Vol.2017-CE-142 No.16
Vol.2017-CLE-23 No.16
2017/12/9

オブジェクト指向プログラミング教育における クラス図作成演習システムの開発

小清水 誓太[1]　高野 辰之[2]　小濱 隆司[3]　宮川 治[3]

概要：オブジェクト指向設計の学習をするとき，学習者はプログラム全体を把握してクラスの設計方法を学ぶ．講義などで演習を通じて学習を行う際には，学習者は対応した関係であるプログラムのソースコードとクラスの設計を理解する必要がある．我々はその理解を深めさせるために，クラス図作成演習として，学習者にソースコードを読ませクラス図へクラス間の関係を演習用紙に記述させる方法を用いている．しかし，手書きによるクラス図作成演習では記述の不明確さや学習者の人数によって，集計が難しい点があり，学習者全体の理解の傾向を把握することは容易ではない．また，学習者のクラス間の把握する過程なども結果からは読み取ることができない．そこで本研究では，学習者の演習結果や演習中の操作を記録するため，クラス図作成演習システムを開発する．本稿では，演習における本システムの利用方法，本システムの実装に関する構成，学習者の演習記録の教授者への提示方法と本システムの評価について述べる．

1. はじめに

オブジェクト指向プログラミングとは，機能を定義しているクラスとその実態を表すオブジェクト同士の相互作用を表現している．オブジェクト指向設計において，再利用性の高いクラス設計をすることは重要である．プログラミングを行う際は，再利用性の高くプログラムの修正に柔軟であるクラスを作成することを意識する必要がある．

クラス間の静的な関係を表すには UML(Unified Modeling Language) のクラス図が用いられる [1]．設計の修正をするにはクラス図を見てクラスの関係や構造を理解するが，初学者にとってクラス図の抽象化された構造を理解することは難しい．多くの教授者は具体的な例を用いて，クラス設計の概念を教授する [2] [3]．しかしモデリングのみでは，具体的なクラス設計を行った後コーディングを行なっていないため，学習者はクラス図とプログラムの繋がりを理解しきれないこともある．

長谷川らは，与えられたクラス図からプログラムの骨格を形式的に導出し，最低限コンパイルが通る状態にするクラスの形式的記述を行なっている [4]．形式的記述を行うことにより UML で示された仕様を満たしているかが判断できるので，クラス図からプログラムの最低限のクラス間の関係が読み取れるといえる．

設計の修正をするには，実装されたプログラムなどからクラス間の関係を読み取らなければならない．また，オブジェクト指向設計の講義をする教授者は，学習者がプログラムの全体を把握しクラス設計を読み取れるかどうかを，確認する必要がある．

そこで，ソースコードからクラス図を作成する演習を行い，学習者がどのようにクラス設計を理解しているのかを把握する．本研究では，ソースコードを読み解き，クラス間の関係をクラス図へ記述する演習のことをクラス図作成演習と定義する．この演習では，教授者がソースコードを用意し，クラス図にあらかじめクラスを配置する．学習者はソースコードを読み，クラス図へクラス間の関係を記述し提出する．

クラス図作成演習は，東京電機大学情報環境学部で開講されているオブジェクト指向設計の中で手書きで行われていた．手書きによるクラス図作成演習では最終的な解答しか得られず，どのようにソースコードを読み，どのようにクラス間の関係を記述していったかなど，学習者の試行錯誤を見ることができなかった．

そこで本研究では，学習者の演習結果や演習中の操作を記録するため，クラス図作成演習システムを開発する．記録されたデータをもとに学習者全体や個人の傾向を分析・提示し，教授者が講義でフィードバックすることを可能と

[1] 東京電機大学大学院情報環境学研究科
Graduate School of Information Environment, Tokyo Denki University
[2] 関東学院大学理工学部
College of Science and Engineering, Kanto Gakuin University
[3] 東京電機大学システムデザイン工学部
School of System Design and Technology, Tokyo Denki University

