接続関係や，部材の各属性間の制約を，建築設計の知識として記述するための言語である．

その記述は以下の四つの部分から構成される．

* これで建築部材モデルを完全に表現できるというわけではない．

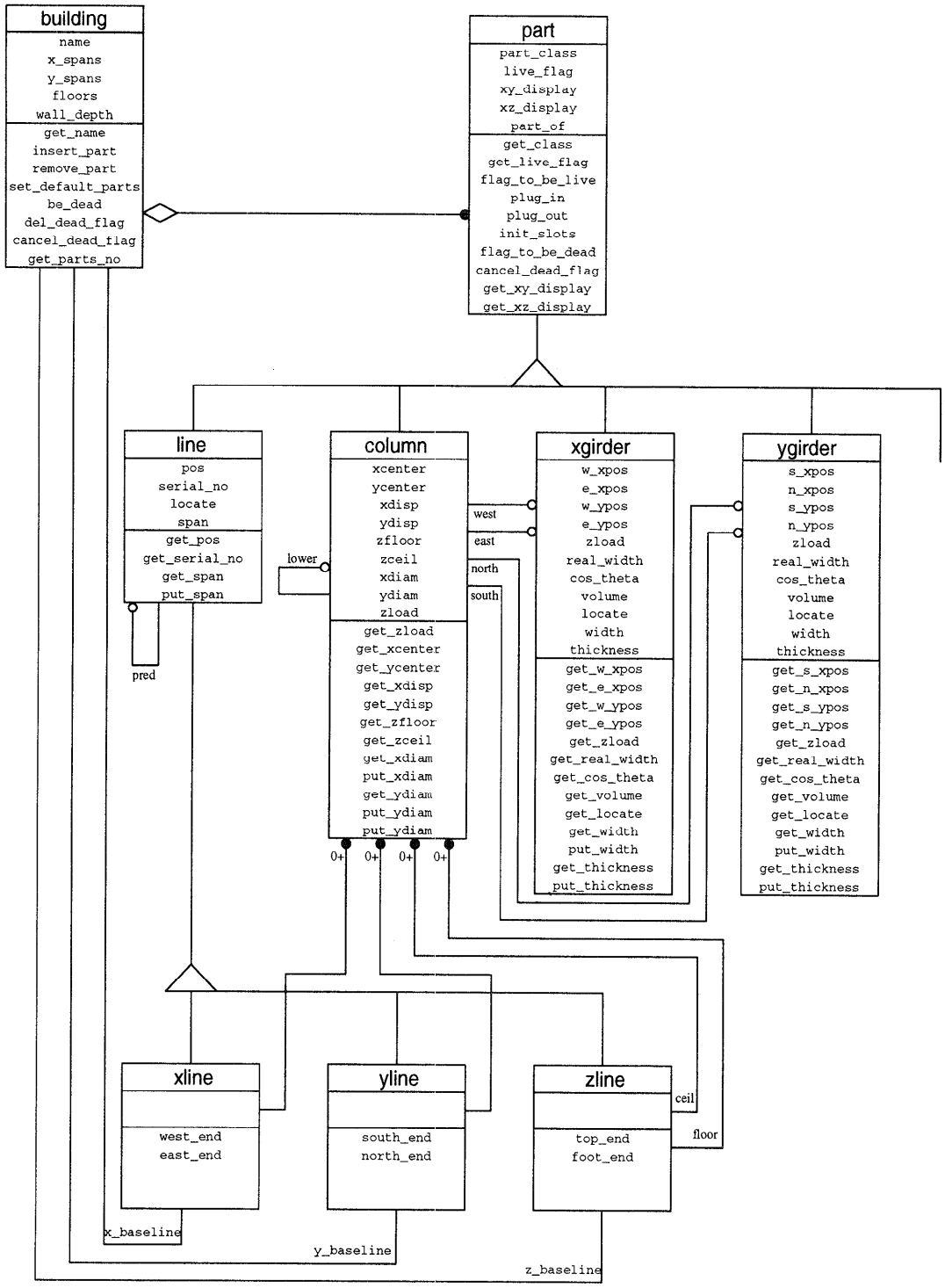


図 4 部材のオブジェクトモデル
Fig. 4 Object model of parts.

