JSiSE Research Report
vol.37,no.5(2023-1)

模写コーディングによる プログラミング学習支援システムの開発

濱田　惇也[*1]，佐々木　整[*1]，佐野　智哉[*1]
*1 拓殖大学工学部

Development of a Learning Support System for Computer Programming Education by “clone coding”

Junya Hamada[*1], Hitoshi Sasaki[*1], Tomoya Sano[*1]
*1 Faculty of Engineering, Takushoku University

小学校でのプログラミング教育が必修化されるなど，プログラミングやコーディングが一層身近なものになってきている．その反面，大学で情報工学を専攻する学生の中には，プログラミングに苦手意識を持つ学生が存在している．主にプログラミング教育は，教授者が文法の説明を行うとともに，説明した内容を含むサンプルプログラムを示し，学習者がそれを入力して動作を確認することが繰り返し行われることが多い．このような学習者が，自力でプログラミングできるほどの能力を習得しているとは限らず，授業で用いているサンプルプログラムの修正はできるが，何もない状態からコーディングすることができない学生も少なくない．このような学習者に対して，具体的な目的を提示して自発的に学習を進められるようにすることを目的とした学習環境の開発を行っており，この学習環境の特徴とプロトタイプを用いて行った評価実験の結果について報告する．

キーワード: プログラミング教育，学習支援システム，模写コーディング，写経型学習

1. はじめに

現在, 日本では IT 人材の不足が問題となっている．経済産業省の試算 (1) によると，2030 年には IT 人材の不足が最大で約 79 万人になる．不足する IT 人材の中でも，ソフトウェア開発ができる人材は，重要なものであるため，その育成が求められている．令和二年度からは，小学校でプログラミング教育が必修化されるなど，大学の情報系の学生でなくてもプログラミングを学ぶようになりつつある．しかし，プログラミングの学習後に，実際にそれを活かしてコーディングする場面がないのが現状である．授業などの学習から，更に知識や技術を伸ばしていくためには，学んだことを活かし，実際に自らの力でコーディングすることで，周辺知識の獲得や様々な経験を積むことが重要である．本来，そのコーディングは自発的に行うことが望ましいが，授業などの学習を終えてすぐには，目標とすべきものが分からず，それぞれが自発的にコーディングを行い，学習を発展させていくことは難しい．自発的にコーディングを行うには，学校の授業などの基礎的学習から移行しやすいような環境を作ることが必要である．

そこで本研究では，基礎を学んだ後に自発的にコーディングを行い，プログラミングスキルを向上できる学習支援システムの開発を行った．

2. プログラミング教育と写経型学習

2.1 プログラミング教育

現在では，小学校からプログラミング学習が行われている．小学校などの初等教育では，Scratch などのブロック型のプログラミング学習ツールが使用されている．しかし，ブロック型言語での学習だけでは，プ

ログラミングができるようになるわけではなく，実際にコーディングを行うテキスト型言語の学習も必要となってくる．その中で，ブロック型言語からテキスト型言語への移行を試みる研究[2]も行われており，初等教育などで行われるブロック型言語での学習から発展させる方法が検討されている．

2.2 写経型学習

大学などで行われているプログラミング教育は,「写経型学習」と呼ばれる学習方法が行われていることが多い[3]．この写経型学習は，教授者がお手本となるコードを学習者に与え，それを学習者が入力し実行，その動作の確認を行うことを繰り返す学習方法である．岡本[4]はこの写経型学習の長所として，“タイピングすることで１つ１つのコード（命令）を確実に確認する”，“自分で打ち込んで実行したという達成感が得られやすい”，“分からないときでも取りあえずそのまま学習を進めることができる”, という三点を挙げている．与えられたコードをそのまま写すという，一見すると学習効果が低そうな学習方法だが, 岡本ら[5]の研究で，文法の理解などで学習効果があるという結果が得られている．

このように，多くのプログラミング教育の現場で行われ，その効果が確認されている写経型学習は，SHAKYO.io[6]などの写経型学習を支援する Web サービスも公開されるなどしており，広く写経型学習のための環境が整いつつある．

