初等プログラミングから設計レベルまでを対象としたオブジェクト指向教育のための支援システムの提案

大城 正典†, 永井 保夫‡

概要

オブジェクト指向開発の教育に於いて，作成中のプログラムの構成要素や，実行中の処理の流れやオブジェクトの振る舞い，特定の意図に基づき設計され意味を付与された構造とその動きを視覚化できれば，教育・学習を効率よく支援できるものと考えられる．たとえば，初等的なオブジェクト指向プログラミングの学習では，クラスやオブジェクトの相互関係などを静的・動的に視覚化することによって，学習者の理解度は高まるものと期待できる．また，オブジェクト指向による設計レベルの学習についても，特定の目的のために設計された比較的小さい構造である典型的なデザインパターンを学習題材とし，個々のデザインパターンの持つ意味を反映した形で視覚化することで，意味を持った構造を作るという設計の本質の理解に役立てることができる．本稿では，初等プログラミングから設計レベルまでの一貫したオブジェクト指向開発の教育を，視覚化によって支援する教育システムを提案する．

A Proposal of Education Support Systems for Basic Programming to Design in Object-Oriented Development

Masanori Ohshiro†, Yasuo Nagai‡

Abstract

In education of object-oriented development, visualization of elements of source programs even under making, flows of processes and behavior of objects, appearances and behavior of significant structures made for a particular purpose may help and improve education and learning efficiently. For instance, in learning basic object-oriented programming, if mutual relations of classes and objects are statically and dynamically visualized and shown to students, these may help and improve their comprehensions. In learning object-oriented design and modeling, typical design patterns are suitable for teaching materials because they are small and modeled for each particular purpose. visualization of these design patterns help students to understand the essence of design, making a structure that has a particular role. In this paper, we propose a education systems using visualization for basic Programming to design in object-oriented development.

† 東京情報大学総合情報学部環境情報学科
Department of Environmental Information, Faculty of Informatics, Tokyo University of Information Sciences
‡ 東京情報大学総合情報学部情報システム学科
Department of Information Systems, Faculty of Informatics, Tokyo University of Information Sciences

1 はじめに

オブジェクト指向プログラミングの教育において，学習者を対象としたプログラミング学習環境の提案 [2, 5] や初中等教育におけるオブジェクト指向プログラミング言語を利用した教育に関する研究 [4] が精力的におこなわれている．

このようなプログラミング教育においては，プログラムの理解を支援することが不可欠であり，静的な手法で解析したプログラムの振る舞い（プログラムのソースコードからフローチャートや実行パスの生成）やプログラムの実行履歴を取得することによる動的な振る舞いを視覚化情報として提供することで学習者の理解を支援する手法が提案されている [6, 7, 8]．

われわれは，オブジェクト指向プログラミングの教育において，クラス定義や動作の様子を視覚化して学習者に提示することが，学習者の理解を深めるのに有効であると考えている．そこで，Java 言語によって書かれたソースプログラム内に定義されたクラスとそのメンバを視覚化 (静的視覚化) し，またプログラムの動作の様子を視覚化 (動的視覚化) する教育支援システムを提案した [10, 11, 13]．これらの視

覚化機能は特にオブジェクト指向プログラミング教育の初期に有効であると考える．

しかしながら，プログラミングの学習のみならず，オブジェクト指向による設計（モデリング）の学習を行う段階においても視覚化による支援は有効であると考える．本研究では，学習者，特に，初学者に対するオブジェクト指向プログラムの学習におけるプログラムの理解支援だけでなく，オブジェクト指向の設計教育を支援するための題材ならびに視覚化機能を検討している [12, 14, 15]．以下では，プログラムの静的情報だけでなく，動的情報を可視化して提供することで，プログラムのふるまいに関する学習を支援するとともに，モデリング（設計）レベルでのプログラムのふるまいの学習も支援することを目的としたプログラミング学習支援システムについて説明する．

2 初等教育における視覚化支援

オブジェクト指向開発の初等教育は，一般にオブジェクト指向言語によるプログラミングから始まる．この段階で学習の助けになる視覚化として，本研究で提案している静的視覚化と動的視覚化の例を示す．

2.1 静的な視覚化

静的視覚化とはソースプログラム上での定義を視覚化したものであり，図 1 は，静的視覚化の例である [10, 11, 13]．ソースプログラム入力欄に入力されたソースプログラムは構文解析され，静的視覚化欄にクラス定義およびそのメンバであるフィールドおよびメソッド (便宜的にコンストラクタを含む) がグラフィカルに表示される．この図に示すように，フィールドは矩形，メソッドは円，クラスは六角形で表示される．

