土木学会論文集 No. 689／I -57, 301-320, 2001. 10

オブジェクト指向構造解析システムの分析と設計

高橋　良和[1]・五十嵐　晃[2]・家村　浩和[3]

[1]正会員　工（修）京都大学助手　工学研究科土木システム工学専攻 (〒 606-8501 京都市左京区吉田本町)
[2]正会員　Ph.D.　京都大学助教授　工学研究科土木システム工学専攻 (〒 606-8501 京都市左京区吉田本町)
[3]フェロー　工博　京都大学教授　工学研究科土木システム工学専攻 (〒 606-8501 京都市左京区吉田本町)

構造解析の知識が深まるにつれ，複雑な構造物の挙動を解析するためのシステムが数多く開発されている．しかしソフトウェアが大規模化するにつれ，システムが複雑化し，保守・管理や新たな手法の追加が困難となってきているのが事実である．本研究では，これら問題を克服するため，オブジェクト指向技術を用いて構造解析システムの分析を行い，構造物・荷重・応答解析の3つのサブシステムに分離したモデルを提案する．これらは単独での拡張や使用が可能であるとともに，互いにメッセージ通信を行うことで，構造解析を行うことができるシステムである．各サブシステム内では，複数のアルゴリズムの適用が可能となるようオブジェクトモデルを構築している．これらの分析・実装を通じて，次世代の構造解析システムが備えるべき雛形を提案するものである．

Key Words : *object-oriented analysis, structural analysis, structure, equation, earthquake, UML*

1. はじめに

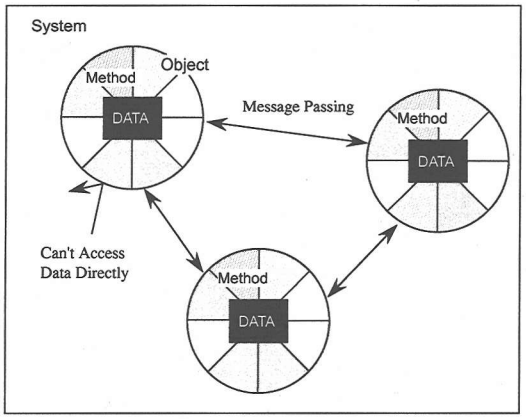
土木構造解析分野に関する知識はますます深まるばかりであり，現在では研究，教育，設計，施工などあらゆる分野においても数値計算は必須のツールとなっているのは言うまでもない．このようなコンピュータのめざましい進歩と普及にささえられ，土木構造工学分野においても大規模なソフトウェアが開発され，より複雑な構造物の挙動を解析できるようになっている．その一方で，システムが複雑化し，保守・管理や新たな手法の追加が困難となってきているのも事実である．情報システムの分野では，大規模で複雑な問題を取り扱う手法として，オブジェクト指向が有効であることが広く認識されている．しかしながらオブジェクト指向が構造解析分野へ適用されてから，まだ約10年の歴史しかない．

工学分野への適用は，まず線形静的有限要素法への適用から始まった．1990年にForde ら[1]は，平面要素を用いた線形解析プログラムに対し，NodeやMaterial, Elementといったクラスを抽出し，グラフィカルな表示によりシステムをモデル化した．わが国でも，1991年に三木ら[2),3)]が，トラス・梁の構造解析におけるオブジェクト指向技術の導入を研究している．

Fenves[4)]をはじめ多くの研究者は，プログラム中のデータを抽象化することの重要性を強調し，オブジェクト指向が工学分野のソフトウェア開発において有用であることを指摘している[5),6)]．従来より構造解析プログラムに用いられていたFORTRANでは，抽象データを効率よく表現できず，SmalltalkやC++などのオブジェクト指向プログラミング言語の利用が必要となってくる．Zimmermann・Pèlerin ら[7)]は，有限要素法におけるオブジェクトを抽出し，Smalltalkによる実装を行っている．FEM理論をほぼそのままプログラムコードで表現できることを示すとともに，動的問題への拡張を行っている[8)]．またC++による実装も行い，FORTRANによる解析と比較し，C++版の方が解析速度は劣るものの，プログラムの拡張やメインテナンスを含めた総合的観点から優れていることを示した[9)]．

有限要素法は各要素を組合せて構造物を表現するため，オブジェクト指向により表現しやすい．オブジェクト指向方法論が確立する以前には，有限要素の和として表現される「構造物」の静的構造のモデル化が研究の中心であった[1),7),10)]．一方で構造物や荷重などの大きな単位だけでなく，非線形問題で主要な役割をなす材料構成則[11),12)]や，行列・テンソル[13),14)]などについても，オブジェクト指向の適用が進められている．

1990年代前半には，数多くのオブジェクト指向方法論が提案されてきた[15),16),17),18),19),20)]．これは，オブジェクト指向が単にプログラミングの技術だけでなく，問題の分析から設計においても有用であることが明らかになってきたからである．この中で有力な方法論であるBooch法[18)]，OMT法[19)]，OOSE法[20)]が統合し，1997年にUML(Unified Modeling Language)が提案された．その後標準化団体であるOMG(Object Management Group)により，UMLがオブジェクト指向モデリング言語標準として承認され，業界標準となった．1999年10月にUML 1.3[21),22)]が発表されている．構造工学分野においても，これら方法論の適用が進み，単にオブジェクト



図–1 オブジェクト指向システム

の静的構造の分析から，動的・機能的モデルを含んだシステムの構築が進んできた[23),24),13),25),26),27)]．石田・多賀ら[28),29),30),31)]は，特に動的・機能的側面に注目し，オブジェクト指向分析を通して，非線形問題や動的問題に対してもその有効性を検証している．

近年，システム設計のキーコンセプトとして，個々のタスクを個別のオブジェクトへ分離することが強調されている．そこで種々の解析に対応できるよう，計算アルゴリズムもまたモデル化し，有限要素とアルゴリズムを分離することで，柔軟なシステムを構築する方法が主流になりつつある[32),33),27)]．非線形計算等のアルゴリズムは「もの」として認識しにくいものであるため，オブジェクト指向分析が進められるようになったのはごく最近であり，線形反復法[34)]，Regula Falsi 法[35)]やナブラ演算子[36)]などの適用がある．これら抽象的な存在に対する研究により，オブジェクト指向の適用範囲が大きく広がったといえる．

本研究では，構造解析システムを構造物・荷重・応答解析の3つのサブシステムに分け，これらが互いにメッセージ通信を行うことで構造解析を行うことができる柔軟なシステムモデルを提案する．特に構造物，応答解析においては，技術の発展に伴う新たな知見の導入を積極的に支援できるモデル化を提案し，次世代の構造解析システムが備えるべき知見を整理する．

2. オブジェクト指向アプローチ

現在広く用いられている構造解析システムのほとんどは，ある機能・アルゴリズムをサブルーチンとして分離し，これらを組み合わせて全体システムを構成する構造化プログラミング手法により作成されてきた．この手法はそのサブルーチンを入れ替えることにより，異なる問題に対応できるシステムを作成できるが，アルゴリズムのみに着目しているため，システム中にデータが分散し，一つの新たな機能の追加が結局全体の修正につながることも多く，保守性，拡張性が困難なものとなる．一方，オブジェクト指向アプローチでは，我々がイメージする「もの」を抽象化したオブジェクトを核としたシステム開発方法論である．このオブジェクトはデータとアルゴリズム (操作，メソッド) をカプセル化したもので，データにアクセスするためには，決められたメソッドを用いなければならないため，システムが堅牢となるだけでなく，機能の拡張に伴い他に与える影響も小さくなる．これらオブジェクト間でメッセージのやり取りを行うことで，システムが駆動する (図–1)．オブジェクト指向は構造化手法の拡張とも言えるが，それ以上にオブジェクト指向ならではの利点がある．

高度に構造化プログラミングを適用したシステムは，拡張性や保守性に関わる問題も小さくなるかも知れないが，アルゴリズムとデータが開発者の視点で切り分けられているため，作成されたシステムを利用者が実際にイメージすることはもはや困難となる．例えば単純な例である行列演算について考えてみる．

$$\mathbf{C} = \mathbf{B}^{\mathrm{T}} \cdot \mathbf{A} \cdot \mathbf{B} \tag{1}$$

上式は3つの行列の積を表しており，科学技術計算中には非常によく出てくる例である．この演算を FORTRAN で記述すると，アルゴリズムのみで表現されるため，我々が容易にイメージできる数式表現とはほど遠いものとなる (図–2)．一方オブジェクト指向ではデータとアルゴリズムを一体として考えられるため，ユーザの視点でシステムを切り分けることができる．行列のデータと演算を一体としたオブジェクトを利用して図–2 と同じプログラムを作成すると図–3 となる．これを見れば，一見してここで行われていることをユーザが理解することは可能であろう．オブジェクト指向は単にプログラミングのみならず，問題の分析から設計，プログラムに至るまで，一貫してユーザの視点によるもの（オブジェクト）をモジュール化したクラスによる表現ができるため，作成されたシステムの可読性はもちろん，保守・拡張性が高いものとなる．

このように行列演算だけを見てみても，科学技術計算におけるオブジェクト指向の優位点は明らかである．では，過去にも数多く研究が進められているものの，未だ構造解析分野で広く受け入れられていないのは何故か? この理由として，研究者は自分の研究分野と直接結び付かないと考えているオブジェクト指向技術を新たに学ばなければならないこと，また学んだとしても，それを実現するためには，研究者が馴染んでいるプログラミング言語である FORTRAN では困難であること，

```
   REAL A(10,10), B(10,5), C(5,5)
   REAL D(10,5)

   DO 10 I=1, 10
     DO 20 J=1, 5
       D(I,J) = 0.0
       DO 30 K=1, 10
         D(I,J)=D(I,J)+A(I,K)*B(K,J)
30     CONTINUE
20   CONTINUE
10 CONTINUE
   DO 40 I=1, 5
     DO 50 J=1, 5
       C(I,J) = 0.0
       DO 60 K=1, 10
         C(I,J) = E(I,J)+B(K,I)*D(K,J)
60     CONTINUE
50   CONTINUE
40 CONINUE
```

図–2 行列演算 (FORTRAN 版)

```
Gen_matrix A(10,10), B(10,5), C(5,5);

C = B.T() * A * B;
```

図–3 行列演算 (オブジェクト指向版)

などが挙げられる．しかし，数値計算による解析が中心となってきた現在，良質なソフトウェアの構築は，新たな研究成果を容易に実現するのを助け，保守・管理に伴う困難を排除してくれる．また，オブジェクト指向により分析された結果は，土木分野の専門家と他分野の専門家（例えばソフトウェア開発者）間の共通のコミュニケーション手法となり，他分野との知識共有を深めることができる．従って今必要なことは，標準化されたオブジェクト指向方法論による構造解析システム開発を行い，その優位性を示すことである．

3. 構造解析システムの分析

構造解析システムに対してオブジェクト指向分析を進めるにあたり，まずその概念モデルを作成する．問題記述から解析システム内外の識別を行い，3 つのサブシステムへと分割した解析システムを提案する．

