ソフトウェア工学 129－12
（2000．11．17）

多視点モデリングのためのCADシステム生成方式へのDB実装

阪中 幹恵[*1] 橋本 正明[*2] 廣田 豊彦[*2] 片峯 恵一[*2] Lukman Efendy[*1]

[*1] 九州工業大学大学院情報工学研究科
[*2] 九州工業大学情報工学部

本稿では、多視点モデリングと、これを実現するための DB の実装について報告する。筆者らは、様々なドメインの仕様記述言語を処理し、CADシステムを自動生成する方式を提唱した。しかし現在の所、生成されたシステムは、異なるドメインのデータを統合管理できない。通常一つのプロダクトは、それを扱うドメインによって、様々な見方をされる。このため、たとえば建築構造設計の専門家は、意匠設計の情報の中から必要なものを抽出し、利用している。この時ドメインの専門家の頭の中では、ドメインの見方が変換されている。多視点モデリングとは、この変換作業を計算機で実現するために、ドメイン毎に独立してデータを考えるのではなく、プロダクトを中心にし、ドメインとの対応をとるもので、DBを用いて実現する。

Implementation of Database in CAD System Generation for Multi-View Modeling

Motoe Sakanaka[*1] Masaaki Hashimoto[*2] Toyohiko Hitora[*2]
Keiichi Katamine[*2] Lukman Efendy[*1]

[*1] Graduate School of Faculty of Computer Science and Systems Engineering,
Kyushu Institute of Technology
[*2] Faculty of Kyushu Institute of Technology

This paper describes multi-view modeling and Implementation of Database for the modeling. We proposed the reference model of CAD system generation, which can process various domain-specific languages. However, existing CAD system generation can't join data of multiple domains. Generally, each domain has own view of the same product. Therefore, in the domain of architectural structure design, designers obtain data from the domain of architecture design. Domain experts exchange domain's views into each other. The multi-view modeling is a way to think of product not in one domain but in all domains. This makes it possible to exchange various domain's views by computers.

1　はじめに

大量生産の時代には,ひとつの仕事をひたすら分けて単純化し、それぞれの業務をそれぞれの専門家が行うことで、業務の合理化,効率化を図ってきた。また、情報技術に関しては、単なる効率化、機械化の手段としてしか捉えられておらず、多くの企業では、企業全体のビジネスと情報化の全体構想を描かずに、販売、経理といった個別の業務機能のみに着目し情報システムの導入・開発を進めて来た。とくに、経営環境が複雑化し、顧客のニーズにすばやく対応することが企業の最大の課題となっている今、各業種で取り扱うべき情報を共有させ、業務全体を一貫して支援する事が重要となる。

また、今日の計算機やネットワークの爆発的な普及により、大量の情報が氾濫しており、情報は、その量だけでは意味をなさず、信頼性や価値が求められるようになってきた。同時に、企業内においても、自社が独自で収集した情報をデータベース化し、知的資産として共有・活用していくことが重要となる。そうした中、データベースの役割は、ますます高まってきている。

本研究は、IBDS（Integrated Building Design System）へ、OODBMS(Object-Oriented DataBase Management Systems)を組み込む事で、データの統合管理を実現する事を目的としている。

以下、2章では、本研究室で現在行っているCADシステムの自動生成とIBDSの概要を述べ、3章で多視点モデリングとその実現方法ついて述べ、4章で多視点モデリングの一環として行った IFC と板金設計用 CAD 間のデータモデル変換実験について報告し、5章では IBDS へのデータベースの実装について述べる。最後に6章でデータモデル変換実験とデータベース実装に関する考察を行う。

