経験知習得のための思考の協調的追体験学習支援システムの開発 -デザインパターンを対象にして-

代表研究者	小尻智子	関西大学システム理工学部　准教授
共同研究者	瀬田和久	大阪府立大学大学院理学系研究科　教授
共同研究者	林佑樹	大阪府立大学現代システム科学域　助教

1 はじめに

デザインパターンは，オブジェクト指向パラダイムでのソフトウェア設計ノウハウを集めたカタログである[1]．よく出会う様々な問題に対して先人らが試行錯誤して得た経験を一般化したものであり，オブジェクト指向設計における経験知とみることができる[2]．先人による創意・工夫のプロセスを経て形式知化されたデザインパターンの学習を通じて，その設計方法の意義や適用条件をより深く理解することは，オブジェクト指向設計における設計意図の習得と捉えられ，パターンを直接使用しない場合であっても優れた設計を行えるようになると考えられている[1]．この意味でデザインパターンはオブジェクト指向の初学者にとってもよい教材であると考えられる．しかし，そこに埋め込まれた設計意図を理解しないまま単に記憶していくのでは，新しい問題に対して自ら創意工夫して設計できるようになるとは考えにくい．

デザインパターンのような経験知が生み出される過程を体験することは，そこでなされた意志決定基準(以後，設計意図と呼ぶ）の理解を促すと考えられる．しかし，成果物のみが示されその生成過程が明記されていることは多くはない．経験知が形成されるまでには様々な生成物を比較し，利点・欠点を検討する過程を経て何らかの基準に基づく選択がなされているが，その基準である設計意図が暗黙になっているために，学習者自身がその過程を推察して意図を理解できるかどうかは学習者の能力に依存する．

デザインパターンの学習支援では，問題を与え，これに対処する設計を学習者が作成することからスタートし，修正を加えていくことでデザインパターンを導出させようとする研究がある[3, 4]．しかし，デザインパターンの設計意図を修得しようとする学習者にデザインパターンを導出させることは必ずしも容易ではない．本研究では，正解としてのデザインパターンを出発点として，機能を維持しつつ望ましくない設計（代替設計）に変更させることで，デザインパターンの設計意図を読み取るような学びを促すことを目的とする．このことで，経験知の形成過程の一部を体験する学びが学習者に課せられることとなり，デザインパターンを教材とした設計意図の学びを促すこととなる．

我々はこれまで，デザインパターンの用いられたクラス図から用いない代替設計を作成するための支援システムを構築してきた[5, 6]．しかし，代替設計を作成させるだけではデザインパターンの利点を明示的に理解させることはできない．そこで，本研究課題では，設計の対象としている問題を拡張する問題設定であるシナリオを与え，作成した他の設計とデザインパターンを用いた設計との比較を促し，デザインパターンに関するより深い理解を促す．また，本学習を実現するために，シナリオを満足するようにデザインパターンを用いた設計と代替設計を変更できる環境を構築する．また，構築した環境を用いた学習を通して，シナリオを用いた設計の良さを考察する学習方法の有効性を明らかにする．

2 設計意図習得のためにデザインパターン学習

デザインパターンは特定の問題に対する合理的な解となる設計カタログである．そこでの特徴的なクラスや関連には，先人達が試行錯誤して確立した結果が示されている．したがって，特徴的なクラスや関連構造を壊しそれらを用いない設計に変更することは，先人が試行錯誤する過程で生成した悪い設計を導出することにつながる．その上で，

- 悪い設計に存在しておらず／しており，優れた設計に存在している／いないものは何だろう？
- 悪い設計で解決できず，良い設計で解決できる問題は何だろう？

を考えることで，デザインパターンの設計に込められた暗黙的な設計意図を考察し，理解を深めることが本研究の着想である．

Adapterパターンを用いた設計（図1，[7]）を例にとり説明する．Adapterパターンは，既存のクラス(Adaptee)を新しいクラスに(Target)差し替えるための設計方法を提供しており，新しいクラスを継承し，メ

ソッド中で既存のメソッドを呼び出す Adapter と呼ばれるクラスを利用することが特徴であるため，新しいクラスを継承する理由を考察することが Adapter パターンを理解する手段の一つとなる．Adapter パターンの利点を理解する変形例として，Adapter クラスを用いず，Targtet で既存のメソッドを直接呼び出す図 2 のような設計が考えられる．図 1 と図 2 を Adapter パターンの利点が発揮できるような問題設定に対して比べることで，継承を組み入れられた設計意図への気づきを促すことができると考えられる．例えば，「さらに Adaptee ではなく，Adaptee2 に置き換えなければならなくなった」というような問題設定が考えられる．この問題設定に対して図 1，図 2 をそれぞれ拡張したクラス図の例が図 3，図 4 のようになる．図 3 では Target の内容を変更しなくてよいが，図 4 では Target にメソッドを追加する必要があるため，図 3 の設計の方が拡張性に優れているといえる．

