有限要素解析プログラム開発における
オブジェクト指向設計の有効性評価に関する研究

富岡 洋一
法政大学大学院工学研究科建設工学専攻修士課程

草深 守人[1)]　竹内 則雄[2)]　武田 洋[3)]
1),2)法政大学工学部土木工学科　3)法政大学工学部システム制御工学科

土木分野で対象とする問題の多くは，諸々の条件が関与する非常に複雑な現象である．これに伴い，プログラムの逐次変更・修正の必要性・難易度は増加の一途を辿っている．
本研究ではオブジェクト指向に基づく有限要素法のプログラミング及びモデル化を行った．使用した言語は UML，Java である．従来からの手法である手続き型指向との比較の結果，オブジェクト指向設計の有効性は認められ，これを用いることの動機も充分に存在するという結論に達した．

1　緒言

現在のソフトウェア工学は，土木分野と密接に結びついている部分が多く，CAD（Computer Aided Design），CAE（Computer Aided Engineering），CAM（Computer Aided Manufacturing）に代表されるように，コンピュータの助けなしに土木工学は成り立たないと言っても過言ではない．
従来までは，科学技術計算におけるソフトウェア工学では効率性，すなわち処理速度が重視された．しかし，土木分野で対象とする問題の多くは，非常に複雑な現象であるため，その都度利用するソフトウェアの逐次変更，修正が求められることが少なくない．
本研究では一つの試みとして，工学の分野で広く用いられている有限要素解析をオブジェクト指向設計に基づく UML によるモデル化，及びオブジェクト指向言語である Java による具体的なプログラミングを行うことによって，今後の拡張が容易なソフトウェアを実験的に開発し，将来への拡張性と生産効率について考察を加える．

2　オブジェクト指向

オブジェクト指向プログラミング／設計（OOP; Object-Oriented Programming, OOD; Object-Oriented Design）は手続き型言語では大きくて複雑なプログラムを作りにくいという反省から生まれた．手続き型言語の問題点は，プログラム全体の構成にある．

2.1　内部データの保守性

手続き型指向の場合は，メインプログラム以下全てのモジュールに全てのデータが雪崩式に流れていく構造になっている．手続き型言語である FORTRAN においては、モジュール（サブルーチン）の呼び出しは，殆どの場合多数の引数が含まれている．各モジュール間の依存関係がオブジェクト指向とは異なり，それが手続き型の欠点となっていた．**図 1** に示したオブジェクト指向においては、球で表した各クラスがその中で必要となるデータ（変数）を格納（データのカプセル化）しているので，互いが「要求－応答」の依存関係になっている．

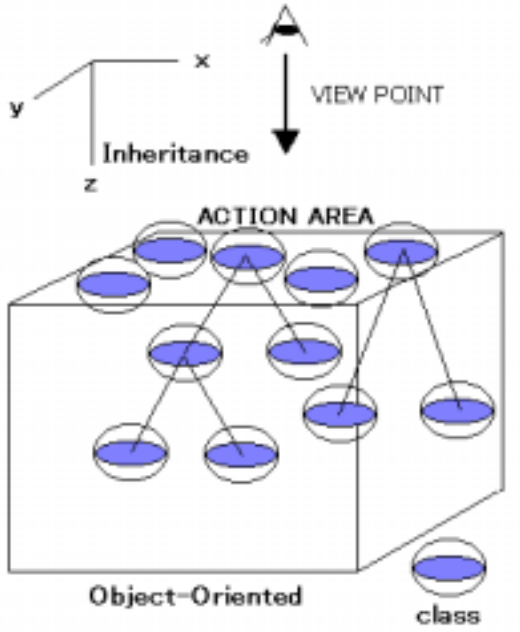


図１　オブジェクト指向イメージ

2.2　拡張性

手続き型で拡張性を考えると，ある関数を新たに追加する時に，その関数を新たに追加したことによる他の関数（モジュール）への影響を考慮しながら，双方のモジュールの変更を余儀なくされていた．そのため，大規模なプログラムになるほど関数の追加が難しくなり，作業の複雑性は指数関数的に高いものとなっていた．