本研究では，UML 1.3 による表記により分析を進める．UML 1.3 記法については，付録に整理してあるので，適宜参照されたい．

(1) 問題記述

本研究で対象とする問題領域は以下の通りである．

- 構造物に外力が作用したときの応答解析を行う解析システムを開発する．
- 外力としては，静的および地震力を考える．これらは時刻歴データとして取り扱う．
- 構造物は二次元を対象とし，その解法として有限要素法，ファイバーモデル，バネモデルが用いることができる．これらを複数組合わせても解析することができる．
- 静的・動的，線形・非線形問題を扱う．

(2) システム境界の識別

a) アクターの識別

アクターとは，システムとインターフェースを取るものである[37]．これは人や他のソフトウェア，ハードウェアなどが挙げられる．本解析システムにおいては，起動するのも，情報を取得するのもユーザであり，アクターは，ユーザーのみである．

b) ユースケースの識別

ユースケースとは，アクターがシステムに望むことを記述したものである．本システムにおけるユースケースの一部を列挙すると，

- 応答解析をする．
- 構造物データを設定する．
- 外力データを設定する．
- 解析結果を取得する．
- 解析法を設定する．

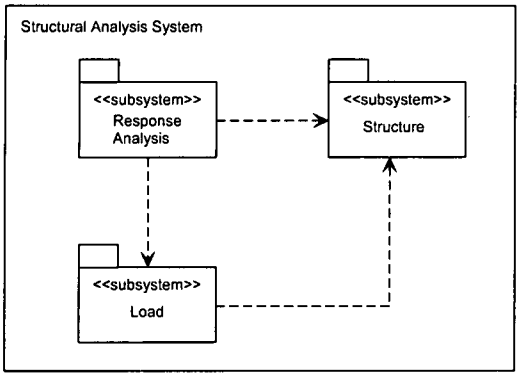
(3) システム内の識別

構造解析システムは，一般に大規模で複雑なものとなる．これらをサブシステムへと分割することができれば，その開発や保守が容易なものとなる．ここで注意すべきことは，システムをほぼ独立な部分へと分割することである．

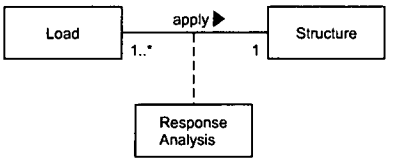
ユースケースを識別する際に，サブシステムの候補として，「構造物」，「荷重」そして「応答解析」が現れた．これらサブシステムの関係として，「構造物と荷重のデータを元に応答解析が行われる」，「地震データは構造物の情報を元に荷重（慣性力）へと変換される」が認識できる．構造解析システムにおけるサブシステムの認識と依存関係を示したものが，**図–4** である．

ここで，これらサブシステムの概要を整理する．

- 構造物サブシステム「Structure」
  構造物に関する情報を保持し，提供する．構造データの設定とともに，与えられた情報（構造データと変形データ）を基に，特性行列を作成する．
- 荷重サブシステム「Load」
  荷重に関する情報を保持し，提供する．力の単位で表される荷重の他に，地震（加速度）による地震力も扱い，その荷重履歴を表す．
- 応答解析サブシステム「Response Analysis」
  方程式を構築し，解を求める．



図–4 サブシステム



図–5 構造解析システムのクラス図

これらサブシステムは各種クラスで構成されるが，代表的クラスの関係をクラス図として表したものが図–5である．構造物は，それ自体では何もしない．ただ「もの」として存在するだけである．荷重（外力）が構造物に作用することによって，構造物は変形する．この変形状態を追跡することが，応答解析である．図–5は，「構造物に（複数の）荷重が作用する」という関連に付随する関連クラスとして，応答解析がある，という形に構造解析をモデル化していることを示している[38),39)]．

以降は，これらサブシステムを詳細に分析し，システムを構築していく．

4. 応答解析サブシステム

本システムにおける応答解析とは，結局のところいかに方程式を構築し，いかに解くか，ということになる．つまりこの方程式が応答解析の核となる．本節では，この方程式を取り上げ，サブシステムより細かいオブジェクトやクラスなどの要素のグループ化メカニズムである，パッケージとしてモデル化し，応答解析サブシステム以外でも使用できるように検討する．

(1) 方程式の表現

運動方程式の解法については，今まで数多くのものが提案されている．代表的なものとしては，Newmarkのβ法やWilsonのθ法の加速度法が挙げられる．また収束計算を必要とする陰的解法の他にもOperator Splitting法のような陽的解法もある．方程式パッケージでは，問題に応じて適当な解法を選択できるようにモデル化を行う．

方程式解法に関する知識を整理すると，線形・動的，線形・非線形方程式は次のように統一的に表現することができることが分かる．

$$\hat{\mathbf{K}}_{n+1}^{(k-1)}\Delta\mathbf{d}^{(k)} = \hat{\mathbf{R}}_{n+1}^{(k-1)} \tag{2}$$

ここで，

$$\hat{\mathbf{K}}_{n+1}^{(k-1)} = \hat{\mathbf{K}}_{dyn} + \mathbf{K}_{n+1}^{(k-1)} \tag{3}$$

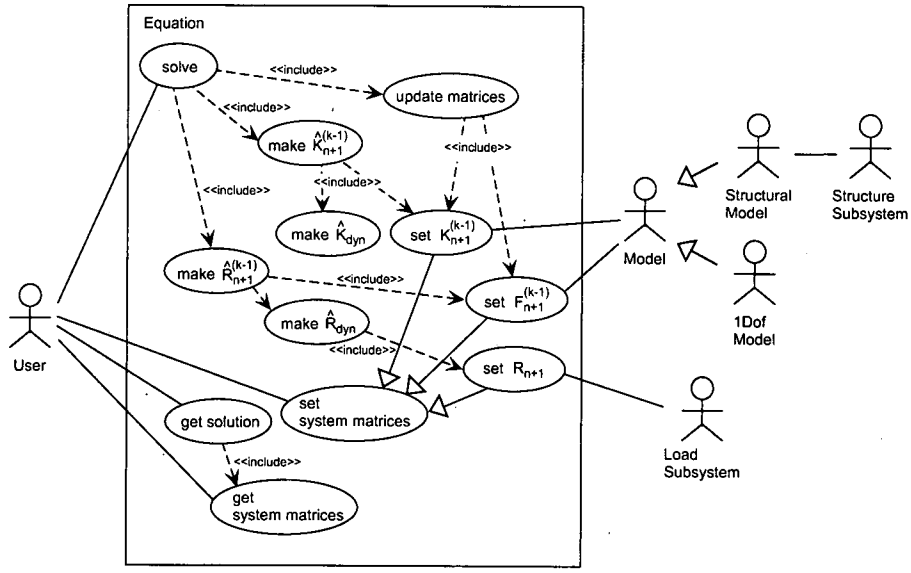
$$\hat{\mathbf{R}}_{n+1}^{(k-1)} = \hat{\mathbf{R}}_{dyn} - \mathbf{F}_{n+1}^{(k-1)} \tag{4}$$

式(2)～(4)は非線形形式で表しており，$\mathbf{K}_{n+1}^{(k-1)}, \mathbf{F}_{n+1}^{(k-1)}$はそれぞれ時刻$t_{n+1}$，繰り返し計算$(k-1)$ステップにおける構造物の剛性マトリクス，内力ベクトルである．また$\Delta\mathbf{d}^{(k)}$は繰り返し計算kステップにおける変位増分ベクトルである．静的問題では，$\hat{\mathbf{K}}_{dyn}$はゼロであり，$\hat{\mathbf{R}}_{dyn}$は荷重ベクトル$\mathbf{R}_{n+1}$そのものとなる．動的問題の場合には，$\hat{\mathbf{K}}_{dyn}$と$\hat{\mathbf{R}}_{dyn}$は共に解法によって決定され，$\mathbf{K}_{n+1}^{(k-1)}, \mathbf{F}_{n+1}^{(k-1)}$以外をまとめた項となる．

以上より，式(3)，(4)の右辺の4つの行列を，各解法アルゴリズムに従い作成することになる．線形問題では繰り返し計算は必要ないので，$k=1$の場合に相当する．解を求めるための基本となる方程式は式(2)であるので，時刻t_{n+1}における最終解$\mathbf{d}_{n+1}$は，$\Delta\mathbf{d}^{(k)}$の足し合わせにより得ることができる．

(2) 要求分析

方程式パッケージに課せられた要求は，「ある時刻において設定された特性行列に基づき解を求める」ことである．その他の要求としては，解の参照や特性行列の設定や参照がある．非線形問題では，剛性は時々刻々と変化し，設定された解析対象モデルにしたがって特性行列が作成されるので，方程式と解析対象モデルとを関連付ける必要がある．このモデルは，構造解析システムにおいては構造物サブシステムに対応するが，特性行列の作成法と非線形の取り扱いを備えている任意のオブジェクトを用いることができるようにすれば，汎用性の高いパッケージが作成できる．そこで方程式を表すオブジェクトと構造物サブシステムの間にModelオブジェクトを作成し，これを介して接続するようにモデル化する．また荷重サブシステムは外力の履歴を保持するものであるが，方程式パッケージはある時刻における解を求めるものであるので，荷重サブシステムとは特性行列を設定するユースケースのみと関連することになる．以上より，方程式パッケージのユースケース図は図–6となる．これより，ユーザ(User)は方程式オブジェクトにのみ線で結ばれていることからも



図–6 方程式パッケージのユースケース図

分かるように，構造物／荷重サブシステムやModelオブジェクトと直接関係しない．また方程式オブジェクトが有するユースケースでは，煩雑な非線形計算部などとは直接関係しないことから，利用性，保守性に富むものとなる．