4.1.1 部材名宣言部

部材名を宣言する．継承する部材がある時は，その部材名を明記する．

```
PART 〈部材名〉［(継承部材名)］
        :
ENDPART
```

4.1.2 接続関係記述部

部材間の接続関係を記述する．これは同時に部材間の存在従属性も表す．また，属性従属性を表すときに他の部材の属性を参照するためにも用いられる．

```
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
```

4.1.3 属性宣言部

ここでは，その部材のもつ属性を記述する．属性には，建築設計の時に設計者が変更する独立属性(user-specified 属性)と，すでに値の決まった他の属性から自動的に値が導き出される従属属性(derived 属性)の2種類がある．

```
USER-SPECIFIED
〈属性名〉〈型〉〈デフォルト値〉
        :
END
 DERIVED
 〈属性名〉〈型〉
      [WHERE〈属性間制約〉]
        :
END
```

〈属性間制約〉はその属性に対する属性従属性を表す関係式であり，つぎの制約記述部で記述することもできる．

4.1.4 制約記述部

属性従属性の記述は以下のようになる．

```
CONSTRAINT
〈属性間制約〉
        :
END
```

```
//
// column class (BDL)
//        柱

PART column ( part )
 PLUG
  xbase_xline INTO xline::connect_col;
  ybase_yline INTO yline::connect_col;
  floor_zline INTO zline::connect_col;
  ceil_zline INTO zline::connect_col;
  lower_col INTO column::upper_col;
 END
 SOCKET
  upper_col TAKE column::lower_col;
  west_xgird TAKE xgirder::east_col;
  east_xgird TAKE xgirder::west_col;
  north_ygird TAKE ygirder::south_col;
  south_ygird TAKE ygirder::north_col;
 END
 USER_SPECIFIED
  xdiam INT DEFAULT default_xdiam; // X方向の柱の幅
  ydiam INT DEFAULT default_ydiam; // Y方向の柱の幅
  live_flag BOOLEAN DEFAULT Live; // Live or Dead
 END
 DERIVED
  xdisp    INT // X方向の柱のずれ
   WHERE xdisp = IF xbase_xline-> xlocate = East_end
            THEN wall_depth / 2 - xdiam / 2
            ELSE BEGIN
                 IF xbase_xline -> xlocate = West_end
                 THEN xdiam / 2 - wall_depth / 2
                 ELSE 0
                END
           - - -
```

図 5 BDL 記述例
Fig. 5 Example of BDL description.

```
//
// column class(PSDL)
//        柱

INFORMATION
 E column //柱(エンティティ型)
  EN -100
 EC .col_is_created..create_col_cmd
 A old_xdiam NUM //x方向の柱の幅(属性)
  = NIL FROM col_is_created
  = NIL FROM col_is_deleted
 A new_xdiam NUM
  = IF exist(modify.col_is_modified.cmd.modify_col_cmd)
     THEN modify.column_is_modified.cmd.modify_col_cmd.xdiam
     ELSE IF exist(deleted.col_is_deleted.cmd.delete_col_cmd)
          THEN NIL
          ELSE IF exist(created.col_is_created.cmd.create_col_cmd)
               THEN Default_xdiam
               ELSE old_xdiam
               ENDIF
          ENDIF
     ENDIF
  - - -

 R xbase_is_connected //柱と基準線の接続(関連型)
  C xbase_xline.column
   RN -1
  C connect_col.xline
   RN -1
 R ybase_is_connected
  C ybase_yline.column
   RN -1
  C connetc_col.yline
   RN -1
  - - -
```

図 6 PSDL 記述例
Fig. 6 Example of PSDL description.