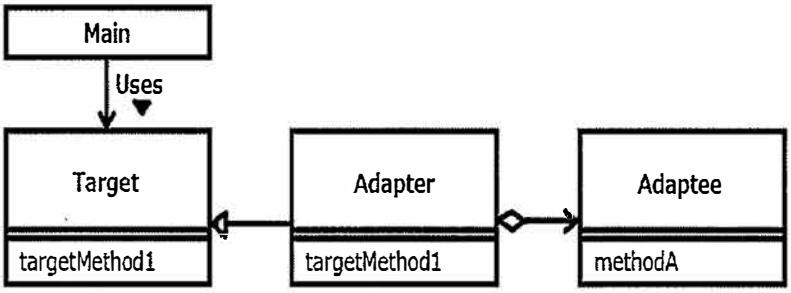

図 1 Adapter パターンを用いた設計

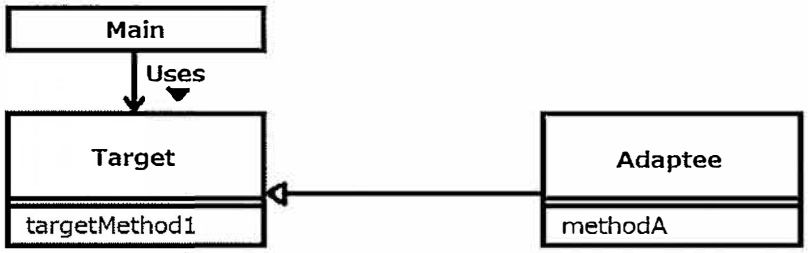

図 2 Adapter パターンを用いない設計

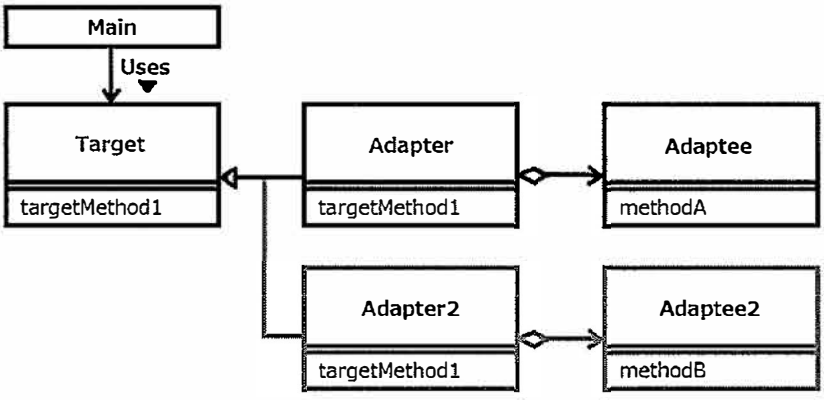

図 3　図 1 を拡張したクラス図

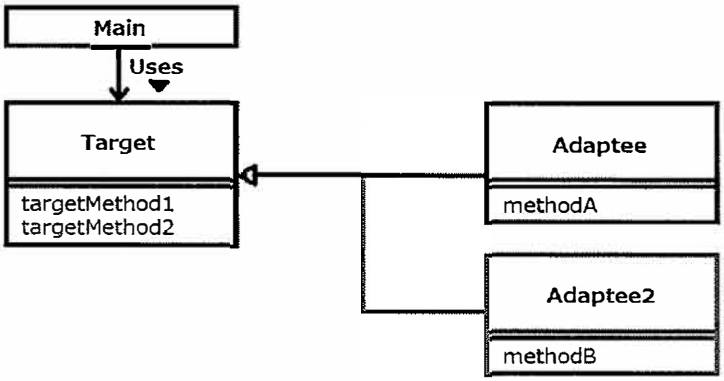

図 4 図2を拡張したクラス図

3 デザインパターン学習支援環境の枠組み

学習者自身で与えられた設計からオブジェクト指向の特徴を減少させた設計を生み出せればよいが，学習者が必ずしもオブジェクト指向の特徴が現れている箇所を理解し，同等の振舞いを果たす設計に変更できるとも限らない．また，設計を変形できたとしても，悪い設計と比較して良い設計の良さを実感できる問題設定を自身で考案できるとは限らない．

本研究では，デザインパターンを用いたプログラム（以後，良い設計とも呼ぶ）を，デザインパターンを用いず等価な振舞いをするクラス図へと変形させることを支援する環境を提供する．また，代替設計を拡張させるような問題設定であるシナリオを明示的に与え，それに対するクラス図の拡張を考えさせ，最終的には利点を解答させる問題（拡張問題）として解答させることで，学習者自身で問題設定を考える負荷を軽減する環境も構築する．図5にシステムの枠組みを示す．

代替設計作成支援機構は，従来研究で開発された機構であり，システムに与えられたデザインパターンを用いた良い設計から代替設計を作成するための支援を提供する．システムは，良い設計に対する代替設計をあらかじめ知識として保持しており，学習者が代替設計と一致しない変形をした場合は，代替設計が生成できるようアドバイスを生成する．なお，先人が試行錯誤した設計は膨大であることが想定され，全ての設計を生成し比較させることは本研究の目的ではない．デザインパターンを理解する上で重要な箇所は，オブジェクト指向の特徴的な設計を含んでいる箇所であるので，オブジェクト指向の特徴を削除した設計を学習者に変形してほしい代替設計として，あらかじめシステムに複数保持することにする．

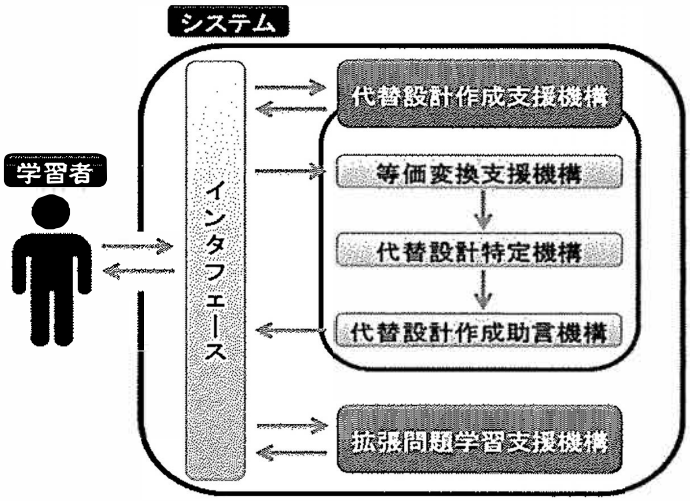

図 5 システムの枠組み

代替設計作成支援機構は以下の機構で構成される.

- 等価変換支援機構

学習者がデザインパターンのクラス図と等価な振舞いをする代替設計を作成できるよう支援する[6].本機構は特定のクラス構造と等価な振舞いをする別のクラス構造を等価変換規則として保持しており，学習者の設計したクラス図に等価変換規則を適用することで，デザインパターンのクラス図と等価な振舞いをするクラス図であるか判定する．等価な振舞いをするクラス図が作成できていない場合は，修正個所を示す.

- 代替設計特定機構

学習者のクラス図がシステムの保持する複数の代替設計のどの代替設計を意図しているかを特定する[5].システムの保持する複数の代替設計と学習者の作成したクラス図との相違の度合いを調べ，最も小さいものを学習者の意図する代替設計と判断する.