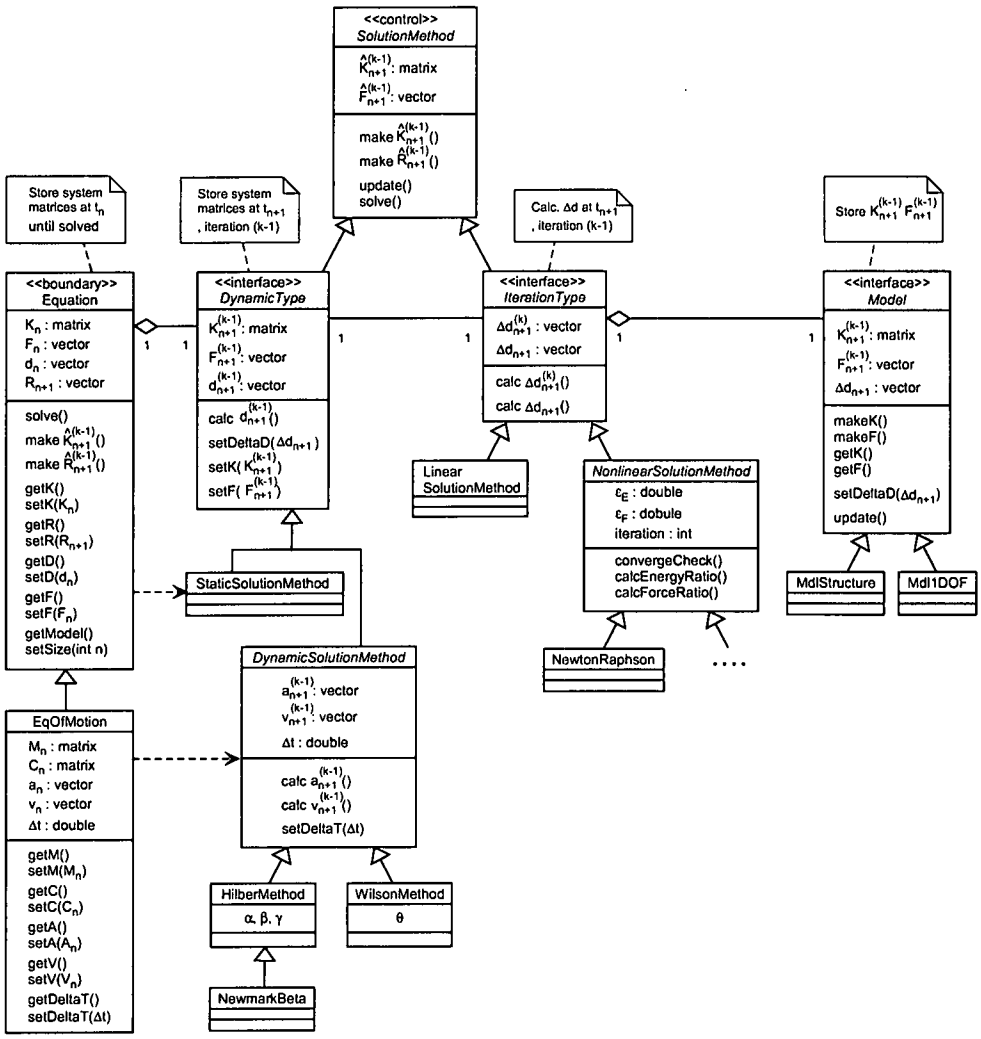
(3) オブジェクト指向分析

方程式パッケージのクラス図を図–7に示す．本パッケージの目的は「解を求める(solve)」ことであるが，この操作には，用いる解法により異なるアルゴリズムが存在する．この操作をモデル化するために，「アルゴリズム自体をカプセル化し，これらを交換可能にする」ために提案されているStrategyパターン[40]を用いてモデル化する．つまり操作solveのアルゴリズムを抽出し，交換可能としたものをSolutionMethodオブジェクトとして作成する．また要求分析の段階で導入したModelオブジェクトは，非線形性を扱うオブジェクトであり，方程式パッケージ内では解を得る時に用いられるので，SolutionMethodを継承したオブジェクト(IterationType)と関連することになる．

非線形計算においては，時刻t_n時の値とともに，収束計算中の値も必要とする．これらを効率的に保持するために，Equationオブジェクトに時刻t_n時の値を保持させておき，非線形計算中に変化する値についてはSolutionMethodオブジェクトが保持するようにモデル化した．

静的と動的問題とで必要となる特性行列の種類が異なるので，この部分をSolutionMethodを継承したDynamicTypeに保持させている．すなわちDynamicTypeには，式(4)における$\hat{\mathbf{K}}_{dyn}$, $\hat{\mathbf{R}}_{dyn}$の算出を受け持たせている．これらを算出するアルゴリズムもまた，用いる解法により異なるので，Strategyパターンを用いて分離している．一方，線形・非線形に関する操作はIterationTypeとしてまとめ，実際の求解操作を担当する．ここではDynamicTypeオブジェクトを通じて得た$\hat{\mathbf{K}}_{dyn}, \hat{\mathbf{R}}_{dyn}$を基に解$\Delta\mathbf{d}$を求めていく．ここで，図–7の代表的クラスについて，特徴を整理する．

- **Equation**
  剛性マトリクス・変位ベクトル・外力ベクトル・内力ベクトルにより構成される釣り合い方程式を表すクラス．ユーザが直接扱うことができる．方程式を解く操作を持つが，実際の求解はSolutionMethodオブジェクト（StaticSolutionMethodに依存）が担当する．これら各特性行列は，方程式を解く前には外力ベクトル以外は時刻t_nにおける値を保持しており，解が得られた後は，時刻t_{n+1}における値を保持することになる．図–7には<<boundary>>とラベル付けしているが，これはユーザとパッケージの境界としての役割を有することを示している．
- **EqOfMotion**
  運動方程式を表すクラス．Equationクラスを継承し，質量マトリクス・減衰マトリクス・加速度ベクトル・速度ベクトルおよび計算時間間隔を持つ．実際の解法はDynamicSolutionMethodに依存する．
- **SolutionMethod**
  計算をコントロールする抽象クラス．$\hat{\mathbf{K}}^{(k-1)}_{n+1}, \hat{\mathbf{R}}^{(k-1)}_{n+1}$を作成するためのデータ，操作のインターフェースを定義する．また計算が収束，終了することを

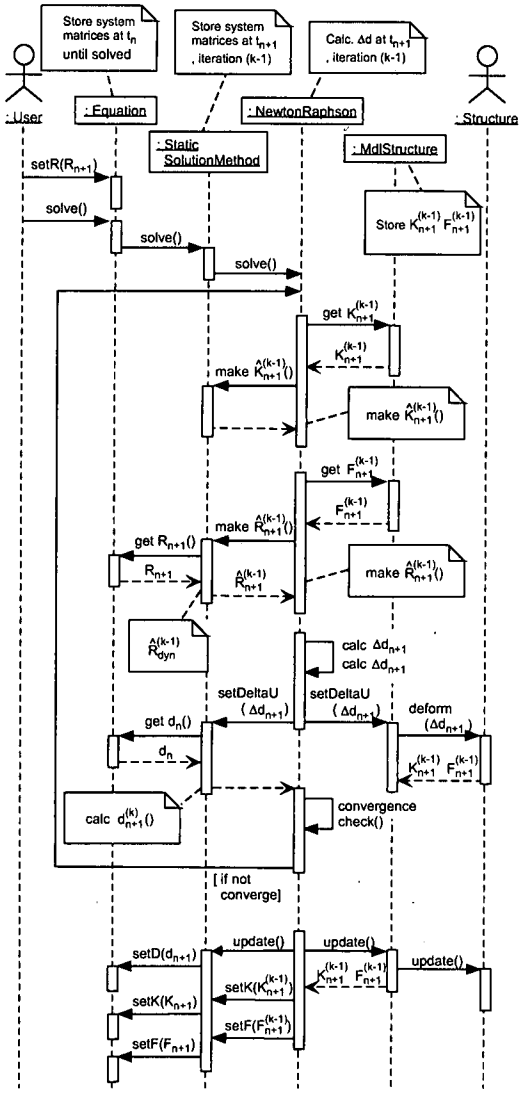


図–7 方程式パッケージのクラス図

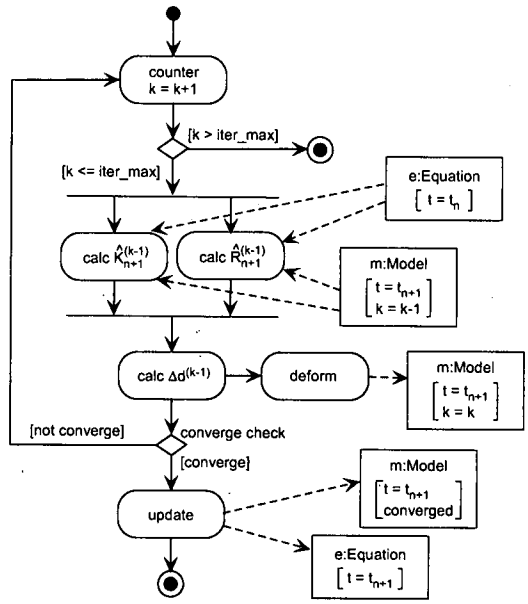
他のオブジェクトに通知するための update メソッドを有する．図–7 には<<control>>とラベル付けしているが，これは実際の求解作業を扱うことを強調するために示している．

- **Model**
  特性行列 $\mathbf{K}_{n+1}^{(k-1)}, \mathbf{R}_{n+1}^{(k-1)}$ の作成アルゴリズムを有するインターフェースクラス．変位増分を与えられると，それに対応した特性行列を作成する．
- **IterationType**
  解法アルゴリズムをカプセル化するインターフェースクラス．SolutionMethod クラスを継承する．DynamicType オブジェクトを用いて $\hat{\mathbf{K}}_{n+1}^{(k-1)}, \hat{\mathbf{R}}_{n+1}^{(k-1)}$ を作成し，解を求める操作の雛形を有する．繰り返し計算ステップにおける $\Delta\mathbf{d}^{(k-1)}$ を得るメソッドを有し，これを Model オブジェクトに通知する．図–7 には<<interface>>とラベル付けしているが，これは線形・非線形アルゴリズムに関するインターフェースを定義していることを示している．
- **DynamicType**
  静的・動的問題により $\hat{\mathbf{K}}_{n+1}^{(k-1)}, \hat{\mathbf{R}}_{n+1}^{(k-1)}$ の作成アルゴリズムが異なるため，その差を吸収するために導入するインターフェースクラス．SolutionMethod クラスを継承する．IterationType::solve() で必要となる $\hat{\mathbf{K}}_{n+1}^{(k-1)}, \hat{\mathbf{R}}_{n+1}^{(k-1)}$ の一部を作成する操作の雛形を定義する．計算ステップ中の $\mathbf{d}_{n+1}^{(k-1)}$ など各特性行列を保持する．図–7 には<<interface>>とラベル付けしているが，これは動的・静的解法に関する求解作業のインターフェースを定義していることを示している．

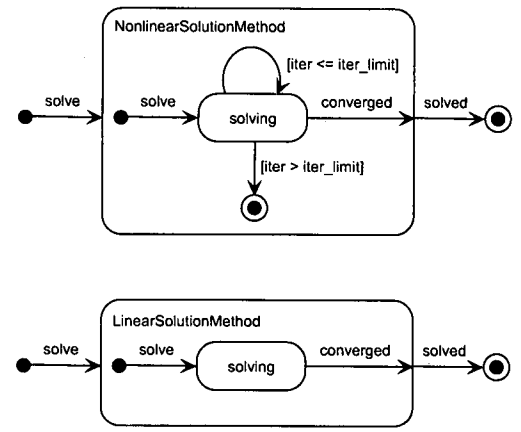
ここで動的振る舞いに着目し，静的問題において方



図–8 方程式パッケージのシーケンス図



図–9 SolutionMethod のアクティビティ図



図–10 Equation パッケージのステートチャート図

程式パッケージが解を求めるアルゴリズムを，図–8 のシーケンス図に示す．求解アルゴリズムの中では，様々なデータが利用，生成されるが，これにより各々のデータをどのオブジェクトが責任を持つかについて整理することができる．すなわち，図中上のノート内にあるように，解を得るまで，時刻 t_n に関するデータを Equation が，繰り返し計算中に変化するデータを SolutionMethod が，特にモデルに関連する剛性行列と内力ベクトルは Model が管理する．このようにモデル化することにより，効果的にデータ分散を図ることができることが分かる．