2.3 写経型学習を終えた後の問題

2.2 で述べた通り，写経型学習は初学者にとって有効な学習方法ではあるが，この学習を続け与えられたコードを入力し実行するだけでは，プログラミングができるようにはならない．写経型学習で一通り文法等プログラミングの基礎を学び終えた後には，課題や目標を, 教授者などが与えてくれるわけではない. また，他の授業等の中でプログラミングが必要になる機会も頻繁にあるわけではないため，学習者自身が自発的にコーディングを行うこと，さらにそれを継続して行くことは難しい. 一方で, 例えば「ゲームが好きなので，ゲームを作りたい」というような学習者が，写経型学習で文法等を修得したとしても，すぐに市販のゲームと同等のものが作成できるようにはならない．著者らの経験上，これは学習者自身も自覚をしており，写経型学習修了後に実際にゲームを作り始める学習者はほとんどいない．そればかりか，プログラミングに対する苦手意識を持ったり，苦手意識が強くなったりする傾向がみられている．

そこで私たちは，このような写経型学習で一通り学習を終えた学習者が，自発的かつ継続的にコーディングを続けて行くことができるように，学習者に身近に感じるものをコーディングの目標に設定することで，自発的なコーディング機会を提供できると考えた．

3. 模写コーディングによる学習と問題点

3.1 模写コーディング

模写コーディング[7]は，既存の Web ページやアプリケーションの画面などの模写対象を見て，同じ見た目のものを作成する学習方法である. 美術における「模写」と比較した模写コーディングのイメージを図１に示す．

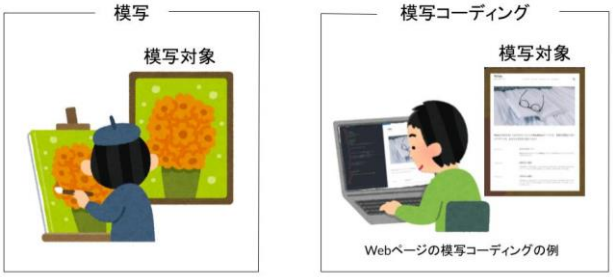


図 1 模写コーディングのイメージ

さらに，模写コーディングと写経型学習の特徴やメリット，デメリットを表１に示す．模写コーディングの大きな特徴として，ソースコードは見ずに，見た目をもとにコーディングをするという点がある．そのため，実際に画面に表示されているものが，どの様に作られているのか，自分で何を書かなくてはいけないかを考え，分からない箇所を自分で調べ実装することを繰り返す必要がある．このような作業を行うため，写経型学習と比べて学習効果が高く，コーディングを

表 1 写経型学習と模写コーディングの違い

	特徴	メリット	デメリット
写経型学習	● サンプルコードを見て書き写す ● 基礎的な文法の学習で多く用いられる	● １つ１つのコードを確実に確認できる ● 自分で行った達成感が得られる ● 分からなくても取りあえず学習を進められる	● 理解しないままコーディングができてしまう ● 書き写しているだけでは学習効果が低い
模写コーディング	● コードを見ずに画面に表示された対象を見てコーディングする ● 見た目が同じようになればコードが同じにならなくてもよい	● 身近なものを模写対象にできるため目標を立てやすい ● 普段使うものを作ることができた時に達成感を得られる ● 写経型学習と比べ学習効果が高い	● 写経型学習を行った人でも難易度が高い ● 対象のコードが見られるわけではないため分からない箇所を解決しにくい ● コーディングとは関係のない作業がある

行うことで様々な経験を積み，さらなる知識やスキルを培うことができる．また，画面に表示されているものと同じ見た目になればよいので，模写の対象とコードが全く同じにならなくてもよいという特徴もある．だが，実際に書いたコードと，対象のコードを比較する作業は，より良い書き方や知識を身に付けることにつながるので重要である．