- 代替設計作成助言機構

学習者が代替設計を作成できていない場合に，代替設計特定機構で特定された代替設計を作成できるようにするための支援をする[6]．代替設計と学習者のクラス図の相違点を特定し，その箇所を指摘したり，代替設計と同じ構造を設計できるようにするための助言を生成したりする.

拡張問題学習支援機構では，本研究課題で開発された機構であり，デザインパターンを用いたクラス図と代替設計を拡張するための拡張問題を与える．デザインパターンは，拡張性に優れていると言われている[7]ため，設計を拡張するような拡張問題が望ましい．拡張問題では，その問題設定を与えた際に変更する必要のある設計箇所を多肢選択形式で解答させる．解答の正誤を判定することで，学習者が良い設計と悪い設計の利点／欠点を的確に把握できているかを判断する．的確に解答できていない場合は変更後のクラス図や具体的なプログラムを提示するなどして，理解を促す．Adapter パターンに関する拡張問題の例を表 1 に示す.

表 1　拡張問題の例

拡張問題：「Adaptee クラス（methodA メソッド）に代わって Adaptee2 クラス（methodB メソッド）を使用しなければならなくなりました．Adaptee2 クラスに対応できるようにプログラムを拡張しましょう」
解答形式：「デザインパターンを拡張したプログラムでは［　①　］において［　②　］を［　③　］するだけでよいが，代替設計を拡張したプログラムでは［　④　］において［　⑤　］を［　⑥　］する必要があり修正量が多くなる.」
選択肢： ①デザインパターンにおける既存のクラス名 ②デザインパターンを拡張した場合の変更対象 （インスタンス宣言時のクラス名，インスタンス宣言時のコンストラクタ名，など） ③変更種別（変更，追加，削除） ④代替設計における既存のクラス名 ⑤代替設計を拡張した場合の変更対象 （インスタンス宣言時のクラス名，インスタンス宣言時のコンストラクタ名，など） ⑥変更種別（変更，追加，削除）

拡張問題のもとでのデザインパターンや代替設計のプログラムの変更点を正しく解答するためには，デザインパターンと代替設計の拡張クラス図の全体像を把握し，その拡張クラス図のプログラムが元のプログラムと比較してどの部分で変更が生じるかを考える必要がある．そこで，拡張問題を解答させる際に，まずデザインパターンと代替設計の拡張クラス図を作成させ，その後プログラムの変更点を解答させる形式をとる．拡張クラス図ができていない場合，または拡張クラス図はできているがプログラムの変更点が正しく解答できていない場合にアドバイスを生成する.

デザインパターンと代替設計のうちいずれかの拡張クラス図ができていなかった場合，拡張クラス図の作成方法を理解できていないと考えられる．そこで，拡張クラス図を作成する際に考えるべきクラス・関連に着目する旨のアドバイスを学習者に与える．それでも正しいクラス図が作成できない場合は，拡張クラス図の具体的な作成方法に関するアドバイスを学習者に与える．また，余分な変形をすると，問題の趣旨に合わ

なくなるだけでなく，作成したクラス図の存在意義がわからなくなる．従って余分な変形が含まれていた場合，余分な変形を無くすためのアドバイスを優先して生成する．

拡張クラス図はできているがプログラムの変更点が正しく解答できていない場合，プログラムとしてどの部分で変更が生じるかを理解できていないと考えられる．そこで，プログラムを拡張する際に変更する可能性のあるプログラム中の箇所を具体的に明示するアドバイスを学習者に与える．それでも変更点を解答できない場合は，変更後のプログラムを明示するアドバイスを学習者に与える．それでも正しく解答できない場合は，解答した変更点の誤り箇所を指摘するアドバイスを学習者に与える．なお，プログラムの変更点の選択肢には「コンストラクタ」「インスタンス」などの用語が含まれており，学習者がそれらの用語を正確に理解していない可能性がある．そこで，変更点の誤り箇所を指摘するだけでなく，「コンストラクタ」「インスタンス」などを使用したプログラムの具体例を提示するアドバイスも同時に与える．

拡張クラス図の作成に関するアドバイス生成パターンのフローチャートを図 6 に示す．まず，学習者の作成したデザインパターン・代替設計の拡張クラス図とシステムの保持する拡張クラス図が一致しているかどうかを判定する．一致していればデザインパターン・代替設計の拡張クラス図は正解となり，学習者にプログラムの変更点の解答を促す．一致していなければデザインパターン・代替設計いずれかの拡張クラス図が誤答であるので，まず余分な変形をしているかどうかを判定する．余分な変形をしていれば，余分な変形を無くすための方法が記述されたアドバイスを生成する（アドバイス C）．余分な変形をしていなければ必要な変形がされていないため，まず拡張クラス図を作成する際に考えるべきクラス・関連に着目する旨のアドバイスを生成する（アドバイス A）．アドバイス A を既に出していれば必要な変形に対するアドバイスを生成する（アドバイス B）．以上のアドバイス生成を学習者がデザインパターン・代替設計両方の拡張クラス図を作成できるまで繰り返す．

プログラムの変更点の解答に関するアドバイス生成パターンのフローチャートを図 7 に示す．学習者の解答したプログラムの変更点とシステム内の解が完全一致しているかどうかを判定し，一致していれば正解となり「Excellent」と出力して終了する．一致していなければ誤答であるので，まずプログラムを拡張する際に考慮すべきプログラム箇所を具体的に明示するアドバイスを生成する（アドバイス D）．アドバイス D を既に出していれば，変更後のプログラムを明示するアドバイスを生成する（アドバイス E）．アドバイス E を既に出していれば，学習者の変更点の誤り箇所を指摘するとともに，「コンストラクタ」「インスタンス」などを使用したプログラムの具体例を提示するアドバイスを生成する（アドバイス F）．

アドバイス A～F に対応するアドバイスはアドバイス・テンプレートで保持する．アドバイス・テンプレートを図 8 に示す．＜対象＞とは変更対象のことで，誤って生成・削除されたクラス・関連が入る．これらは学習者の作成したデザインパターン・代替設計の拡張クラス図とシステム内の各々の拡張クラス図との差分をとることにより取得できる．＜要素＞とはインタフェース，継承などのクラスのプロトタイプを指す．＜誤り箇所＞とは学習者のプログラムの変更点の解答で誤っている選択肢の番号のことである．これは学習者の解答したプログラムの変更点とシステム内の解を比較することにより取得できる．

