JSiSE Research Report
vol.37,no.6(2023-3)

プログラミング演習における学習者の進捗状況の把握を目的とした評価指標の提案

田中 空来[*1], 香山 瑞恵[*2], 新村 正明[*2], 舘 伸幸[*2]
*1 信州大学大学院, *2 信州大学

A Research on Assessment Metrics for Keeping track of Learners' Progress in Programming Exercises

Sora Tanaka[*1], Mizue Kayama[*2], Masaaki Niimura[*2], Nobuyuki Tachi[*2]
*1 Graduate School of Science & Technology, Shinshu University, *2 Shinshu University

本研究の目的は初学者向けのプログラミング演習で，学習指導が必要である学習者を同定し，指導者の学習指導をサポートすることである．そのため，模範解答との類似度と相違度に基づく評価指標を提案する．本稿では，まず，プログラムに対応する抽象構文木（AST）から算出した類似度と相違度の計算方法を示す．次に，これらの指標と一般的なプログラムメトリックスとの関係を整理する．この結果に基づき，提案指標の妥当性を考察する．また，学習指導が必要な学習者の進捗状況について，どのようなパターンが存在するのかを考察する．最後に，進捗状況可視化システムの概要の説明と可視化シミュレーションの評価を行う．

キーワード: プログラミング演習，進捗状況，類似度，相違度，抽象構文木

1. はじめに

高等教育機関での専門情報教育において，プログラミングは基本技術であり，多くの大学でプログラミング演習が行われている．プログラミング演習は，開始時に指導者（講師，Teaching Assistant）が学習者に課題を与え，学習者は，わからない点を指導者に質問しつつ，コーディングを行うという授業形式が一般的である[1]．また，コーディングの終了タイミング，もしくは，課題の提出時間になると，指導者にソースコードを提出し，チェックを受ける．このように，プログラミング演習では，学習者が個別に課題に取り組むため，学習者の能力差によって進捗状況に大きな差が生まれることが多い[2]．実際に，講義や演習の内容がわからない状態でも質問をせずに，手が止まってしまう学習者が多く存在している[3]．そのため，指導者は学習者からの質問に対応するだけでなく，学習指導が必要である学習者を把握し，状況に応じた指導を行うことが重要である．特に，初学者向けのプログラミング演習では，理解不足などの状態を早期発見することは非常に重要である．しかし，プログラミング演習では，多数の学習者に対して，少数の指導者が対応するため，学習者全員のコーディング状況を把握することは困難である．そこで，演習中の学習者のコーディング過程を分析し，進捗状況を可視化するツールが提案されている[4-6]．これらの研究では，コーディング行数，コーディング時間，エディターの操作数，エラーメッセージの内容，実行結果や実行ログなどの学習者の行動に基づく情報を利用している場合が多い．しかし，ソースコードの内容や構造などの質的な評価に基づく指標を考慮している研究は少ない．これに対して，本研究では，学習者が作成したソースコード（以下，学習者コード）の質的な評価指標の開発を試みてきた[7]．

本研究の目的は，大学で行われている初学者向けのプログラミング演習で，学習指導が必要である学習者を同定することで，指導者の学習指導をサポートすることにある．そこで，本稿では，ソースコードの質的な評価指標となる，類似度と相違度を提案する．これらの指標と，一般的なプログラムメトリックスである，循環的複雑度とソースコードの行数を用いたプログラムの評価を行い，学習指導が必要な状態について考察した．また，指導者と学習者に対して進捗状況を表示するツールのプロトタイプを開発した．

本研究におけるリサーチクエスチョンは以下の 3 点である．

1) 学習指導が必要な学習者の進捗状況とはどのような状態であるか
2) 学習指導が必要な学習者の進捗状況を特定のパターンに分けることができるのか
3) 学習指導が必要な学習者の進捗状況を演習の過程で検出できるのか

2. 学習者の進捗状況について

リサーチクエスチョン 1)「学習指導が必要な学習者の進捗状況とはどのような状態であるか」について，本研究では，次のように仮定する．プログラミング演習において，学習者に対する学習指導が必要でない状態とは，課題内容やアルゴリズムを理解しており，自力でコーディングすることができ，かつ，学習者の解答内容が模範解答に近い場合であるとする．

一方，学習指導が必要な状態として，主に次の 2 点が考えられる．1 点目は，課題内容やアルゴリズムが理解できず，手が止まっている状態である．2 点目は，課題内容やアルゴリズムはある程度理解しており，コーディングもしているが，模範解答には近づいていない（解答の方向性が間違っている）状態である．ここで，1 点目の状態については，コーディング行数や実行ログなどの学習者の行動に基づく評価指標を用いることで，学習指導が必要な学習者を検出できる可能性が高い．2 点目の状態については，コーディング行数や実行ログのみでは，学習指導の必要の有無を判断することは困難である．そこで，本稿では，ソースコードの構造を抽象構文木（図 1）を用いて比較することで，ソースコードの質的な評価指標を具体化する．これらの指標を用いることで，2 点目の状態の学習者の同定を試みる．

3. 提案する評価指標について

提案する評価指標は「類似度」と「相違度」である．類似度は「学習者コードと模範コード（後述）が構造的に一致するノード（後述）の割合」，相違度は「学習者コードの中で模範コードに含まれないノードの割合」とした．また，相違度は 1 から類似度を引いた値では