同様に SolutionMethod の求解におけるアルゴリズムを，他のオブジェクトの状態変化を示しながらアクティビティ図により表現したのが図–9 である．方程式パッケージでは，各データを機能に応じたオブジェクトに分散させているので，このような単純なアクティビティで全ての問題に対応することができる．

次に Equation が取りうる状態に着目し，線形・非線形問題をどのように扱っているかを示したのが図–10 である．アクターとの境界にある Equation オブジェクトの求解自体は，線形・非線形に関係なく，求解要求のイベントが生じると Equation は求解状態になり，解を得るか解を得ない最終状態へと移行するだけである．ただし Equation の求解操作は，実際には IterationType がコントロールしているが，NonlinearSolutionMethod と LinearSolutionMethod の部分を取り替えれば同じ図になるように，Strategy パターンによるモデル化により線形・

```
 1: /*-------------------------------------------------------------------------*/
 2: void Equation::solve() {
 3:  sMethod->solve();                          // DynamicType へとメッセージを転送
 4: }
 5: /*-------------------------------------------------------------------------*/
 6: void DynamicType::solve() {
 7:  iterationType->solve();                    // IterationType へとメッセージを転送
 8: }
 9: /*-------------------------------------------------------------------------*/
10: void LinearSolutionMethod::solve() {
11:  setUp();                                   // 初期設定
12:  calcDeltaD();                              // K_{n+1}, F_{n+1} を計算し, Δd_{n+1} を計算
13:  model->setDeltaD(deltaD);                  // Δd_{n+1} を構造モデルへと通知
14:  storePreData(deltaD);
15:  storeDataAtT(deltaD);                      // 時刻 t_{n+1} におけるデータを保存
16:  model->update();                          // 構造モデルの諸量を更新
17: }
18: /*-------------------------------------------------------------------------*/
19: void NonlinearSolutionMethod::solve() {
20:  setUp();                                   // 初期設定
21:  iteration = 0;                            // 繰り返し回数初期値
22:  accum_dD = 0;                             // Δd_{n+1} 初期値
23:  do {                                      // 繰り返し計算開始
24:   iteration++;
25:   calcDeltaD();                            // K^{(k-1)}_{n+1}, F^{(k-1)}_{n+1} を計算し, Δd^{(k)}_{n+1} を計算
26:   checkEnergyConv();                       // エネルギー収束判定指標の計算
27:   accum_dD += deltaD;                      // Δd_{n+1} を計算
28:   model->setDeltaD(accum_dD);              // Δd_{n+1} を構造モデルへと通知
29:   storePreData(accum_dD);                  // 繰り返し回数 k におけるデータを保存
30:   checkForceConv();                        // 荷重収束判定指標の計算
31:   if(iteration>=20) {                      // 繰り返し回数による発散判定
32:    cout << "\nDivergence..." << endl;      // エラーメッセージ
33:    exit(1);                                // 強制終了
34:   }
35:  } while(convergenceCheck()!=1);           // 収束判定
36:  storeDataAtT(accum_dD);                   // 時刻 t_{n+1} におけるデータを保存
37:  model->update();                          // 構造モデルの諸量を更新
38: }
39: /*-------------------------------------------------------------------------*/
```

図–11 Equation パッケージのメソッド solve() の実装

非線形計算の差を吸収している．これにより，Equation オブジェクトを利用するユーザーは，異なるアルゴリズムであっても，全く同じように利用することが可能となる．

(4) 実装

a) Equation クラスライブラリ

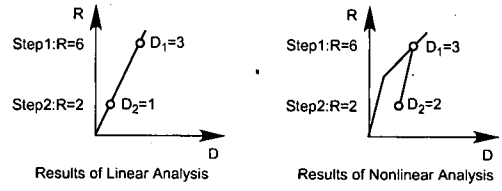
分析・設計された方程式パッケージを C++言語により実装した．クラス構造は，分析した結果である図–7 をそのままプログラムへと変換することで実現できる．

ここでは，Equation パッケージのキーメソッドである，求解メソッド solve の実装について説明する．全プログラム中で，メソッド solve に関する部分を抜き出したのが図–11 である．図中には説明のため，行番号を付加している．ここで，10 行，19 行にある Linear/NonlinearSolutionMethod は，IterationType を継承したクラスであり，線形／非線形解法の求解メソッドを具体化しているものである．ここでプログラム中によく現れている記号「::」,「->」の後ろはメソッド名を表し，その前はそれぞれそのメソッドを有するクラス，オブジェクトを表している．また「//」は注釈である．その他 C++言語の詳細については本論文の範囲を超えるので，成書に譲る [例えば41),42)]．

まず，ユーザとのインターフェースとなる Equation では，先の分析にもあるように，具体的な解法は持っていない．従って，Equation にメッセージ solve が送られると (2 行目)，DynamicType（ここでは sMethod）へと委譲し (3 行目)，この DynamicType 内でもまた，IterationMethod へと委譲している (7 行目)．ここで sMethod や iterationType は，プログラム実行時に具体的なオブジェクトと関連づけられるものであり，LinearSolutionMethod(10 行目) あるいは NonlinearSolutionMethod(19 行目) のメソッドが実行される．つまり Equation などには直接アルゴリズムを持つようにしていないため，新たなアルゴリズムを追加しても，これらのクラスは全く修正する必要はない．

線形／非線形解法アルゴリズム (10 行目／ 19 行目より) を見ると，図–8 のシーケンス図等と対応していることが分かる．非線形の取り扱いについては，Model を通じて Equation パッケージの外で実施されるので (13

```
 1: #include <fstream.h>                                  // 各種定義の読み込み
 2: #include "equation.h"
 3: #include "mdlusr.h"
 4: #include "hyst.h"
 5:
 6: int main()                                           // メインプログラム開始
 7: {
 8:   Hysteresis *hyst2 = new HysLinear(1.0, 2.0);       // 線形履歴モデルの定義
 9:   Hysteresis *hyst2 = new HysBilinear(1.0, 4.0
                                    , 3.0, 6.0);        // バイリニア履歴モデルの定義
10:
11:   Equation *eqL = new Equation(new MdlDof1(hyst1)
                  , new LinearSolutionMethod
                  , new StaticSolutionMethod);          // 静的線形方程式の設定
12:   Equation *eqN = new Equation(new MdlDof1(hyst2)
                  , new NewtonRaphson
                  , new StaticSolutionMethod);          // 静的非線形方程式の設定
13:
14:   int num_dof = 1;                                  // 自由度設定
15:   Col_vector R(num_dof), D(num_dof);                // 荷重，変位ベクトルの定義
16:
17:   // ------------------------ Step 1: -----------------------------
18:   R[1] = 6.0;                                       // 荷重設定（載荷）
19:   eqL->setR(R); eqN->setR(R);                       // 荷重ベクトルを方程式に設定
20:
21:   eqL->solve();                                     // 線形方程式を解く
22:   eqN->solve();                                     // 非線形方程式を解く
23:
24:   D = eqL->getD(); D.print("D_L");                  // 線形方程式から解を得て表示
25:   D = eqN->getD(); D.print("D_N");                  // 非線形方程式から解を得て表示
26:
27:   // ------------------------ Step 2: -----------------------------
28:   R[1] = 2.0;                                       // 荷重設定（除荷）
29:   eqL->setR(R); eqN->setR(R);                       // 荷重ベクトルを方程式に設定
30:
31:   eqL->solve();                                     // 線形方程式を解く
32:   eqN->solve();                                     // 非線形方程式を解く
33:
34:   D = eqL->getD(); D.print("D_L");                  // 線形方程式から解を得て表示
35:   D = eqN->getD(); D.print("D_N");                  // 非線形方程式から解を得て表示
36:
37:   return 0;                                         // 解析正常終了
38: }
```



図–12 静的線形・非線形解析プログラム例と結果

行目や28行目)，非線形解法アルゴリズムも単純なプログラムで実現できている．また本プログラムでは行列クラスライブラリを利用しているので，科学技術計算で多用される行列演算を効率よく記述できる（例えば27行目はベクトルの加算）．

b) 利用例

方程式パッケージを用いた，一自由度静的線形および非線形解析プログラムの全リストを図–12に示す．また実行結果を下部に示す．ここでは従来は全く異なる表現をせざるを得なかった線形，非線形プログラムがどの様に表現されるかを示すため，プログラム中に2つの方程式を表現している(11,12行目)．また非線形性を扱うオブジェクトとして，構造物サブシステムのところで後述する履歴オブジェクトを用いた(8,9行目)一自由度モデルオブジェクト(MdlDof1)を設定している(11,12行目)．

まず17行からは初期状態から載荷された時の変形を求めているが，21，22行目を見ると分かるように，線形・非線形のいずれの問題に対しても全く同じように表すことができる．

また27行以降は，載荷後除荷した状態を表している．このように履歴を伴う解析では，従来は履歴データをアルゴリズムに伝える必要があり，必要なデータもまた履歴特性により異なっていたため，非常に煩雑にならざるを得なかった．しかし本システムでは，履歴情報については履歴オブジェクトが管理しているため，方

程式オブジェクトやユーザが意識する必要がない．従って 22 行目と 32 行目を比べれば分かるように，載荷状態に関わらず，解を得るには全く同一の表現を取ることができる．

このように，プログラム中に方程式が陽な形で現れ，解法に関わらず同一の表現を取ることができることは，実際の数式と直接対応づけることができ，可読性が高いコードを作成できる．

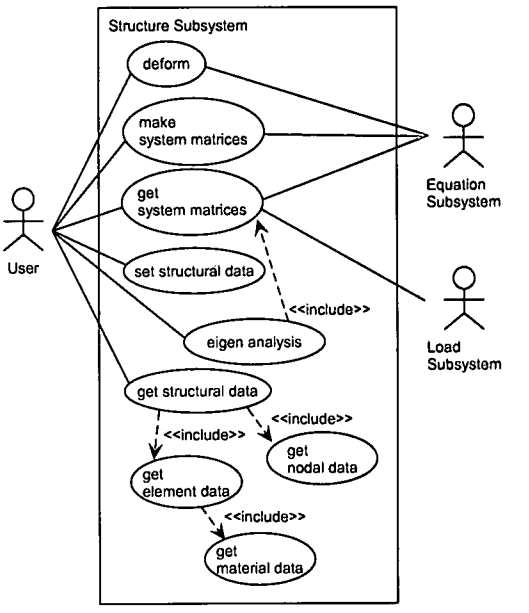
5. 構造物サブシステム

構造物はオブジェクト指向の適用に関する研究の当初より，モデル化が積極的に進められてきた．ただし従来の研究では，FEM のみを対象としているものがほとんどであり，構造物のモデル化は比較的簡潔にすることができた．しかし特性行列作成法については，有限要素法の他に，ファイバーモデル手法をはじめ数多くあり，今後さらに新しい方法が提案されるであろう．本論文では，これら種々の方法を効率よく，同時に扱えるようモデル化を進める．