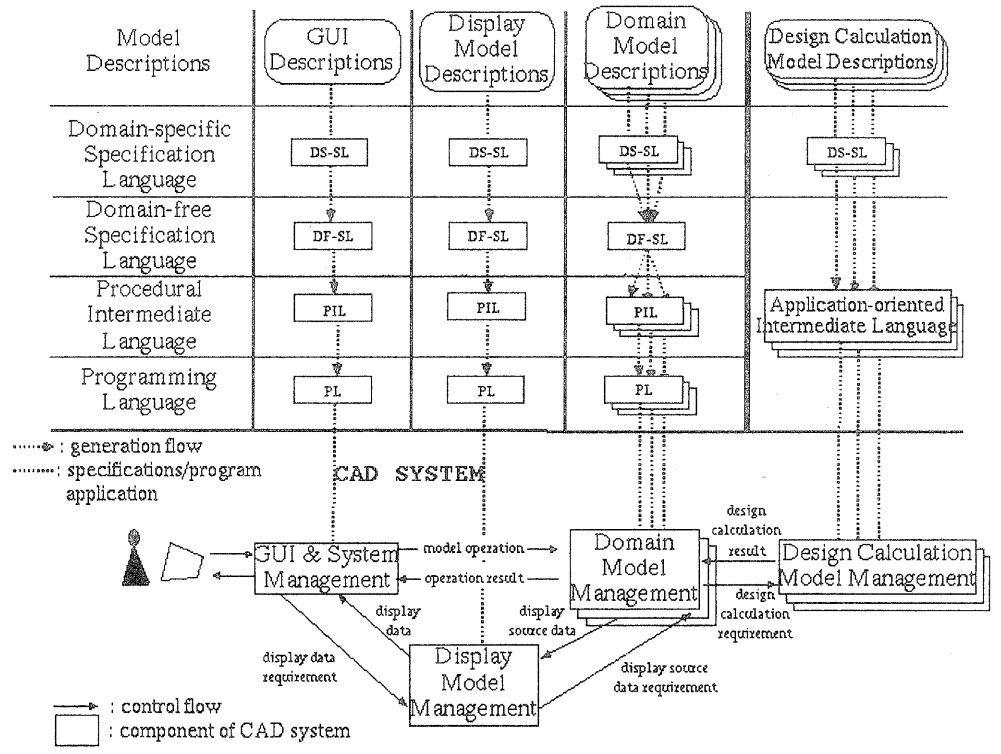

図1 CAD生成リファレンスモデル

2 リファレンスモデルに基づくCADシステムの自動生成

2.1 リファレンスモデル

本研究室では、CAD生成リファレンスモデルを提唱している(図1)[1]。このモデルに基づき、我々は自動的にCADシステムIBDSを生成するコンパイラを開発している。IBDSの生成は、まず、ドメイン特化仕様記述言語により記述されたCADシステムの仕様に、字句解析、構文解析を行い、中間言語1(DS-SL:Domain-specific Specification Language)を生成する。さらに中間言語1に意味処理を行い、中間言語2(ドメインフリーの仕様記述言語:DF-SL:Domain-free Specification Language[2])に変換する。

中間言語2は、非手続き型であるので、手続きを組み込み、中間言語3(PIL:Procedural Intermediate Language:)に変換し、最後にプログラミング言語(PL:Programming Language)が生成される。

ドメインフリーの仕様記述言語は、汎用的なオブジェクトモデルに基づいており、属性の計算の他に、オブジェクトやリンクの生成と消去を行うための制約も付加している。オブジェクトモデルに制限を加えたドメイン特化モデルや、オブジェクトモデルの要素を組み合わせたドメイン特化モデルはオブジェクトモデルによって表現可能である。そのため、様々なドメインの仕様をドメインフリーの仕様記述言語に変換し、統合することが可能となる。したがって、ドメインフリーの仕様記述言語に対しIBDSジェネレートプログラムを開発することによって、様々なドメインに特化した仕様記述言語による記述から、IBDSの自動生成を行うことができる。

2.2 CADシステム自動生成

IBDSとは、分業化された建築物設計支援システムの作業を統合して支援するために、オブジェクト指向かつ概念モデルを適応したCADシステムである。

我々は、信頼性があり、高品質のアプリケーションソフトウェアを開発する効果的な方法について研究している。一般に、プログラミング言語を用いてシステムを開発する場合、ドメインの専門家自身によって、直接プログラムにドメイン知識を追加したり変更したりすることは困難である。このため、プログラマは、専門家との仕様書を介したコミュニケーションによって、ドメインの知識を獲得している。あるドメインに適応した高品質なソフトウェアを開発するためには、そのドメインの専門家自身がシステムの要求仕様書を記述することが望ましい。そのために、我々は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の規定を行っている。現在のところ、建築構造設計ドメインに特化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BDL(Building Design Language)を規定した。さらに、BDLで記述された仕様からIBDSの自動生成が可能である。