Vol.2017-CE-142 No.16
Vol.2017-CLE-23 No.16
2017/12/9

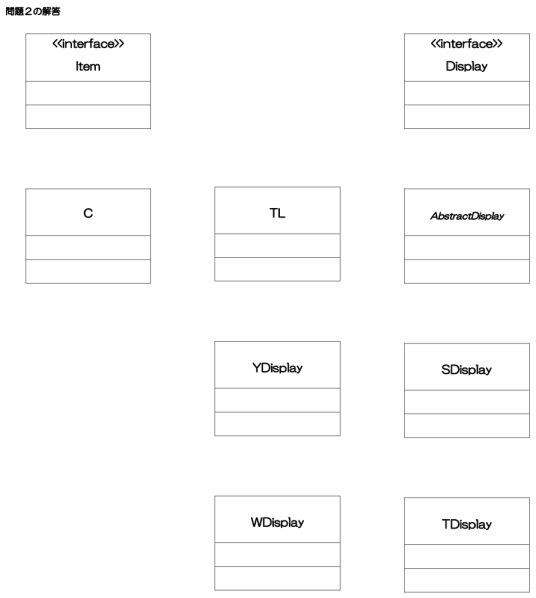

図 1　クラス図作成演習の解答用紙

する．

2. 関連研究

オブジェクト指向設計において学習者にクラス図を用いて概念を教授するシステムはこれまで多く開発されている．

増元らは，対象世界の構造をクラス図として記述するシステムを開発している．記述を通して，初学者にはどのような記述の誤りが生じるのかを分析している [2]．

田中らは，クラス図作成時の編集工程を編集イベント単位で収集し，再現するツールを開発している．田中らのクラス図作成演習は，問題記述からデータモデリングを行う演習を指している [3]．

本研究は，ソースコードからクラス間の関係を読み取ることの練習が目的でありモデリングが目的ではない．

3. クラス図作成演習

本研究のクラス図作成演習は，オブジェクト指向に基づいて設計されたソースコードからクラス図を作成する演習である．プログラムに修正を加える場合，ソースコードからクラスの内容やクラス間の関係を理解する必要がある．そのため，オブジェクト指向設計の講義では，ソースコードからクラス図を作成する演習を行っている．教授者は，オブジェクト指向に基づいて設計したソースコードを用意し，クラス図のクラスを配置する．この演習は学習者がクラス間の関係の理解しているかを確認するため，図 1 のようにクラス図の状態やメソッドは省略している．

クラス間の関係を表 1 に示す．クラスとクラスを結ぶ矢印は継承，実装，関連，依存の 4 種類存在し，それぞれの矢印が集約しているかどうかを表すことができるので，計

表 1　本研究のクラス図で用いるクラス間の関係

表記	関係	説明
	継承	汎化ー特化関係にある
	実装	インタフェースを実装
	関連	インスタンスとして関連
	依存	利用関係にある
	集約	全体とその一部

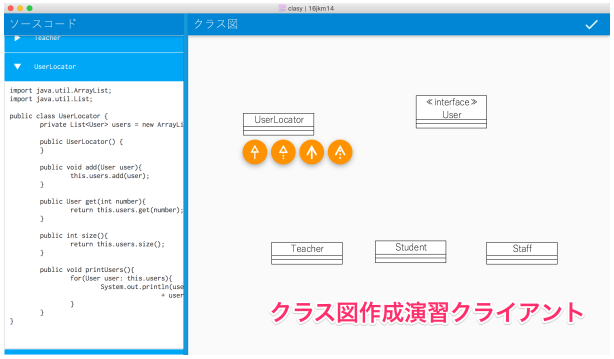

図 2　演習画面

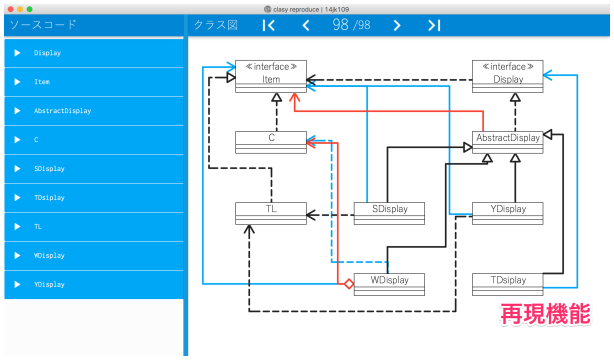

図 3　再現ツールで操作を最後まで再現した画面

8 種類の矢印を記述することができる．矢印には始点と終点があり，クラスからクラスへ結ぶ．矢印の終点が同じクラスを指している場合，黒点を記述し矢印を接続することを許可している．点線矢印の切れ間や線の長さは指定していない．

採点方法は，教授者が模範解答を作成し学習者の解答と比較する．不足や余剰している矢印ごとに減点する．クラス図には省略可能な矢印が存在する．学習者が省略可能な矢印を記述した場合，減点は行わない．

問題用紙のソースコードにはクラス設計と関係がない実行用のソースコードが写っているが，定期試験の一部としてクラス図作成演習を行っていたためである．クラス図作成演習を行う際には実行用ソースコードは読まない．