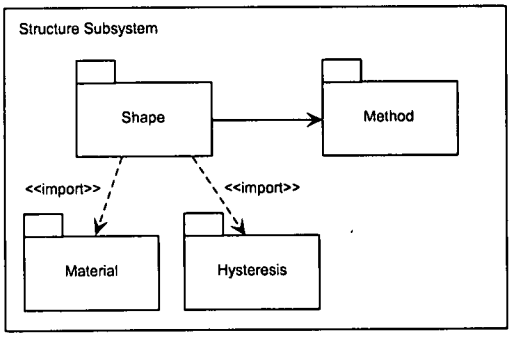
(1) 要求分析

本論文におけるモデル化では，応答解析に関する知識は，応答解析サブシステムに委譲してあるため，従来システムにおける Solver に相当する知識を本サブシステムに含める必要はない．したがって構造物サブシステムに課せられた要求は，「現実の構造物の形状をできるだけ忠実にモデル上で再現すること」および「状態に応じて，設定されたモデリング手法に基づき，特性行列を作成すること」である．その他要求に関するとともにユースケースを表したのが，**図–13** である．

この中でポイントとなるユースケースは「特性行列を作成する (make system matrices)」であるが，各手法ごとに独自のユースケースを作成するのは，煩雑となる．モデリング手法により特性行列の作成法には種々の方法があるとしても，我々が構造物をまず認識するのは，その「形状」であり，これは特性行列作成法と関係がない．そこで本論文では，構造物サブシステムを，まず形状 (Shape) とモデリング手法 (Method) のパッケージへと分離し，Strategy パターンによりモデル化する[43](**図–14**)．**図–14** 中には，材料 (Material) と履歴 (Hysteresis) モデルもまたパッケージとして分離している．これは材料，履歴パッケージは，本論文で作成するシステムのみならず，他のシステムにおいても利用できる汎用的なものであると考えるため，単独での使用ができるようにするためである．以降，本論文では形状およびモデリング手法パッケージについて分析を進める．



図–13 構造物のユースケース図



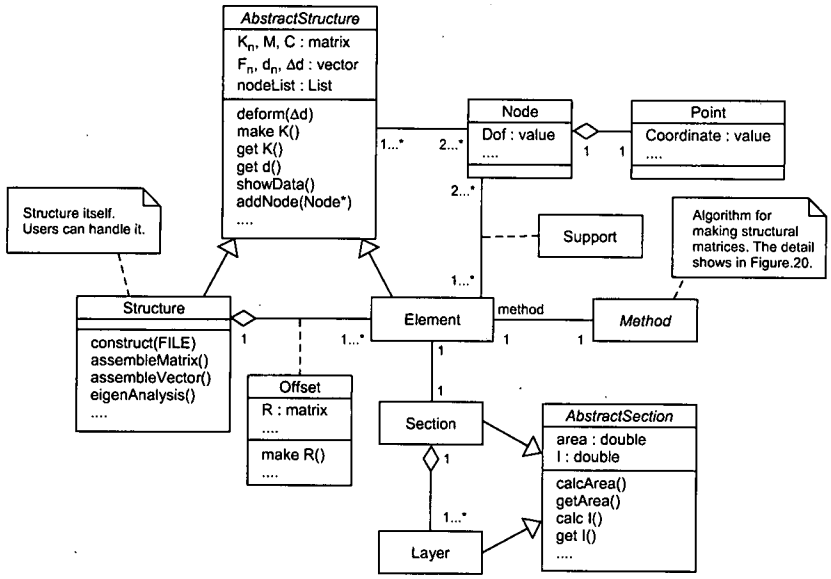
図–14 構造物サブシステム

(2) 形状パッケージに関する分析

現実世界における構造物は，例えば橋梁では，橋脚や桁のような種々の部材の集合体である．ただし，これら実構造を解析するにあたっては，解析モデルへと簡略化するのが一般である．本形状パッケージでは，この段階の構造形態を対象としている．

ここではフレーム構造物を対象とする．従って構造物は線部材で構成される．形状におけるオブジェクトを抽出すると，構造物に対して Structure クラスを，部材に対して Element クラスが認識でき，Structure が Element を複数保持する形でモデル化する．また Structure と Element には属性と操作において次のような共通点がある．

- 剛性マトリクスや等価節点力ベクトル等の特性行列を作成する
- 複数の節点を有している

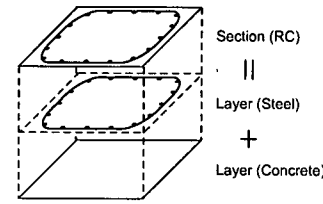


図–15 形状に関するクラス図 (その 1)

- 変形する

従って，Structure と Element を汎化した親クラス (AbstractStructure) を作成し，両者はこれを継承する形でモデル化する．以上の内容についてモデル化したものが図–15 である．ここで，主要なクラスについて整理する．

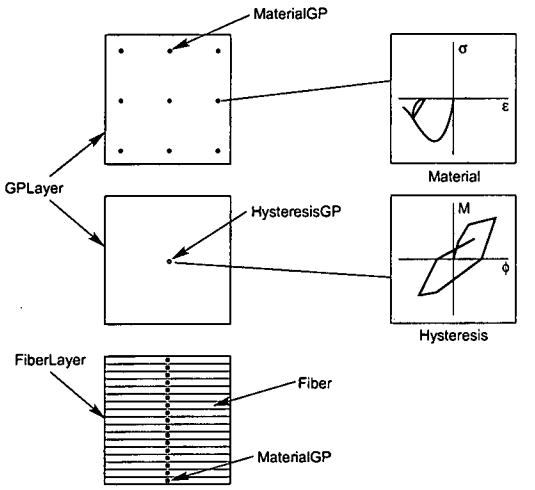
- **AbstractStructure**
  構造物一般を表す抽象クラス．構造物の座標を決定するための節点のリストや，特性を表す剛性マトリクスや等価節点力ベクトル・質量マトリクスなどを属性として保持する．また，それらの特性行列を作成する，変形する等の操作を持つ．
- **Structure**
  構造物を表現するオブジェクトクラス．自分を構成する部材のリストを持っている．特性行列を作成する際には，各部材ごとに作成された特性行列を重ね合わせて作成する．固有値解析を行うことができる．
- **Element**
  構造物を構成する部材を表すオブジェクト．解析方法パッケージ (Method) を用いて剛性マトリクスと等価節点力ベクトルを作成する．また，質量マトリクスも作成する．自分自身の断面形状 (Section) に関する情報を保持している．
- **Point**
  絶対座標を保持するクラス．
- **Node**
  節点を表すクラス．自由度に関する情報を有する．構造物や部材はこの Node の集合として表現することができるが，その座標は Point クラスを参照することにより得ることができる．



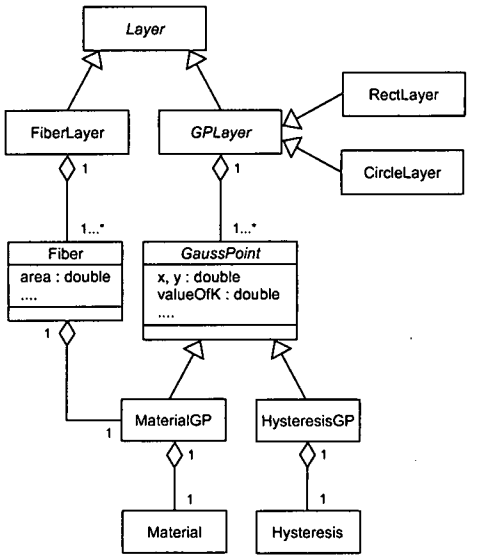
図–16 Section と Layer の関係

- **Section**
  部材の断面を表す．断面積，断面 2 次モーメントなどの断面に関するデータを有する．複数の Layer により構成され，この Layer を参照して断面積等を算出する．複数の Layer により断面を表すことで，複合構造に対応することができる．
- **Layer**
  断面内における 1 つの材料により表される層構造を一般化したもの (図–16)．断面積等を算出する操作を有する．
- **Offset**
  全体座標系と部材座標系とのオフセットを表す．回転行列を作成する操作を有する．

次に図–15 の Layer 以降の分析を進める．複数の解析法を適用できるようにするためには，Layer に異なる非線形特性を関連づける必要がある．そこで図–17 に示すように，非線形特性と断面を結び付けるものとして Gauss 点を設け，断面は FEM では材料と関連する



図–17 GaussPoint オブジェクトの概念図

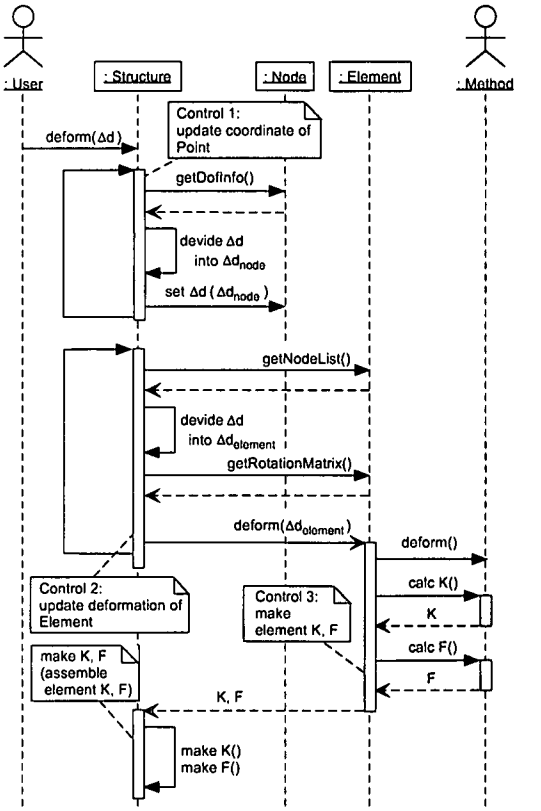


図–18 形状に関するクラス図 (その 2)

Gauss 点を複数個，モーメント–曲率法では履歴と関連する Gauss 点を 1 個保持するようにモデル化する．ファイバーモデル解析の場合には，断面は複数のファイバーにより構成され，ファイバーは材料特性の他に面積の情報も有する．