3 多視点モデリングへの対応

3.1 データ共有の必要性

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。各設計者は、相互に他の設計者の助けを必要とし、たとえば、意匠設計を進めるには、部材の寸法や空調設備の機種、高架水槽の位置などがある程度決まっていなくてはならない。このため、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くべきであるが、現在は分業を徹底させ、流れ作業で設計を行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く、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。そこで見込み違いが起こると設計変更が必要となるので、余裕の多い設計を行う傾向になっている。また、従来は分業化された各設計で独自に設計支援システムを用いおり、それぞれの設計で使われているシステムはデータの互換性がなかった。このため、従来から設計計算書やCAD図面等などにより、設計者自身が手作業で情報の交換を行ってきた。さらに、それぞれ独自に変更や修正を行うため、設計情報の整合性を保

つのが困難となっていた。ところが、建築物の設計は、発注者の新たな要求や施工上の問題点を解決するため、再設計が頻繁に行われている。設計変更や再設計を行うには、時間がかかるとともに、専門外の設計者の協力が必要となり、さまざまな設計図書間の整合性を管理するのに手間取る。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。

このような問題を改善するには、設計作業の流れ全体を一貫して支援し、それぞれの設計作業に必要な設計情報を共有させることが必要となる。

3.2 多視点モデリング

通常プロダクトは、それを扱うドメイン毎にさまざまな見方をされる。前述したように、複数ドメイン間で協調して作業を行う場合、ドメイン間でのデータ交換は不可欠となる。このような場合、ドメインの専門家は、提示された他のドメインのデータの中から必要な情報だけを抽出し、利用している。この時、作業者の頭の中で、ドメインの見方の変換が行われているのである。この変換作業を計算機で実現するために、多視点モデリングを行わなければならない。

多視点モデリングとは、従来のようにドメイン毎に独立してデータを考えるのではなく、プロダクトを中心にデータを考えるものである。多視点モデリングにより、複数のドメインの間でデータを統合的に扱う事ができるようになる。

3.3 多視点モデリングのためのデータベース

現在の IBDS では、全てのデータをディスクではなく、メモリ上に置いて実行している。しかし、データが膨大になった場合、それらのデータはデータベースに保存されなければならない。

また、多視点モデリングは、データベースを利用して実現する。つまり、プロダクトデータと、プロダクトとドメインとのデータ変換をメソッドとしたオブジェクトをデータベースで管理する。

我々が提唱した CAD 生成リファレンスモデル(図1)では、様々なドメインのデータは、一旦ドメインフリーのモデルに変換されるため、このドメインフリーモデルのデータをデータベースに保存すれば、様々なドメインの間のデータを同じ形式で保存する事が可能となる。

このドメインフリーの仕様記述言語の基になっているドメインフリーのモデルをベースに、ドメインに特化したモデルとプロダクトの変換メソッドを加えたものをデータベースで保存することで多視点モデリングを実現する。

ドメインフリーのモデルはオブジェクトモデルに基づいているので、オブジェクト指向データベースならば、容易に適応できる。そのため、本研究では、現在、オブジェクト指向データベースの実装を行っている。

多視点モデリングを実現するためには、プロダクトとしてどういう属性を持てば良いか、また、プロダクトとドメイン特化モデルはどう対応するかを考えなければならない。あるプロダクトの見方が異なるドメイン間でどのように違い、どのように関連しているのかを知るため、データモデル変換実験を行った。

4 データモデル変換実験

IBDSは、意匠、構造、設備、積算の4つの支援システムを、データベースを介して統合するものである。それぞれの業務毎に、業務に特化した概念モデルを持っているため、それぞれの業務間で連携して作業を行う場合、業務間でデータモデル変換は必要不可欠である。