この問題は，オブジェクト指向言語である Java のクラスの概念を用いることにより解消される．クラスは，図１に見られるように個々が独立したオブジェクト（クラス）として扱われる．また，全てのオブジェクトは一元的に影響しあう．

図１のｚ方向には，継承関係によるブラックボックスを示した．ユーザ，或いはクライアントプログラマは平面のみを意識しながらの作業を行うことができる．設計側プログラマは，ｚ方向で表された範囲において，任意の時点でクラスの変更・追加を行うことができる．つまり，ユーザ，クライアントプログラマと関連する部分とそうでない部分，見える部分と見えない部分を完全に切り離すことによって互いの管理する領域を分担している．逆に，ユーザからは平面の部分しか見ることができないし，見る必要が無い．

あるオブジェクトから継承関係にある目的のオブジェクトへの通信を考えると，要求を受けた時点で対話先のオブジェクトは継承関係にある目的のオブジェクトを自動的に呼び出して，適切な応答を返す．即ち，ｚ方向のオブジェクトの選択は，要求の受け手側のオブジェクトに任せることになる。オブジェクト指向設計，或いは保守・管理のし易い設計は，このような関連分離（Separation of Concern）の考え方に基づく設計が理想と考える．

３　有限要素解析システムのモデリング

有限要素解析システムを開発するに当たって，共通の参照モデルによる抽象化を初期設計段階にて行う．作成されたダイアグラムは対象とするシステムを表現し，これによって「ユーザープログラマ」間の意思疎通が容易なものとなる．また，各モジュールの詳細や内容はプログラマ個人に任せたまま，設計者側はシステムを全体として捕らえることが可能である．

本研究では OMG（Object Management Group）によって提案された汎用言語である UML（Uniformed Modeling Language）を用いることにした．特定の言語に依存しないことも UML の特徴の一つである．

図２には，場合分けすべき要素形状の継承関係を示した．継承関係におけるスーパークラスは Element クラスとなっている．ユーザは下層にどのような要素形状に対応したクラスが用意されているかを意識することなく，スーパークラスである Element クラスに「要素形状は？（要求）」と指示を出すだけで，要求の受けて側である Element クラスは自動的に適切な応答を返す．

図３では，有限要素解析中の通信関係をアソシエーションを用いて表示した．白抜きのアソシエーションロールは合成（アグレゲーション）による関係を表しており，ロール名とマルチプリシティには，必要となるマトリクス名，または変数とその個数を示した．ここでは，図２に見られるような継承関係は用いておらず，アップキャストを伴わない合成によるコードの再利用を目的とした設計となっている．

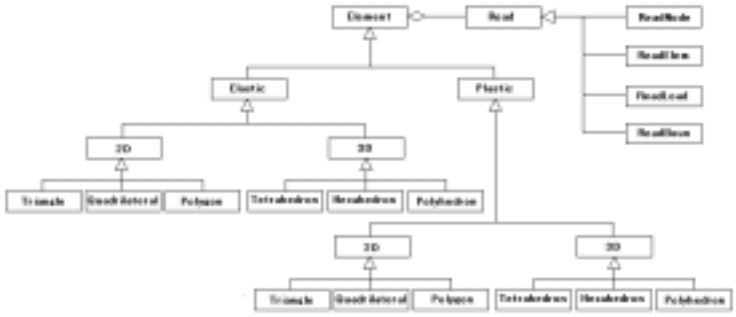


図２　要素間継承関係

図３　有限要素解析モデル

以上のように設計段階において，共通の参照モデルを用いて以後のプログラム作業を概念化することにより，プログラマ間の意思疎通が容易になるだけでなく，必要な項目の追加，修正を迅速に行うことが可能であり，また，このような設計規約を前もって定義することが，ソフトウェア開発において最も重要な作業行程の一つと考える．