模写コーディングのメリットとしては，先ほど述べた学習効果が高い点の他に，普段から使うような身近のものを模写の対象にできる点がある．模写コーディングでは，既存の Web ページやアプリケーションの画面を模写対象にすることが可能であるため，課題を与えられなくても学習者自身で目標を立てやすい．さらに，普段使うような Web ページやアプリケーションの画面を完成させることができた時には，サンプルプログラムを入力する写経型学習とは違う達成感を得られるというメリットもある．

本研究では，私たちが普段から使用するようなページを目標にすることができる「Web ページの模写コーディング」に着目し，開発するシステムで扱うことにした．

3.2 Web ページの模写コーディング

Web ページの模写コーディングでは，Web ページのソースコードを取得できることもあり，ソースコードと書いているコードを比較することが可能である．それにより，学習者がコーディングで行き詰まった部分の問題解決につながると考えた．これは，身近な Web ページを目標にすることができるという点も含めて，学習をする上で重要である，学習者のモチベーションの維持につながると考えた．また，Web ページの模写コーディングは，Web ページ制作の仕事の状況と似ているという特徴がある．Web ページ制作では，デザインカンプと呼ばれる見本を見ながらコーディングを行うが，Web ページの模写コーディングでも既存の Web ページを見本とし，その見本を見ながらコーディングをするため類似する点があると言える．さらに，模写コーディングで作成した Web ページを，ポートフォリオとして使用している例があるため，模写コーディングを行うことは，将来的にも役立つのではないかと考えている．

3.3 模写コーディングによる学習の問題点

私たちは，模写コーディングを行う上での問題点は二点あると考えている．一点目は，コーディングとは関係ない作業が多いという点である．模写コーディングを行う際には，模写コーディングを行うページを決めた後に，ページで使用されているテキストや画像などの素材の準備が必要となる．この作業があるため，学習者は模写コーディングを行うページを決めてから，すぐにコーディングを始められない．

二点目は，一からコーディングしなくてはいけないため，学習者によっては難易度が高すぎてしまうという点である．写経型学習によって，文法の基礎的学習

を行ってきた学習者でも，一からコーディングしプログラムを完成させることは非常に難易度が高い．これまで行ってきた写経型学習では，ソースコードが与えられ，全体像から細かい部分まで分かった状況でコーディングしてきた．しかし，模写コーディングになったとたん，ソースコードではなく完成した状態の見た目のみが与えられるため，コードの全体像から細かい部分まで学習者が一から考えコーディングする必要がある．そのため，いきなり自発的にコーディングし学習を進めていくことは難しい．

そこで，一点目の問題点に対しては，コーディングとは関係ない作業を自動化することで，学習者がすぐにコーディングを始められ，模写コーディングに集中できるのではないかと考えた．また，二点目の問題点に対しては，学習者が自発的にコーディングを行うことが重要である．そのため，学習者が一からすべてコーディングするだけでなく，ヒントやコードの大まかな全体像を与えることで，部分的なコーディングも可能になり，これまで行ってきた写経型学習と模写コーディングとのギャップを減らすことができる．これにより，学習者が自発的にコーディングに取り掛かりやすくなると考えている．

本研究では，模写コーディングによる学習の二点の問題点に対して，Web ページで使用されているファイルの一括取得と，取得したプログラムを学習者に応じて加工することを可能にした，模写コーディングによる学習を支援するシステムの開発を行った [8] [9]．

4. 開発した学習支援システムの使用例

本研究で開発したシステムは，「模写コーディング」による学習を支援するものである．また，本システムでは，大学の授業などで写経型学習を行い，文法の基礎知識を習得している人を学習対象としている．また本システムでは，HTML，CSS，JavaScript のファイルを加工し提供するが，本研究では，変更箇所が分かりやすく学習者のモチベーション維持につながり，文法が HTML，JavaScript に比べて複雑ではないという点でコーディングに取り掛かりやすいと考え，CSS ファイルの加工，提供から着手した．