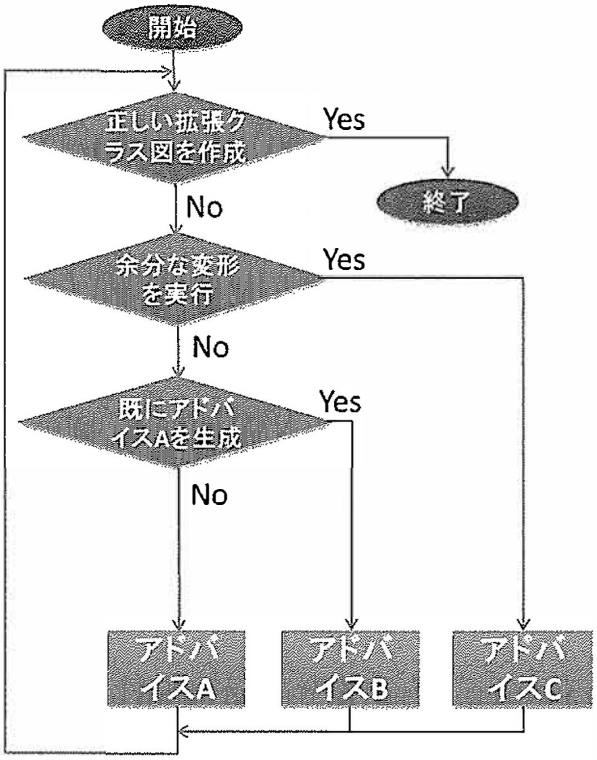

図 6　拡張クラス図の作成に関するアドバイス生成のフローチャート

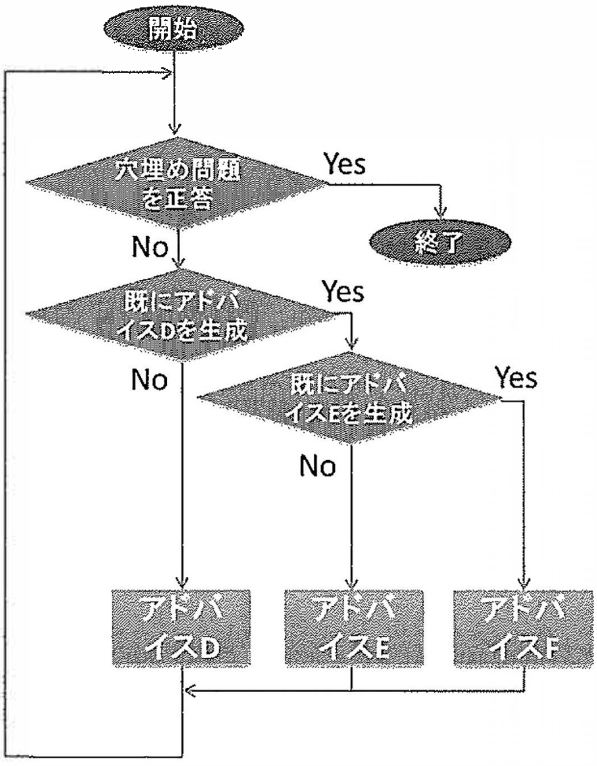

図 7　プログラムの変更点の解答に関するアドバイス生成のフローチャート

(A) 拡張クラス図の実現のため，*<対象>*に着目して考えてみましょう．

(B) 拡張クラス図の実現のため，*<対象>*を(追加／削除)してみましょう．

(C) *<対象>*の(追加／削除)はその *<要素>*の存在意義に関わるような変形ではないので，余分な変形です．
*<対象>*を(削除／追加)してみましょう．

(D) 拡張する際に考慮しないといけない箇所を青色で提示します．具体的にどう変更するかを考えてみましょう．
(※該当するプログラム箇所を青色で表示)

(E) 変更後のプログラムを赤色で提示します．最後に「プログラムの拡張」ウィンドウで，プログラムの具体的な変更箇所をまとめてみましょう．
(※変更後のプログラムを赤色で表示)

(F) プログラムの具体的な変更箇所をまとめてみましょう．
リスト *<誤り箇所>*を考え直してみましょう．
プログラム中，newの直後にあるのが「コンストラクタ名」，インスタンス宣言時にインスタンスの直前にあるのが「クラス名」，インスタンスを使ってメソッドを呼び出しているものが「インスタンスが呼び出すメソッド名」です．
(例) Frame w = new DrawFrame();
w.setdrawframe();
※Frame：クラス名　　DrawFrame：コンストラクタ名
w：インスタンス名　　setdrawframe：メソッド名

図 8　拡張問題に関するアドバイス・テンプレート

4 プロトタイプ・システム

図 9 に代替設計作成のためのインタフェースを示す．問題選択部で学習者が学習したいデザインパターンを選択すると，クラス図表示部に選択されたデザインパターンを用いた良い設計が表示される．エンティティ編集部ではクラスを，関連編集部では関連を文字形式で編集できるようになっている．再描画ボタンをクリックすることで，編集された内容をクラス図表示部に再描画できる．アドバイス生成ボタンをクリックした際，学習者の作成したクラス図に誤り箇所が含まれていればアドバイス表示部に正しい解答をさせるためのアドバイスが表示される．

学習者が代替設計を作成できると，良い設計と学習者の作成した代替設計のクラス図（図 10），および拡張問題（図 11）が表示される．図 10 のクラス図は自由に変形することが可能であり，学習者が拡張問題の解答を考えるための補助として活用する．拡張問題提示インタフェースでは，変更点選択リストから問題の解答が選択できるようになっている．アドバイス生成ボタンを押すと，学習者の解が正しいか否かを判定し，もし学習者が適切な解を生成できていない場合は助言を生成する．図 12 にアドバイスの例を示す．拡張問題で変更が生じる個所が青色で，変更後のプログラムが赤色で記述されている．