４　Java による有限要素解析プログラミング

ここでは，UML によってモデル化されたシステムを，オブジェクト指向言語である Java によって具体的にプログラミングを行う．

４.１　ポリモルフィズムに伴う継承関係を用いた設計

継承関係にあるクラス（図２）と対話するに当たって，**リスト１**に示すような selecter()メソッドをクラス中に用意することによって，サブクラスではなくスーパークラスとの対話を行うことができる．

この方法は，コンパイル時にはどのクラスが呼び出されるのかは分かっておらず，オブジェクトの実際の型に基づいて実行時に結合が行われるので，遅い結合（動的結合，Dynamic Binding）と呼ばれる．

ここでは，selecter()の引数として Read 型のハンドルが渡され，それ以上のことをコンパイラは知ることができない．しかしマトリクス m にはサブクラスである ReadNode クラスによって呼び出されたマトリクスが格納される．これは，実行時に selecter()メソッドによって自動的に ReadNode クラスが呼び出されたためである．

ここで，逆の場合，即ち直接サブクラスを呼び出すような場合を考えてみる（**リスト２**）．

この欠点の第一は，コーディング量が増えること，第

二に，selecter()のようなメソッドを新たに追加したい場合や，ReadNode のような新たなクラス（型）が増えるたびに，その型に固有のメソッドを付け足さなくてはならないので，既存のクラスのコード変更が必要となることである．

リスト１

```
<Example code for Dynamic Binding>
class GrossStiffnessMatrix {
    Matrix m;
    ReadNode rn;
    GrossStiffnessMatrix() {
        rn = new ReadNode();
        m = selecter(rn);
    }
    public Matrix selecter(Read r) {
        return r.get();
    }
      …
```

リスト２

```
<Example code for Dynamic Binding off>
class GrossStiffnessMatrix {
       …
  public Matrix selecter(ReadNode rn) {
    return rn.get();
  }
  public Matrix selecter(ReadElem re) {
    return re.get();
  }
  public Matrix selecter(ReadBoun rb) {
    return rb.get();
  }
      …
```

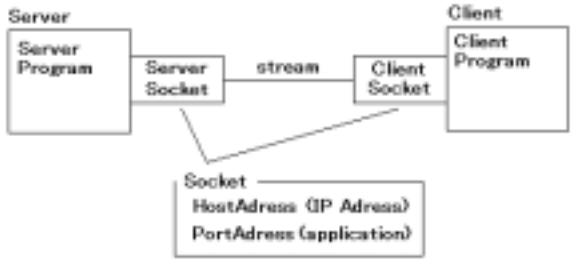


図４　クライアント／サーバ方式

４．２　ネットワーキングによるソフトウェア利用

Java では，オブジェクトはローカルでも，ネットワークを通じて遠隔でも実行可能な TCP/IP プロトコル用の拡張可能なライブラリを備えている．また，アーキテクチャに依存しないコード転送が可能であり，アプリケーションは再コンパイルせずに複数プラットフォームで動作可能である．

ネットワーク環境として**図４**に示す，クライアント／サーバ方式によってシステムを構築した．サーバとクライアントはデータ転送の終点であるソケットによって結ばれており，IP アドレスとポートアドレスに対応している．ユーザは，IP アドレスによりサーバの指定を行い，ポートアドレスによってアプリケーションの指定を行う．クライアントからの要求が発生するとデータストリームが発生し，パケットが転送される．

有限要素解析システムを運用する際は，実際の解析はホストコンピュータ上にて行われるため，クライアントコンピュータのハードウェアは解析に必要な性能を有していなくとも，遠隔操作によるソフトウェアの利用が実現できる．

