Eclipse 視覚化プラグインによる総合的なプログラミング教育支援システム

大城 正典[1,a)]　永井 保夫[1,b)]

概要：著者らが作成しているプログラミング教育を支援するための視覚化機能を持った Eclipse プラグインの機能とその活用法について提案する．現時点では本システムは Java 用に作成され，静的視覚化機能と動的視覚化機能を備えている．静的視覚化機能では，ソースコード上の文法的要素とその相互関係をプログラミング作業にともなってリアルタイムに視覚化し，ソースコードの静的な内容を理解する能力および文法にそったソースコードを書く能力の向上を支援する．この機能を利用して，提示された仕様にしたがってクラス定義を段階的に行わせる演習方法と，各学習者のクラス定義の進行状況をリアルタイムにモニタする機能について紹介する．また，動的視覚化機能はプログラムの動きを視覚化するもので，アルゴリズムの動作する様子やオブジェクト間のデータのやりとりなどをリアルタイムに観察することができ，プログラムの動的な側面に対する学習者の理解を助ける．本論文ではその一例として Obsever パターンを題材とした UML シーケンス図をともなう動的視覚化について提案する．これによって，Obsever パターンの複雑な動きの把握とともにシーケンス図の理解を助けることが期待される．最後に，これまで提案してきた視覚化機能を利用した学習方法の有効性を，プログラミングエージェンシーを仮定して考察した．

キーワード：プログラミング教育，視覚化，Java, Eclipse，Observer パターン，エージェント

A General Support Systems for Programming Education using Visualization Plug-in

Ohshiro Masanori[1,a)]　Nagai Yasuo[1,b)]

Abstract: In this paper, we propose functions and uses of our Eclipse plug-in for visualization to support programming education. The system is made for Java Programming Language and has static visualization and dynamic visualization. The static visualization view shows syntax elements in a source code and their mutual relationships as corresponding figures. The figures are drawn and erased when users write and erase the syntax elements in real time. In this way, the static visualization is able to improve understanding syntax elements in source code and ability to writing them. We introduce step-by-step class definition lesson using this static visualization. Teachers can monitor each learner's progression of the lesson. The dynamic visualization function shows control flows and movement of data in a running program in real time, therefore learners can watch them to understand the dynamic aspect of the program. We introduce dynamic visualization of observer pattern with a UML sequence diagram. We expect that the dynamic visualization helps learners to understand complex behavior of observer pattern and meaning of sequence diagrams. In the last of this paper, we assume that programming is executed by programming agency and discussed the efficiency of our systems and methods.

Keywords: Programming Education, Visualization, Java, Eclipse, Observer Pattern, Agent

[1] 東京情報大学
Uniersity of Information Sciences, 4–1 Onaridai Wakaba-ku, Chiba, Chiba, Japan
[a)] ohshiro@rsch.tuis.ac.jp
[b)] nagai@rsch.tuis.ac.jp

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 [1], [2] 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 [3] 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欠かせない．静的に解析したプログラムの振る舞い（ソースコードからフローチャートや実行パスを生成）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した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 [4], [5], [6]．著者らは，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 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 [7], [8], [9], [10], [11], [12], [13], [14], [15]．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．また，学習者がモデリングなどのオブジェクト指向設計を学び始める段階においては，複数のクラス・オブジェクト間の関係やメソッドの呼び出し関係，それらが動作している様子を観察することで，学習者にとっては複雑性の高い設計の内容も理解しやすくなるものと思われる．

しかし，我々は本研究において大きな問題に気がついた．それは，視覚化によってプログラムの静的・動的側面を理解できたとしても，プログラミングや設計といったソフトウェアを作成する能力の向上に直接繋がるわけではない，という点である．ソフトウェア作成能力を向上するには，適切な視覚化をともなった演習方法も必要になると考えられる．

たとえば，本システムは当初からプログラムのコーディングと同時にリアルタイムに静的視覚化を行う機能を持っているが，実際には学習者が適切なコーディング手順に従っていないと，視覚化によるサポートも効果を発揮できない．それどころか視覚化サポートの有無にかかわらず，プログラミング能力がいつまでも向上しないスパイラル状態に陥る可能性がある．2 章ではこの問題について詳しく述べ，解決する演習方法を提案する．また，設計の学習に際しても有用なクラス間関係の理解を UML 図や本システムの視覚化によってサポートするだけではなく，実際にそれらのコンセプトを応用する演習でも視覚化のサポートを行えると，設計能力の向上により役立つと思われる．その一例として Observer パターン [16] に関する演習方法を 3 章で提案する．

本研究では今後も継続して視覚化機能を備えた学習支援システムを考案・実装していく予定だが，視覚化機能をともなった有効な演習方法を考案・検討する際，プログラミングを人間がどう行うかというモデル化の必要性を強く感じた．そこで 4 章では，プログラミングを行うエージェンシー [17] を仮定し，本システムの諸機能がどのように個別のエージェントの働きを支援するかを議論し，また今後の指針についても検討した．