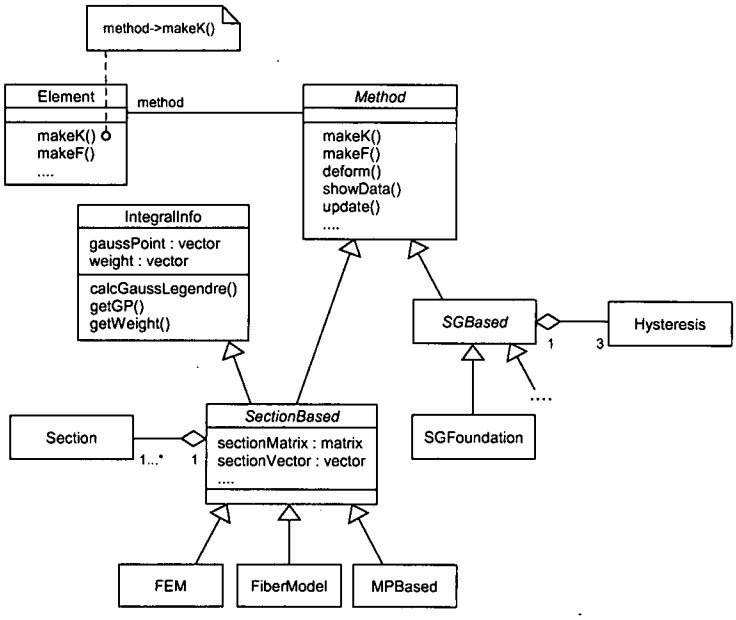
このように，非線形特性に関する部分まで分析を進めてくると，形状と解析方法を完全に分離することが困難となってくるが，必要最小限になるようモデル化を行った．以上の分析より，Layer 以降の形状に関するクラス図は図–18 のようになる．

- **FiberLayer**
  任意の断面の幾何学的形状をファイバーの集合として表すためのレイヤー．複数のファイバーを保持している．各ファイバーの面積の和から断面積を算出する．
- **GPLayer**
  計算点としてガウス点を用いる抽象レイヤー．複数のガウス点を保持しており，数値積分を用いて断面積等を算出する．
- **GaussPoint**
  数値積分するための計算点を表すもの．断面内における位置を属性として持ち，状態に応じた履歴情報を有する．
- **MaterialGP**
  材料の情報を持ったガウス点．
- **HysteresisGP**
  履歴特性の情報を持ったガウス点．
- **Fiber**
  ファイバーモデルの基本構成物．断面積を属性として持ち，材料に関する情報と断面内における位置を MaterialGP より得る．



図–19 形状パッケージのシーケンス図

これらオブジェクトが，ユーザーから変形命令を受けた場合のシーケンス図を図–19 に示す．変形は，まず構造物に変位増分ベクトルが与えられることにより生じる．この変位増分を，各節点や部材に関する増分へ



図–20 モデリング手法パッケージのクラス図

と変換し (図中 Control 1,2)，部材変位増分をまた，部材に対して変形に関するメッセージを送ることで対応する (Control 2)．個々の部材オブジェクトは，モデリング手法パッケージへ部材特性行列の作成を依頼し，値を受け取る (Control 3)．このようにモデリング手法パッケージは一種のブラックボックスとなっており，構造物はどのような手法と関連づけられているかを知る必要はない．形状パッケージが知っていることは，変形履歴に応じた節点の位置と特性行列のみである．このようにモデル化することで，新たなモデリング手法が追加されても本パッケージに何ら影響はない．

(3) モデリング手法パッケージに関する分析

今回対象とするモデリング手法は，有限要素法[44]・ファイバーモデル解析[45]・モーメント–曲率モデル・バネモデル (荷重–変位モデル) である．知識整理によって，有限要素法 (FEM)・ファイバーモデル手法 (FiberModel)・モーメント-曲率モデル (MPBased) の 3 つについては，部材断面における特性行列を作成さえすれば，軸方向の積分により部材特性行列を作成する，というアルゴリズムは同一であることが分かる．バネモデル (SGBased) については，部材特性行列の段階まで他のモデリング手法と共通性はない．モデリング手法に関する知識整理をもとに作成したクラス図を**図–20** に示す．本図は**図–15** 中 Method 以降を詳細分析した結果を示している．

モデリング手法パッケージでは具体的に特性行列を作成することが目的となる．そこでモデリング手法オブジェクトの親オブジェクトとして，Method クラスを作成し，モデリング手法一般に共通する属性や操作を保持させる．具体的なモデリング手法は，Method クラスを継承している FEM などである．Element オブジェクトは，特性行列を作成するに当っては，Method オブジェクトを通じて行うことができるので (**図–20** 上のノート : Element::makeK() 参照)，新たなモデリング手法が導入されても，形状パッケージに及ぼす影響はない．

具体的なモデリング手法オブジェクトについては，有限要素法・ファイバーモデル手法・モーメント–曲率モデルの親クラスとして SectionBased クラスを作成する．形状パッケージでは，Element オブジェクトと Section オブジェクトは一対一の関係であったが，モデリング手法パッケージでは軸方向の断面の非線形履歴を保持する必要があるため，Element を通じて得た Section をコピーし，複数の Section オブジェクトを保持している．また，数値積分を行うので，積分点や重みを計算する役割を持つ IntegralInfo クラスを作成し，SectionBased クラスはこれも継承する．バネモデルは SGBased クラスで表現し，基礎バネや軸方向・軸直角方向・回転方向のそれぞれに特化したサブクラスにより表現する．

各モデリング手法オブジェクトの動的振る舞いについては，個々のアルゴリズムに従う．

(4) 実装

a) Structure クラスライブラリ

分析・設計した構造物サブシステムを C++言語により実装した．クラスの静的構造については，方程式パッケージと同様，分析結果をほぼそのままプログラムへ

```
 1: /*-------------------------------------------------------------------*/
 2: Col_vector Structure::deform(Col_vector du) {
 3: //----------------------
 4: // 図 19 の Control 1 に相当
 5: //----------------------
 6:  Col_vector disp(3);
 7:  for(int i=1; i<=NumNode; i++) {            //-------------------------
 8:   Node *node = (*NodeList)(i);              //
 9:   for(int j=1; j<=3; j++) {                // Node が有する自由度を
10:    if(node->getDof()->value(j)==0) {       // 参照しながら，構造物変位
11:     disp[j] = 0.0;                         // 増分ベクトル Δd を Node 移動
12:    } else {                                // 変位量 Δd_node に変換する
13:     disp[j] = du[node->getDof()->value(j)]; //
14:    }                                       //
15:   }                                        //-------------------------
17:   node->getPosition()->addCoordinate(disp[1], disp[2]); // Node を移動
18:  }
19: //----------------------
20: // 図 19 の Control 2 に相当
21: //----------------------
22:  Col_vector inc_disp(6); disp.setsize(6);
24:  ObjectList<Node> *nodeList;
25:  for(int i=1; i<=NumElem; i++) {
26:   nodeList = ((*ElementList)(i))->getNodeList();// Element の NodeList を得る
27:   for(int j=1; j<=nodeList->getSize(); j++) {  //--------------------
28:    for(int idegree=1; idegree<=3; idegree++) { //
29:     int irow = (((*nodeList)(j))->getDof())    //
30:              ->value(idegree);                 // Element を構成する Node の
31:     if(irow==0) {                              // 自由度を参照しながら
32:      inc_disp[3*(j-1)+idegree] = 0.0;          // 構造物変位増分ベクトル
33:      continue;                                 // Δd を部材変位増分ベクトル
34:     }                                          // Δd_element へと変換する
35:     inc_disp[3*(j-1)+idegree] = du[irow];      //
36:    }                                           //
37:   }                                            //--------------------
38:   disp = ((*ElementList)(i))->getR()*inc_disp; // 回転行列 R を乗じて部材
                                                      座標系による増分に変換
39:                                               //--------------------
40:   ((*ElementList)(i))->deform(disp);          // 部材に変形を通知
41:  }
42:  return du;
43: }
44:
45: /*-------------------------------------------------------------------*/
46: Col_vector Element::deform(Col_vector du) {
47: //----------------------
48: // 図 19 の Control 3 に相当
49: //----------------------
50:  deltaD = du;
51:  method->deform();              // モデリング手法オブジェクトに変形を通知
52:  makeF();                       // method に従い内力ベクトルを設定
53:  makeK();                       // method に従い剛性行列を設定
54:  return deltaD;
55: }
56:
57: /*-------------------------------------------------------------------*/
```

図–21 Structure モジュールのメソッド deform() の実装

変換している．ここでは Structure サブシステムのキーメソッドである，変形メソッド deform の実装について図–21 に示す．ここではオブジェクト分析結果をどのように実装しているかを示すため，敢えてプログラム全文を載せている．プログラム中に用いられている変数等を全て解説するのは困難であるため，ここでは図–19 や注釈を参照しながら全体の流れを理解して頂きたい．

Structure に変形要求メッセージと共に変位増分ベクトルが通知されると (2 行目)，まず Structure 内で各節点 (Node) の移動量と各部材 (Element) の部材座標系に変換した変形量を算出する (3~39 行目)．Structure の行うべき作業はここまでであり，後の作業は Element へと委譲している (40 行目より 46 行目が呼び出される)．このことで，新たな部材オブジェクトが追加されたとしても，形状パッケージ内の Structure を変更する必要はないことを補償している．

一方 Element に変形要求メッセージが通知されると，特性行列を作成する作業となるが，その作成については，Element に関連づけられているモデリング手法パッケージ (Method) へと委譲しており (51 行目)，詳細は Element は関係ない．このことにより，新たなモデリング手法が追加されたとしても，形状パッケージを変更する必要はない．

このようにして，部材やモデリング手法に関する追

```
 1: #include <fstream.h>
 2: #include "structure.h"
 3:
 4: int main()
 5: {
 6:   Structure *structure = new Structure(1); // ID:1 の structure の作成
 7:   structure->construct("input.dat");       // structure の設定
 8:   structure->showData(cout);               // structure データの表示
 9:
10:   structure->makeF();                      // 内力ベクトルの作成
11:
12:   Col_vector D(structure->getNumDof());
13:   D[1] = 1.0;                              // 変位ベクトルの設定
14:
15:   structure->deform(D);                    // structure の変形
16:
17:   structure->makeK();                      // 剛性行列の作成
18:   structure->makeF();                      // 内力ベクトルの作成
19:
20:   return 0;                                // 正常終了
21: }
```

図–22 構造物オブジェクトを用いたプログラム例

加や変更に変化を受けにくいサブシステムを実現している．

b) 利用例

図–22 に構造物オブジェクトを用いたプログラム例を示す．設定ファイル (input.dat) 内に構造物データおよび解析手法が記述してある (7 行目)．本サブシステムは，変形状態に応じた特性行列を作成するが (10 行目や 18 行目)，過去の変形履歴は構造物オブジェクト内部で管理されているため，ユーザはそれを意識する必要はなく，全く同じ表現を用いることができる．

6. 荷重サブシステム

荷重サブシステムは，外力一般を表し，構造物に作用するものとして考える．構造解析問題における役割としては，ある時刻における荷重を構造物に作用させることを表現することである．

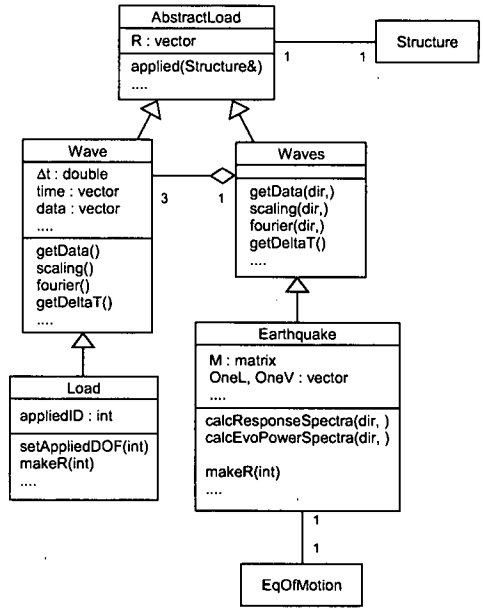
(1) オブジェクト指向分析

a) 静的荷重 (Load)