リスト３

```
<Example code for Server Program>
import java.io.*;
import java.net.*;

public class FEMServer {
  int port;
  Stress stress;
  FEMServer() {
    port = 1120;
    stress = new Stress();
    eqstress = stress.getEquivalentStress();
      …
  }
  public void mode_Stress() {
    try {
      ServerSocket ss = new ServerSocket(port);
      while(true) {
        Socket s = ss.accept();
        OutputStream os =
                      s.getOutputStream();
        DataOutputStream dos =
                new DataOutputStream(os);
      …
  }
  public static void main(String[] args) {
    FEMServer fems = new FEMServer();
    fems.mode_Stress();
  }
}
```

リスト３に，サーバクラスのコードの一部をを示した．

インターフェースでポートアドレスを宣言し，コンストラクタ内で任意のポート番号を指定している．本来，サーバプログラムでは，アプリケーションプロトコルに対応したポート番号を使うが，本システムでは有限要素

解析のみなので，任意のポート番号を宣言している．もしくは，実行メソッドよりコマンドライン引数として取りこむことも考えられるが，現段階では実用を考えたものではないので，コンパイル時定数にとどめた．

メソッド mode_Stress()内では，while 文で無限ループに入ることによって，常時クライアントからのアクセス待機状態に入る．クライアントからの要求が発生すると，データストリームが発生し，パケットが転送される．一連の解析が終了すると，ストリームが閉じて，再び待機状態に入る．

４.３　ブラウザ上での表示例

開発したプログラム FEMA（Finite Element Method Applet）のコードの一部を**リスト４**に示す．作動環境は OS に Windows，ブラウザとしてアプレット（JDK1.2）を用いた．

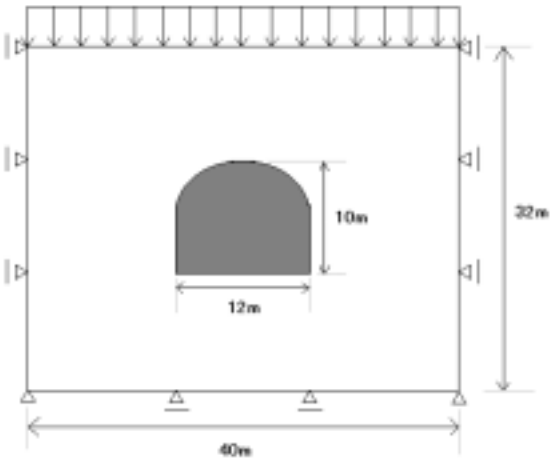
リスト４

```
<Example code for Client Program>
import java.applet.Applet;
import java.awt.Graphics;
import java.io.*;
import java.net.*;
/*
  <applet code = "FEMA"
   width = 800 height = 600>
  </applet>
  */
public class FEMA extends Applet {
  int port;
  Polygon[] p;
  int[] rgbs = {  …  }
      …
  public void init() {
    port = 1120;
    String server = "127.0.0.1";
    try {
      Socket s = new Socket(server, port);
      InputStream is = s.getInputStream();
      DataInputStream dis =
                new DataInputStream(is);
      …
  }
  public void paint(Graphics g) {
      …
  }
}
```

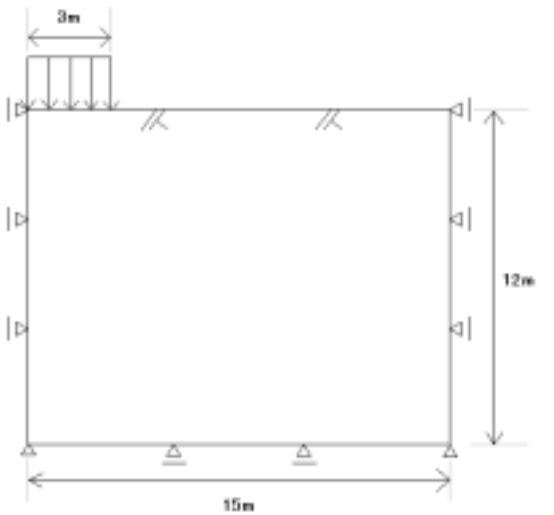
FEMA クラスは Applet からの継承関係にある．初期化メソッド init()において，ソケット生成と同時にホストの IPアドレス, ポートアドレスを宣言している. さらに，データストリームを開きホストとの通信を開始する．アプレットが開始されると, 実行メソッド paint()が起動し，ホストから得たパケットを元に，解析結果を表示する．