2. 構造指向による段階的なコーディングガイド機能

本研究の静的視覚化機能は，利用者が Eclipse 上でソースプログラムを書いていくと，リアルタイムでソースプログラム内の構造および構造間の関係を JDT プラグインの機能を使って分析し，静的視覚化ビュー内に描画するというものである．その際，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．

このように，テキストエディタ上で定義された内容を即時に視覚化することで，ソースプログラム内の構造を明確に把握しやすくなり，ソースコードを書いている間も何かしらの成果物をフィードバックして学習者を飽きさせず，手応え感を与えることもできる．そして，ソースコード内にあるべき構造を作っていくというプログラミングの基礎を意識させることが期待できる．

しかし，教材のサンプルプログラムの要点を明確に視覚化して学習者に伝えることによって学習者の理解の助けになることは期待できるものの，ソースプログラムを最初からコーディングする能力に直接結びつくことは期待できない．

われわれは経験上，サンプルプログラムを 1 行目から順に書き写すという書法から抜け出せない初心者が多くいることを知っている．このような書き方は，プログラミング修得者のコーディング手順と大きく異なっている．通常，プログラミング修得者は，括弧を伴う制御文，クラス定義やメソッド定義の記述時に，対応する括弧を先に書いて外側の構造を完成させてから，その後中身を埋めるという構造指向的な書き方を行う．これは，プログラミング修得者のコーディング時の思考の流れそのものでもある．しかし，このような書き方を指導した後でさえ，1 行目から順に写していく初心者を一定数見ることができる．原因としては，ソースコード上に存在する文法に基づく構造を把握できていないために，1 行目から順に写していくという書き方しかできない，という可能性が考えられる．そしてこのような初心者は適正な書き方ができないために，構造の把握力が低迷するという悪循環に陥っているように見受けられる．ソースコード上の構造は文法の適用結果であるので，これは，文法を把握することができない状態に陥って

いるとも言える．本システムのソースコードをリアルタイムに静的視覚化する機能も，1 行目から順にプログラムを書くという書き方に陥っている初心者の場合，文法や構造を把握する能力の向上効果は期待できない．

そこで，われわれは初心者のコーディング能力，特に文法的構造を書くことができる能力を高めるために，段階的なガイダンスによって学習者にソースプログラムを完成させていく機能を本学習支援システムに追加した．また，学習の実効性を上げるために学習者がどこまでソースコード内の構造を書けたかをモニターする機能を用意した．

2.1 モニタリングシステム概要

本システムのサーバ・クライアント構造を図 1 に示す．問題は教員間で共同製作・共有・再利用可能なように問題データベースにソースコードとして登録可能にしておく．文法的に新しい内容を学習した後，関連した問題を 5〜10 問程度用意することを想定している．Eclipse 上で本システムのプラグインから演習モードを起動すると，データベースに接続して問題用ソースコードを読み込む．学習者は Eclipse 上で問題を解くが，ソースコード内の構造ごとに完成させるように指示される．学習者がソースコード内の各構造を完成させると，本プラグインはその進捗状況をサーバへ送信する．進捗状況は以前われわれが開発したシステム [18] によってモニタすることが可能になっている．教員は Web ブラウザで Ajax を利用した自動更新によって，ほぼリアルタイムに各学習者のおおよその進捗具合 (色表示) と，どの問題のどの構造まで定義できたか (数字表示) を確認することができる．図 2 にその表示例を示す．

2.2 段階的なコーディング演習の例

クラスを定義する問題の場合，本システムを使って行われる段階的なコーディング演習の手順は次の様になる．図 3 は，問題の元となるソースコードの例で，問題文は javadoc コメントと同じ箇所に独自のコマンド`@q#`を使ってあらかじめ埋め込んだ情報とソースコードの解析結果から生成される．基本的なコマンドの形式は

`@q#`番号 問題文

となっていて，0 から始まる番号は学習者に書かせる順番で，同じ番号はどれから書いても良いことを表す．特に，`@q#0` は問題全体の主旨を含む導入用の問題文となる．番号が省略された場合は，後述するように学習者が書くのに適した順番が自動で指定される．`@q#`コマンドの問題文部分が省略された場合や`@q#`コマンド自体が省略された場合，その要素を説明する javadoc コメントの 1 行目の記述と構文解析の結果を使って自動で問題文が生成される．型名・要素名を含まない説明記述の場合は，構文解析の結果から判明した型名・名前を挿入する．たとえば，int 型の age というフィールドの場合，以下の説明記述