4. システム概要

本システムのユーザは学習者と教授者である．学習者は

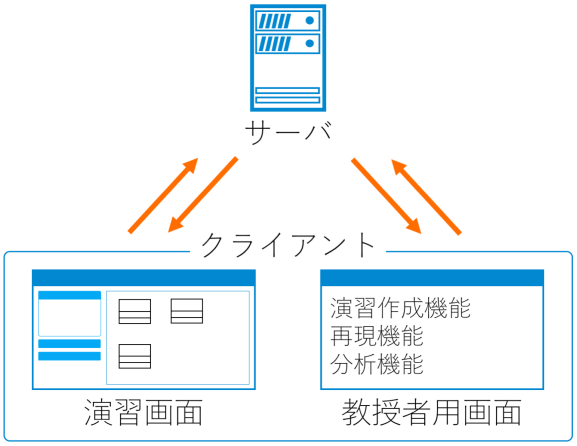

図 4　演習画面

演習を行い，教授者は演習作成と学習者の解答結果の分析を行う．学習者が演習を行う画面を図 2 に示す．演習画面では，ソースコードの開閉，矢印の記述，矢印の削除，集約の付加，解答の提出ができる．演習中にシステム内で行った操作は保存される．

教授者は，演習を作成する機能と学習者の試行錯誤を再現する機能，正解率や解答データの特徴を確認することができる分析機能を使うことができる．

教授者は演習作成機能を使ってクラス図作成演習を作成する．演習はシステムにソースコードをアップロードして作成する．クラス図作成演習システムは手書きの演習を行っていた時と同じ採点基準を設けるため，教授者がクラスの配置を決定しクラス間の関係の模範解答を作成し保存する．演習にはオープン/クローズの状態がある．学習者はオープンしている状態の演習を解答することができる．

再現機能で学習者の操作を最後まで再現した画面を図 3 に示す．学習者を対象としてデータを読み込むと，その学習者が演習中にどのような操作を行ったかを再現することができる．この機能は学習者の一操作ずつを進める戻すという形式で再現をしている．学習者の最後の操作では，不足や余剰している矢印に色をつけて最終的にどこに誤りがあるのかを確認できる．

分析機能では同じ演習を受けた学生全体で，正解者の人数や全体的に多く誤っている矢印を確認することができる．教授者はこの結果をみて，講義でフィードバックをすることが可能となる．

5. システム詳細

本システムは図 4 のようにサーバクライアントで構成される．クライアントには学習者が利用するクラス図作成演習クライアントと教授者が利用する演習管理クライアントがある．演習管理クライアントには演習作成機能，再現機能，分析機能が存在する．

それぞれのクライアントは多様なプラットフォームで同様に動作させるため，JavaFX [5] を用いて実装を行なった．また，サーバは RESTful Web サービスとして開発するため，JAX-RS を実装している Jersey [6] を用いて実装を行なった．

5.1 クラス図作成演習クライアント

クラス図作成演習クライアントでは学習者がクラス図作成演習を行う．演習画面では左側にソースコード，右側にクラス図が表示されている．また，画面右上に提出ボタンが配置されている．

本システムの演習で使用するクラス図は，クラス間の関係を読み取ることに重点を置くため，クラスはあらかじめ配置し，クラス間の関係を消している．表 1 でクラス間の関係の種類を示す．

クラス図作成演習の矢印は，シャフトが実線と破線が存在する．また，アローヘッドは論理積 (∧) の形と白抜き三角形 (△) の形が存在する．シャフトの始点は，ひし形になる場合がある．

クラス間の関係の矢印は，垂直か水平に直線で記述することができる．任意の場所に角度 90 度の曲げを作ることができる．クラスを選択すると，クラス間の関係の矢印ボタンが表示される．記述する矢印を決定すると，選択されたクラスが始点クラスとなりオレンジに背景色が変化する．そして，終点のクラスを選択すると矢印が記述できる．曲げを作るには，クラス図背景をクリックする．クリックされた場所で矢印を曲げることができる．

ソースコードとクラス図のセパレータは可動式で，ソースコードの表示幅やクラス図の表示幅を変更することができる．

はじめは，各ソースコードが閉じている．クラス間の関係を読み解くには適切なソースコードを展開して確認する必要がある．学習者がクラス間の関係を適切に読み取れているのかを確認するため，ソースコードの開閉の操作を記録している．

クラス図作成演習クライアントの画面には，ログイン画面，教授者選択画面，演習選択画面がある．教授者選択画面はクライアントの初回起動時に，学習者が履修する教授者（講義を履修する教室など）を選択することができる．サーバに保存されている教授者一覧から，学習者が履修する教授者を選択する．選択された履修する教授者の情報は，クライアントが保存されているディレクトリと同じディレクトリにクライアントデータとして保存される．

演習選択画面はクライアント起動時に，行いたい演習を選択する．演習画面を開くためには，必ず演習を選択しなければならない．サーバに保存されている演習一覧から，学習者が演習を選択する．選択する演習は，教授者がアップロードした演習のみを選択することができる．