〈属性間制約〉は代入式ではなく関係式である．

BDL の記述例を図5に示す．

4.2 プログラム仕様記述言語 PSDL

プログラム仕様記述言語 PSDL(Program Specification Description Language)は，事務処理分野のソフトウェア再利用を目的とし，記述性，形式性，理解性，拡張性に重点をおいたプログラム仕様を記述するための言語である[5]．このため，概念データモデルの一つである ER(Entity-Relationship)モデルに，従属性制約を付加した形をとっている．

PSDL プログラム仕様は，プログラムの入出力データで表された情報の枠組を定める情報層，その入出力データの構造を記述するデータ層，入出力ファイルへのアクセス方法を定めるアクセス層から構成されている．しかし，概念モデルは情報層で用いられているので，情報層の記述のみを記述して，他の言語との比較に用いた．

本論文の実験で記述した PSDL 仕様のうち，柱クラスの一部を図6に示す．

5. 記述実験

BDL を評価するため，その対比として PSDL および C++をとりあげた．C++は著者らが開発した CAD 実験システム[25]のソースコードであり，これまでに述べた概念モデルが可能な限り反映されたものになっている．同一の建築物オブジェクト PSDL および BDL で記述し，C++のソースコードとも比較し，考察した．

記述対象である建築物のオブジェクトには，

- X 方向，Y 方向，Z 方向の 3 種類の基準線
- 柱
- X 方向，Y 方向の 2 種類の梁
- 建築物そのものを表すオブジェクト

がある．また，各オブジェクトは属性値を持ち，オブジェクトの生成とオブジェクトの消去，属性値の変更の 3 種類のコマンドがある．

純粋に概念モデルの記述に関して比較を行うために，C++の記述から画面表示用の属性とメソッドは除き，PSDL は情報層のみを記述した．

建築設計の知識を表す存在従属性と属性従属性が，それぞれの言語でどのように記述されるのかについて以下で説明する．

5.1 存在従属性

存在従属性は BDL ではプラグとソケットを記述するだけでよい．図7(a)は梁と柱の存在従属性の BDL による記述例である．PSDL では関連型を記述するが，

```
PART column( part )
  SOCKECT
    west_xgird TAKE xgirder::east_col;
  - - -
ENDPART
PART xgirder( part )
  PLUG
    east_col INTO column::west_xgird;
  - - -
ENDPART
```

(a) BDL

```
R west_xgird_is_connected
  C west_xgird.column
    RN -1
  C east_col.xgirder
    RN -1
  RC (east_col.xgirder:
        created.xgird_is_created.cmd.create_xgird_cmd.w_x_sn
      == west_xgird.column:
        xbase.xbase_is_connected.col.xline.new_sn)
  && (east_col.xgirder:
        created.xgird_is_created.cmd.create_xgird_cmd.w_y_sn
      == west_xgird.column:
        ybase.ybase_is_connected.col.yline.new_sn)
  && (east_col.xgirder:
        created.xgird_is_created.cmd.create_xgird_cmd.w_z_sn
      == west_xgird.column:
        ceil.ceil_is_connected.col.zline.new_sn)
```

(b) PSDL

```
class column {
  xgirder *east_gird;
  - - -
};
class xgirder {
  column *west_col;
  - - -
};
column::plugin( int plug, part *pt ) {
  switch( plug ) {
    case PLUG_EAST_GIRDER:
      east_gird = (xgirder *)pt;
      break;
    - - -
  }
}
x_girder::x_girder( column *w, column *e ) {
  west_col = w;
  west_col -> plugin( PLUG_EAST_GIRDER, this );
  - - -
}
```

(c) C++

図 7 存在従属性の記述例
Fig. 7 Example of existence dependency description.