4.1 実験内容

今回、IFCモデルとドメインフリーモデル(オブジェクトモデル)と板金設計用CADの間のデータモデル変換を行った。 具体的には、IFCモデルの中から、部材(壁と開口)の形状に関する部分を抜き出し、IBDSで表示させ(図2)、その中から壁の開口の大きさを抽出し、板金設計用CADで読み込み、建築部材が設計され、自動的に出入り口用板金の設計を行う(図3)というものである。

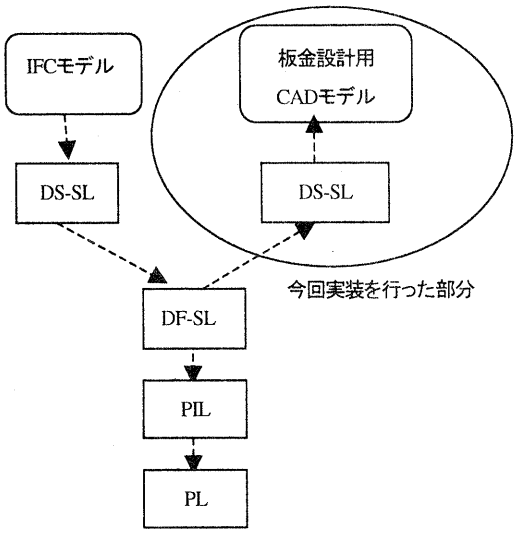

図4 モデル変換実験の流れ

モデル変換の流れは図4のようになっており、既にIFC から汎用モデルへの変換は実装済みである[3]ので、今回は主に汎用モデルから板金設計用 CADモデルへの変換を行った。

4.2 IFCモデル

IFC(Industry Foundation Classes)とは、建築物の設計から施工,管理,廃棄というライフサイクル全般で使用できるような、共有データモデルでありIAI(International Alliance for Interoperability)により規定されたものである。

IFCでは、建物を構成するすべての要素のシステム的な表現方法の仕様定義やプロジェクトモデルのデータ構造を提示されている。モデルの基本要素はエンティティとなっており、リレーショナル・モデルをベースとしたモデルであるが、継承などのオブジェクト指向言語の特徴も持っているモデルである。

4.3 板金設計用CADモデル

板金設計用CADモデルは木構造モデルである。基本要素はフィーチャーで、フィーチャーの寸法の間に、制約関係を定義することができる。フィーチャーとは、板金設計用 CAD モデルの基本要素であり、データムとソリッドから成る。データムとはモデル構

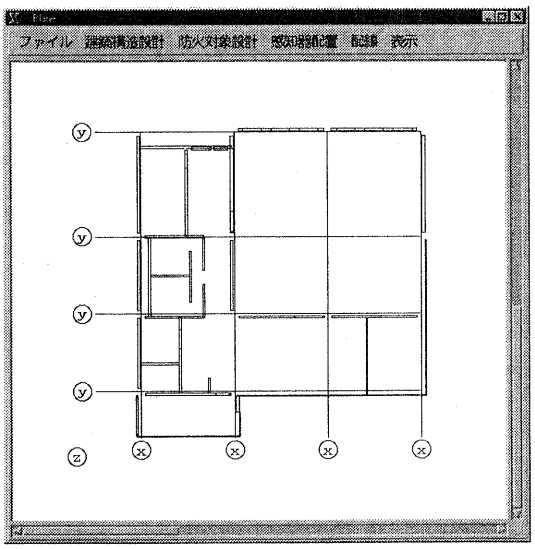

図2 IBDS におけるのIFC データの表示

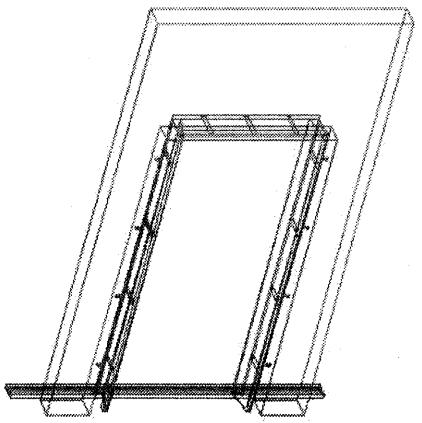

図3 自動設計された出入り口板金

築の参照として利用されるもので、平面、軸、点、座標系などを言う。ソリッドとはモデルの形状をあらわすもので、カット突起などを言う。板金設計用CADでは、フィーチャーが複数組み合わさって1つの部品となり、さらに部品が複数組み合わさって最終的な製品が完成する。