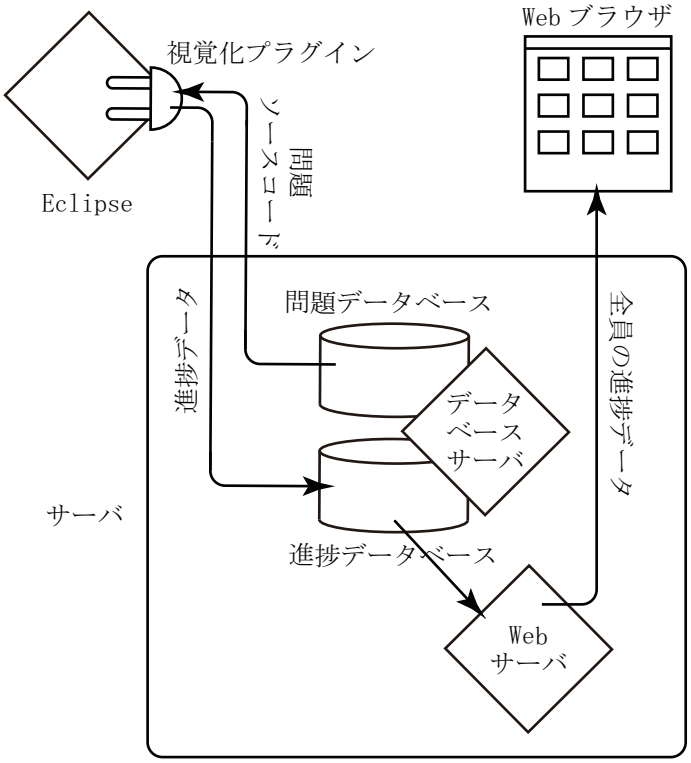

図 1 本システムのサーバ・クライアント構造
Fig. 1 Schema of server-client relation of the system.

図 2 Web ブラウザ上でのリアルタイム進捗状況表示
Fig. 2 A display of monitoring progression of students on Web browser.

```
/**
 * 人間を表すクラス。
 *
 * @q#0 人間を表す public な Person クラスを定義しよう。
 * このクラスは以下のメンバを持つ。
 * @q#mlist
 */
public class Person {
    /**
     * 年齢を表す。
     */
    int age;

    /**
     * age フィールドを初期化するコンストラクタ。
     */
    Person( int age ) {
        this.age = age; // @q# 前段階で作成したコンスト…
     }

    /**
     * main メソッド。
     */
    public static void main( String args[ ] ) {

    }
}
```

図 3 問題となるソースコードの例
Fig. 3 A sample source code for a question.

年齢。
年齢を表す。
年齢を表すフィールド。
から，いずれも
年齢を表す int 型の age フィールド
というように補間されて問題文に使用される．なお，クラスの説明文の中では，そうして補間されたメンバ記述のリストが，@q#mlist によって展開可能である．また，メソッド内部の処理などについては，
ソースコード // @q# 説明文
や
// @q#begin 説明文
ソースコード
// @q#end
という形で問題文を設定できる．これらの問題文の中で，@q#this というコマンドで対象のソースコード部分自体を学習者に提示できる．例えば，
this.age = age; // @q# @q#this と記述せよ。
とすれば，学習者には
"this.age = age; と記述せよ。"
と提示される．このような指示に対する学習者の入力は，元ソースコードの内容と白空白を無視したマッチングで正解・不正解を検査される．なお，プロジェクトの作成，ファイルの作成・追加の指示は自動的に挿入される．

本システムによる演習開始をメニューから選択するとログイン画面が表示される．ログインするとデータベースに接続する．なお，あえてログインしない場合は自習形式となり，クライアント側にあらかじめ用意された自習用の問題を使うことになる．ログイン後，演習時間内であれば最初の問題から順次別ウィンドウで表示される。単純なクラス定義問題 (図 3) から生成された問題の表示例を図 4 に示す。この指示に対して学習者が Java プロジェクトを作成し，指定されたソースファイルをプロジェクトに追加すると，JDT を利用した解析により最初の第 1 段階は終了したと判断され，問題表示は次の段階へ進む。

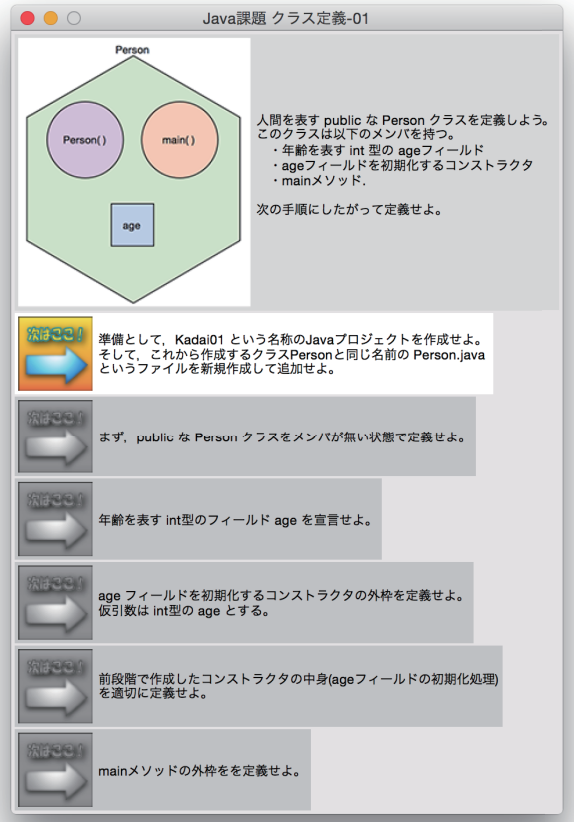

図 4 段階的なコーディング演習の問題表示例
Fig. 4 An example of step-by-step instructions for coding.