実際にどのインスタンス間に関係があるのかは関連存在従属性制約で記述しなければならない．図7(b)に PSDL による柱と梁の記述を示す．RC が関連存在従属性制約であり，柱と梁が同じ X，Y，Z 基準線を参照

```
PART xgirder ( part )
  DERIVED
    w_xpos INT
      WHERE w_xpos = west_col->xcenter + west_col->xdiam / 2;
    - - -
ENDPART
```

(a) BDL

```
E xgirder
  A new_w_xpos NUM
    = IF exist(created.xgird_is_created.cmd.create_xgird_cmd)
         || (west_col.east_xgird_is_connected.east_xgird.column.new_xcenter
          != west_col.east_xgird_is_connected.east_xgird.column.old_xcenter)
         || (west_col.east_xgird_is_connected.east_xgird.column.new_xdiam
          != west_col.east_xgird_is_connected.east_xgird.column.old_xdiam)
       THEN west_col.east_xgird_is_connected.east_xgird.column.new_xdiam
       ELSE old w_xpos
       ENDIF
    - - -
```

(b) PSDL

```
xgirder::get_w_xpos( ) {
  if( w_xpos == NIL_INT )
    w_xpos = west_col->get_xcenter( ) + west_col->get_xdiam( ) / 2;
  return w_xpos;
}
```

(c) C++

図 8 属性従属性の記述例
Fig. 8 Example of attribute dependency description.

するときに両者を関係づけている．C++では，コンストラクタで部材の接続関係を表すポインタを設定し，デストラクタで自身に依存する部材を削除することになる．図7(c)に，梁が生成されたときに柱と接続するメソッドの一部を示す．

5.2 属性従属性

属性従属性はBDLでは関連する属性間に成立する等式を記述する．たとえば，X方向の梁の西側のX座標は，西側に接続されている柱の中心のX座標と柱の径に依存し，**図8**(a)のように記述される．PSDLでは属性値は単一代入であるので，属性値の更新を記述するためには，newとoldの二つの属性を用意する必要がある．図8(b)がBDLに対応するPSDLの記述を示している．C++では属性は必ずメソッドを介してアクセスされるので，そのメソッドの中で属性値を計算すればよい．図8(c)が対応するC++の記述である．ただし，ここには記していないが，独立属性を変更したときにそれに依存する属性の再計算を起動するための機構が別に必要である．

6. 考 察

6.1 言語の比較

前章で述べた実験の結果を分析して，最小性，記述性，理解性，拡張性，形式性，適用性の6個の評価基準[26]に従って，BDLとPSDL，C++を相互に比較する．

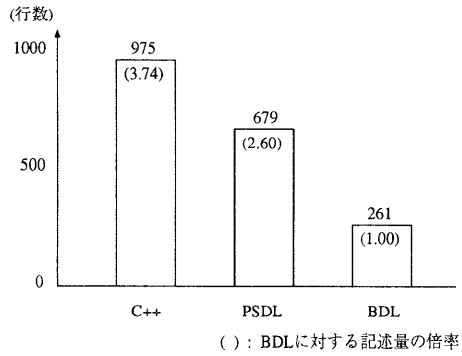


図 9 建築部材モデル記述量の比較
Fig. 9 Number of lines in building part model descriptions.

6.1.1 最 小 性

最小性とは，同じ内容の仕様をなるべく少ない行数で書き表すことができる性質である．

各言語の建築部材モデルの記述量は**図9**のようになった．同図から分かるように，BDLの最小性は他の言語と比較して，数倍優れている．この差の要因を以下に述べる．

プログラミング言語であるC++は，オブジェクト指向の考え方にのっとって情報隠蔽を行うため，オブジェクトの外部から属性へアクセスするのにメソッドを備える必要がある．さらに，C++ではメソッド宣言部が実現部とは別に必要になるが，他の言語ではメソッド実現部に相当する記述のみで済んでいる．これらがC++の記述量が多くなる要因である．

BDLでは，PSDLの関連型記述に相当している接続関係記述部が，同時に存在従属性制約を意味するなど，建築設計に特化しているので，記述量が少なくなる．一方，事務処理分野を対象にしたPSDLにはそのような特化は許されていない．

また，PSDLでは属性の単一代入方式のため，5.2節の例で示したように，1) 各属性にnewとoldの2種類を用意しなければならない，2) 属性値変更のトリガーも記述する必要がある．それに対して，BDLはTMS(Truth Maintenance System)方式を想定しているので，トリガーの記述は不要である．マンマシン・インタフェース・プログラムが属性値変更のトリガーを受け持つようになっている．