図 5　演習管理クライアントのホーム画面

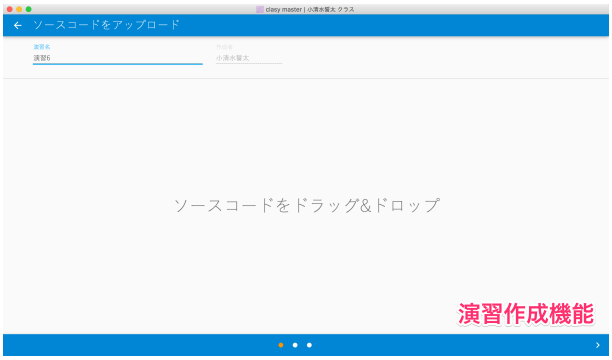

図 6　ソースコードアップロード画面

図 7　ソースコードアップロード後のリストビュー

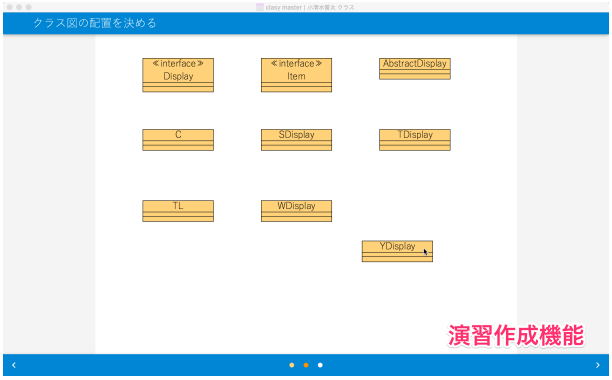

図 8　クラス配置画面

5.2　演習管理クライアント

教授者は演習管理クライアントを使って，クラス図作成演習の作成，学習者の解答の再現，分析を行う．図 5 は，

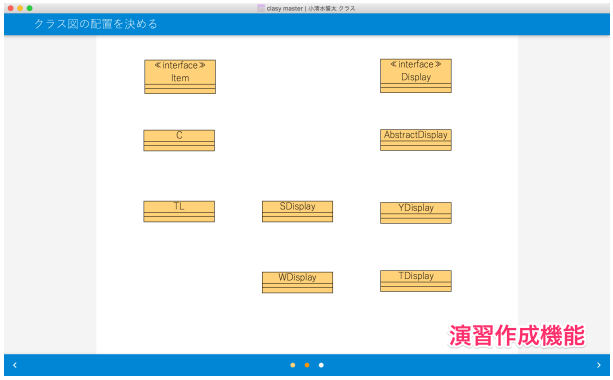

図 9　任意のクラス配置

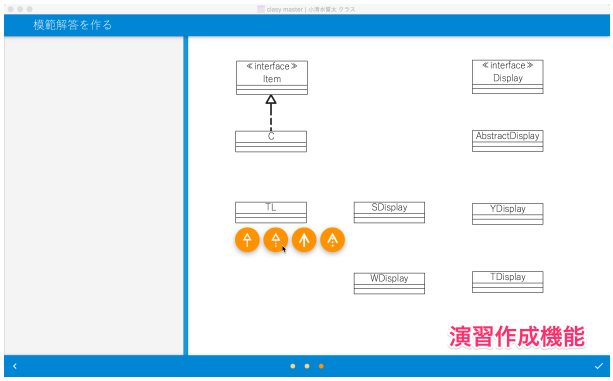

図 10　模範解答作成画面

演習管理クライアントのホーム画面である．演習作成機能で作成した演習はホーム画面にリスト形式で表示される．演習はオープンとクローズの状態を持つため，教授者は事前準備をすることや演習の締め切りをすることができる．教授者はオープンとクローズを切り替えて運用する．

ホーム画面左上にある演習追加ボタンを押すと，演習作成機能が開くことができる．また，ホーム画面右上にあるメニューボタンから再現機能や分析機能を開くことができる．

5.2.1　演習作成機能

演習作成機能を使ってクラス図作成演習を作成する (図 6)．演習を作成する手順は，ソースコードのアップロード，クラスの配置，模範解答の作成の 3 ステップとなっている．

ソースコードを読み込ませることで，リストビューが表示される (図 7)．次に自動的にクラス図情報を生成し，それぞれのクラスを配置する (図 8)．クラス図を作成するにあたり講義ではインタフェースを上にまとめることが多いため，インタフェースとクラスに分けてインタフェースの自然順序順，続いてクラスの自然順序順に 3 列に自動的に配置する．教授者は図 9 のようにクラス図のクラスを講義や演習を進めやすい任意の配置にすることもできる．

教授者は，模範解答としてクラス間の関係を記述する．記述の仕方は，クラス図作成演習と同じ方法である (図 10)．クラス間の関係の解答はソースコード解析により自動生成も可能であるが，省略可能なクラス間の関係を省略するか