本システムの使用例を紹介する．まず学習者は，ログイン画面からシステムにログインする．

4.1 模写用素材の取得と加工

システムにログインした後に表示される図 2 のページでは，学習者が実際に模写を行う既存の Web ページを選択する．

図 2 模写用素材の取得画面

模写を行う Web ページの選択は，図 2 のページ上部にあるテキストボックスに，作成したい Web ページの URL を入力することで行う．また，本システムはあらかじめ模写サイト [10] [11] [12] のページを登録してあるので，学習者は URL を直接入力するだけでなく，それら登録されたページを選択することもできる．

模写を行う Web ページを決定すると，システムは自動的にその Web ページで使用されている各種のファイルを，図 3 のように Web ページで使用されているディレクトリ構造を保ったまま取得する．ディレクトリ構造を保ったまま取得することで，学習者は，ファイルの場所などを変える必要がなく，すぐにコーディングに取り掛かることができる．

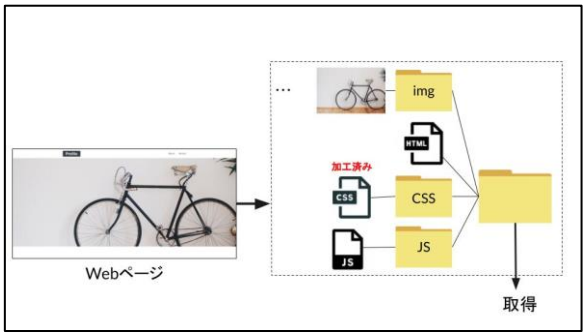
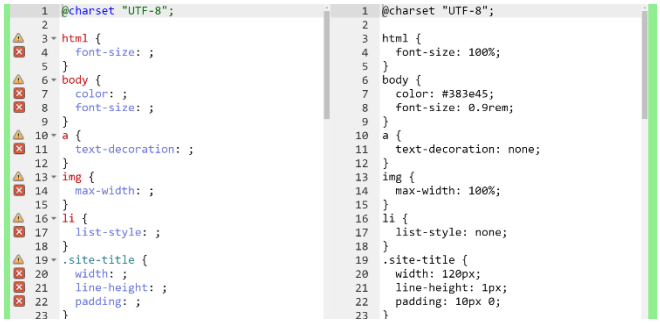


図 3 ファイル取得のイメージ

また，取得した Web ページ（CSS ファイルなど）は，部分的に削除を行って学習者に提示することもできる．図 2 のページ右下のチェックボックスで，加工内容を選択し，「白紙状態」，「セレクタの削除」，「プロパティの値の削除」，「選択したプロパティの削除」を自動で行う．これによって，学習者は全く何もない状態から HTML や CSS ファイルを作成するのではなく，部分的に欠落したファイルを元に，欠落した部分を埋めていくという，写経と模写の中間的な学習からスタートすることもできる．このファイルの加工例を図 4 に示す．



```
@charset "UTF-8";

html {
  font-size: ;
}
body {
  color: ;
  font-size: ;
}
a {
  text-decoration: ;
}
img {
  max-width: ;
}
li {
  list-style: ;
}
.site-title {
  width: ;
  line-height: ;
  padding: ;
}
```

```
@charset "UTF-8";

html {
  font-size: 100%;
}
body {
  color: #383e45;
  font-size: 0.9rem;
}
a {
  text-decoration: none;
}
img {
  max-width: 100%;
}
li {
  list-style: none;
}
.site-title {
  width: 120px;
  line-height: 1px;
  padding: 10px 0;
}
```

図 4 ファイル加工例（プロパティ値削除）

図 4 の右側は模写対象の Web ページ[11]で使用されていた CSS そのものであり，左側は本システムによって一部のセレクタが削除されたものである．コーディング時には，基本的にオリジナルのコードではなく，Web ページの画面を見ながらコーディングすることを想定しているが，ここではコードの比較がしやすいように，左右に並べて表示している．

4.2 エディタへのコーディング

学習者がコーディングを始める際には，図 3 に表示されている「エディタを表示」ボタンを押す．その後 4.1 のように素材を取得した Web ページを，図 5 のようにプルダウンメニューから選択する．