サブクラスを定義した場合は，スーパークラスから継承した部分はグループ化されて視覚化され，そこからスーパークラスに向けて継承関係を表す矢印線が引かれる．

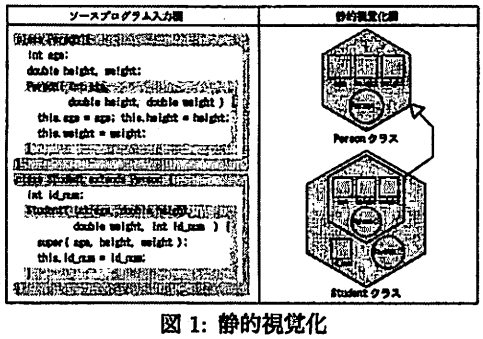

図 1: 静的視覚化

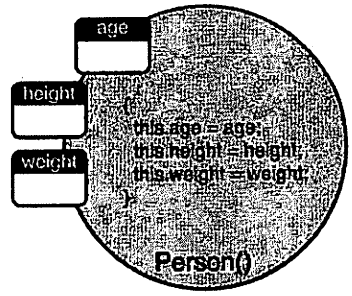

図 2: メソッドの静的視覚化

メソッドについては，引数・動作定義も視覚化されるが (図 2)，これらは表示領域が狭い場合は省略される (図 1)．

その他，通常のメソッド，抽象メソッド，final メソッド，抽象クラス，インタフェイスについてはオプションによって図 3 に示すようにも表示可能である．オブジェクト指向プログラミングでは，外部に公開される部分は (一般的な意味での) インタフェイス部分になり，非公開な部分は実装部分になることを理解することが重要である．図 3 に挙げた表示形式は，その点を考慮したものになっている．

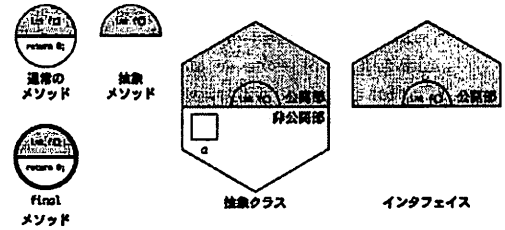

図 3: 通常メソッド・final メソッド・抽象メソッド・抽象クラス・インタフェイス

図 3 では，通常のメソッドはサブクラスで実装部分であるボディ部分がオーバライドによって置き換わる可能性があるために，シグネイチャ部分とボディ部分が線で分割されている．final メソッドはシグネイチャ部分とボディ部分が不可分であり，まるごとサブクラスに継承されることを示すために，太線で全体が囲まれている．抽象メソッドはボディ部分が欠如した不完全なメソッドであることを示すために，シグネイチャ部分のみで表示される．抽象クラスは公開部分と非公開部分に分割されて表示され，インタフェイスは公開部分のみを持つ形で表示される．

静的視覚化の表示は，ソースプログラムの編集過程で可能な限りリアルタイムで行う．例えば，クラス定義が形式的に完了すれば，そのクラスが視覚化欄に新たに表示される．エディタ上でクラス定義内にテキスト入力カーソルがある場合は，視覚化欄のそのクラスがクローズアップされる．書き直し等でクラス定義が壊れた場合は，視覚化欄の該当クラスは消滅する．

オブジェクト指向プログラミングの教育で難しい点としては，作成したプログラムを実行するまでに書かなければならない(学習しなければならない)ものが多く，学習者が「何かを作っている」という達成感を得るまでに飽きてしまう，ということが挙げられる．このように，ソースプログラムを作成している時点でもリアルタイムに成果物が表示されることで，作成中のプログラムを実行する前から「何かを作っている」という実感を与え，学習者の注意力・集中力を持続させる．

また，クラス定義やメソッド定義が形式的に完了，すなわち定義のボディ部分である対応する括弧が書かれた段階で即時に視覚化されるので，「対応する括弧を先に書く」という構造を意識した書き方を促進する効果も望めると考える．

また，視覚化欄のクラスやメンバをクリックするとクローズアップされ，ソースプログラム上の対応する定義部分がハイライトして，ソースプログラムと図の同一性を示す．また本稿では，視覚化欄に表示された各要素がドラッグされたとき，ハイライトするとともに若干振動する仕様を新たに追加した．振動は短時間で減衰しやがて元の位置に静止してハイライトも解除される．これは，学習者に体感的フィードバックを与え，学習者の興味をひく効果をもたらすことを意図している．

2.2 動的な視覚化

動的視覚化とはプログラムの動作する様子を視覚化したものであり，図 4 は，動的視覚化の例である[10, 11, 13].