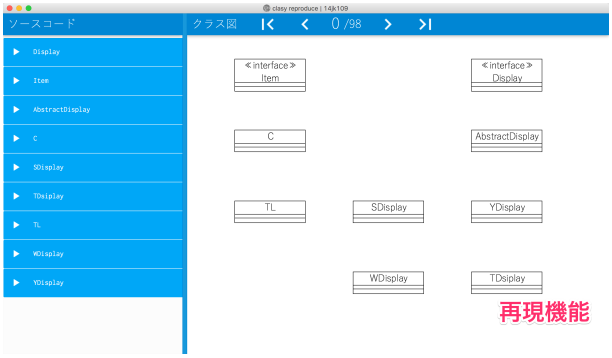

図 11　再現ツールで学習者一人を選択した画面

図 12　ソースコードを開く操作を再現した画面

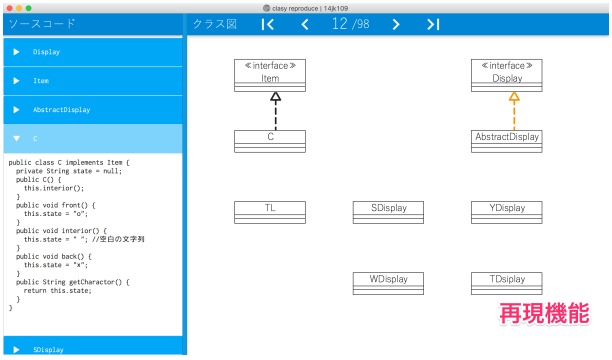

図 13　模範解答に存在する矢印と現在のステップで記述した矢印

どうかは教授者が決定するため模範解答は教授者が作成する．

模範解答作成画面の右下にあるチェックボタンを押すと，演習の作成が完了する．演習はクローズされた状態で作成される．

5.2.2　再現機能

再現機能はそれぞれの学習者がどのように演習を行なっているかを再現することができる (図 11)．画面の構成は左側にソースコード，右側にクラス図，上部のツールバーに学習者の操作ステップ数が表示されている．学習者の操作ステップを進めるたり戻したりすることで，どのような試行錯誤を行なったか確認することができる．ステップを進めたり戻したりするボタンは，最初のステップに戻るボタン，1 ステップ戻るボタン，1 ステップ進めるボタン，最後のステップに進めるボタンの 4 つがある．

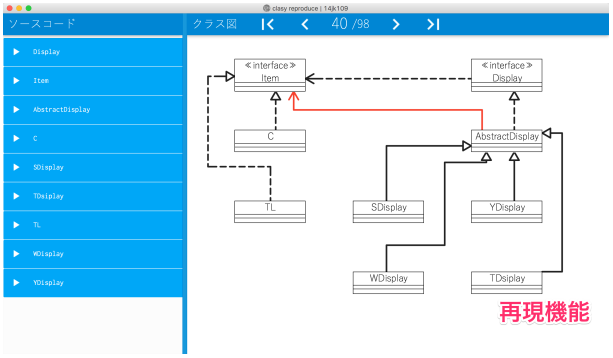

図 14　模範解答にない矢印

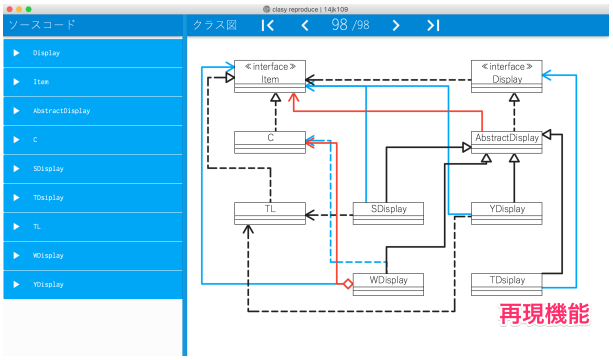

図 15　学習者が記述していない矢印

図 12 に，学習者がソースコードを展開したステップを表す画面を示す．学習者がどのソースコードを開いたか一目でわかるように，ソースコード名の部分がハイライトされている．もし教授者がこのステップを変えずに他のソースコードを確認したい場合，他のソースコードを開くことができる．このとき，学習者がこのステップで展開したソースコードのソースコード名はハイライトされ続けるが，教授者が確認したいソースコードが展開される．

再現中の矢印は，現在のステップで記述した矢印はオレンジ色，正しい矢印は黒色で表示する (図 13)．また，模範解答に存在しない矢印は次のステップから赤色で表示する (図 14)．学習者が削除をした矢印は，削除したステップで非表示になる．最後のステップの時，学習者が記述していない矢印 (模範解答に存在する矢印) を青色で表示する (図 15)．

