Vol.2022-CE-167 No.7
2022/12/3

コード - クラス図間対応理解補助システムの開発と評価

蟻坂 太士[1,a)]　久野 靖[1,b)]　中山 泰一[1,c)]

概要：プログラミングにおいてオブジェクト指向型の考え方は広く使われている．一方で，オブジェクト指向型言語は，手続き型と比べ難しいといわれる．その原因の一つとして，クラスや関係性の概念の理解が難しいことがあげられる．その難しさを軽減するため様々な可視化によるアプローチがある．その中でも，静的な関係を示すために UML(Unified Modeling Language, 統一モデリング言語) のクラス図が用いられる．コードとクラス図の対応を把握することがオブジェクト指向型言語を習得するうえで重要と考え，コードとクラス図間の対応関係の理解を補助するシステムを開発した．本システムではコードを書かせることに重点を置き，オブジェクト指向の考え方を用いてコードを書けるよう補助する動作設計を行った．大学生に対し行ったシステムの使用感に関するアンケート調査を行った結果，本システムを用いることでコードとクラス図の対応関係や自身の書いているコードの構造を意識できることが示唆された．

キーワード：オブジェクト指向，UML，クラス図

The System to Assist in Understanding the Correspondence Between Code and Class Diagrams

Arisaka Taishi[1,a)]　Yasushi Kuno[1,b)]　Yasuichi Nakayama[1,c)]

Abstract: Object-oriented programming (OOP) is commonly used. However, it is more difficult than procedural programming. One of the reasons is that it is difficult to understand the concepts of classes and relations used in OOP. There are various visualization approaches to alleviate this difficulty. UML (Unified Modeling Language) class diagrams are used to visualize classes and static relationships. Understanding the correspondence between code and class diagrams is important for learning OOP. A system was developed to assist in understanding the correspondence between code and class diagrams. This system focuses on getting students to write code, and the behavior was designed to help them write code using the object-oriented approach. In an evaluation experiment conducted on university students, we conducted a questionnaire survey on the usability of the tool. As a result, it is suggested that this system makes users aware of the structure of their own code and the correspondence between code and class diagrams.

Keywords: Object-oriented, UML, class diagram

1. はじめに

プログラミングにおいてオブジェクト指向型言語が広く使われている．オブジェクト指向型言語は，手

[1] 電気通信大学
UEC, Chofu, Tokyo 182–8585 , Japan
[a)] a2131011@edu.cc.uec.ac.jp
[b)] y-kuno@uec.ac.jp
[c)] nakayama@uec.ac.jp

続き型と比較して難しいといわれる．その原因の一つとして，クラスや関係性という概念の理解の難しさが挙げられる．特にプログラミング初学者はコードとクラスや関係性との対応を把握しておらず，メソッドを関数と混同するなどして，手続き型のような単調なコードを書いてしまう事がある [1]．富永らの研究 [7][8][9] のように，クラスや関係性を可視化することで，学習の理解を深めようという試みが行われている．静的な関係を示すために UML(Unified Modeling Language，統一モデリング言語) のクラス図が用いられる．コードとクラスや関係性の対応を把握することはオブジェクト指向型言語を習得する上で重要である．コードからクラス図を生成する機能やクラス図からコードを生成する機能を用いて，コードとの対応を把握する補助を行うツールがある．本研究では，これらの機能を組み合わせたシステムの提案と評価を行う．また，オブジェクト指向型言語である Java を対象に，クラス，インターフェース，継承関係，実装関係，関連関係を取り扱う．この理由に関しては，類似・関連研究の項で示す．

既存のエンジニアリングツールはコード生成機能によってユーザーのコードを直接編集する場合がある．しかし，ユーザーがコードとクラス図の対応を把握しきれていない初学者の場合，編集箇所の把握や編集された内容の理解が難しい．そこで，より学習に適したクラス図の可視化・生成システムを開発し評価することを目的とする．

クラス図からコードを生成する際，ユーザーのコードを直接編集せず，プレビューとして生成したコードを提示する．プレビューを確認しながらユーザー自身がコードを修正していく．これにより，編集箇所を明確化するとともにコードとクラス図の対応を意識してコード記述を行うことができ，コードとクラス図の対応理解が深まると考えた．