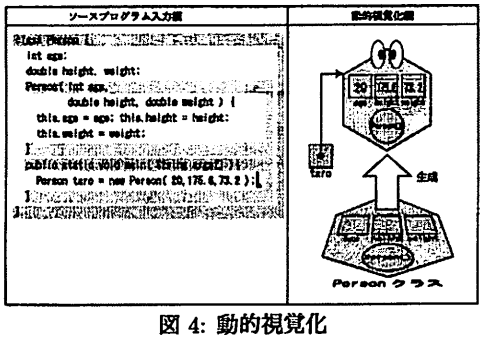

図 4: 動的視覚化

ここでは，ソースプログラム上で実行カーソルのあるオブジェクト生成の場面が表示されている．オブジェクト指向言語には，クラス自体をオブジェクトとして扱うものとそうでないものがあるが，Java は後者に属する．そのため本システムの動的視覚化では，クラスはオブジェクトの設計図としてとらえることができるように，あたかも机の上に置かれている設計図を斜め上から見ているかのような姿で表示されている．

それに対し，個々のオブジェクトは同じ設計図(クラス)から生成される，個性と主体的な振る舞いを持った実体であることを強調するために擬人化されて表示されている．

メソッド(コンストラクタを含む)の呼び出しでは，呼び出したメソッドに移動してアニメーションによる動作表示を継続することができる．たとえば，図 4 におけるオブジェクトの生成過程では，図 5 のようなコンストラクタ内の初期化過程が表示される．この例は，コンストラクタによってフィールド age が初期化される段階を表している．

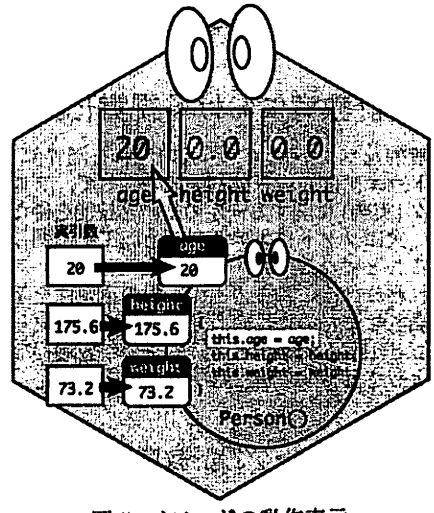

図 5: メソッドの動作表示

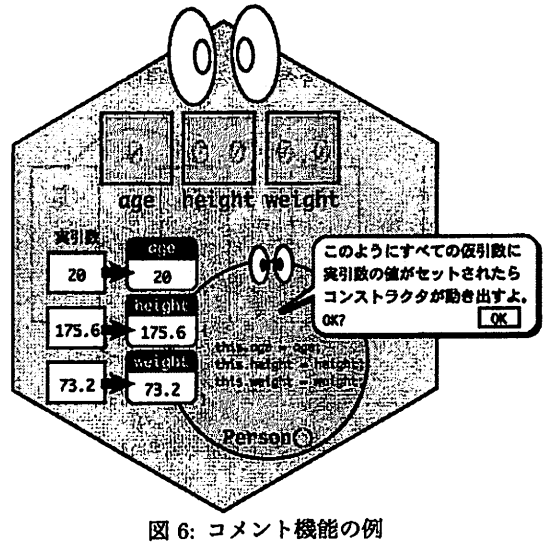

図 6: コメント機能の例

コンストラクタによる初期化が完了していない状態のオブジェクトは，「目」の瞳が白いままになっており，コンストラクタの動作が完了すると黒くなる．これにより，学習者はコンストラクタによる初期化を意識することができる．またこの図からも分かる

とおり，メソッドも処理を行う主体であるので，擬人化して表示している．

本システムは，特定の構文要素や実行アニメーションの中に，あらかじめコメントを設定することが可能である．これにより，教育者は学習者に理解して欲しい要点を強調して解説することができる．たとえば，図 6 は，コンストラクタの動作に関するコメントが表示された例である．このコメントのように閲覧者に確認を求めるコメントが表示された場合は実行はそこで一時停止し，閲覧者が"OK"ボタンを押すことで実行が次に進む．

3 設計教育における視覚化支援

一般的なソフトウエア開発の流れを図 7 に示す．ソフトウエア開発教育の初等段階として，プログラミング (実装) に関する一定の知識と技能を得た学習者は，次にソフトウエア開発の上流工程に関して学習することになる．

われわれは，これらの上流工程の重要性を学習者に認識させるためには，設計と実装を繰り返し学習することが有効であると考え，この学習過程を視覚化によって支援する方法を提案した [14, 15].