FEMA を用いた解析例として, 次の二通りを設定した．

１）空洞モデル．（**図５**）
解析領域幅 40m, 高さ 32m. 掘削断面幅 12m, 深さ 10m. 上部より 2.5(tf/m2)の等分布荷重を載荷.
２）基礎支持力モデル．（**図６**）
解析領域幅 15m，高さ 12m．基礎幅 3m．基礎圧力 10(tf/m2)を加える．



これらの二つの例題に対するブラウザ上での作動状況



を，デスクトップイメージとして**図７，図８**に示す．

図５　空洞モデル

図６　基礎支持モデル

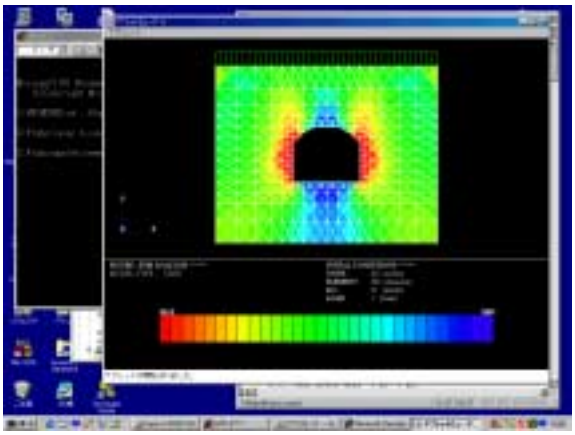


図７　空洞モデル解析結果

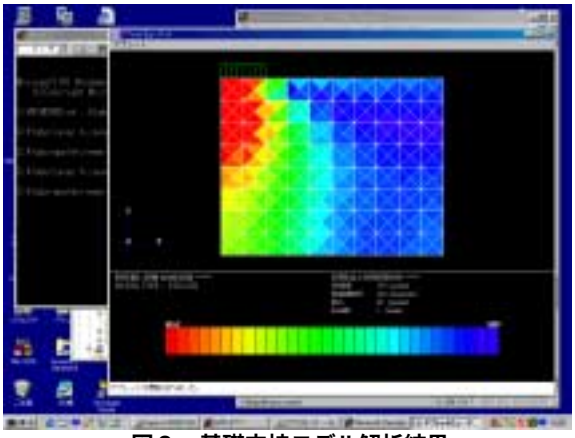


図８　基礎支持モデル解析結果

５　結言

５.１　オブジェクト指向を適用した有限要素解析

有限要素解析を例に，オブジェクト指向設計によるシステム開発を行うことの動機が十分に存在することが明らかとなった．

システムの設計段階において，対象とする問題をオブジェクト指向設計に基づいて，汎用言語でのモデル化を行うことにより，プログラマは客観的に全体を捕らえることができる．

開発段階においては，定型的な処理を格納したクラスライブラリを活用することによって，後の新たなシステム開発において，既存のリソースをシームレスに組み込み可能である．また，クラスを追加する際は，手続き型の樹形図に見られるような層構造は認められず，既存のモジュールの変更を行うことなくサブクラス以下をブラックボックスとして扱うことが可能である．

実際の運用段階においては，オブジェクト指向言語である Java にはネットワーキングに関する機能が予め言語レベルで備わっているため，遠隔環境においての開発，運用が可能である．また，クライアント／サーバ方式におけるネットワーキングによって，運用に絶え得るサーバを設置することにより，ユーザ側のハードウェア機能的側面の問題も解決されると考える．