5.2.3　分析機能

分析機能は学習者全体の演習ごとの正答率や解答の傾向を表示する．サーバに保存された演習と学習者たちの解答データを読み込む．

矢印の正誤判定は，模範解答に存在するかどうかである．模範解答の矢印の始点クラス，終点クラス，クラス間の関係，集約がすべて一致している矢印が，学習者の解答の中に存在しているかどうかで判定する．そのため，矢印の長さや曲げの位置は判定に加味されない．例えば，模範解答と一致しているが学習者が全く同じ矢印を 2 つ以上記述した場合，1 つ正解，それ以外を不正解とする．

前章までに省略可能なクラス間の関係の矢印について述べた．本システムでは省略可能なクラス間の関係を記述した場合も，模範解答に存在していなければ不正解としている．このことに関しては，省略可能な場合にも対応するため，設定可能にしている．

学習者全体で正解している矢印や，模範解答にない矢印，不足している矢印をまとめて表示するため，学習者が陥りやすい間違いを確認することができる．演習終了後，すぐに結果が反映されるので講義中にフィードバックを行うこともできる．

学習者が矢印を記述する時，ソースコードからクラス間の関係を読み取っているかを確認するため，始点クラスのソースコードを開きながら矢印を記述しているかを確認することができる．

5.3 サーバ

サーバは，それぞれのクライアントで利用するデータを管理する．管理するデータは，演習作成機能で作成した演習データ，クラス図作成演習クライアントで学習者が解答する解答データがある．

演習データは，演習名，教授者名，演習のオープン/クローズ，ソースコードリスト，クラス図のクラスリスト，模範解答，作成時刻が含まれる．クラス図のクラスリストには，演習作成時に選択したソースコードを解析しクラスリストを自動生成する．それぞれのクラスの座標を記録することで作成される．演習のオープン/クローズの切り替えは，指定の URL に演習名とオープンかクローズかどうかを送信することで行うことができる．

解答データは，演習名，学習者の学籍番号，操作履歴，提出時刻が含まれる．操作履歴はクラス図作成演習クライアントで行った操作が時系列順に並べられている．この操作履歴をもとに学習者の解答データを作成する．

データベースは JavaDB と JPA(Java persistent API) を用いた．

6. 実験

実験の目的は，演習のデータをサーバに送信できるか，また運用するにあたりデータ形式に問題がないか，解答データの傾向を確認することである．

本研究で開発したシステムを利用して，クラス図作成演習を行い，解答や操作履歴を収集した．

被験者はオブジェクト指向設計のクラス図の表記法を学習中の大学 3 年生 13 名，オブジェクト指向設計のクラス図の表記法を学習済みの大学 4 年生 13 名と大学院生 2 名，合計 28 名である．被験者には，UML 作成演習やシステムの利用方法を示してから，約 30 分間の演習を行った．

今回の演習では，9 つのソースコードを見ながらクラス間の関係の矢印を 12 本記述する．クラス間の関係を読み解き，矢印を引いていく．

表 2　実験用ソースコードのクラス間の関係

始点クラス	終点クラス	関係	集約
C	Item	実装	なし
TL	Item	実装	なし
AbstractDisplay	Display	実装	なし
SDisplay	AbstractDisplay	継承	なし
TDisplay	AbstractDisplay	継承	なし
WDisplay	AbstractDisplay	継承	なし
YDisplay	AbstractDisplay	継承	なし
TDisplay	Display	関連	なし
SDisplay	Item	関連	なし
WDisplay	Item	関連	あり
YDisplay	Item	関連	なし
Display	Item	依存	なし

9 つのソースコードは Item インタフェース，Display インタフェース，C クラス，TL クラス，AbstractDisplay クラス，SDisplay クラス，TDisplay クラス，WDisplay クラス，YDisplay クラスである．クラス間の関係を表 2 に示す．

矢印の正誤判定は，始点や終点となるクラス，クラス間の関係の種類が模範解答にある矢印と一致している矢印を正解としている．また演習の正誤判定は，模範解答の矢印のみが全てが記述されている時のみ正解としている．

収集したデータは学習者の試行錯誤を再現するためにソースコードの開閉，矢印の記述，集約の付加，削除である．ソースコードは一つしか開けない仕様になっているため，すでに開いているソースコードがある状態で別のソースコードを開こうとすると開いている状態のソースコードが閉じる操作も記録される．

本実験で得た解答データから分析を行う．分析内容は，学習者の演習結果や演習中の操作の記録などから，それぞれの解答が正解であるか，正しい矢印は何本引かれているか，正しくない矢印は何本引かれているかなどを確かめる．