この流れを組み込んで設計したシステム実装し，大学生を対象にシステムの使用感に関するアンケート調査の形式で行った．用意した仕様書に従い，本システムを用いてコードとクラス図を作成してもらう形式で行った．結果として，コード記述の際にクラス図との関係を意識できた，コードとクラス図を同時に見ることができ自身のコードの構造が把握しやすかったなどの意見が得られた．結論として，本システムを用いることでコードとクラス図の対応関係や，自身の記述しているコードの構造を意識させることができたことが示唆された．一方で，初心者向けのシステムとしては提案機能の不足や UI の問題があり改良の余地があることも示唆された．

以下，第 2 章では，類似・関連研究や参考とした既存ツールについて本研究との関連を述べ，第 3 章では本研究で開発したシステムの設計を述べる．第 4 章では，システムを実際に使用した実験の内容について述べる．第 5 章では，実験の結果をまとめ，結果をもとに考察を行う．第 6 章では，本研究をまとめ，今後の展望について述べる．

2. 既存のツールや類似・関連研究

2.1 既存のツール

2.1.1 Visual Studio

Microsoft が開発･提供している統合開発環境 (IDE) である [3]．プロジェクト内のファイルからクラス図を作成する [クラスダイアグラムで表示] という機能を持つ．クラスやメソッド等の追加した場合，スケルトンコードやファイルが自動的に作成される．コードの追加はファイルを直接書き換える形で行われる．

クラス図の表現，データの入力方法，一部機能を参考にした．コードの生成が新規ファイルの作成や自動的な上書きによって行われる点が本研究で開発したシステムと異なる．

2.1.2 PlantUML

シーケンス図，ユースケース図，クラス図，オブジェクト図などを素早く作成するためのコンポーネントである [5]．構文に従ってテキストファイルを作成し，そこからクラス図を生成する機能を持つ．

クラス図の表現方法やコード表現を参考にした．実際のコードから生成できるわけではなく，専用の言語でクラス図を書いていくシステムである．

2.1.3 UML Doclet

ソースコードで提供されるフリーソフトウェア．Java ソースコードから Javadoc を使用し UML クラス図形式の HTML を埋め込んだドキュメントを生成するツール [6]．UML に基づいたクラス図を生成することができ，標準 Javadoc 形式の HTML と相互リンクができる．

クラス図からのコード生成機能などは持たない．

2.2 類似・関連研究

背景で，クラス図を可視化することでクラスの概念をより深く理解させようとする試みがされていることについて触れた．その中でも，Java やクラス図を扱ったシステムや研究を挙げる．

2.2.1 Java 演習支援システム TooDex

香川大学の富永らによって設計，開発，検証され

ている演習支援システム [7][8][9]．オブジェクト指向によるチームでの設計と実装を実践的に体験ができ，より初級的な学生を対象とした教育支援やより応用的な演習を提案している．

単調な手続き型のコードを書いてしまうことを防ごうという目的で一致する．本システムでは個人向けのシステムとして開発を行い，コード記述を行いながらオブジェクト指向型の特性を学べるシステムを目指している．

2.2.2 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

小清水らによる東京電機大学情報環境学部で行われていた手書きのクラス図作成演習をデジタル化したシステム [10]．学習者の演習結果や演習中の操作を記録する目的で作成された．実験として，ソースコードを解析しクラス図のオブジェクトに関係の矢印を追加していく演習でシステムを用いている．結果として，学生の解答や操作を記録し，正誤に関する傾向を分析することに成功している．

2.2.3 UML のクラス図と Java プログラムとの関係抽出について

宮崎大学の薮谷らによるクラス図を利用したテストによる Java プログラムの信頼性向上を目的とする，クラス図の要素と Java ソースコードとの対応関係の抽出と考察 [2]．UML クラス図で表現される関係が Java コードのどのような表現と対応するのかをまとめた．

本システムで扱うクラス，インターフェース，継承（汎化），実装（実現，洗練），関連関係の 5 種を扱う事を決定するために参考にした．Java では UML クラス図の集約関係はコンポジション関係コード上，差がないため区別することができない．また，関連関係と集約関係もコード上の違いがない．集約関係は，関連関係の中でも概念的に全体と一部の関係にある場合の関係である．しかし，コード上は区別ができないので，本システムでは関連関係にまとめることとした．また，依存関係はコードから抽出するのが困難であるため，省略した．