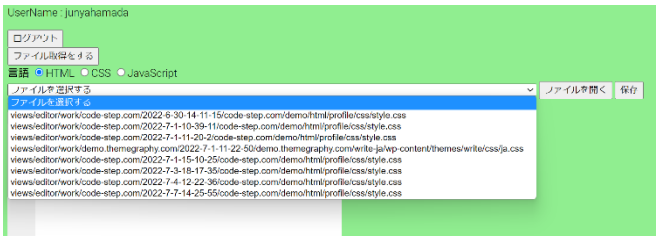


図 5 コーディングを行うファイルの選択

選択後，「ファイルを開く」ボタンを押すと，エディタにコーディングを行う CSS ファイルが表示される．模写コーディングは，既存のものをお手本としてコーディングを行うので，本システムでは図 6 のように，エディタと模写対象となる Web ページのスクリーンショットを，コーディングしながら見やすいように横に並べることを可能にした．

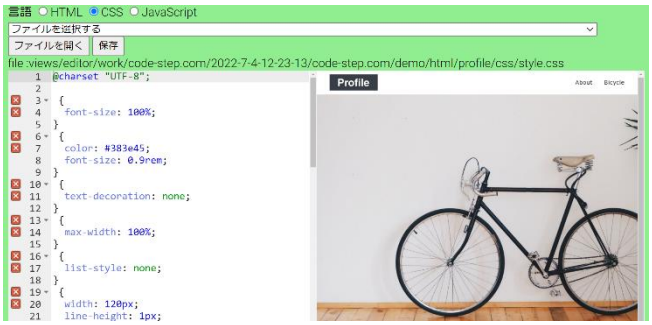


図 6 システムのエディタ画面

また，CSS の記述には HTML のタグやクラス名などが必要となるため，HTML のソースコードも参照可能とした．また，模写コーディングではソースコードを見ずにコーディングを行うが，コーディングで行き詰まり，完成させることができない場合のために，CSS のソースコードも参照可能にし，写経型学習から始めることも可能である．それにより，学習者のモチベーションの維持につながると考えている．

また，学習者のコーディング中は，コードが変更された時に自動で保存が行われるようになっており，システムとしての利便性も高めている．

4.3 ページの比較

Web ページの模写コーディングでは，作成しているページと既存の Web ページの見た目を比較しながら

完成させていくが，本システムでは図 7 のように，作成しているページと既存の Web ページを横に並べることで，ページの比較を行いやすくしている．

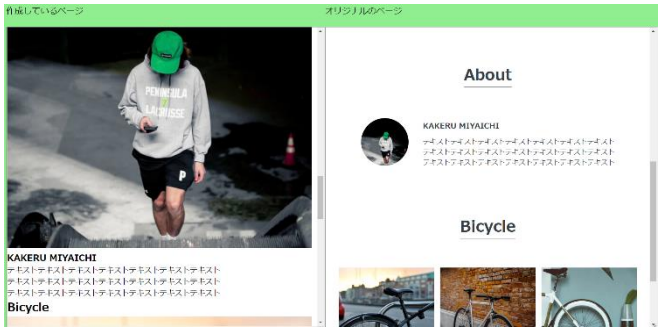


図 7 作成中のページと既存のページの比較

学習者は，コーディングとページの比較を繰り返すことで，模写対象である既存の Web ページの見た目に近づけていく．

5. 評価実験

5.1 評価実験の概要

2022 年 7 月に，拓殖大学工学部情報工学科の四年生 6 名の協力を得て，評価実験を行った．この 6 名全員は，二年次に情報メディア実験という Web ページ作成の基礎を学ぶ 1 回 180 分の授業を 15 回受講している．システムの説明を行いながらシステムに用意してある Web ページの模写を，プロパティの値を削除した CSS ファイルに限定して 30 分間コーディングし，その感想を収集した．模写対象の CSS ファイルは 151 行あり， そのうち 55 箇所のプロパティの設定がなされている．

しかし，被験者のうち 2 名は履修後 1 年半以上経過していることもあり，CSS をはじめ Web ページ作成の理解が十分ではなく，写経型学習から行うべき状態であった．そのため，本システムで模写コーディングを行う学習者の対象でないと判断し，システムを利用した感想のみヒアリングをすることとした．