3.1 ソフトウェア設計教育に関する課題

現状の情報教育では，学習者に対してソフトウエア開発の上流工程の重要性を認識させるには不十分であると考えられる．その原因のひとつとして，UML クラス図・オブジェクト図などの設計レベルでの表現と，実際のソースプログラムとの対応に対する理解が不十分である場合が多いことが挙げられる．つまり，ソフトウエア開発の流れに於いて設計および実装とその繋がり (図 7-B) を把握できていないために，それ以前の上流工程 (図 7-A) とも関連づけて理解することができず，結果としてソフトウェア開発における上流工程の重要性を意識するまでに到らないと考えられる．

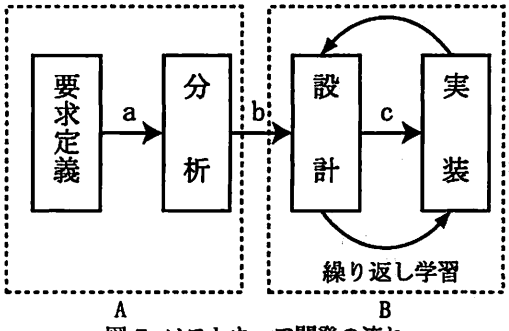

図 7: ソフトウェア開発の流れ

そこで，設計と実装 (図 7-B) の学習を静的・動的視覚化によって支援する方法を検討した．視覚化と適切な題材を用い，設計と実装を交互に繰り返して学習する (図 7) ことによって，学習者が設計と実装の対応を十分理解できれば，要求定義・分析のステージと設計・実装のステージの結びつきを深く理解し，上流工程の重要さを意識することに結びつくと考えた．

3.2 学習題材としてのデザインパターン

設計・実装のステージを繰り返して学習する際，適切な題材の選定も重要である．典型的なデザインパターンのいくつかは，特定の目的のために設計された複数のクラス・オブジェクトからなる小さめの構造を持つことから，オブジェクト指向による設計の学習に適していると考えられる．

特に Strategy パターンは，Gamma ら [1] の紹介した 23 個のデザインパターンの内 14 個が Strategy パターンの構造を持っている等，多くのデザインパターンの基本的な構造として用いられている [3]．そこで，本研究では Strategy パターンとそのバリエーションであり状態遷移の管理に利用される State パターンを，視覚化システムを用いた学習の教材として採用した．

3.3 Strategy パターンの視覚化

まず，Strategy パターンの視覚化機能について示す．図 8 は，Strategy パターンの UML クラス図である [1]．Strategy パターンの構造的特徴として，Context 型オブジェクトから Strategy 型オブジェクトへの参照があり，Context 型オブジェクトのメソッドから Strategy 型オブジェクトのメソッドへの呼び出しがあること (委譲) が挙げられる．

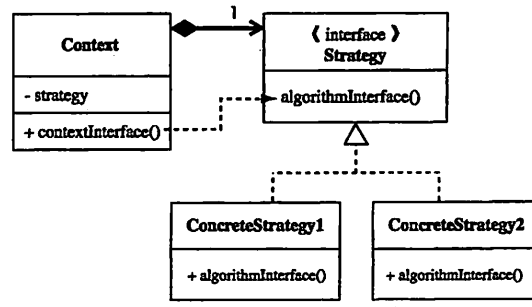

図 8: Strategy パターン

Strategy パターンの視覚化を行う場合，この委譲を自動で検出して視覚化を行う．図 9 は，Strategy パターンの静的視覚化例である．Strategy パターンを構成するクラス群がひとつの枠にまとめられ，パターン名が表示される．また，ConcreteStrategy に相当する具体クラス群もひとつにまとめられる．また，各クラスが Strategy パターンのどのクラスに相当するかが明示される．図 10 は，Strategy パターンの動的視覚化の例であり，図 9 のソースプログラム

に対応している．Strategy パターンを構成するオブジェクト群がひとつの枠にまとめられ，パターン名が表示される．また，その時点で存在する Strategy インタフェイス (または抽象クラス) を実装 (または継承) する具体クラスのオブジェクト群はひとつの枠にまとめられ，そのうち現在 Context 型オブジェクトから参照されているオブジェクトが左端に配置され，参照関係を表す線で Context 型オブジェクトのフィールドと結ばれている．これにより，その時点での移譲先オブジェクトを視覚的に把握することが可能である．