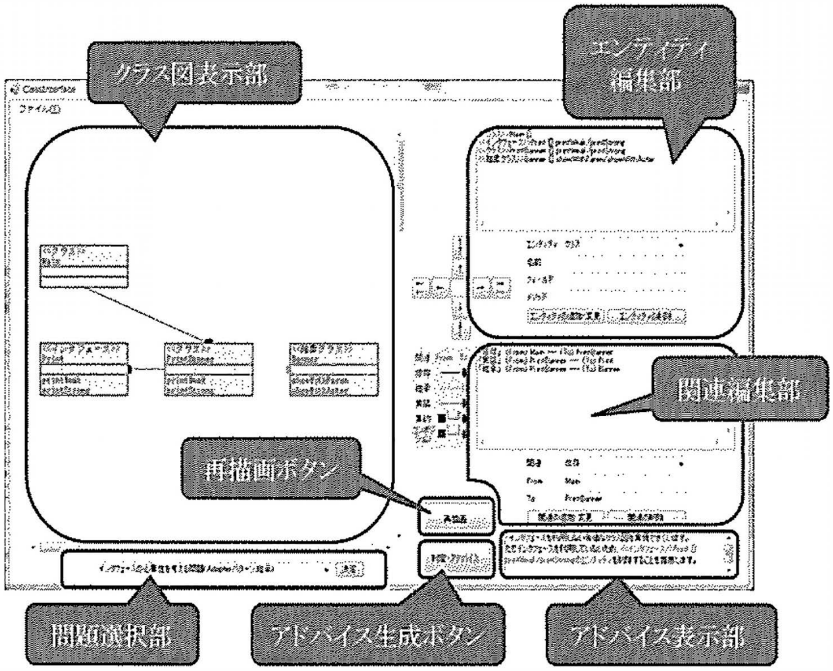

図 9　代替設計作成支援インタフェース

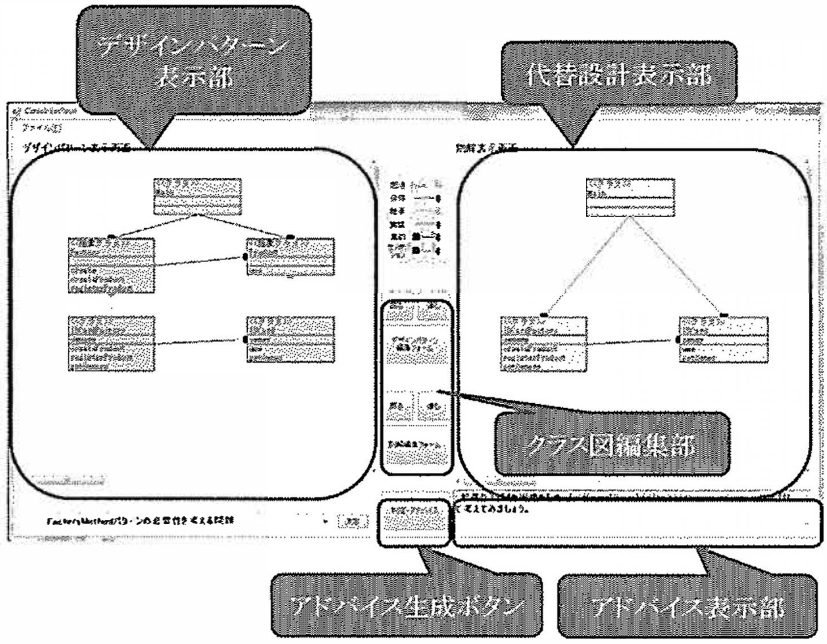

図 10 クラス図拡張インタフェース

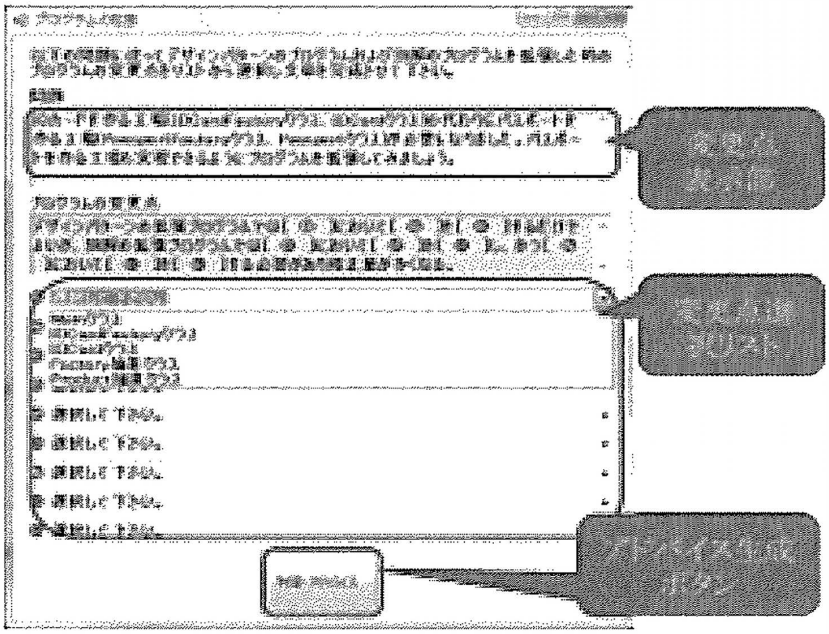

図 11　シナリオと拡張問題提示インタフェース

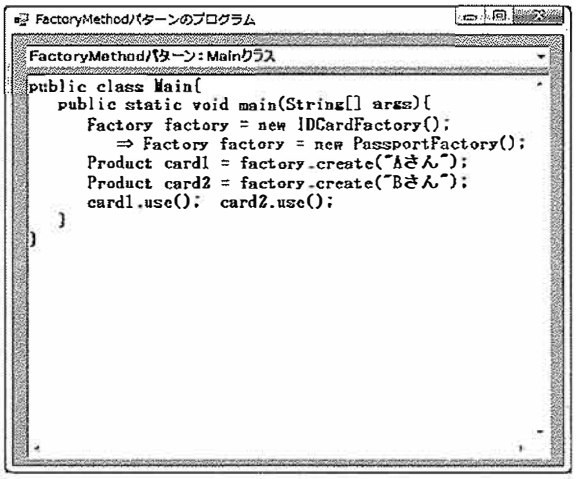

図 12　アドバイス例

5 評価実験

5.1 シナリオに基づいた設計比較の有効性の評価

本研究課題で構築したプログラム拡張問題学習環境を備えたデザインパターン学習支援システムの有効性を検証することを目的に評価実験を実施した．被験者は，オブジェクト指向プログラミングを学習し，かつ作成した経験はあるが，デザインパターンの設計意図を深く理解していない大学院生 9 名である．本実験では，あらかじめ代替設計を準備し，その代替設計とデザインパターンを用いた設計を比較させることで，デザインパターンの良さを理解できるかどうかを調査した．