3. 設計

3.1 システムの概要

初学者向けのシステムとしての開発を目指すため，以下の 3 点を重視し開発する．

- 編集箇所の明確化
- クラスや関係性とコードの対応の明確化
- その対応を意識したコード記述

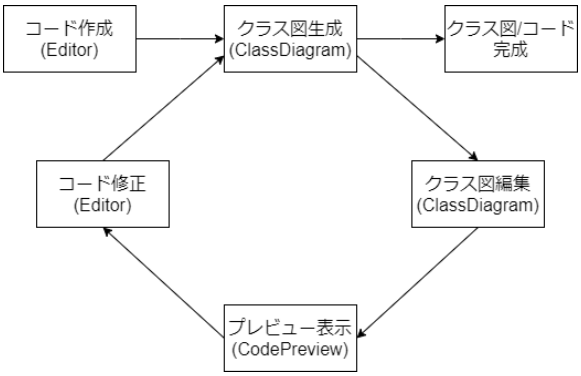

図 1 システムの流れ

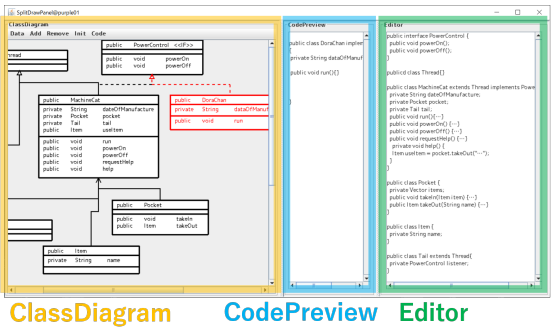

図 2 システムの全体図

クラス図の生成システムの基本的な機能として，コードを編集する Editor 部分とクラス図を編集する ClassDiagram 部分を持つシステムを設計した．クラス図をコード化した際，ユーザーのコードを直接書き換えない仕組みとして，CodePreview を設計する．これによりユーザーにクラス図を意識したコード記述をさせることができると考える．システムを使用する際の流れを図 1 に，システムの全体図を図 2 に示す．

3.2 コード作成/修正

コード作成とコード修正では，完成させる Java コードの編集を行う．Editor を用いて，ユーザーはコードを作成，修正する．

Editor に入力されたコードからクラス図を生成する．本システムがコードから抽出できるクラス図は，クラス，インターフェース，継承関係，関連関係，実装関係の 5 種である．クラスについて，class キーワードで宣言されるクラスを対象とする．インターフェースについて，interface キーワードで宣言されるインターフェースを対象とする．継承関係について，extends キーワードで宣言される継承関係を対象とする．関連関係について，new キーワードで宣言されるクラス名を型に持つ変数と，ネストされたク

ラスを対象とする．実装関係について，implements キーワードで宣言される実装関係を対象とする．

3.3 クラス図生成

クラス図生成では，Editor に入力されたコードからクラス図の生成を行う．ClassDiagram の Init メニューの init ボタンでユーザーはコードからクラス図を生成する．

クラス図として生成できるコードは類似・関連研究 2.2.3 で述べた理由から，クラス，インターフェース，継承関係，関連関係，実装関係に絞り，クラス図における表現は UML を参考に簡略化したものを扱う．

3.4 クラス図編集

クラス図編集では，クラス図の修正を行う．クラスや関係性の追加と削除機能を用いて修正を行う．クラス図編集で用いるボタンを表 1 に機能と共にまとめる．

表 1 機能一覧（ClassDiagram）

ボタン (メニュー)	機能
Set(Data)	クラスのデータ設定
DoubleClick* (Class/Interface)	クラスのデータ設定
Interface(Add)	インターフェースの追加
Class(Add)	クラスの追加
Generalization(Add)	継承関係の追加
Association(Add)	関連関係の追加
Implementation(Add)	実装関係の追加
rmConnect(Remove)	関係の削除
rmClass(Remove)	クラスの削除
init(Init)	ソースコードからクラス図を生成
GenerateCode (Code)	クラス図の変更部分からコードの概形を生成

*DoubleClick のみボタンではなく操作である．

3.5 プレビュー表示

プレビュー表示では，クラス図からスケルトンコードを生成する．ユーザーは CodePreview 部分に表示されたスケルトンコードを確認しながら自身のコードを修正していく．

CodePreview は本システムの特徴となる部分であり，このプレビュー表示を介することで設計の目標であった，編集箇所の明確化，クラスや関係性とコードの対応の明確化，対応を意識したコード記述の 3 点を実現できると考える．プレビュー表示を確認しながら自身のコードを修正することで，コードを修正箇所の明確化ができる．また，コードを修正する際，クラス図を意識しながら修正する必要が生まれる．これにより，コードとクラス図の関係をより理解することにつながると考える．