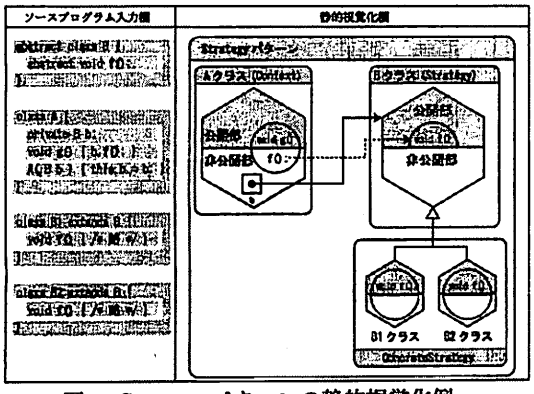

図 9: Strategy パターンの静的視覚化例

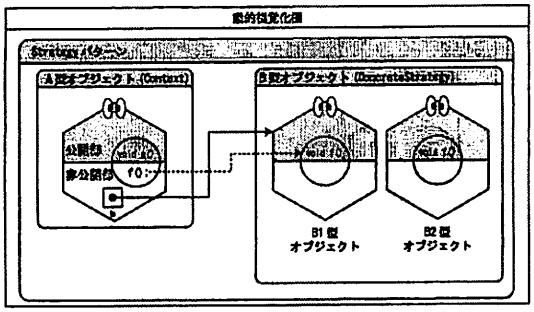

図 10: Strategy パターンの動的視覚化例

3.4 State パターンの静的視覚化

次に，State パターンの視覚化について述べる．図 11 は，State パターンの静的構造の概要である [1]．State パターンは Strategy パターンの構造を持つので，まず Strategy パターンとして認識される．State パターンとして認識するためには，人為的な方法とクラス名の命名規則による自動的な方法がある．人為的な方法は，図 9 のように Strategy パターンとして表示された後で，パターン全体を囲む枠の左上に表示されているパターン名 ("Strategy") をクリックし，表示されるメニューから State パターンとして解釈するように指示する．または，"Abstract" から始まり"State"で終わる名前のクラスがある場合は，これを State パターンに於ける State 抽象クラスとして解釈し，全体を State パターンとして解釈し直す．図 12 は，State パターンの標準的な静的視覚化の例である．

State パターンの静的視覚化には，図 12 で示した標準的なものの他に，状態遷移表モード (図 13) と状態遷移図モード (図 14) が用意されている．これらは，例えば B1 クラスの f() の定義内にある

```
if( E1 ) {
  この状態の動作
  A 型オブジェクト.setB(B2 型オブジェクト);
}
if( E2 ) {
  この状態の動作
  A 型オブジェクト.setB(B4 型オブジェクト);
}
```

などの状態遷移を制御するコードの定義を認識して生成される．また，状態遷移がそれ以上行われない状態を終了状態として認識する (図 13 および図 14 では B5 で，二重線で表示されている)．なお，初期状態は動的に決定されるので静的視覚化では明示はしない．

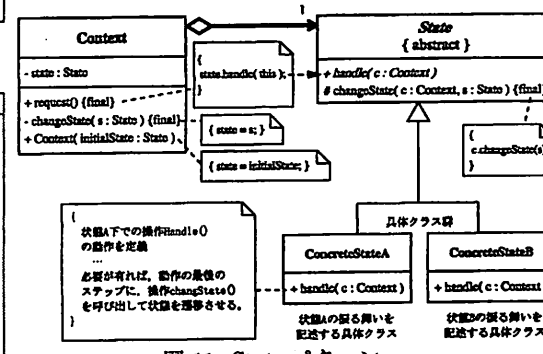

図 11: State パターン

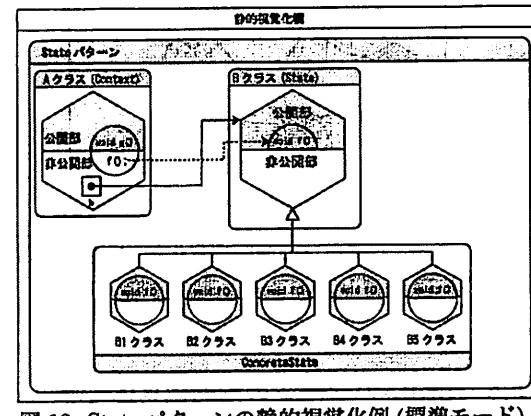

図 12: State パターンの静的視覚化例 (標準モード)