初めに，デザインパターンの理解度を確認するため事前テストを実施した．事前テストの大問 2 問で構成される（それぞれⅠ・Ⅱとする）．Ⅰは，インタフェースを使用した構造を包含した Adapter パターンを含むクラス図を用いた．Ⅱは，抽象クラスや継承を使用した構造を包含した FactoryMethod パターンを含むクラス図を用いた．小問はそれぞれ 2 問ずつである．Ⅰにおいて，1 問目はインタフェース Component の役割を問う問題である．2 問目はそのインタフェース Component を使用する構造上の理由を問う問題である．Ⅱにおいても同様に，1 問目は抽象クラス Factory の役割を問う問題で，2 問目はその抽象クラス Factory を使用する構造上の理由を問う問題である．これらの問題は，インタフェースや抽象クラスのプログラムの中身や，それらを使用する設計意図を被験者が理解しているかどうかを確認するために設けた．このとき，Adapter パターンや FactoryMethod パターンを用いた各々のプログラムの全体像を被験者に把握できるように，各ク

ラスのプログラムを記述した資料を配布し，それらを参照しながらテストに解答させた．

また事前テストと同時に，デザインパターンに関する意識調査のため事前アンケートに回答させた．事前アンケート項目を表 2 に示す．設問は 2 問である．設問①はデザインパターンの学習経験を，設問②はデザインパターンを使用したプログラミングの経験を問う項目で，いずれも「ある」「ない」の 2 段階で評価させた．

次に，被験者に対し，プログラム拡張問題に基づいたデザインパターンの学習方法をスライドを用いて説明した．まず，簡便なクラス図とプログラム拡張問題の例をあげ，それに対するプログラムの変更点の解答方法を説明した．また，プログラムの変更点を解答する際に考えるべき拡張クラス図，拡張プログラムについて説明し，その後，本問題のプログラムの変更点の解答例についても説明した．

説明後，被験者にシステムを利用して拡張問題を学習してもらった．まず，あらかじめシステムの持たせておいたデザインパターンおよび代替設計を提示し，各々の拡張クラス図を作成させた．正しい拡張クラス図を作成後にプログラムの変更点を解答してもらった．その際，システムの操作方法を口頭で説明するとともに，操作方法を示すプリントを配布した．問題は全部で 2 問であり，1 問目は Iterator パターンを用いたクラス図中の Aggregate インタフェースおよび Iterator インタフェースの必要性を考える問題，2 問目はの Adapter パターンを用いたクラス図中の Adapter クラスの必要性を考える問題である．

学習後，デザインパターンの理解度の変化を確認するため事後テストを実施した．事後テストはインタフェースを使用した構造を包含した Prototype パターンと，抽象クラスや継承を使用した構造を包含した TemplateMethod パターンのクラス図を用いて，事前テストと同様にそれぞれ Product インタフェース，AbstractDisplay 抽象クラスの役割とそれを使用する構造上の理由を問う問題を出題した．このとき，Prototype パターンと TemplateMethod パターンを用いたプログラムの全体像を被験者に把握できるように，各々のパターンのプログラムを記述した資料を配布し，それらを参照しながらテストに解答してもらった．問題形式は事前テストと同様である．事前テストと事後テストの結果を比較して，学習効果が得られたかどうかについて評価した．

最後に，本学習支援手法の性能を評価するため事後アンケートに回答してもらった．事後アンケート項目を表 3 に示す．設問①～③はそれぞれ 1（低評価）から 4（高評価）の 4 段階で，設問⑤は 1（低評価）から 5（高評価）の 5 段階で評価してもらった．その他，本システムに関する意見・感想・提案事項を自由記述してもらった．

表 2　事前アンケート項目

設問	内容
1	デザインパターンを学習したことがあるか
2	デザインパターンを使ってプログラムを作成したことがあるか

表 3　事後アンケート項目

設問	内容
1	プログラムの変更点を考えることでデザインパターンの設計の良さは理解できたか
2	新しいデザインパターンを学習するときに設計を拡張してみようと思うか
3	システムからのアドバイスはプログラムの変更点を考える上で理解できたか
4	システムは操作しやすかったか

事前アンケートの結果について述べる．事前テストと同時に実施したデザインパターンに関する意識調査のためのアンケートの結果を表 4 に示す．設問①「デザインパターンを学習したことがあるか」で，学習したことがあると回答した被験者は 9 人中 7 人であった．一方，設問②「デザインパターンを使ってプログラムを作成したことがあるか」で，作成したことがあると回答した被験者は 9 人中 1 人にとどまった．したがって，デザインパターンのプログラムの構造を学習した経験はあっても，それを自ら使用して新たなプログラムを作成した経験がない被験者がほとんどである．

次に事前テストと事後テストの結果について述べる．表 5 にその結果を示す．事前テストについて，I（インタフェースを使用した構造を含むデザインパターン）の 1 問目「インタフェースの役割を問う問題」では，被験者 9 人中 1 人が正しく説明できていたが，それ以外の 8 人は問題の意図と異なる説明をしているか，白紙解答であった．II（抽象クラスや継承を使用した構造を含むデザインパターン）の 1 問目「抽象クラスの役割を問う問題」でも，被験者 9 人中 3 人が正しく説明できていたが，それ以外の 6 人は問題の意図と異な