4. 実験

4.1 実験内容

システムとしての使用感に関する評価と改善を行うため，本学の学生 4 名を対象に本システムを実際に利用してもらう実験を行った．

オブジェクト指向型言語初学者向けのシステムとして開発を進めたが，今回の実験の対象者は Java 等のオブジェクト指向型言語を学習済みの学生である．教育的効果を調査する前段階として，システムの使用感や初学者向けのシステムとして成立するかを確かめる目的で実験を行った．実施にシステムを使ってもらった後，アンケート調査による評価実験を行った．

実験に際して，マニュアル，仕様説明，完成予想図 (コードとクラス図)，アンケートを作成した．マニュアルでは基本的なシステムの使用の流れを説明し，仕様説明に沿いながらシステムを使ってコードとクラス図を作成してもらった．仕様説明は継承や実装関係を 1 から 2 個含んだ簡単な構造を文章で説明したものとなっている．(例：public なインターフェース PowerControl を作る．PowerControl のメソッドは 2 つ．public な戻り値型 void の powerOn()，public な戻り値型 void の poworeOff() .) 完成予想図はユーザーが作成したコードとクラス図との比較に用いた．アンケート内容は 4.3 に後述する．

4.2 実験の流れ

実験は以下のような流れで行った．

(1) マニュアルを読んでもらう
(2) 仕様説明を読んでもらう
(3) マニュアルを読みながら，システムを利用して，仕様説明に従った Java コードとクラス図を作成してもらう
(4) 完成予想図として用意したコードやクラス図と，自身の作成したコードやクラス図を比較してもらう
(5) アンケートに回答してもらう

4.3 アンケート内容

実施したアンケートの内容を表 2 にまとめる．システム全体の使用感や，システムがコードとクラス

表 2 アンケート内容

システムについて	
1	コードとクラス図の対応をより理解することにつながったと思いますか.
2	オブジェクト指向型言語初学者向けのシステムとして役立つと思いますか.
3	全体的にシステムは使いやすかったと思いますか.
4	不便に感じた点や良かった点があればお教えください. (自由記述)
5	システムに必要と思われる機能があればお教えください. (自由記述)
6	致命的なバグ等ありましたらお教えください. （自由記述）
コードとクラス図の対応について	
7	コードとクラス図は正しく対応していたと思いますか.
8	コードの内容がクラス図で十分に表現されていたと思いますか.
9	クラス図表現で本システムに必要と思うものがあればお教えください.（自由記述）
CodePreview について	
10	CodePreview によりコードとクラス図の対応を意識できたと思いますか.
11	生成されたスケルトンコードは役に立ったと思いますか.
用意された仕様説明に含まれる要素について	
12	実施のコードで必要な要素を満たしていたと思いますか.
13	(質問 12 で 3 以下をつけた場合) どのような要素が必要かお教えください.（自由記述）
その他	
14	上記の質問以外でご意見等何かあればお教えください.（自由記述）

図の対応を理解する補助に役立つかなどを 5 段階と自由記述形式で評価してもらった．質問内容ごとに 5 つの項目，システムについて，コードとクラス図の対応について，CodePreview について，用意された仕様説明について，その他，に分けた．

5. 結果と考察

5.1 アンケート結果

実験で実施したアンケートの結果を載せる．5 段階評価は 5 に近づくにつれて高評価，1 に近づくにつれて低評価である．

質問 1，2，3，7，8，10，11，12 の結果は表 3 にまとめる．

5.1.1 システムについて

質問 1「コードとクラス図の対応をより理解することにつながったと思いますか」，質問 2「オブジェクト指向型言語初学者向けのシステムとして役立つと思いますか」，質問 3「全体的にシステムは使いやすかったと思いますか」についての結果を表 3 に示す．

質問 4「不便に感じた点や良かった点があればお教えください」では次のような回答が得られた．不便に感じた点を表 4 に，よかった点を表 5 にまとめる．

質問 5「システムに必要と思われる機能があればお教えください.」では表 6 のような回答が得られた．

質問 6「致命的なバグ等ありましたらお教えください」では表 7 ような回答が得られた

5.1.2 コードとクラス図の対応について

質問 7「コードとクラス図は正しく対応していたと

表 3 アンケート結果 (5 段階評価)

質問番号	各人数				
	1	2	3	4	5
1	0	0	0	0	4
2	0	1	1	2	0
3	0	0	2	1	1
7	0	0	1	0	3
8	0	0	1	2	1
10	0	0	1	1	2
11	0	0	2	0	2
12	0	0	0	2	2