5.2 実験結果

この本実験で，被験者がどの程度プロパティの値を入力できたのか，3 名の被験者の結果を表 2 にまとめた．

表 2 被験者が入力したプロパティ値の数

	入力した プロパティの値
被験者 A	8 / 55
被験者 B	1 / 55
被験者 C	4 / 55

実験で用いた CSS ファイルには，授業内で扱わなかったプロパティも多かったため，コーディングが思うように進まなかったが，最もプロパティ値を入力することができた被験者 A は他の 2 人の被験者と比べ，知らないことを調べ，すぐに解決していた．本システムでは，オリジナルのコードを参照し写経型学習をすることも可能だが，被験者はコードを参照せず，知らないことはインターネットを使い調べていた．

また，本実験で被験者から，実際にシステムを使用して出た意見を表 3 にまとめた．

表 3 被験者からの意見

肯定的な意見	否定的な意見
● 部分的なコーディングから始められる（4 名） ● CSS から学習を始めることでモチベーションの維持につながるのでは	● ページの比較がしにくい（2 名） ● 学習履歴を管理できると良い ● ソースコードを参照できても良いのか

5.3 評価と考察

得られた意見の中で良かった点としては，ファイルを白紙の状態からコーディングするよりも，部分的なコーディングから始められたほうが学習しようと思う，といったような本研究の目的に近い意見を，実際にコーディングを行った 4 名の被験者から得た点である．また，システムに対してではないが，CSS から学習することで，学習者のモチベーションの維持につながるのではないかという意見を得られ，CSS ファイルの加工，提供から始めるという本研究のアプローチとして

良かったと感じた．また，被験者 A は，他の被験者よりコーディングの経験が豊富だったため，分からない部分を調べ，すぐにコーディングできており，経験を積むことの重要性を再認識することができた．

改善点としては，作成したページと既存のページとの比較がしにくいという意見を，実際にコーディングを行った被験者の半数から出ており，早急に改善する必要があると感じた．さらに，これまでにどのようなページを学習したのかなどの学習履歴を管理できるとよいという意見も出た．学習履歴に関しては，実際に加工して学習者に提供された CSS ファイルのパスは確認することができるが，完成させることができたのかなどの情報は管理することができていないため，継続して開発していく必要があると感じた．また被験者からは，ソースコードを簡単に参照することができても良いのかというコメントが出たが，本システムではソースコードを参照することができる設計になっているため問題はない．学習の中でわからない部分をわからないままにしておくことは良くないため，ソースコードと学習者が書いているコードとの比較を可能にし，さらに学習者によっては写経型学習から始められるようになっている．

一方で，本実験でコーディングを行うことができなかった 2 名からは，白紙の状態のファイルではなく加工されたファイルであれば，実際にコーディングを行ってみたいという意見が出され，本システムの学習者の対象外ではあるが，良い感触を得ることができた．

実験時間に関しては，模写コーディングを紹介している動画 [13] で行われていた CSS ファイルの編集時間を参考にしたが，被験者は経験が浅かったためセレクタ，一箇所ほどしかコーディングすることができなかった．想定内の結果ではあったが，継続的に本システムを利用することで得られる意見もあるのではないかと感じた．

評価実験を通して，本システムで提供したファイルに対しての評価は高かったが，学習支援システムとしての改善点が多く見られたため，今後の課題としていきたい．

6. 今後の課題

今後の課題として挙げられるのは，まずは本研究で取り掛かることができなかった，HTML，JavaScript ファイルの加工，提供である．本システムは，Web ページの模写コーディングを支援するため，CSS のみのファイル加工，提供では不十分である．HTML，JavaScript の学習も可能にすることで，将来的に学習者が一から Web ページ作成を行うことができる環境づくりを進める．また，他言語のファイル加工を行うことで，Web ページに限らずソースコードを取得でき，画面に表示されるものであれば応用可能であるため，本システムの価値が高まるのではないかと考えている．