7. 結果

被験者が提出したデータはサーバに人数分記録され，欠けは無かった．値の確認はサーバに送信する前のスクリーンショットとサーバに記録されたデータを比較して確認した．

模範解答通りの解答を提出できた学生はいなかった．記述された矢印の平均は 10.82 本で，そのうち正しい矢印は平均 7.54 本記述されていた．模範解答にない矢印の平均は 3.29 本，足りていない矢印の平均は 4.46 本であった．模範解答と同じ矢印をすべて記述できた学生もいたが，その学生は余分な矢印も記述していた．

表 3 のに模範解答にある矢印を記述した本数を示す．正

表 3　実験結果: 模範解答にある矢印を記述した本数

始点クラス	終点クラス	関係	集約	本数
SDisplay	AbstractDisplay	継承	なし	26
C	Item	実装	なし	26
YDisplay	AbstractDisplay	継承	なし	26
WDisplay	AbstractDisplay	継承	なし	26
TDisplay	AbstractDisplay	継承	なし	26
TL	Item	継承	なし	25
AbstractDisplay	Display	実装	なし	25
Display	Item	依存	なし	8
SDisplay	Item	関連	なし	7
YDisplay	Item	関連	なし	6
TDisplay	Display	関連	なし	5
WDisplay	Item	関連	あり	5

表 4　実験結果: 模範解答にない矢印を記述した本数

始点クラス	終点クラス	関係	集約	本数
WDisplay	C	関連	あり	7
TDisplay	Item	依存	なし	5
SDisplay	TL	関連	なし	5
YDisplay	Item	依存	なし	4
YDisplay	TL	関連	なし	4
TDisplay	Display	依存	なし	3
TDisplay	Item	関連	なし	3
SDisplay	Item	依存	なし	3
Item	WDisplay	関連	あり	3
AbstractDisplay	Item	関連	なし	3
SDisplay	AbstractDisplay	実装	なし	2
TDisplay	AbstractDisplay	実装	なし	2
Display	Item	関連	なし	2
YDisplay	Item	継承	なし	2
WDisplay	C	継承	あり	2
YDisplay	AbstractDisplay	実装	なし	2
WDisplay	AbstractDisplay	実装	なし	2
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

しい矢印の中で一番多く記述したのは始点 SDisplay から終点 AbstractDisplay の集約のない継承の矢印で 26 本あった．26 本記述されている矢印は 5 種類あり，始点 SDisplay から終点 AbstractDisplay の集約のない継承の矢印，始点 C から終点 Item の集約のない実装の矢印，始点 YDisplay から終点 AbstractDisplay の集約のない継承の矢印，始点 WDisplay から終点 AbstractDisplay の集約のない継承の矢印，始点 TDisplay から終点 AbstractDisplay の集約のない継承の矢印である．正しい矢印で学習者が記述した矢印の中で一番少なかったのは，始点 WDisplay から終点 Item の集約のある関連の矢印，始点 TDisplay から終点 Display の集約のない関連の矢印であった．

表 5　実験結果: 模範解答に存在する矢印を記述できなかった本数

始点クラス	終点クラス	関係	集約	本数
TDisplay	Display	関連	なし	23
WDisplay	Item	関連	あり	23
YDisplay	Item	関連	なし	22
SDisplay	Item	関連	なし	21
Display	Item	依存	なし	20
TL	Item	実装	なし	3
AbstractDisplay	Display	実装	なし	3
SDisplay	AbstractDisplay	継承	なし	2
C	Item	実装	なし	2
YDisplay	AbstractDisplay	継承	なし	2
WDisplay	AbstractDisplay	継承	なし	2
TDisplay	AbstractDisplay	継承	なし	2

```java
public class WDisplay extends AbstractDisplay {
  private Item[] line0 = {new C(), new C(), new C()};
  private Item[] line1 = {new C(), new C(), new C()};
  private Item[] line2 = {new C(), new C(), new C()};
  public WDisplay() {
    this.line0[1].back();
    this.line1[0].back();
    this.line1[2].back();
    this.line2[1].back();
  }
  public int height() {
    return 3;
  }
  public int width() {
    return 3;
  }
  public Item get(int x, int y) {
    if(y == 0) {
      return this.line0[x];
    } else if(y == 1) {
      return this.line1[x];
    } else if(y == 2){
      return this.line2[x];
    }
    return null;
  }

}
```

図 16　WDisplay.java のソースコード

表 4 は模範解答にない矢印を記述した一覧の一部である．一番多く間違えて記述しているのは，始点 WDisplay から終点 C の集約のある関連の矢印であった．全ての結果は付録に載せた．