静的荷重オブジェクトは荷重データの履歴を表すため，内容はデータ列にすぎない．荷重オブジェクトは，構造物に作用してはじめて，荷重としての意味が生じてくる．これは構造解析においては荷重ベクトルという形で表現され，このベクトル作成のためには，静的荷重オブジェクトは構造物への作用点を知っていなければならない．

b) 地震荷重 (Earthquake)

地震波は加速度データの列であって，荷重ではない．しかし地震が構造物に作用すると，慣性力として地震荷重が発生する．それ故，地震荷重オブジェクトが荷重ベクトルを作成するためには，作用する構造物の質量行列などの構造データを知っている必要がある．



図–23 Load サブシステムのクラス図

一般に地震波は NS, EW, UD の 3 成分で表現される．これらはそれぞれ一方向の時刻歴である．我々はこの時刻歴の性質を知るために，このデータを用いてフーリエ解析などの計算を行っている．したがってまず最初に波形クラス (Wave) を作成し，3 つの Wave オブジェクトを保持するものとして Waves クラスを作成する．

地震波の性質は時刻歴やフーリエ解析だけでなく，応答スペクトルや非定常パワースペクトル，必要強度スペクトルなどで表現される．したがって地震クラス (Earthquake) は Waves クラスを継承して，独自に応答スペクトル等のメソッドを有するものとしてモデル化する．またこれらの計算には 1 自由度の運動方程式を解く必要

```
 1: #include <fstream.h>
 2: #include <stdlib.h>
 3: #include "structure.h"
 4: #include "load.h"
 5: #include "earthq.h"
 6:
 7: int main()
 8: {
 9:   // 構造物の設定
10:   Structure *structure = new Structure(1);  // ID:1 の structure の作成
11:   structure->construct("str_inp.dat");      // structure の設定
12:   structure->showData(cout);                // structure データの表示
13:
14:   // 静的荷重の設定
15:   Load* load = new Load;                    // load の生成
16:   load->setWaveDataWithTime("static.dat");  // データ列の読み込み
17:   load->setAppliedDOF(5);                   // 自由度番号 5 に作用する
18:   load->applied(*structure);                // 構造物に作用する
19:
20:   // 地震荷重の設定
21:   Earthquake* earthq = new Earthquake;      // earthquake の生成
22:   earthq->setOneDirWaveDataWithTime(NS, "kobe_ns.dat"); // データ列の読み込み
23:   earthq->applied(structure);               // 構造物に作用する
24:
25:   // 427 番目のステップ，時刻における荷重ベクトルの生成
26:   load->makeR(427);
27:   earthq->makeR(427);
28:
29:   // 地震オブジェクト単体の使用
30:   earthq->calcResponseSpectra(NS, "response.dat", 0.05);
31:
32:   return 0;
33: }
```

図–24 荷重・地震オブジェクトを用いたプログラム例

があるので，Earthquake クラスは方程式パッケージを再利用することにより求解作業を行うように分析・設計を行った．従って本サブシステム内に改めて方程式の求解作業に関する知識を書く必要はない．

以上の分析結果による荷重サブシステムのクラス図を，図–23 に示す．

(2) 利用例

荷重サブシステムを用いた利用例を，図–24 に示す．このプログラム中には，静的および地震荷重が表れており (15,21 行目)，構造物に作用している様子を示している (18,23 行目)．作用した結果，解析条件に合致した荷重ベクトルを作成することができる (26,27 行目)．

またプログラム中には地震パッケージの単独での使用も表現されており，このように応答スペクトル等の計算にも用いることができる (30 行目)．

7. 解析システムによる数値解析例

以上のように分析を行ってきたオブジェクト指向構造解析システムは，3 つのサブシステムとしてモデル化し，それぞれ単独での利用，拡張が可能であることを示してきた．ここで構造解析システムとしては，種々の問題に対して柔軟に対応できるよう，解析システム本体はメインプログラムとして実装し，個々のサブシステムはクラスライブラリとして整理する方式を取る．

作成したシステムを用いた地震応答解析プログラムの全リストを図–25 に示す．ここでは，構造物オブジェクト (12 行目)，地震オブジェクト (18 行目) そして運動方程式オブジェクト (25 行目) が生成され，これらが組み合わされることにより，解析システムが構成されているのが分かる．

このように本システムでは，各ユーザが対象とする問題に関してプログラムを作成する必要があるが，構造解析に特化したクラスライブラリを用いることで，高度な解析においても，図–25 のように可読性に富むプログラムを作成することができる．また各サブシステムについても，オブジェクト指向分析に基づいて実装されているため，その改良や再利用は容易なものとなっている．

8. まとめ

本研究では，高機能だが複雑になりつつある構造解析システムに対し，オブジェクト指向分析を行い，次世代構造解析システムを構築した．本研究を通じて，得られた知見は以下の通りである．

1. 構造解析システムを，「構造物サブシステム」・「荷重サブシステム」・「応答解析サブシステム」の 3 つのサブシステムに分離し，これらが互いにメッセージ通信を行うことで解析が進行するシステムを提案した．これを用いて作成されたプログラム

```
 1: #include <fstream.h>
 2: #include "equation.h"
 3: #include "structure.h"
 4: #include "mdlstr.h"
 5: #include "load.h"
 6: #include "earthq.h"
 7: #include "method.h"
 8:
 9: int main()
10: {
11:   // 構造物 (橋梁) の設定
12:   Structure *bridge = new Structure(1);        // 橋梁の生成
13:   bridge->construct("5span.dat");              // データの読み込み
14:
15:   bridge->eigenAnalysis();                     // 固有値解析
16:
17:  // 地震の設定
18:   Earthquake *earthq = new Earthquake;         // 地震の生成
19:   earthq->setOneDirWaveDataWithTime(NS,  "kobe_ns.dat"); // データの読み込み
20:
21:  // 地震が橋梁の作用
22:   earthq->applied(*bridge);
23:
24:  // 方程式を設定
25:   EqOfMotion *equation;
26:   equation = new EqOfMotion(new MdlStructure(bridge)
                              , new NewtonRaphson(
                                new NewmarkMethod(0.25)));
27:   equation->setM(bridge->getM());              // 質量行列を構造物オブジェクト
                                                   // から得る
28:   equation->setDeltaT(earthq->getDeltaT());    // 計算時間間隔を地震オブジェクト
                                                   // から得る
29:   int size = bridge->getNumDof();              // 総自由度数を構造物オブジェクト
                                                   // から得る
30:   // 地震応答解析
31:   Col_vector R(size), acc(size), dsp(size);    // 各種行列の生成
32:   ofstream ACC("accout.dat");                  // 出力ファイルの設定
33:   ofstream DSP("dspout.dat");
34:   for(int j=0; j<earthq->getNumData(); j++) { // 地震応答解析開始
35:     R = earthq->makeR(j);                      // 地震荷重より荷重ベクトルを得る
36:     equation->setR(R);                         // 方程式に荷重ベクトルをセット
37:
38:     equation->solve();                         // 方程式を解く
39:
40:     acc = equation->getA()+earthq->getData(NS,j)*One;  // 絶対加速度応答を得る
41:     dsp = equation->getD();                            // 変位応答を得る
42:
43:     ACC << ' ' << earthq->getTime(j) << ' ' << dsp << endl; // 解の出力
44:     DSP << ' ' << earthq->getTime(j) << ' ' << acc << endl;
45:   }
46:
47:   return 0;                                    // 正常終了
48: }
```

図–25 地震応答解析プログラム例

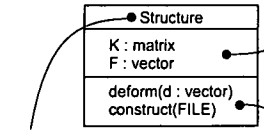
中には，我々がイメージできるオブジェクト (構造物など) が直接現れるため，可読性に富み，保守・管理が容易となる．このことは，新たな解法が求められる研究分野のみならず，システム全体の意味を容易に理解することができる点で，教育分野においても有利となる．

2. 「応答解析サブシステム」では，従来積極的にモデル化されていない方程式パッケージを核としたシステム構築を行った．ここで開発したパッケージは，他のクラスに影響を与えることなく，種々のアルゴリズムを利用できる．また単独での使用が可能なため，再利用性にも優れている．
3. 「構造物サブシステム」を，形状と解法パッケージへと分離することで，種々の解法を組み合わせた場合でも，全く同様に取り扱うことができるシステムを構築した．
4. 「荷重サブシステム」は，構造物オブジェクトと作用することで適切な荷重ベクトルを作成することができるよう構築した．また地震オブジェクトは単体でも使用することができ，応答スペクトルの計算等，他の研究領域でも使用することができる．
5. 構造解析システムの分析において，UML による表記を用いることにより，構造解析分野の研究者のみならず，システムエンジニアを含む情報工学分野の研究者とも，知識の共有を図ることができる．

謝辞：最後に，本研究を進めるにあたり京都大学防災研究所澤田純男助教授との討論は非常に有意義であった．ここに深く感謝の意を表します．

<<クラス>>

同一の属性，操作，関係などを共有しているオブジェクトの集合を記述したもの．



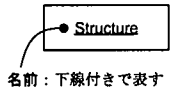
属性：クラスのデータを表す

名前：要素を識別するための文字列（斜体の場合は抽象要素を表す）

操作：クラスのサービスを表し，そのクラスの任意のオブジェクトがリクエストを受けて，そのオブジェクトの振る舞いに影響を与える

<<オブジェクト>>

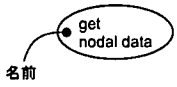
抽象概念の具体的な現れ．状態と振る舞いがカプセル化された実体で，クラスのインスタンス．



名前：下線付きで表す

<<ユースケース>>

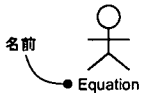
システムが実行する一連のアクションの組み合わせを，記述したもので，アクターに対して観察可能な意味のある効用や価値を結果として与えるもの．