6.1.2 記述性と理解性

記述性とは，思考に現れる概念を加工することなくそのまま記述できる性質である．理解性とは，記述されたものを見て概念の形成が容易にできる性質である．このため，思考に現れる概念と言語仕様が密接に

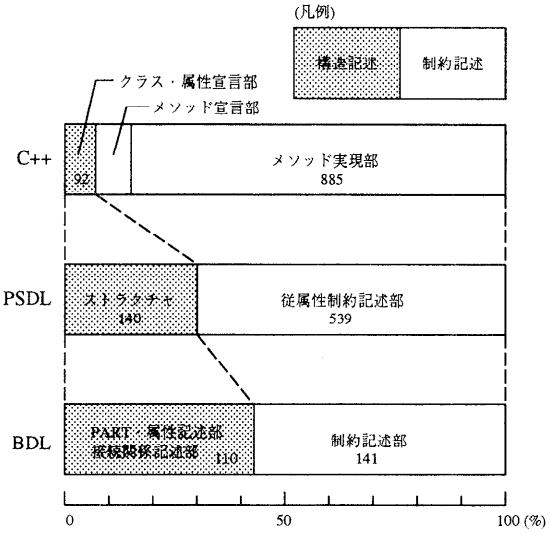


図 10 構造記述と制約記述の割合
Fig. 10 Ratio of structure description and constraint description.

関係していれば，記述性と理解性に優れている．

（1） 最小性との関係

思考に現れる概念と言語仕様が密接に関係していれば，最小性がよくなる傾向にある．このため，最小性がよければ，記述性と理解性がよいものと考えられる．

（2） 構造記述の割合

全体の記述を構造記述と制約記述に分けると，構造記述の方が制約記述よりも記述しやすく理解しやすい．このため，構造記述の割合が大きいほど有利である．各言語の構造記述の割合は**図 10** に示すとおりであり，BDL が最も大きい．

記述量の絶対値で比較すると，各言語の構造記述の差は小さい．つまり，制約記述の量によって，図 9 に示すように全記述量に差ができ，図 10 に示す割合に大きく影響している．

（3） 関係の記述方法

現実の建築物を記述すると，仕様の規模は大きくなる．その仕様全体を記述したり理解するには，その部分と部分との関係を記述したり理解することが，作業に大きな割合を占めるようになる．このため，関係の記述方法が記述性と理解性に大きな影響を与える．

ところが，C++では関係の定義は明示しておらず，ポインタによって表現するのである．一方，PSDL では関連型を用いて明示的に記述し，BDL でもプラグとソケットによる接続関係を用いて明示的に記述している．また，PSDL と BDL を比較すると，PSDL の関連型はエンティティの間の抽象的な対応性を表すのみである．一方，BDL の接続関係は，各建築部材の間の接続という具体的な関係を表している．このため，関係の記述方法から見ても，BDL の記述性と理解性が優れている．

6.1.3 拡 張 性

拡張性とは，記述に現れている概念を少し変更した時に，その変更の影響が及ぶ範囲の小さい性質である．

BDL や PSDL のような仕様記述言語は，思考に現れる概念と密接に関係している．一方，C++のようなプログラミング言語は概念を手続きに分解して記述している．このため，変更の影響が及ぶ範囲が小さくならず，拡張性に不利な傾向がある．

次に BDL と PSDL を比較する．BDL が持っている接続関係には依存・被依存の方向性がある．一方，PSDL の関係型にはそのような方向性はない．このため，BDL では，属性計算の内容によって，記述の変更が波及する方向を推定することができる．たとえば，柱は基準線に，梁は柱にその存在を依存している．このため，梁の属性である座標は，柱の座標に依存する．このとき，柱の座標に関して記述を変更した場合，その影響のチェックは，柱に依存する部材について行えばよい．この場合，梁がそれに当たり，基準線の記述はチェックする必要がない．

一方，各部材にかかる重量計算を行う場合は，柱にかかる重量は座標とは逆に梁に依存する．このとき，梁の重量について記述を変更した場合，その影響のチェックは柱について行わなければならない．