始点となっているクラスを開けている状態で，模範解答にある矢印を記述した回数の合計は 174 本，始点となっているクラスを開けている状態で，模範解答にない矢印を記述した回数の合計は 167 本，始点となっているクラスを開けていない状態で，模範解答にある矢印を記述した回数の合計は 47 本，始点となっているクラスを開けていない状態で，模範解答にない矢印を記述した回数の合計は 91 本であった．

8. 考察

ソースコードのクラス名の部分に必ず記述がある継承と実装の関係は矢印の正答率が高かった．関連と依存の関係を表す矢印の正答率が低かったことから，ソースコードの決まっていない部分から関係を読み取ることは難しいと言える．

模範解答にない矢印で多かったものは，始点 WDisplay から終点 C の集約のある関連の矢印であった．WDisplay のソースコード (図 16) を見ると，インスタンス変数の型は Item でインスタンスは C を代入されているためこの間違いが多かった．しかし，この矢印は省略可能の矢印に含まれており，始点 WDisplay から終点 Item の集約のある関連の矢印で解答するのが正解としている．

始点となっているクラスを見ながら矢印を引くより，見ずに引くと間違えやすいという傾向が見られた．

9. まとめ

本研究では，学習者の演習結果や演習中の操作を記録するため，クラス図作成演習を開発した．本システムでは，学習者がおこなったクラス図作成演習の結果や演習中の操作の記録をもとに，試行錯誤を記録し教授者へ提示することができる．今後は，実際に講義で利用し教授者がフィードバックをし，さらなるシステムの改善を行う．

参考文献

[1] Object Management Group : UML 仕様書, ASCII, 2001.
[2] 増元健人，香山瑞恵，小形真平，橋本昌巳 : クラス図を用いた基礎的概念モデリングにおける誤り分析に基づく初学者向け誤り自動検出機能の開発，情報処理学会研究報告，Vol.2015-SE-187 No.15(Mar. 2015).
[3] 田中昂文，橋浦弘明，櫨山淳雄，古宮誠一 : クラス図作成演習における学習者の編集過程の細粒度分析，電子情報通信学会信学技報，KBSE2014-54,2015-03.
[4] 長谷川伸，松田承一，高野辰之，宮川治 : プログラミング入門教育を対象としたリアルタイム授業支援システム，情報処理学会論文誌，Vol. 52 No. 12 3135-3149 (Dec. 2011).
[5] Oracle : JavaFX Rich Internet Applications Development, Oracle(オンライン)，入手先 [http://javafx.com/](参照 2017-08-11).
[6] Jersey : RESTful WEB service in Java(オンライン)，入手先 [https://jersey.github.io/](参照 2017-8-11).

付　　録

表 A·1 実験結果: 模範解答にない矢印を記述した本数

始点クラス	終点クラス	関係	集約	本数
WDisplay	C	関連	あり	7
TDisplay	Item	依存	なし	5
SDisplay	TL	関連	なし	5
YDisplay	Item	依存	なし	4
YDisplay	TL	関連	なし	4
TDisplay	Display	依存	なし	3
TDisplay	Item	関連	なし	3
SDisplay	Item	依存	なし	3
Item	WDisplay	関連	あり	3
AbstractDisplay	Item	関連	なし	3
SDisplay	AbstractDisplay	実装	なし	2
TDisplay	AbstractDisplay	実装	なし	2
Display	Item	関連	なし	2
YDisplay	Item	継承	なし	2
WDisplay	C	継承	あり	2
YDisplay	AbstractDisplay	実装	なし	2
WDisplay	AbstractDisplay	実装	なし	2
TL	Item	継承	なし	2
C	Item	継承	なし	2
AbstractDisplay	Display	継承	なし	2
WDisplay	Item	依存	あり	2
Item	AbstractDisplay	関連	あり	2
WDisplay	C	依存	なし	2
YDisplay	TL	関連	あり	2
YDisplay	TL	実装	あり	1
YDisplay	TL	継承	なし	1
AbstractDisplay	Display	依存	なし	1
SDisplay	TL	関連	あり	1
TL	C	実装	なし	1
Item	YDisplay	関連	あり	1
TDisplay	Item	継承	なし	1
TL	Display	関連	なし	1
Display	TDisplay	関連	なし	1
SDisplay	Item	実装	なし	1
WDisplay	Item	継承	なし	1
C	WDisplay	関連	あり	1
Item	YDisplay	関連	なし	1
YDisplay	TL	依存	なし	1
SDisplay	Item	継承	なし	1
WDisplay	Item	依存	なし	1
SDisplay	TL	継承	なし	1
Item	SDisplay	継承	なし	1
AbstractDisplay	Item	依存	なし	1
WDisplay	Item	関連	なし	1
Item	Display	依存	なし	1
WDisplay	C	関連	なし	1
SDisplay	TL	依存	なし	1
Item	SDisplay	関連	あり	1