名前

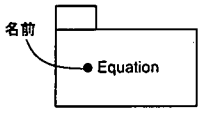
<<アクター>>

ユースケースのユーザがユースケースと相互作用するときに演じる，一貫性を持った役割．



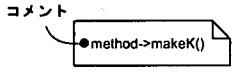
<<パッケージ>>

要素をグループに整理するための汎用のメカニズム．



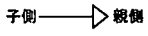
<<ノート>>

要素に付ける，制約またはコメントを表すシンボル．



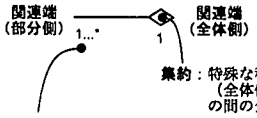
<<汎化>>

特化された要素（子）のオブジェクトを汎化された要素（親）に代入できるような関係．子は親の性質を継承し，より特化した要素となる．



<<関連>>

要素間の意味的な結合を表し，両者の集合関係を表す．

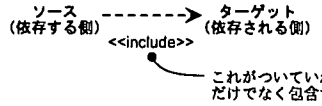


集約：特殊な種類の関連で，集約オブジェクト（全体側）とコンポーネント（部分側）の間の全体－部分関係を表す．

多重度：1つの集合で許される基数の範囲を指定するもの．
*は無限を表す．

<<依存>>

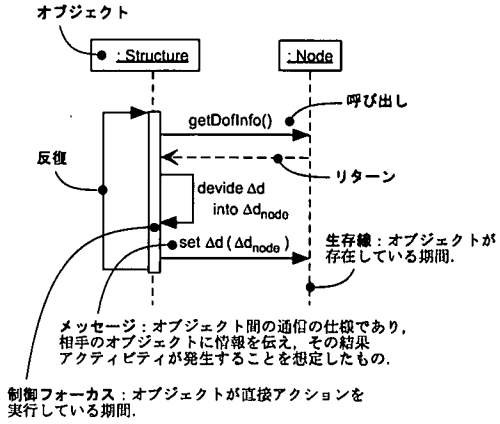
2つのものの間の意味的な関係で，一方の変更が他方に影響を与えることを表す．



これがついていれば，依存するだけでなく包含することを意味

<<相互作用>>

複数のオブジェクトの間で交換されて目的を遂行する一連のメッセージを包含している振る舞い．



<<状態マシン>>

オブジェクトが，その生存期間を通じ，イベントへの応答として通過する状態のシーケンスを，それらイベントへの応答と一緒に指定する振る舞い．

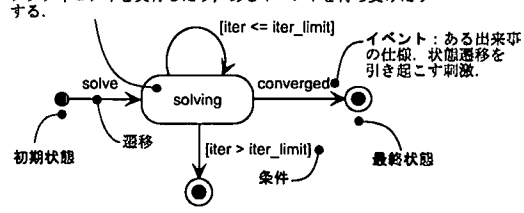


図-26 UML 記法

付録：UML 記法

図-26 に，本論文で用いた UML 1.3 による記法について整理する．

参考文献

1) Bruce W.R. Forde, Ricardo O. Foschi, and Siegfried F. Stiemer : Object-oriented finite element analysis, *Computer and Structures*, Vol. 34, No. 1, pp. 355–374, 1990.

2) 三木光範, 杉山吉彦, 内田雄治：オブジェクト指向によるはりの変形解析，日本機械学会論文集（A 編），57 巻 541 号, pp. 212 – 217, 1991.

3) 三木光範, 杉山吉彦, 内田雄治：オブジェクト指向によるトラスの変形解析，日本機械学会論文集（A 編），57 巻 541 号, pp. 218 – 223, 1991.

4) Gregory L. Fenves : Object-oriented programming for engineering software development, *Engineering with Computers*, Vol. 6, pp. 1–15, 1990.

5) Miller, G.R. : An object-oriented approach to structural analysis and design, *Computer and Structures*, Vol. 40, No. 1, pp. 75–82, 1991.

6) John W. Baugh Jr. and Daniel R. Rehak : Data abstraction in engieering software development, *Journal of Computing in Civil Engineering, ASCE*, Vol. 6, No. 3, pp. 282 – 301, 1992.

7) Thomas Zimmermann, Yves Dubois-Pèlerin, and Patricia Bomme : Object-oriented finite element programming : I. governing principles, *Computer Methods in Applied Mechanics and Engineering*, Vol. 98, pp. 291 – 303, 1992.

8) Yves Dubois-Pèlerin, Thomas Zimmermann, and Patricia Bomme : Object-oriented finite element programming : II. a prototype program in Smalltalk, *Computer Methods in Applied Mechanics and Engineering*, Vol. 98, pp. 361 – 397, 1992.

9) Yves Dubois-Pèlerin and Thomas Zimmermann : Object-oriented finite element programming : III. an efficient implementation in C++, *Computer Methods in Applied Mechanics and Engineering*, Vol. 108, pp. 165 – 183, 1993.

10) Pidaparti, R.M.V. and Hudli, A.V. : Dynamic analysis of structures using object-oriented techniques, *Computer and Structures*, Vol. 49, No. 1, pp. 149–156, 1993.

11) Ph. Menétrey and Th. Zimmermann : Object-oriented nonlinear finite element analysis: Application to J2 plasticity, *Computer and Structures*, Vol. 49, No. 5, pp. 767–777, 1993.

12) Foerch, R., Besson, J., Cailletaud, G., and Pilvin, P. : Polymorphic constitutive equations in finite element codes, *Computer Methods in Applied Mechanics and Engineering*, Vol. 141, pp. 355 – 372, 1997.

13) Jun Lu, D.W. White, Wai-Fah Chen, and Dunsmore, H.E. : A matrix class library in C++ for structural engineering computing, *Computer and Structures*, Vol. 55, No. 1, pp. 95–111, 1995.

14) Boris Jeremić and Stein Sture : Tensor objects in finite element programming, *International Journal for numerical methods in engineering*, Vol. 41, pp. 113–126, 1998.

15) Sally Shlaer and Stephen J. Mellor : *Object-Oriented Systems Analysis : Modeling the World in Data*, Yourdon, 1988.

16) Peter Coad and Edward Yourdon : *Object-Oriented Analysis*, Yourdon, 1991.

17) Peter Coad and Edward Yourdon : *Object-Oriented Design*, Yourdon, 1991.

18) Grady Booch : *Object-Oriented Analysis and Design with Applications, Second Edition*, Addison-Wesley, 1994.

19) James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen : *Object-Oriented Modeling and Design*, Prentice Hall, 1991.

20) Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Overgaard : *Object-Oriented Software Engineering : A Use Case Driven Approach*, Addison Wesley, 1992.

21) Grady Booch, James Rumbaugh, and Ivar Jacobson : *The Unified Modeling Language User Guide Version 1.3*, Addison-Wesley, 1999.

22) James Rumbaugh, Ivar Jacobson, and Grady Booch : *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1999.

23) Dietrich Hartmann, Arnd Fischer, and Peter Holéwik : Object oriented modeling of structural systems, In *International Conference on Computing in Civil and Building Engineering*, Vol. 5th, pp. 78 – 85, 1993.

24) Ziga Turk, Tatjana Isaković, and Matej Fishinger : Object-oriented modeling of design system for RC buildings, *Journal of Computing in Civil Engineering, ASCE*, Vol. 8, No. 4, pp. 436 – 453, 1994.

25) Mackie, R.I. : Using objects to handle complexity in finite element software, *Engineering with Computers*, Vol. 13, pp. 99 – 111, 1997.

26) Friedrich, J. : Object-oriented model and code for the visual examination of subsurface structures under historical buildings, *Computer and Structures*, Vol. 69, pp. 85–94, 1998.

27) Archer, G.C., Fenves, G., and Thewalt, C. : A new object-oriented finite element analysis program architecture, *Computer and Structures*, Vol. 70, pp. 63–75, 1999.

28) 石田栄介, 新美勝之, 福和伸夫, 中井正一, 多賀直恒 : 部分法を活用した有限要素動的構造解析のオブジェクト指向プログラミング, 構造工学における数値解析法シンポジウム論文集, 第 17 巻, pp. 471 – 476, 1993.

29) 石田栄介, 新美勝之, 福和伸夫, 中井正一 : 静的線形有限要素解析のオブジェクト指向分析と設計, 構造工学論文集, Vol. 40B, pp. 243 – 251, 1994.

30) 福和伸夫, 小磯利博, 田中清和, 石田栄介 : オブジェクト指向による構造物 - 地盤系の地震応答問題の分析, 応用力学連合講演会, 第 43 回, pp. 257 – 260, 1994.

31) 多賀直恒 : オブジェクト指向による地震都市防災システムに関する研究, 科学研究費研究成果報告書 研究課題番号 05452249, 1996.

32) Michael D. Rucki and Gregory R. Miller : An algorithmic framework for flexible finite element-based structural modeling, *Computer Methods in Applied Mechanics and Engi-*

neering, Vol. 136, pp. 363 – 384, 1996.

33) Rucki, M.D. and Miller, G.R. : An adaptable finite element modelling kernel, *Computer and Structures*, Vol. 69, pp. 399–409, 1998.

34) Are Magnus Bruaset and Hans Petter Langtangen : Object-oriented design of preconditioned iterative methods in Diffpack, *ACM Transactions on Mathematical Software*, Vol. 23, No. 1, pp. 50 – 80, 1997.

35) Phillips, J.B., Price, G., Fry, S., Arcziscewski, T., DeMonsabert, S., and Menawat, A.S. : An object-oriented approach to numerical methods: the Regula Falsi method for solving equations with tight tolerances for environmental applications, *Journal of Hazardous Materials*, Vol. B:63, pp. 145 – 162, 1998.

36) 棚橋隆彦, 中井太次郎 : オブジェクト指向 FEM のための離散化ナブラ演算子, 日本機会学会論文集 (B 編), Vol. 62, No. 595, pp. 204 – 212, 1996.

37) Geri Schneider and Jason P. Winters : *Appling Use Cases*, Addison-Wesley, 1998.

38) 高橋良和, 五十嵐晃, 家村浩和 : 耐震工学へのオブジェクト指向技術の適用に関する 1， 2 の考察, 土木情報システム論文集, Vol. 5, pp. 123–130, 1996.

39) Yoshikazu Takahashi, Akira Igarashi, and Hirokazu Iemura : Application of object-oriented approach to earthquake engineering, *Journal of Civil Engineering Information Processing System , JSCE*, Vol. 6, pp. 271–278, 1997.

40) Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides : デザインパターン, SOFTBANK, 1995.

41) Bjarne Stroustrup : *The C++ Programming language*, Addison-Wesley, 1984.

42) Bjarne Stroustrup : *The C++ Programming language, Second Edition*, Addison-Wesley, 1991.

43) 藤田亮一, 高橋良和, 家村浩和 : オブジェクト指向構造解析システムにおける 構造物モジュールの構築, 土木情報システム論文集, Vol. 6, pp. 119–126, 1997.

44) Bathe, K.J. : *Finite Element Procedures*, Prentice Hall, 1996.

45) Ristić, D., Yamada, Y., and Iemura, H. : Nonlinear behaviour and stress - strain based modeling of reinforced concrete structures under earthquake induced bending and varying axial loads, KUCE No.88 - ST - 01, Kyoto University, 1986.

(2001. 2. 15 受付)

OBJECT-ORIENTED ANALYSIS AND DESIGN OF STRUCTURAL ANALYSIS SYSTEM

Yoshikazu TAKAHASHI, Akira IGARASHI and Hirokazu IEMURA

A variety of computer programs has been developed in order to analyze complicated structures. As the number of functions in their programs increases, it becomes difficult to manage the programs. The fundamental problem lies in the fact that they are designed in a computer-oriented manner which is totally different from our intuition. Therefore the object-oriented approach is efficient to overcome the above mentioned problem. In this study, the three-subsystem model for the structural analysis system is proposed. The system is characterized by the structure, load and analysis modules passing messages each other. This model is more flexible and more useful than the conventionally used structure-based model for the earthquake response analysis problem as well as the structure analysis problem.