学習者に記述させる順番は，修得者が実際にコーディングする順番に近くなるように配慮した．そのため，基本的に先に外側の構造から書かせ，外側の構造が完成した後に内側の構造を書かせるものとする．クラス定義の場合は，空のクラスを書かせてからメンバを書かせる．メンバを書く順番については，@q#番号コマンドで強制的に変更することもできるが通常は以下の様に自動設定される．初心者を対象としていることから，まず実装部分となりやすいフィールドの宣言から先に書かせる．同じ優先順位のものについては，ソースコード上の順番に準じて問題文が提示される．全フィールドの宣言が書けたら，コンストラクタ，他メソッドの順で書かせる．以上の規則に従って，ソースコードを完成させるまでの状態遷移の仕組みが State パ

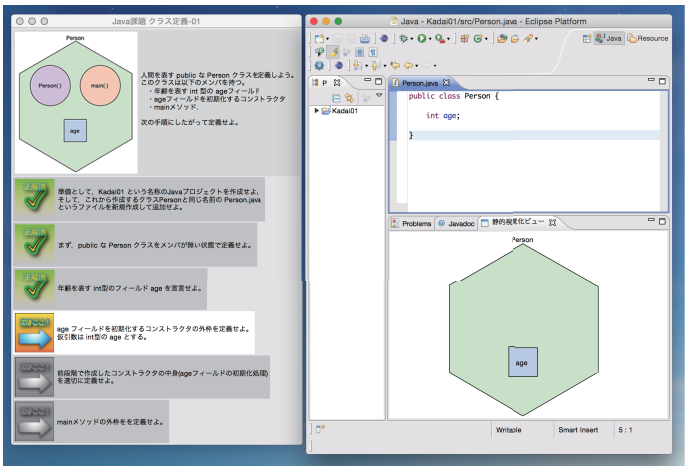

図 5 段階的なコーディング演習の経過例
Fig. 5 A progress of coding exercises with the step-by-step instructions.

ターンを使って管理される．図 5 は，さらにクラス Person の定義とフィールド age の定義まで終了した時点の表示である。対応する正解入力例を図 6 に示す。演習中を通じて，事前に定められた状態遷移に反したコーディングが行われた場合は，警告を表示して正しい順序で書くように促す．進捗が滞っている学習者がいれば，教員が Web ブラウザ上のモニタで簡単に発見でき，どこで行き詰まっているかを知って即時にアドバイスすることが可能である．

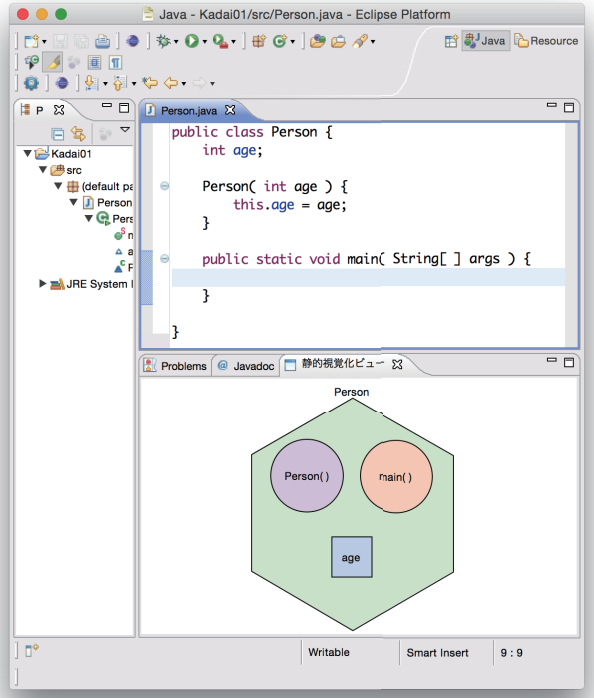

図 6 クラス定義のソースコードの正解例
Fig. 6 An example of correct source code.