次に作成したページと既存のページの比較についてである．評価実験では，実際にコーディングを行った被験者の半数から，比較がしにくいという意見があった．この点に関しても優先して取り組むべきだと考えている．現段階では実装できていないが，作成したページと模写の対象であるページとの見た目の差分を検出し，学習者に結果を提示することを考えている．ページが完成したかを決めるのはあくまで学習者ではあるが，検出された差分で間違いを指摘することはシステムとして必要なことだと考えているため，ページが比較しにくいという問題解決のためにも，差分検出を実装していく．

他にも，評価実験で得た学習履歴の管理や，システムとしての使い勝手などの意見に関しても今後の課題として挙げられる．評価実験に関しても，実験方法や被験者の人数に関しての指摘があったため，引き続き行っていく必要がある．

7. おわりに

本稿では，自発的にコーディングを行う，模写コーディングによる学習を支援するシステムの開発と，現在大学などで行われている写経型学習，本研究で扱った模写コーディングについて述べた．本研究では，写経型学習を行ってきた学習者が模写コーディングに移行しやすくするための，部分的なコーディングから始められるようなファイルの生成，学習者がすぐにコー

ディングを始められるようにするための，模写コーディングに用いる素材の自動取得を行う機能の実装をすることができた．また，評価実験を行ってシステムの評価と考察を行い，プログラミングの学習支援システムとしての評価と課題が得られた．

参 考 文 献

(1) 経済産業省：“IT 人材需給に関する調査”，https://www.meti.go.jp/policy/it_policy/jinzai/houkokusyo.pdf（2022 年 1 月 20 日確認）

(2) 市川大祐：“プログラミング導入教育におけるブロック型言語からテキスト型言語への移行の試み”，日本教育工学会第 34 回全国大会，pp.219-220，2018

(3) 岡本雅子，喜多一：“プログラミングの「写経型学習」における初学者のつまずきの類型化とその考察”，滋賀大学教育学部付属教育実践総合センター紀要，第 22 巻，pp.49-53，2014

(4) 岡本雅子：“写経プログラミングをめぐる終わりそうもない論争”，情報処理学会，情報処理 2018 年 1 月号，p.81，https://www.ipsj.or.jp/magazine/9faeag0000005al5-att/5901peta_1.pdf，（2022 年 11 月 8 日確認）

(5) 岡本雅子，村上正行，吉川直人，喜多一：“プログラミングの写経型学習過程を対象にしたつまずきの分析とテキスト教材の改善：作業の自律的遂行と作業を介した理解のための支援と工夫”，京都大学高等教育研究，第 19 号，pp.47-57，2013

(6) SHAKYO.io，https://shakyo.io/，（2022 年 11 月 8 日確認）

(7) 服部洋介：“模写コーディングでスキルアップしよう！”，ARCHETYP blog ，2020-02-06，https://www.archetyp.jp/blog/replication-coding/，(2022 年 6 月 27 日確認)

(8) Hitoshi Sasaki and Junya Hamada：“An Attempt to Improve the Coding Skill through the Clone Coding”，The 6th International Conference on E-Society, E-Education and E-TechnologyHitoshi Sasaki and Junya Hamada：“Development of a coding skill training system using “clone coding””，World Conference on Computers in Education 2022

(9) “【初心者向け】おすすめ模写サイト 5 選”，Develop Myself，2022-05-08，https://yaponkalog.com/replication-website-beginner/#toc4，（2022 年 7 月 25 日確認）

(10) “【難易度別】初心者にもオススメな模写コーディングにうってつけな無料サイト 9 選”，ぴょんなことから for programmer，2022-06-30，https://maipyon.net/programming/copy-coding-site/，(2022 年 7 月 25 日確認)

(11) “【模写コーディング】おすすめ練習サイト【入門編～上級編】”，Codestep，2022-07-04，https://code-step.com/coding-recommend/，(2022 年 7 月 25 日確認)

(12) Web の神様：“【超入門】初心者必見！模写コーディングをやってみた　実践編【HTML・CSS コーデイング】”，YouTube，2020-07-04，https://youtu.be/RBLkXAG4QTM，（2022 年 6 月 27 日確認）