4.4 モデルの対応

モデル変換において、もっとも考慮しなければならないのは、リレーションシップの対応である。ERモデルでは、どんなリレーションも表すことができるが、オブジェクトモデルでは、集約、リンク、継承の3種類に限定される。そのため、ドメイン特化モデルにおけるリレーションが、オブジェクトモデルにおいて、集約、リンク、継承の、どの概念に変換させるかが、大きな問題となる。板金設計用CADモデルにおいては、アセンブリ関係は、集約関係に、配置拘束関係と、寸法参照関係はリンクに対応する(表1)。

表1 モデルの対応

IFCモデル	CADモデル	オブジェクトモデル
ERモデルベース	木構造モデル	オブジェクトモデル
部材クラス	アセンブリ フィーチャー データム ソリッド	クラス
属性	属性	属性
リンク	アセンブリ関係	集約
	配置拘束関係 寸法参照関係	リンク
継承	なし	継承

5 データベースの実装

本研究では現在、IBDS へ、オブジェクト指向データベースの実装を行っている。これは、Java 言語で実装されており、Java プラットフォームで動作可能である。

図1にもある通り、IBDS は、オブジェクト指向プログラミング言語により実装される。今の所、Inside Prolog と呼ばれるオブジェクト指向 Prolog 言語と、C++言語の自動生成が可能である[4]。今回、OODBMS の導入のため、Java 言語の自動生成を実装した。

現在、自動生成された Java プログラムを基に、データベースへの保存プログラムを手動により作成中である。このデータベースは、スタンドアロンの Java アプリケーションである。これがサーバとなり、データベースを利用するアプリケーションは、クライアントとなる。それらは、UNIX ソケットを通して通信を行う。クライアントアプリケーションは、RMI(Remote Method Invocation)を通して動作する。このデータベースの RMI は、Java-RMI と似ているので、簡単に用いることができるようになっている。クライアントアプリケーションは、プロキシオブジェクトを用いてのみデータベースオブジェクトと接続する事ができる。

OODB を使用するにあたり、データベース化するクラスは、そのクラス自身のほかに、関係するクラスとインタフェース定義しなければならない。たとえば、Car オブジェクトを保存する場合、まず、Car というクラスを定義しなければならない。しかし、すべてのデータベースオブジェクトは RMI によってコントロールされるので、まず、Car オブジェクトのメソッドインタフェースを定義する(図6)。これらのインタフェースメソッドは、RMI を通してのみ呼ばれるため、このインタフェースがデータベースの RMI である事を示すため、Remote クラスを継承する必要がある。また、あるメソッドによってオブジェクトの状態が更新される場合、データベースの方も、オブジェクトの状態を更新しなければならない。このため、プログラマはこのようなメソッドに対しては、メソッドの終わりに/*update*/、あるいは//update とコメントを記すか、メソッドの名前を*_update とするなどして明示的にシステムに分かるようにしなければならない。

次に、CarImpl クラスを定義する必要がある(図 7)。データベースオブジェクトはこのクラスのインスタンスとなる。このクラスは、Car インタフェースの実態を実装するクラスである。このクラスがデータベースオブジェクトである事を示すために, DBObject クラスを継承する必要がある。

最後に、CarImpl クラスのためのプロキシクラスを定義する必要がある(図8)。プロキシクラスはデータベースに、データベースオブジェクトのメソッド呼び出しを通知するものである。