このように，ソースプログラムを強制的に構造単位で完成させる様に書かせることで，学習者がソースプログラム内の構造を意識した書き方を行えるように補助することが可能であると思われる．これは，修得者の思考の流れに沿った書き方を追体験させることでもある．

3. UML シーケンス図と連携した動的視覚化

コーディングや設計を行うためには，完成したプログラムでどのようにオブジェクトが振る舞うかを想像できることが重要と思われる．学習者の段階では，複数のクラスが係わる複雑性の高いプログラムにおいて，ある目的のために複数の関連するオブジェクトが特定の役割を持ちながらどのように相互作用するかを把握することは難しいと言える．関連する複数のオブジェクトを抽出してその相互作用を視覚化したものに UML シーケンス図がある．しかし，UML シーケンス図では実際に動作する様子を動的に観察できるわけでは無いし，UML クラス図とも連携していないので，その相互作用がどのように生み出されているのかが把握しづらい．そこで，本システムの動的視覚化機能を UML シーケンス図に導入することで，関連するオブジェクトの相互作用を動的に観察できるようにし，さらにソースコードと静的視覚化図とも連携して理解できる機能を提案する．そして，そこで学んだ考え方を応用して実際に自分で作成したプログラム上でも，同様の相互作用が機能するか動的に観察できるようにする．題材としては，Observer パターンを用いる。

3.1 UML シーケンス図の動的視覚化過程

基本的にシーケンス図は，時間軸にそったオブジェクト間のメッセージの呼び出し，データのやりとりなどを中心に図示している．そこで，動的視覚化にあたって，ライフライン上を上から下に関連するオブジェクトが降下しながらシーケンス図を描画していく形とした．図 7 は，関連するオブジェクトが，シーケンス図を描画する前の状態である．なお，Observer パターンからサブジェクトオブジェクトとオブザーバーオブジェクトは，名称の末尾に Subject と Observer がついているクラスのオブジェクトの中で，参照関係にあるものを図示に用いる．また，シーケンス図の性質より，関連するメソッドのみをオブジェクトの中に提示する．

オブザーバーオブジェクト aObserver がサブジェクトオブジェクト aDataSubject の setState() メソッドを呼び出すスコープに処理が入ると，シーケンス図上で両オブジェクトがライフラインを引きながら降下し始める．これ以降，メソッドのうち関連するもの以外は半透明表示になる．そして，aObserver が実際に aDataSubject.setState() を呼び出すと，メッセージを表すラベル付き矢印が発生し，aDataSubject.setState() まで伸張する．同時に，矢印の根元から aObserver のライフラインにそって実行仕様を表す

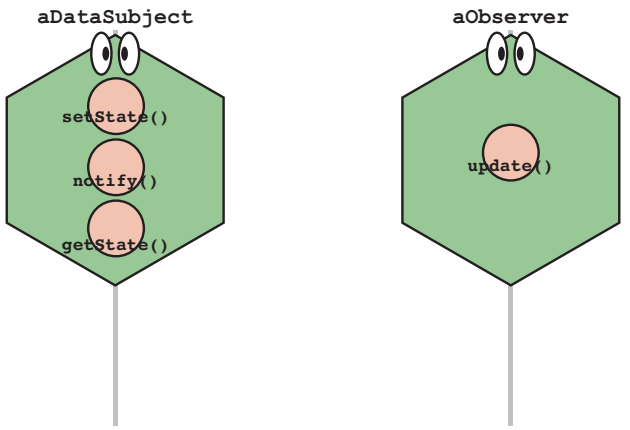

図 7 シーケンス図描画開始前
Fig. 7 This figure shows status of drawing a sequence diagram.

矩形が下向きに伸び始める (図 8)。なお，このとき本システムの静的視覚化によって表示された図でも，該当メソッドを呼び出したことを示す矢印が表示される．本システムの動的視覚化機能では，常にソースコード上でどこを実行しているかが示されるので，ソースコード上で何が起こっているかも同時に観察できる．以下同様に，シーケンス図上でのできごとは静的視覚化図とソースコードでも確認可能である．

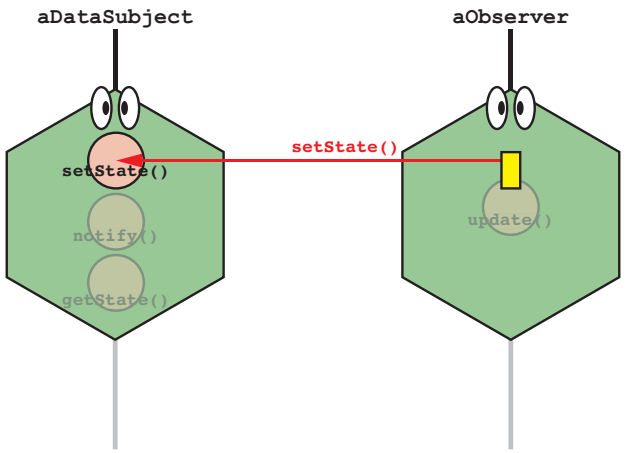

図 8 メッセージと実行仕様の発生
Fig. 8 A message and an execution specification are created.