BDL ではさらに，接続関係記述部が同時に存在従属性制約を意味するので，接続関係を変更すれば，同時に存在従属性制約も変更されたことになる．以上に述べたように，拡張性も BDL が優れている．

6.1.4 形 式 性

形式性とは，言語を計算機で処理できる性質である．その例として，プログラムの自動生成や仕様の検証がある．PSDL はその仕様から，COBOL や C 言語のソースコードを自動生成できる[27),28)]．また，C++のソースコードの自動生成も研究中である[29)]．BDL についても，C++のソースコードを自動生成する研究を行っている[30)]．

BDL では，接続関係にあるソケットとプラグの記述に矛盾がないかをジェネレータがチェックをしてくれる．このような機能は，C++のコンパイラや PSDL のジェネレータには備わっていない．

6.1.5 適 用 性

適用性とは，なるべく広い問題領域を書き表すことができる性質である．

表 2 記述言語の評価
Table 2 Evaluation of description languages.

	C++	PSDL	BDL
最小性	×	△	○
記述性と理解性	×	△	○
拡張性	×	△	○
形式性	○	○	○
適用性	○	△	×

BDL は，建築物の構造に特化しているので，適用性が一番弱くなっている．たとえば，建築物の構造計算において，各部材の属性値を算定するための準備計算は BDL で記述できるが，地震層せん断力算定[31]には BDL には存在しない**階**という概念が必要であるため，その算定を BDL で記述することはできない．

一方，C++や PSDL にはこのような問題はない．

6.2 BDL の評価

前節の比較の観点について，各言語の評価を**表 2** にまとめる．表 2 から分かるように，BDL はその適用性を狭め，最小性を増すことにより，記述性・理解性や拡張性を増している．

BDL は，特別なプログラミング知識を必要としないので，設計者自身による設計知識の記述が期待される．これらのことから，それぞれの作業に特化した言語を規定し，設計者自身に知識を記述させることは，設計支援システムの要求分析に有効な手段である．

6.3 ドメインに特化した言語の特徴

本研究では，建築 CAD のプロトタイプの開発を目的として，要求分析ならびに仕様化を行った．その結果，応用ドメインに特化した概念モデル記述言語の特徴として以下の三つの点が明らかになった．これらの特徴はシステム開発サイクル（表 1）のフェーズ 3 以降で，記述言語が改訂された場合でも保存されると考えられる．

第一に，ドメインモデルの構造が言語に反映されることである．ER モデルやオブジェクト指向モデルは汎用性のあるモデルであるが，それが応用ドメインのモデルとして最適というわけではなく，最適なモデルを選択することによって，言語の最小性，記述性，理解性を増すことができる．すでに述べたように，BDL においては，プラグとソケットが大きな役割を果たしている．

第二に，制約記述の集約ならびに制限がある．特定のドメインにおいては任意の制約が記述できる必要はなく，それを制限することによって，拡張性や形式性の面ですぐれた言語となる．BDL では，部材の存在従属性制約が接続関係と一緒になっており，さらに属性従属性制約が接続関係上に限定されている．

第三に，適切な計算機構の選択がある．データフロー計算が中心となるようなドメインでは単一代入が有利であるが，挿入，削除や属性変更が重要であるときには，単一代入はむしろ記述が困難になる．また制約記述を限定したことによって，必要な再計算をシステムが自動的に起動することが可能になる．このようにドメインにふさわしい計算機構を備えることによって，特に記述の最小性を高めることができる．

7. おわりに

本論文では，建築 CAD に特化した概念モデル記述言語 BDL の特徴を分析し，ドメインに特化した概念モデル記述言語のあり方について考察した．

BDL は，建築の専門家にも容易に理解でき，知識の追加や修正ができることを目指して開発をすすめている言語であり，建築物の構成要素である部材の接続関係に基づいて部材の存在従属性が自動的に仮定され，さらに属性従属性の記述も容易であるという特徴を持っている．