6 考察

6.1 データモデル変換実験

今回のデータモデル変換実験により、IBDS において一旦ドメイン特化モデルからドメインフリーのモデルへの変換規則を規定しておけば、容易に他のドメイン特化モデルへと変換できる事が確認できた。今後は、板金設計用CADモデルから汎用モデルへの変換と、汎用モデルから IFC モデルへの変換を規定する必要がある今回の実験により、IFCモデルと板金設計用CADと、オブジェクトモデルの間の対応関係は明らかになったが、データモデルの対応関係はまだ明らかにはなっていない。例えば、今回は建築ドメインにおける壁の開口の幅や高さの値によって出入り口板金の幅や高さが決まる。多視点モデリングを実現するためには、こうした属性の間の対応関係を明らかにしていく必要がある。

6.2 データベース実装

データベース化するためにはいくつかの問題がある。例えば、オブジェクトの状態を更新するメソッドに関してはコメントを記さなければならない。また、1つのオブジェクトに対して、インタフェースとプロキシクラスを定義しなければならないので、プログラムのソースコードの量も数倍に増えてしまう。また、独自のクラスを継承したりしなければならないため、プログラムのクラス構造を変更しなければならない。さらに、本来そのクラスがサブクラスである場合、Java には多重継承機能がないため問題となる。現在データベース保存プログラムを手動で作成しているが、この部分も自動生成するプログラムを作成する。さらに、実装する OODBMS を選択できるようにし、それに伴い生成されるプログラムも変更されるようにする。また、データベースにデータを保存する場合と、メモリ上において実行させる場合との速度比較も行い、ユーザが必要に応じてデータベースにデータを保存できるようにする。

7 おわりに

本稿では、我々が進めている IBDS 自動生成の研究を紹介し、多視点モデリングについて述べた。そして、その一環として行ったデータモデル変換実験と、データベースの実装について報告した。今回のデータモデル変換実験で、建築以外のドメインも IBDS により支援できる事が示されたが、多視点モデリングを実現するにはまだ結果の分析が不十分である。データベースの実装に関しては、いくつか問題があるため、それを解決していかなければならない。今後はデータベースの実装を中心に行っていきたい。

参考文献

[1] Lukman Efendy. A Reference Model of CAD System Generation from Various Object Model-Based Specification Description Languages Specific to Individual Domains. 電子情報通信学会英文論文誌, 2000.

[2] 筒井 雄一郎. ドメインフリーの仕様記述言語とその実装 学士論文,九州工業大学,1999.

[3] 中島 健. 建築 CALS の実装実験におけるデータモデル変換の研究 修士論文, 九州工業大学, 1998.

[4] 甲斐 俊文. 非手続き型仕様記述言語処理系の手続き生成部の作成 学士論文, 九州工業大学、1999

```
//import the Remote-Interface
import DB.Remote;

//define our Car-interface

public interface Car extends Remote {

    // two methods to set/get the name of a car
    public void setName (String name); /*update*/
    public String name();
}
```

図6　インタフェース Car の定義

```
import DBObject;
import java.util.*;

public class CarImpl extends DBObject implements Car {
    // some members
    String _name;

    // a constructor without arguments
    public CarImpl() {
        _name = new String ("");
        }
    // implement the interface methods
    public String name() {
        return _name;
        }
    public void setName (String name) {
        _name = name;
        }
    }
```

図7　CarImpl クラス定義

```
import.*;
import.core.ObjectID;
import DB.core.Lock;

public class CarImpl_Proxy
        extends OzoneProxy
        implements Car {
    public CarImpl_Proxy() {
        super();
        }
    public CarImpl_Proxy (ObjectID oid, OzoneInterface link) {
        super (oid, link);
      public java.lang.String name () {
        Object result = null;
        try {
            Object[] args = {};
            result = link.invoke (this, "name", "",
                                  args, Lock.LEVEL_READ);
            }
        catch (Exception e) {
            e.fillInStackTrace();
            throw new UnexpectedException (e.toString()); }
        return (java.lang.String)result;
        }
    public void setName (java.lang.String arg0) {
        Object result = null;
        try {
            Object[] args = {arg0};
            result = link.invoke (this, "setName",
                                  "java.lang.String",
                                  args, Lock.L EVEL_WRITE);
            }
        catch (Exception e) {
            e.fillInStackTrace();
            throw new UnexpectedException (e.toString()); }
        }
}
```

図8　Car_Proxy クラスの定義