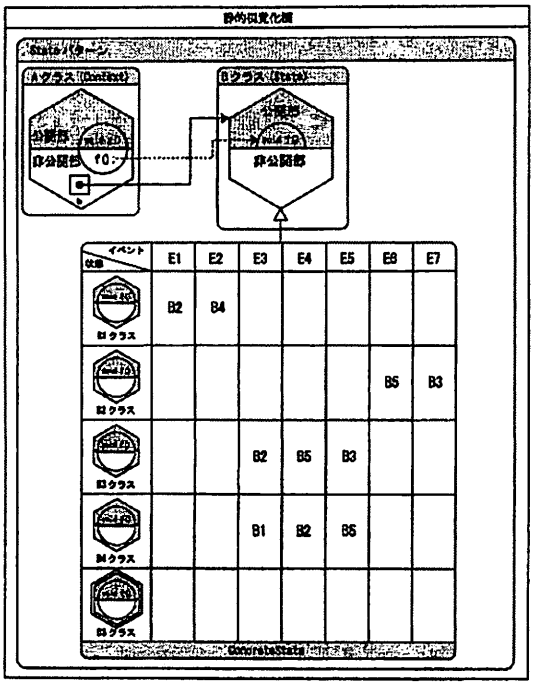

図 13: State パターンの静的視覚化例 (状態遷移表モード)

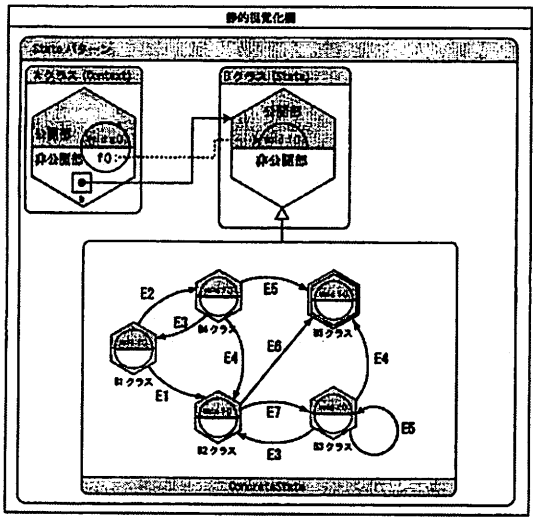

図 14: State パターンの静的視覚化例 (状態遷移図モード)

3.5 State パターンの動的視覚化

State パターンの動的視覚化の表現も標準モード (図 15), 状態遷移表モード (図 16), 状態遷移図モード (図 17) が用意されている. これらの例では, 現在の状態を表現しているのは B4 型オブジェクトであり, Context 型オブジェクトに相当する A 型オブジェクトが, B4 型オブジェクトを参照していることが把握できる. また, 状態遷移表モードでは, 参照している状態オブジェクトの行が強調されている.

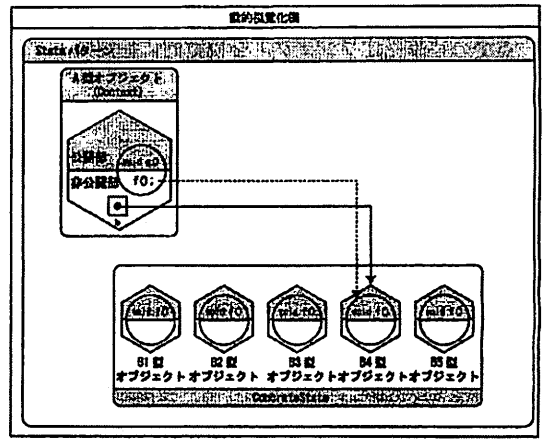

図 15: State パターンの動的視覚化例 (標準モード)

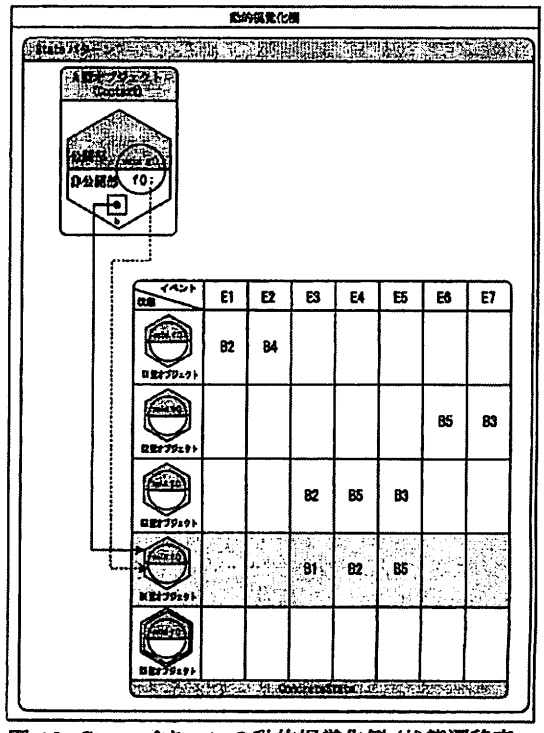

図 16: State パターンの動的視覚化例 (状態遷移表モード)