具体的に BDL の特徴を明らかにするために，仕様記述言語 PSDL およびオブジェクト指向プログラミング言語 C++と比較した．評価の方法として，建築物のオブジェクトを各言語で記述し，考察を行った．その結果，ドメインに特化した記述言語では，ドメインモデルの構造が言語に反映されており，制約記述や計算機構もドメインモデルに最適なものが選択されていることで，記述言語の最小性，記述性と理解性，拡張性，形式性などが高まっていることが明らかになった．

著者らは現在建築設計用 CAD のプロトタイプの開発をすすめているが，BDL はそのための言語としてまだ不十分なものであり，今後プロトタイプの開発の進行に伴って拡充する必要がある．また，現在の BDL は部材とそれらの静的な相互関係だけを記述しているが，設計者が設計対象に対して施す動的な作用の記述も必要であり，汎化タスクのような概念を導入する必要がある．

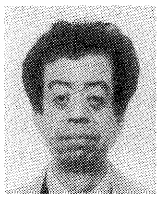
謝辞 本研究の記述実験にご協力いただいた大学院学生千原博司君（現在キヤノン(株)勤務）ならびに佐藤俊孝君に感謝します．また，詳細なコメントを頂いた査読者の方に感謝します．

参考文献

1) Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorensen, W.: *Object-Oriented*

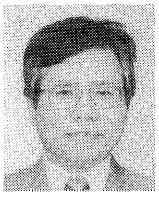
Modeling and Design, Prentice Hall (1991)(邦訳:『オブジェクト指向方法論—モデル化と設計』羽生田栄一監訳, トッパン(1992)).
2) Bell, R.: Choosing Tools for Analysis and Design, *IEEE Software*, Vol. 11, No. 5, pp. 121-125 (1994).
3) Prieto-Diaz, R.: Domain Analysis for Reusability, *Proc. IEEE Computer Software and Application Conference*, pp. 23-29 (1987).
4) 長澤 勲: ポスト・マスプロダクション・パラダイムとIMS, 精密工学会知識工学とCAD専門委員会第23回例会 (1993年6月).
5) 橋本正明: データ中心のプログラム仕様記述法, 井上書院 (1988).
6) Hashimoto, M. and Hirota, T.: A Case Study on Conceptual and Formal Modeling of Software, *Joint Conference on Software Engineering '93*, pp. 333-340 (1993).
7) DeMarco, T.: *Structured Analysis and System Specification*, Yourdon Inc. (1979)(邦訳:『構造化分析とシステム仕様』高梨智弘ほか訳, 日経マグロウヒル社 (1986)).
8) Chen, P. P.: The Entity-Relationship Model—Toward a Unified View of Data, *ACM Trans. Database Syst.*, Vol. 1, No. 1, pp. 9-36 (1976).
9) Coad, P. and Yourdon, E.: *Object-Oriented Analysis*, 2nd edition, Prentice-Hall (1991)(邦訳:『オブジェクト指向分析(OOA)』羽生田栄一監訳, トッパン (1993)).
10) Hashimoto, M. and Okamoto, K.: A Set and Mapping-based Detection and Solution Method for Structure Clash Between Program Input and Output Data, *Proc. IEEE Computer Software and Application Conference*, pp. 629-638 (1990).
11) (社)日本設計学会: 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6年度)報告書 (1991).
12) 伊藤公俊: 設計対象物のメンタルモデル, 人工知能学会誌, Vol. 7, No. 2, pp. 203-211 (1992).
13) 中島裕生: 設計における知識表現とモデリング, 人工知能学会誌, Vol. 7, No. 2, pp. 212-218 (1992).
14) 吉川弘之: 一般設計学序説—一般設計学のための公理的方法—, 精密機械, Vol. 45, No. 8, pp. 20-28 (1979).
15) 吉川弘之: 一般設計学, 機械の研究, Vol. 37, No. 1, pp. 108-116 (1985).
16) Brown, D. C. and Chandrasckaran, B.: Knowledge and Control for a Mechanical Design Expert System, *IEEE Computer*, Vol. 19, No. 7, pp. 92-100 (1986).
17) 梅田政信, 長澤 勲, 樋口達治, 永田良人: 設計計算のプログラム書法, 電子情報通信学会技術研究報告(人工知能と知識処理), Vol. 91, No. 315, pp. 25-32 (1991).
18) 望月雅光, 長澤 勲, 梅田政信, 樋口達治, 小島崇司: 公差解析のための知識表現言語とそのプログラミング手法, 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. 41-48 (1994).
19) 山口秀行, 長澤 勲, 梅田政信, 櫻井尚子: 発電プラント向け熱交換器設計知識の整理と体系化, 第12回設計シンポジウム講演論文集 (1994).
20) 伊藤公俊, 持田恵作: 機械設計者の概念モデルとしての設計対象多視点モデル, 人工知能学会研究会資料(知識ベースシステム), Vol. 88, No. 1, pp. 52-61 (1988).
21) Sriram, D., Logcher, R., Wong, A. and Ahmed, S.: An Object-Oriented Framework for Collaborative Engineering Design, Sriram, D., Logcher, R. and Fukuda, S. (eds.), *Computer-Aided Cooperative Product Development* (*LNCS 492*), pp. 51-92, Springer-Verlag (1991).
22) 平沢岳人, 山崎雄介: 部品概念とその知識表現に関する研究, 第16回情報・システム・利用・技術シンポジウム, pp. 205-210 (1993).
23) 渡辺 俊, 渡辺仁史: 建築設計のための知識表現モデルに関する研究, 日本建築学会計画系論文報告集, No. 443, pp. 71-78 (1993).
24) 長澤 勲, 手越義昭, 牧野 稔: IBDS: 建築物の統合化設計支援システム, 情報処理学会論文誌, Vol. 30, No. 8, pp. 1058-1067 (1989).
25) 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 (1993).
26) Liskov, B. H. and Zilles, S. N.: Specification Techniques for Data Abstraction, *IEEE Trans. Softw. Eng.*, Vol. SE-1, No. 1, pp. 7-19 (1975).
27) 橋本正明: 非手続き型言語と入出力データの構造不一致, 情報処理学会論文誌, Vol. 29, No. 12, pp. 1141-1150 (1988).
28) 岡本克巳, 橋本正明: 入出力データの構造不一致検出解決法に関する研究, 情報処理学会論文誌, Vol. 33, No. 5, pp. 707-716 (1992).
29) 山崎充彦, 廣田豊彦, 橋本正明: 仕様記述言語PSDLからオブジェクト指向言語への変換, 第48回情報処理学会全国大会論文集, 1G-10 (1994).
30) 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 (1994).
31) 海野哲夫: 構造学再入門—II, 彰国社 (1982).