表 4 不便に感じた点

- 削除やデータ保存のテンポが悪い
- DiagEditor が移動できない
- コード上で定義したクラスを型として使えない点
- クラス図の領域が広がると関係 (矢印の一部) が見えなくなる点
- もっとカラフルなら関係がより分かり易い

表 5 良かった点

- コードからクラス関係がしっかりとクラス図として表示される
- クラスや関係の追加削除が簡単にできる
- コードの記述とクラス図の矢印が対応しているため，使い続けるクラス図が読めるようになってくる
- クラス図を通して直観的にコードの編集が行える点
- 図とコードを同時に確認しながら書ける UI が直観的でよい
- クラスを選択した際，つながった関係がハイライトされる点

表 6 質問 5 の回答

- 変数やメソッドを static にするかどうかの設定
- 自分で実装したクラスを戻り値型に持つメソッドもコード生成されること
- クラス図の追加，削除やコードが生成したコードが変更されていないことなどを伝える UI
- Undo/Redo 機能
- クラス図を選択すると対応したコード部分も色をハイライトする機能
- プルダウン式の可視性選択画面
- 先頭に数字の名前や，予約語を名前にできないようにする機能

表 7 質問 6 の回答

- 原因不明だが，一度作成したクラスデータを再度編集すると，それまでのデータが消えてしまうことがある

思いますか」，質問 8「コードの内容がクラス図で十分表現されていたと思いますか」についての結果を表 3 に示す．

質問 9「クラス図表現で本システムに必要と思うものがあればお教えください」では表 8 ような回答が得られた．

表 8 質問 9 の回答

- Editor やライブラリで定義されたクラスや型の表示

5.1.3 CodePreview について

質問 10「CodePreview によりコードとクラス図の対応を意識できたと思いますか」，質問 11「生成されたスケルトンコードは役に立ったと思いますか」についての結果を表 3 に示す．

5.1.4 用意された仕様説明に含まれる要素について

質問 12「実施のコードで必要な要素を満たしていたと思いますか」についての結果を表 3 に示す．

質問 13「(質問 12 で 3 以下をつけた場合) どのような要素が必要かお教えください」についての結果は，評価が少なくこの質問に対する回答はなかった．

5.1.5 その他

質問 14「上記の質問以外でご意見等何かあればお教えください」についての結果は，評価が少なくこの質問に対する回答がなかった．

5.2 考察

アンケートの結果を元に考察を述べる．

5.2.1 システムについて

表 3 の質問 1 を見ると回答者数は少ないが，4 名全員が段階 5 と回答したことが分かる．本システムを用いることである程度のコードとクラス図の対応理解を補助することができると考えられる．

表 3 の質問 2 から 2 名が段階 4 を，2 名は段階 3 以下の評価をつけたことが分かる．オブジェクト指向型言語初学者向けとしては，UI や機能面から利用がまだ難しいと考えられる．

表 3 の質問 3 から 2 名が段階 4 以上，2 名が段階 3 と回答したことが分かる．このことから，使いやすさとしては改善の余地があると考えられる．

質問 4，質問 5 の回答から，データの入力操作画面の使いにくさと，入力や表示ができないデータの存在，誤ったデータを保存できてしまう事が問題と分かる．質問 2 の結果と合わせて考えると，より初学者向けにするために，より使いやすい UI，誤入力を防ぐ仕組みやどのような操作を行ったか表示する事などが必要と考えられる．

質問 6 の回答から，分かりにくい仕様は細かいものでも，実験の際にマニュアル等ですべて説明しないといけないことがわかった．

5.2.2 コードとクラス図の対応について

表 3 の質問 7 から，全員が段階 3 以上と回答し，3 名が段階 5 をつけたことが分かる．また，表 3 の質問 8 から，全員が段階 3 以上と回答し，3 名が段階 4 以上をつけたことが分かる．現在表示できるコードやクラス図の対応付けは正しく行えていると考えられる．

質問 9 の回答から，自分で定義したクラスの型名や，Vector や配列，List などを使うことの多いライブラリから呼び出される型名の必要性を感じる人がいる事が分かった．これらのコードをデータとして扱うことのできる機能を追加すべきと考えられる．

5.2.3 CodePreview について

表 3 の質問 10 を見ると全員が段階 3 以上をつけ，2 名が段階 5 をつけたことが分かる．CodePreview はコードとクラス図の対応を意識させるのに役に立つと考えられる．