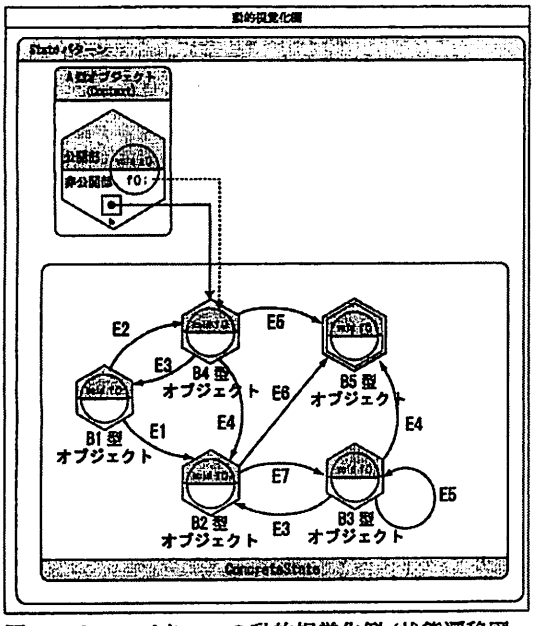

図 17: State パターンの動的視覚化例 (状態遷移図モード)

3.6 実際の教育時における使用法

オブジェクト指向開発の教育のために，次の様な授業科目の流れを提案する．

(1) プログラミング入門科目
(2) オブジェクト指向入門科目
(3) オブジェクト指向設計学習科目
(4) ソフトウェア開発学習科目

まず，科目 (1) でクラス定義，メソッドの動き，オブジェクトのふるまいの確認に本システムを使用する．次に，科目 (2) で情報隠蔽，継承，実装，抽象クラス，抽象メソッド，インタフェイスなどの学習に本システムを用いる．科目 (3) では，UML クラス図・オブジェクト図，クラス間関係，委譲，オブジェクトコンポジションの学習を行った上で，設計の初歩として Strategy パターンとその応用としての State パターンを学習する．

このとき，次の様な手順を考えることができる．まず教員が，典型的な例題で仕様と状態遷移表・状態遷移図を学習者に見せ，対応するソースプログラムと静的に視覚化された図を提示し，その後，プログラムを動作させて動的に視覚化された図を学習者に観察させる．更に学習者にも自身の手で動作させ，視覚化された図を観察させる．

次に，別の課題として仕様と状態遷移表・状態遷移図を与え，学習者自身が静的・動的に視覚化された図を意識しながらソースプログラムを作成し，動作確認を行う．更に，本システムを利用せずにソースプログラムから状態遷移表・状態遷移図を作成し，また逆に本システムを利用せずに仕様および状態遷移表・状態遷移図からソースプログラムを作成する．

このような手順を，Strategy パターンを基礎とするその他の代表的なデザインパターンの学習についても実施し，複数のクラスを関係を持たせて設計し，実行時にオブジェクト間の関係を活かすことでどのように問題を解決できるかを理解し，設計の重要さを認識させる．その後，科目 (4) では提示された問題に対して，要求仕様作成を行い，設計，実装へと進む一貫したソフトウェア開発の流れを学習する．

4 UML クラス図・オブジェクト図との比較

本システムの第 1 目的は，学習者の注意力を維持ししつつ理解を助ける事である．そのために，仕組みや動作が理解しやすいように表現を工夫している．例えば，データ，処理，オブジェクトを形状で区別したり．サブクラスオブジェクトにスーパークラスのオブジェクトが一部分として含まれていることを表示して継承の理解を助けるようにしている．また，モジュールであるクラスの公開部分 (インタフェイス部分) と非公開部分 (実装部分) を明示し，情報隠蔽や抽象クラス，インタフェイスの本質的な理解を助けるように意図されている．その他，本稿で提案した State パターンの視覚化のように，特定の目的のための特殊な視覚化も採用可能である．また，動的視覚化を当初から考慮しているのも，本システムの特徴である．

一方，UML クラス図・オブジェクト図は，汎用性のある設計記述を可能にするためのものであり，表現も意図的に制限されているため，文字表現が主体で，視覚的な特徴も乏しく慣れを必要とする．また，「なぜこうなっているのか」という理由が分かりづらい点も問題として挙げられる．例えば，UML では「なぜサブクラスのオブジェクトはスーパークラスのオブジェクトとして振る舞えるのか」という理由は説明しづらいが，本システムでは「サブクラスのオブジェクトはスーパークラスのオブジェクトを内蔵していて，スーパークラスのオブジェクトの能力を受け継いでいるため」と言うことが視覚的に説明できる．