５.２　速度の問題

科学技術計算に対しては，コンパイラ型の短所である速度の問題は依然残ってはいるものの，遠隔環境におけるソフトウェア開発・運用，Java のポリモルフィズムの機能などの諸々の利点は，処理速度の問題点を十分相殺するものと思われる．さらにこの問題は，昨今見られるような急激なハードウェアの進化（ムーアの法則；半導体チップの集積度 18 ヶ月で 2 倍，ギルダーの法則；通信網の処理能力 1 年で 2 倍）を考えると，手続き型と相対的には十分ならずとも，実用的な実行速度は実際の現場での計算が快適に行える程度に解消されるものと思われる．

５.３　今後のソフトウェア開発に関して

本研究では，オブジェクト指向言語として Java によるコーディングを行った．しかし，このような作業は依然としてプログラミング言語に依存することになり、それらの言語はいずれ必ず陳腐化する．そこで，初期設計段階において汎用言語によるモデル図を作成した．この作業によって，後のプログラマへの意思伝達は，より容易なものになるはずである．数値解析技術も含めてあらゆるモデル図は青写真のようなもので，それらを具体的なプログラムにするのは個々の技術者である．個々のソフトウェアの設計図は彼らの頭の中にしか存在せず，それをプログラムの解読という作業をしながら理解しようとすることは非常に困難で退屈な作業である．結局は，現時点で最も優れている言語を用いてもそれはプログラマレベルでの解決でしかない．

手続き型言語であっても，従来まではそうしてきたように今後も大規模プログラミングは可能かもしれない．しかし，それは非常に限られた，狭い範囲での管理に留まらざるを得ない．常時修正・変更の絶えない実際の現場でソフトウェアとして利用することを考えると，開発における効率化が重要な点であることは間違いない．そもそも，ソフトウェアの土木分野への利用は，複雑で到底手作業で処理できないような問題への単純な解決の手段として利用されるはずである．それが，ソフトウェア開発自体が建造物の建設以上に複雑になってきているように思われる．今後は，従来から存在する膨大な技術計算におけるアルゴリズムを利用すると同時に，容易に拡張可能なソフトウェア設計が重要と考える．今にも壊れそうな巨大なプログラムを限られた範囲によって作成するのではなく，蓄積されてきたアルゴリズムを共通の財産として蓄えることにより，後に作成するであろう大規模プログラムに組立て部品の一部になるように設計する方が，進化する生物のように柔軟なものになると考える．

有限要素法に限らず，様々な数値解析において，オブジェクト指向における設計のパターンが無限に存在し，研究段階は未だ初期段階である．それらとの比較・検討を行った上で，どのモデルが最良であるか，十分吟味されるべきと考える．

参考文献

Bruce Eckel，Java プログラミングマスターコース株式会社ピアソンエデュケーション，1999

キーワード．
有限要素解析，オブジェクト指向設計，UML，Java

Summary.

Effectiveness of Object-Oriented Design on Finite Element Method Programming

Yoichi TOMIOKA　Morito KUSABUKA　Norio TAKEUCHI
Department of Civil Engineering, Faculty of Engineering, HOSEI University

Hiroshi TAKEDA
Department of System Control Engineering, Faculty of Engineering, HOSEI University

A number of objects in civil engineering are affected by diverse conditions therefore those phenomena are really complicated in structure. That has brought with it following increase necessity of programming alteration and degree of difficulty modification.
In this study, Finite Element Analysis was modeled by Uniformed Modeling Language on Object-Oriented Design and programmed by Java. Model the target by all-purpose language, then end user and programmer are able to catch the system objectively and that is easy to understand rightly, smoothly each other. On programming phase, it has discovered that following faculties, Matrix operation class library for FEM, useful of polymorphism on inheritance as a code recycle, remote development and handling on networking, and so on those couldn't achieve until now, help the way brake through.
The result from comparison of Procedure-Oriented and Object-Oriented is that there were signs of effectively to use Object-Oriented and enough motive to adopt it.

Keywords.
Finite Element Analysis, Object-Oriented Design, Uniformed Modeling Language, Java