表 3 の質問 11 と質問 10 を比較すると段階 4 が 1 名減っていることが分かる．このことから，スケルトンコードに表現したデータには不十分な点があると考えられる．

5.2.4 用意された仕様説明に含まれる要素について

図 3 の質問 12 から，2 名が段階 4，2 名が段階 5 と評価しているため，簡単なコードに含まれる要素に関しては用意した仕様説明でカバーすることができていると考えられる．

しかし，仕様説明は，システム評価用のために用意したものである．初学者向けの評価を行う場合は，より適切なコードや仕様を作成する必要があると考

えられる．

質問 13 と質問 14 には回答がなかった．

6. 終わりに

本研究では，オブジェクト指向型言語初学者向けのコードとクラス図の対応を理解する補助を行うシステムとしてコード-クラス図間対応理解補助システムの開発を行った．クラス図の編集個所の明確化，クラスや関係とコードの対応の明確化，対応を意識したコード記述，の 3 点を重視したシステムの設計を行った．クラス図からコードを生成する際に，スケルトンコードのプレビュー表示を見ながら自身のコードを自身の手で修正していくことで，編集個所の明確化を行うとともに，対応を意識したコード記述を行う流れを実現した．システムの使用感に関する評価実験を行った結果，本システムはコードとクラス図の対応を理解する補助を行うシステムとして一定の効果を持つことが示唆された．プレビュー表示による学習効果について，コードとクラス図の対応を理解する際に有用であることが分かった．

一方で，プレビュー表示されるスケルトンコードに関しては不足している部分が多くあることが分かった．UI や操作性，ツールとしての側面にも改善の余地があることが分かった．

本システムがコードとクラス図の対応を理解するために一定の効果を持つことが示されたが，初学者向けのシステムとしての改善点が見つかった．ユーザーに向けた提案や表示を行う機能や，誤入力・操作を防ぐための機能などが必要だと考えられる．初学者がよりオブジェクト指向の利点を生かしたコード記述を行うためにデザインパターンが役立つと考え，ユーザーへの提案機能として，デザインパターンを簡単に呼び出せる機能やハイライトする機能の実装を検討している．

また，今回の実験はシステムの使用感に関する評価と改善を目的とした．そのため，オブジェクト指向型言語をある程度扱ったことがあり，クラス図についても若干の知識がある者を対象として行った．今後は，システムを改良したのち，初学者を対象に実験を行う必要がある．

参考文献

[1] 木間塚 達, 野田 夏子：クラスとオブジェクトの扱いの意識付けに着目した Java 学習支援ツールの提案，研究報告コンピュータと教育（CE），2018-CE-146，No.3，pp1-6，(2018).
[2] 藪谷 悠介, 片山 徹郎：UML のクラス図と Java プログラムとの関係抽出について，宮崎大学工学部紀要，Vol.33， pp.375-382, (2004).
[3] Microsoft：Visual Studion，Web，<https://visualstudio.microsoft.com/ja/vs/>(参照 2022-10-24).
[4] Eclipse Foundation：Eclipse，<https://www.eclipse.org/> (参照 2022-10-24).
[5] Arnaud Roques：PlantUML の概要，PlantUML，<https://plantuml.com/ja/> (参照 2022-10-24).
[6] ObejectClab：UML Doclet，オブラブ，<http://objectclub.jp/download/uml_doc> (参照 2022-10-24).
[7] 辻健人, 富永浩之：オブジェクト指向における段階的な開発スタイルへの誘導する初級 Java 演習の支援システムの試作」情報処理学会 第 80 回全国大会，Vol.80，No.1， pp.817-818 (2018).
[8] 石井 怜央, 辻 健人, 富永 浩之：オブジェクト指向における段階的な開発とテスト駆動を誘導する応用 Java 演習の支援システム 〜 整列算法のコードの修正と拡張を題材とする演習進行の仮想実験とユーザ評価 〜，信学技報, Vol.118，No.46，pp.11-16 (2018).
[9] 石井 怜央, 辻 健人, 富永 浩之：段階的詳細化によるオブジェクト指向開発を誘導する Java 演習支援システムの出題採点機能の実装，情報処理学会 第 81 回全国大会， pp.541-542 (2019).
[10] 小清水 誓太, 高野 辰之, 小濱 隆司, 宮川 治：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, 研究報告教育学習支援情報システム（CLE），Vol.2017-CLE-23，No.16，pp.1-8 (2017).