(平成6年 7 月29日受付)
(平成6年11月17日採録)



廣田　豊彦（正会員）

1954年生．1976年京都大学工学部電気工学第二学科卒業．1981年同大学大学院工学研究科博士後期課程研究指導認定退学．工学博士．1981年京都大学情報処理教育センター助手．1988年九州工業大学情報科学センター助教授．1989年同大学情報工学部知能情報工学科助教授，現在に至る．プログラム開発環境，プログラムテスト支援，CADシステムなどの研究に従事．著書「Cプログラミングの基礎」(培風館)ほか．日本ソフトウェア科学会，人工知能学会各会員．



橋本　正明（正会員）

昭和21年生．昭和43年九州大学工学部電子工学科卒業．昭和45年同大学院修士課程修了．同年日本電信電話公社電気通信研究所勤務．昭和63年ATR通信システム研究所出向．平成5年九州工業大学情報工学部教授．従来オペレーティングシステムや，データベース設計，仕様記述言語，ソフトウェア自動作成等の研究実用化に従事．著書「データ中心のプログラム仕様記述法」(井上書院)．工学博士．電子情報通信学会，IEEE等各会員．

長澤　勲（正会員）

昭和19年生．昭和42年九州大学工学部電子工学科卒業．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/CAM，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，日本設計学会，日本ロボット学会各会員．