る説明をしているか，白紙解答であった．また，Ⅰの2問目「インタフェースを使用する構造上の理由を問う問題」やⅡの2問目「抽象クラスを使用する構造上の理由を問う問題」では，正しく説明できていた被験者がそれぞれ9人中0人，9人中1人と，1問目よりもさらに正解率が低かった．このように，事前テストの段階ではデザインパターンの概念が理解できていない，あるいは概念は理解できていてもなぜその構造をしているかが理解できていなかったことがうかがえる．

一方，事後テストでは，Ⅰの1問目，Ⅱの1問目とも，被験者9人中5人が問題の趣旨に沿ってよく説明できており，「メソッド名のみを定義し，そのメソッドの具体的な処理をサブクラスで定義させる」といった趣旨の解答が多かった．ただ，正しく解答できなかった被験者も多く見受けられた．この原因の一つに，各クラスの役割を理解させる支援ができていなかったことがあげられる．したがって，デザインパターンの設計の良さだけでなく各クラスの役割を思考させる支援環境が必要だと考えられる．Ⅰの2問目やⅡの2問目では，正しく説明できていた被験者がそれぞれ9人中7人，9人中9人と，1問目よりもさらに正解率が高く，「インタフェース（抽象クラス）を利用することで，サブクラスと他のクラスの実装が切り離されるため，プログラムを拡張する際に他のクラスの変更を最小限に抑えられる」といった趣旨の解答が多かった．これらの結果から，本システムによる支援機能がデザインパターンの設計の良さを理解させる上で有効であったことが明らかになった．

最後に，事後アンケートの結果を表6，7に示す．①「プログラムの変更点を考えることでデザインパターンの設計の良さは理解できたか」と，②「新しいデザインパターンを学習するときに拡張プログラムを作成してみようと思うか」では，いずれも被験者全員が3か4と回答していた．詳細な意見を聞いたところ，「既存プログラムを拡張することの意義を理解することができた」「デザインパターンを用いたプログラムの設計の良さをより深く理解することができた」という意見がみられた．これらの結果から，プログラムの拡張を通じた学習方法の有効性を多くの被験者に認識してもらえたことが明らかになった．さらに「システムで具体的なプログラムが見れることにより，クラス図の全体像が把握しやすかった上，プログラムの変更箇所が考えやすかった」という意見もあった．このことから，拡張クラス図の作成や拡張問題の解答を通じた学習において，全てのクラスのプログラムを提示する機能によって，元のクラス図や拡張クラス図の全体像を把握させることができ，プログラムを読みながらプログラムの具体的な変更箇所をスムーズに考えさせることができたと考えられる．③「アドバイスはプログラムの変更点を考える上で理解できたか」では，アドバイス機能を全く使用していない被験者が2と回答していたが，それ以外の被験者8人が3か4と回答しており，「プログラム中で変更点をアドバイスする所が見やすく，本機能で理解がより深まった」という意見もあったことから，プログラム中の変更点を色を変える機能が，被験者の学習を適切に支援できていたことがうかがえる．④「システムは操作しやすかったか」では，被験者9人中8人が4か5と回答しており，システムのユーザビリティは改善されていたことが明らかになった．但し，「エンティティはドラッグ&ドロップで動かせるようにしてほしい」など，システムの操作に改善を求める意見もあったため，今後も引き続きシステムのユーザビリティの改善をしていく必要がある．

表4　事前アンケートの結果

設問	選択肢	人数
①	学習したことがある	7人
	学習したことがない	2人
②	作成したことがある	1人
	作成したことがない	8人

表5　事前テスト・事後テストの結果

テスト		事前		事後	
		1問目(役割)	2問目(理由)	1問目(役割)	2問目(理由)
1問目	正しく説明できた	1人	0人	5人	7人
	間違えた説明をしていた	6人	5人	2人	1人
	全く説明できなかった	2人	4人	2人	1人
2問目	適切な代替設計を作成できた	3人	1人	5人	9人
	適切でない代替設計を作成していた	4人	2人	2人	0人
	全く作成できなかった	2人	6人	2人	0人

表 6　事後アンケートの結果 (質問①～③)

設問	選択肢 (1：低評価⇔4：高評価)			
	1	2	3	4
①	0 人	0 人	5 人	4 人
②	0 人	0 人	5 人	4 人
③	0 人	1 人	5 人	3 人

表 7　事後アンケートの結果 (質問④)

設問	選択肢 (1：低評価⇔4：高評価)				
	1	2	3	4	5
④	0 人	0 人	1 人	4 人	4 人

5-2 代替設計の作成およびシナリオに基づいた設計比較の有効性の評価

本研究課題で構築したプログラム拡張問題学習環境に，これまで我々が開発してきた代替設計作成環境を統合し，被験者が自身で代替設計を作成したうえでその拡張問題を考えるという一連の学習方法の有効性を検証するための評価実験を実施した．被験者は，デザインパターンについて学んだことがある大学生 5 名である．

初めに，デザインパターンの理解度を確認するため事前テストを実施した．事前テストには Bridge パターンを用いた．事前テストの問題を表 8 に示す．1 問目は，Bridge パターンを用いた設計を見せ，このデザインパターンが優れた設計である構造上の理由を問う問題である．2 問目は，代替設計を作成させる問題である．これらの問題は，被験者がデザインパターンの設計の良さを理解しているかを問うために設けた．このとき，各クラスのプログラムを記述した資料を配布し，それらを参照しながらテストに解答させた．次に，被験者に対しシステムの操作方法について説明した．その後，被験者にシステムを利用してもらい，提示されたデザインパターンを利用したクラス図の代替設計を作成させた．また，代替設計が作成できた後は拡張問題を提示し，プログラムの変更点を解答してもらった．問題は 1 問で，FactoryMethod パターンに関する問題である．本実験では，最終的にすべての被験者が拡張問題を正しく解答することができた．システムを用いた学習後，事後テストを実施した．事後テストは，事前テストと同様の問題を，Observer パターンを用いたクラス図に対して聞いた．

表 8　事前テストの問題

以下に示すのは Bridge パターンのクラス図です．
1.　このデザインパターンがなぜ優れた設計だといえるか，構造上の理由を説明して下さい．
2.　このデザインパターンと同じ動作をするが設計の良さを低下させる代替設計のクラス図を作成して下さい．