サブジェクトオブジェクト aDataSubject が内部状態を変化させる操作に従って，自分自身の notity メソッドを呼び出す (図 9)．Observer パターンにおいては，これがオブザーバーオブジェクトへ状態の変化を通知する契機となる．

サブジェクトオブジェクト aDataSubject は，自身が参照しているオブザーバーオブジェクト aObserver の update メソッドを呼び出す (図 10)．これによってオブザーバーオブジェクトはサブジェクトオブジェクトに状態変化が起こったことを知る．

オブザーバーオブジェクト aObserver は，変化のあった

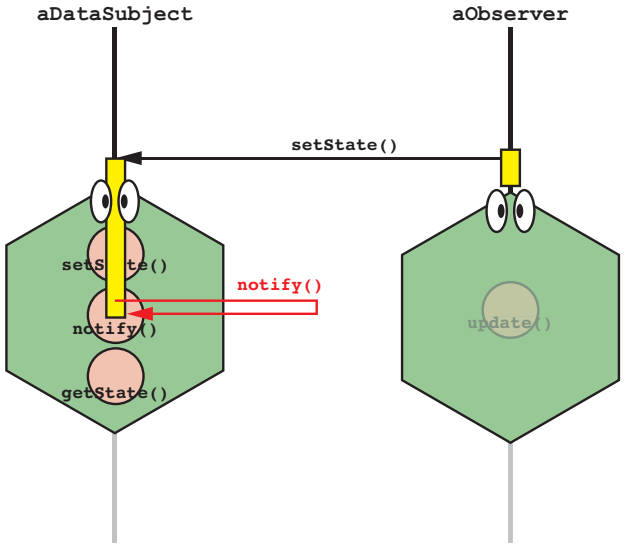

図 9 サブジェクトオブジェクトが nootify メソッドを呼び出す
Fig. 9 The subject object calls its notify method.

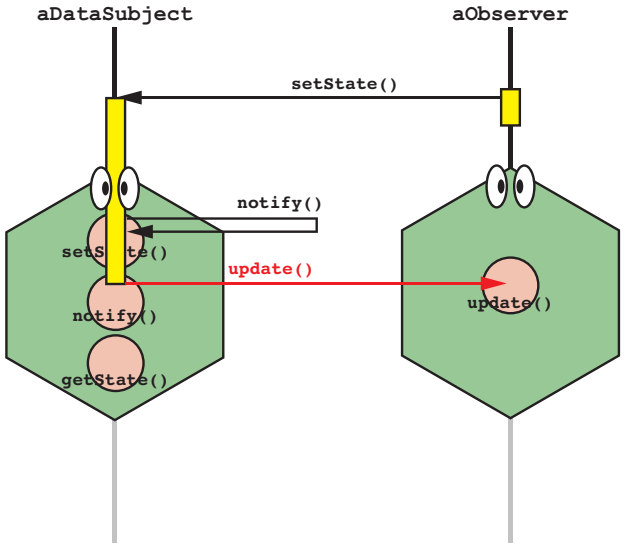

図 10 サブジェクトがオブザーバーの update メソッドを呼び出す
Fig. 10 The subject calls update method of an observer.

サブジェクトオブジェクトの状態を取得するために，サブジェクトオブジェクトの getState メソッドを呼び出す (図 11)．以上で，Observer パターンの主要な動作は一通り終了し，シーケンス図の作成も完了する (図 12)．2 回目以降の実行では，1 回目で出来上がったシーケンス図上をなぞるように動作する．

3.2 動的シーケンス図を使った演習方法

実際の演習では，Observer パターンを使った一例 (具体 Subject クラスと具体 Observer クラスを用意する) について，ソースコードとそれを本システムによって静的視覚化した図，UML クラス図を学習者に提示しながら，プログラムを動作させ，具体 Subject クラスと具体 Observer クラスのオブジェクトの相互作用を，前述した動的シーケン

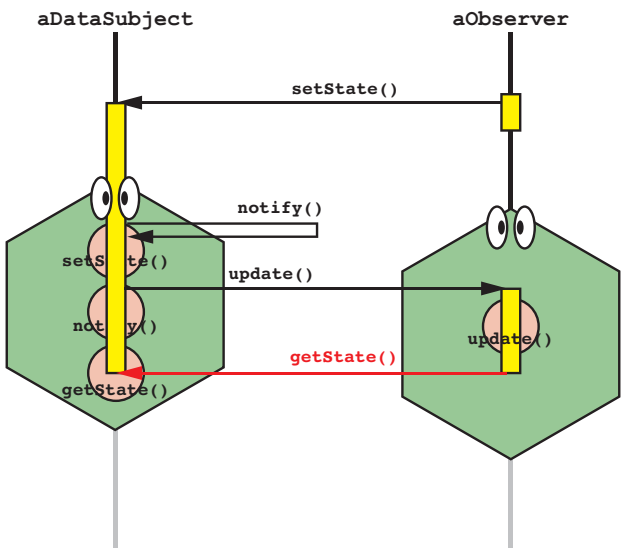

図 11 オブザーバーがサブジェクトの getState メソッドを呼び出す
Fig. 11 The observer calls getState method of the subject.