しかし，両者は対立するものではなく，学習段階や動作の視覚的確認等に本システムを使用し，汎用的な記述が必要な場合には UML 図を利用するなど，それぞれの特長を活かした利用が考えられる．また，UML 図の学習に本システムを用いることも有効であると考える．

5 おわりに

本稿では，これらの視覚化による教育支援システムを，オブジェクト指向開発の初歩から設計レベルまで一貫した教育の支援システムを，教育方法まで含

めて提案した．現在，実装をEclipse上のプラグインとして試みている．今後は，本システムを実際にプログラミング入門科目からオブジェクト指向設計学習科目までの一連の授業に使用し，効果を検証したい．さらに，UMLのクラス図，ステートマシン図などによる設計レベルの振る舞いの表現については，UMLの表記によるチェックだけでなく，SPIN/Promelaによるモデル検査[9]を行うことを考えている．本システムでは，このようなモデル検証の結果を反映した設計モデルに基づきプログラムを修正できる機能の実現についても検討していく．

参考文献

[1] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley Professional, Nov. 1994.

[2] 萩庭崇, 永田守男,"オブジェクト指向言語のための視覚的プログラム支援環境," 情報処理学会ソフトウエア工学研究, 96-4, 1996.

[3] 大城正典, "Java と C++によるデザインパターン入門 総括編, " pp.67-76, C マガジン, ソフトバンク, 東京, Nov. 1999.

[4] 兼宗進, 中谷多哉子, 御手洗理英, 福井眞吾, 久野靖, "初中等教育におけるオブジェクト指向プログラミングの実践と評価," 情報処理学会論文誌, Vol.44, No.SIG 14 (PRO18), 2003.

[5] 長慎也, 甲斐宗典, 川合晶, 日野孝昭, 前島真一, 筧捷彦,"Nigari-Java 言語へも移行しやすい初学者向けプログラミング言語," 情報処理学会研究報告 2003-CE-71 (3), 2003.

[6] 喜多義弘, 川添貴識, 片山徹郎,"初心者を対象にした Java プログラム自動可視化ツールの実現に向けて," 信学技報 SS2004-46, 2005.

[7] 谷口孝治, 石尾隆, 神谷年洋, 楠本真二, 井上克, "プログラム実行履歴からの簡潔なシーケンス図の生成手法," コンピュータソフトウエア Vol.24, No.3, pp.153-169, 2005.

[8] 竹下彰人, 片山徹郎,"シーケンス図を用いた実行履歴の可視化による Java プログラムの理解支援に関する考察," 信学技報 SS2006-63, 2006.

[9] 萩谷昌己（監修）, 吉岡信和, 青木利晃, 田原康之, "SPIN による設計モデル検証," トップエスイー実践講座 3, 近代科学社, 2008.

[10] 大城正典, 永井保夫, "オブジェクト指向プログラムの視覚化によるプログラミング教育システム," 電子情報通信学会 2009 総合大会講演論文集 情報システム講演論文集 1, pp.212, Mar. 2009.

[11] 大城正典, 永井保夫, "オブジェクト指向プログラムの静的・動的側面を視覚化するプログラミング教育支援システム," 情報処理学会第 71 回全国大会講演論文集 (4), 4-413, 4-414, Mar. 2009.

[12] 永井保夫, 大城正典, "モデリングを考慮したソフトウエア教育のためのオブジェクト指向プログラミング教材の検討," 電子情報通信学会 2010 総合大会講演論文集, D-15-25, Mar. 2010.

[13] 大城正典, 永井保夫, "オブジェクト指向プログラムの視覚化によるプログラミング教育システムの改良," 電子情報通信学会 2010 総合大会講演論文集 情報システム講演論文集 1, pp.170, Mar. 2010.

[14] 大城正典, 永井保夫, "情報視覚化を活用したオブジェクト指向プログラミング教育支援システムの提案," 電子情報通信学会 技術研究報告 Vol. 110,No. 453 教育工学 pp.131-136, Mar. 2011.

[15] 大城正典, 永井保夫, "オブジェクト指向プログラム視覚化教育システムにおけるデザインパターンの視覚化サポート, " 電子情報通信学会 2011 総合大会講演論文集 情報システム講演論文集 1, pp.139, Mar. 2011. http://cho.is.meisei-u.ac.jp/SSS2011POST/