事前・事後テストの結果を表 9 に示す．事前テストについて，2 問とも正しく解答できていた被験者が 1 名いたが，1 問目は 3 人が，2 問目は 4 人が正しく解答することができていなかった．したがって，2 問とも正しく解答できていた被験者 1 人を除き，残りの 4 人は事前テストの段階でデザインパターンがなぜその構造をしているかが理解できていなかったことがうかがえる．一方，事後テストでは，事前テストで 2 問とも正しく解答できていた被験者 1 人に加え，1 問目は残りの 4 人全員が問題の趣旨に沿ってよく説明できていた．2 問目は 4 人中 3 人が適切な代替設計を作成できていた．これらの結果から，提案したデザインパターンの学習方法および本システムによる支援機能がデザインパターンの設計の良さを理解させる上で有効であったことが明らかになった．

表 9　事前・事後テストの結果

テスト		事前	事後
1 問目	正しく説明できた	2 人	5 人
	間違えた説明をしていた	1 人	0 人
	全く説明できなかった	2 人	0 人
2 問目	適切な代替設計を作成できた	1 人	4 人
	適切でない代替設計を作成していた	1 人	1 人
	全く作成できなかった	3 人	0 人

最後に，アンケートに回答してもらった．アンケート項目を表 10 に示す．設問①・④はそれぞれ 1（低評価）から 4（高評価）の 4 段階で，それ以外の設問は自由に記述してもらった．なお，設問②は設問①で 3 または 4 と回答した被験者のみ記述してもらった．

表 10　アンケート項目

①　代替設計作成やプログラムの変更点を考えることでデザインパターンの設計の良さは理解できたか．
②　本実験のどの段階でデザインパターンの設計の良さを理解できたか．
③　デザインパターンの設計の良さを理解する上でなぜ代替設計を作成する学習をしたと思うか．
④　新しいデザインパターンを学習するときに代替設計を作成してみようと思うか．

アンケートの結果を示す．設問①・④の結果を表 11 に，設問②・③の結果を表 12 に示す．①では被験者全員が，④では被験者 5 人中 4 人が 3 か 4 と回答していた．④で 2 と回答した被験者 1 人は，事前・事後テストともに全ての問題で正しく解答できており，デザインパターンの設計の良さを既に十分理解していた．したがって，本実験での学習から習得した知識が少なかったため，学習効果が少なかったと考えられる．また，②の結果から，代替設計を作成するプロセスでデザインパターンの設計意図を理解できる被験者もいるが，拡張問題で良い設計が有効に機能する場面に触れることで初めて設計意図を習得できた被験者もいたことが明らかになった．さらに，③では，被験者全員がデザインパターンの設計の良さと代替設計の悪さを実感するためだと回答していた．これらの結果から，多くの被験者に代替設計作成や問題設定の拡張を通じた学習方法の有効性を認識してもらえたことが明らかになった．

表 11　アンケートの結果（①，④）

設問	1	2	3	4
①	0 人	0 人	4 人	1 人
④	0 人	1 人	3 人	1 人

表 12　アンケートの結果（②，③）

②	
代替設計の作成終了時	2 人
拡張問題の解答時	2 人
拡張問題のアドバイス取得時	1 人
③	
● デザインパターンでは仕様の変更が容易であることを確認するため ● 代替設計が仕様の変更に弱いことを実感するため ● 設計の良し悪しをプログラム変更点の比較によって実感するため	

6 おわりに

本研究では，オブジェクト指向プログラミングにおける経験知の一つであるデザインパターンを対象として，デザインパターンを用いたクラス図を試行錯誤的に変形することでデザインパターンの設計意図を理解する学習手法を提案し，その学習を体験できるシステムを構築した．学習者が設計上の利点を下げる代替設計を作成できるように，学習者の代替設計を判定し，必要に応じてアドバイスを生成する機構を構築した．さらに，代替設計の対象としている問題を拡張した拡張問題を提示することで，良い設計の設計意図を実感できるようなシステムを構築した．評価実験の結果，デザインパターンが優れた設計である理由やデザイン

パターンの利点を下げる代替設計を多くの被験者が解答できるようになったことから，代替設計作成に基づいた学習方法は有効であったことが明らかになった．

提案した学習方法ではデザインパターンの設計意図を理解させることはできても，デザインパターンを使いこなせるようになるための支援はできていない．デザインパターンを適用している問題を抽象化することができれば，現在取り組んでいる問題にデザインパターンをできるか否かを判断できるようになると考える．したがって，様々な問題に対して，デザインパターンの使用可否とその理由を考えることのできる学習支援システムも，今後考案していきたい．

【参考文献】

[1] アラン・シャロウエイ，ジェームズ・R・トロット著，村上雅章訳：「デザインパターンとともに学ぶ オブジェクト指向のこころ」，ピアソン・エデュケーション (2005)

[2] マイケル・ポランニー：「暗黙知の次元」，筑摩書房 (2003)

[3] Stephen WEISS, "Teaching Design Patterns by Stealth", Proc. of SIGCSE 2005, pp.492-494, 2005.

[4] Neishia PILLAY, "Teaching Design Patterns", Proc. of SACLA (2010)

[5] 大江洋希，小尻智子，瀬田和久：「別解作成に基づいたデザインパターン学習における学習者の別解特定機構の構築」，情報処理学会第 76 回全国大会講演論文集，pp.4-551－4-552 (2014)

[6] Hiroki OOE, Tomoko KOJIRI and Kazuhisa SETA: "Learning Support System to Facilitate Redesigning for Understanding Software Design Patterns". Proc. of ICCE 2012, pp.61-65 (2012)

[7] 結城浩，“Java 言語で学ぶデザインパターン入門”，ソフトバンククリエイティブ株式会社 (2004)

〈発 表 資 料〉

題 名	掲載誌・学会名等	発表年月
代替設計との拡張性の比較に基づいたデザインパターン設計意図理解支援システムとその評価	電子情報通信学会技術研究報告，Vol. 114, No. 53, pp. 31-36	2014 年 5 月
Learning Method for Understanding Design Policy of Object-oriented Design and its Meta-learning Support System	Proc. of 22th International Conference on Computers in Education, pp. 129-131	2014 年 12 月