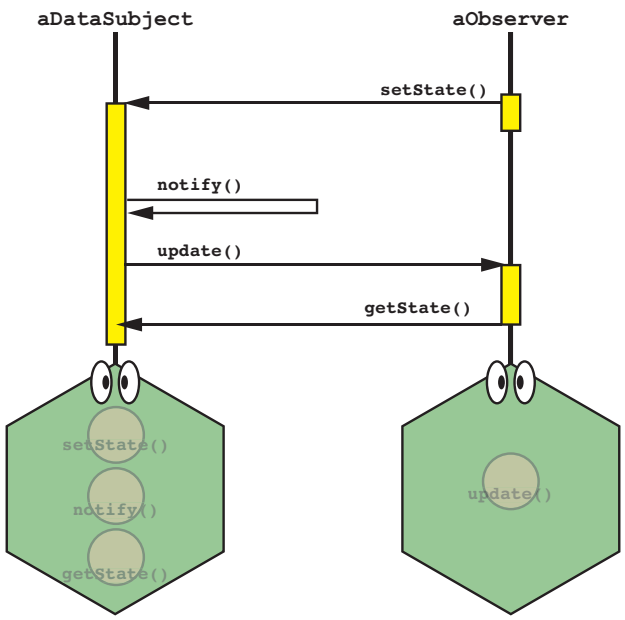

図 12 Observer パターンの動作を表すシーケンス図が作成される．
Fig. 12 Completion of UML sequence diagram for Observer pattern.

ス図を使って観察させる．そして，Observer パターンを使った別の処理を設計させる応用課題を学習者に与える．このとき，例とは異なる独自の具体 Subject クラスと具体 Observer クラスを作成させる．課題が完成したら，自作した処理を実行させ，ソースコード，静的視覚化図，UML クラス図とともに動的シーケンス図上でのオブジェクトの相互作用を観察・確認させる．

4. プログラミング能力とエージェント

われわれは，視覚化を用いたプログラミング学習支援について，いくつかの提案・開発をしてきた．2 章で紹介した構造指向による段階的なコーディングガイド機能などをはじめ，その多くが初学者向けである．一方で，3 章で提案したシーケンス図の動的視覚化や，以前の論文で提案した State パターンの静的視覚化・動的視覚化 [12] は設計を学習する段階まで理解が進んだ学習者向けと言える．これらを大学でのプログラミング・ソフトウェア開発の学習に広範囲に適用可能であろう．

しかし，このようにプログラミング学習の手法を考案する際，実際には想定したほどの効果が生まれない場合も多い．たとえば本論文でも述べたように，静的視覚化と動的視覚化だけではプログラムに対する理解度は向上しても，プログラムを作成する能力の向上は期待できない．考案した機能や手法の実効性を検証することは難しく，現場で使用した後でのアンケートや試験の結果から行うしかないのが現状である．考案・開発のコストが高いだけに，教育方法の考案段階から指針となる考え方の必要性を強く感じる．

Minsky は，「心はたくさんの小さなプロセスからできている」とし，その小さなプロセスのことをエージェントと呼んだ [17]．そして，グループ化・階層化された複数のエージェントが「自分が何をしているか知っている」場合，それをエージェンシーと定義した．われわれは，プログラミングという思考過程を行っている主体をエージェンシーと捉え，それを構成する 3 つのエージェントとして，「仕様を考えるエージェント」，「コードを理解するエージェント」，「与えられた仕様をコード化するエージェント」が存在すると仮定した (図 13)．「コードを理解するエージェント」は，プログラミング言語の文法を理解する役目も負っていると思われる．これらのエージェントが対話をすることによってプログラミングが行われているとすれば，各エージェントの働きと対話の過程を，何かしらの支援システムによって追体験させることで，各エージェントとその対話能力を育成させれば，プログラミング能力を高められるのではないかと考えられる．

たとえば，本システム全般に使用されている視覚化機能の特徴は，文法によらずに「仕様を考えるエージェント」の働きを支援することと言える．また，視覚化された情報とソースコードを関連づけて表示することで「コードを理解するエージェント」の成長を補助することが期待できる．2 章で紹介した構造指向による段階的なコーディングガイド機能で行う演習は，「仕様を考えるエージェント」の働きを追体験させ，その指示に従ってコーディングすることで「与えられた仕様をコード化するエージェント」の働きを訓練する，ということである．3 章で提案した動的シーケンス図を使用した演習方法は，「仕様を考えるエージェント」によってプログラムの実際の振る舞いの一面を理解し，それをソースコードとともに観察することで，「コードを理解するエージェント」の働きを支援することになると思われる．そして，自作課題によって「与えられた仕様を

コード化するエージェント」を鍛えることができるものと思われる．

実際にプログラミングを行うエージェンシーがどのようなエージェントの相互作用によって成立しているかは不明だが，プログラミング学習用の手法や支援システムを考案する上で，その解明もしくは信頼度の高いモデル構築の試みは重要と思われる．著者らは入門段階から設計学習の段階まで広範に使える視覚化支援システムの構築を目指しており，支援・強化すべきエージェントの働きを見逃していないかを吟味する上でもこのようなモデルが必要である．今後のこの方面の研究発展を期待したい．

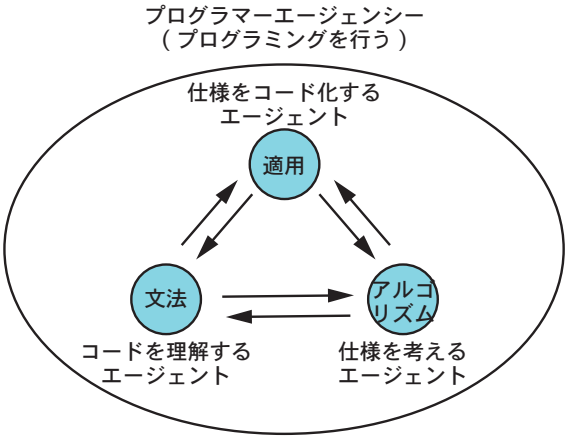

図 13 仮定したプログラマーエージェンシー
Fig. 13 An assumed explanation for programmer agancy

5. おわりに

本論文では，単なる視覚化機能だけでは効果は限定的であるという問題点，特にソフトウェア作成能力の向上は望めないという点について述べ，それを解決するために考案した機能・演習方法をふたつ紹介した．2 章では，学習者に仕様を提示した上でソースコード内の構造ごとにソースコードを書かせると同時に視覚化表示し，その進捗状況をモニタできるシステムを紹介した．また，3 章では複雑な関連性を持つオブジェクト間の振る舞いについて，シーケンス図をプログラムの動作に従って動的に生成・視覚化するとともに，静的視覚化図とも連携させて表示することで理解を助け，その上で応用製作課題を行う演習方法について提案した．最後に，プログラミング学習支援システムの妥当性を検討するために，プログラミングという行為を複数のエージェントの働きであると仮定し，本システムの各機能が各エージェントの能力と協調力を成長させる可能性について述べた．今後は，これまで開発してきた本システムの各機能を授業などで幅広く利用し，教育的効果などの検証を行っていくとともに，有効と思われる視覚化機能と演習方法の組み合わせを検討していく予定である．

参考文献

[1] 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，Vol. 96, No. 4, pp. 25–32 (1996).
[2] 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コンピュータと教育，Vol. 103, No. CE-071, pp. 13–20 (2003).
[3] 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), pp. 58–71 (2003).
[4] 喜多義弘，川添貴議，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 SS，Vol. 104, No. 570, pp. 19–24 (2005).
[5] 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ェア，Vol. 24, No. 3, pp. 153–169 (2007).
[6] 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 SS，Vol. 106, No. 426, pp. 43–48 (2006).
[7] 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，p. 212 (2009).
[8] 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)，pp. 4–413, 4–414 (2009).
[9] 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，p. 170 (2010).
[10] 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，信学技報教育工学，Vol. 110, No. 453, pp. 131–136 (2011).
[11] 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，p. 139 (2011).
[12] 大城正典，永井保夫：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，情報処理学会 情報教育シンポジウム 論文集 SSS2011，pp. 59–66 (2011).
[13] 大城正典，永井保夫：Eclipse を用いたオブジェクト指向プログラミング教育支援視覚化システムの設計と実装，信学技報教育工学，Vol. 112, No. 500, pp. 185–188 (2013).
[14] 大城正典，永井保夫：モニタ機能と可視化機能を持った構造指向による漸次的なプログラム作成学習システム，信学技報教育工学，Vol. 113, No. 482, pp. 31–34 (2014).
[15] 大城正典，永井保夫：段階的コーディングガイド機能およびモニタ機能を持つオブジェクト指向プログラミング教育のための視覚化支援システムの提案，信学技報教育工学，Vol. 114, No. 260, pp. 53–58 (2014).
[16] Gamma, E., Helm, R., Johnson, R. and Vlissides, J. M.: *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, Reading, Massachusetts (1994).
[17] マーヴィン・ミンスキー：心の社会，産業図書 (1990).
[18] 大城正典，山川裕子，J., M. K.，松下孝太郎，布広永示：プログラミング学習支援システム CAPTAIN における学習状況把握機能の開発，日本教育工学会「教育実践を指向した学習支援システム／一般」研究会，pp. 81–84 (2010).