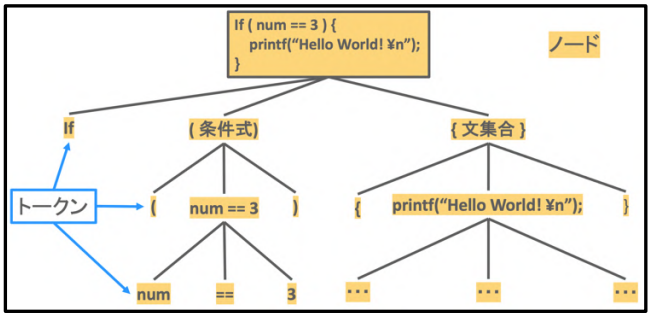

図 1: 抽象構文木の例

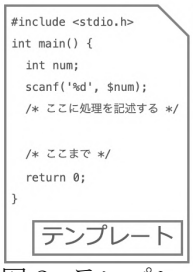

図 2: テンプレート(t)の例

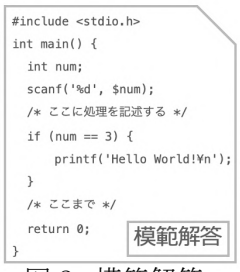

図 3: 模範解答(a)の例

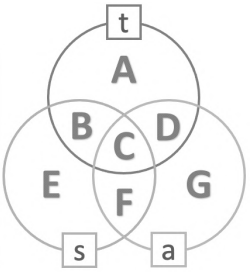

図 4: 各コードの関係性

ない．以下，3.1 でこれらの評価指標の設計概要を示し，3.2 で計算方法を示す．

3.1 設計概要

本研究では，抽象構文木（Abstract Syntax Tree，以下，AST）のノード単位でソースコードを分析する．ここでは，ソースコードの解析ツールとして Gum Tree[8,9]を利用する．Gum Tree は 2 つのソースコードを入力として与えることでそれぞれの AST を生成し，AST 間でノードの一致，挿入，削除，更新，移動を検出するが，ソースコードの構造が破綻していると，AST が生成できない場合がある．本研究では，このうち一致と挿入を使用している．Gum Tree に与えるソースコードは，テンプレート（以下，t），模範解答（a），スナップショット（s）の 3 種とする．t と a は指導者があらかじめ用意しておく．学習者は t を用いてコーディングを始める．図 2 の t には学習者が処理を記述する範囲がコメントで示されており，学習者はその中のみ編集を行うことが期待される．図 3 の a は t に処理が追記された模範解答例である．s は編集途中の学習者コードであり，編集が加えられた場合に一定間隔で自動保存される．

3.2 計算方法

図 4 に，t，a，s に含まれるノードの関係を示す．A

～G を Gum Tree で解析されたノードの部分集合とした場合，それぞれ，t＝{A，B，C，D}，a＝{C，D，F，G}，s＝{B，C，E，F}となる．このうち，E は s にのみ存在するノードであり，F は s と a にのみ存在するノードである．t には，基本的に A，B のノードは存在しないが，一部の課題で t を編集可能であるため，t の要素に含んでいる．類似度と相違度の計算式は図 4 の要素を用いて次の(1)式，(2)式のように表される．両指標の値域は，[0, 1]である．

$$\text{類似度} = \frac{\{A, F\}}{\{A, B, F, G\}} \quad (1)$$

$$\text{相違度} = \frac{\{D, E\}}{\{A, D, E, F\}} \quad (2)$$

4. プログラムメトリックスの概要

一般的なプログラムメトリックスについて説明する．

4.1 Jaccard 係数

Jaccard 係数は集合の類似度を計算するメトリクスである．2 つの集合の和集合に対する積集合の割合で計算され，値域は[0, 1]である．図 4 の a と s の Jaccard 係数は次の(3)式のように表される．

$$J(a, s) = \frac{|a \cap s|}{|a \cup s|} = \frac{\{C, F\}}{\{B, C, D, E, F, G\}} \quad (3)$$

また，1 から Jaccard 係数を引いた値である Jaccard 距離は，集合の非類似性を表す．値域は[0, 1]である．a と s の Jaccard 距離は次の(4)式のように表される．

$$dj(a, s) = 1 - J(a, s) = \frac{|a \cup s| - |a \cap s|}{|a \cup s|} \quad (4)$$

4.2 循環的複雑度

循環的複雑度（Cyclomatic Complexity，以下，CC）は，プログラムの複雑度を測るメトリックスであり，ソースコード内の線形独立な経路の数のことである．例えば，if 文や for 文などの分岐のないソースコードの場合，その経路は 1 つしかないため，CC は 1 である．また，ソースコードに 1 つの if 文が含まれる場合は，if 文が真となる場合と偽となる場合の 2 つの経路が存在するため，CC は 2 となる．

4.3 ソースコードの行数

ソースコードの行数（Lines of code, LOC，以下，行数）は，ソフトウェアの規模を表すメトリックスのひとつである．テキストファイルとしての行数を物理 LOC（physical LOC）という．本研究では，空行やコメント行を除いた行数である論理 LOC（logical LOC）を使用する．

5. 提案指標の妥当性の検証

評価指標と Jaccard 係数, CC, 行数との関係を示す．

5.1 実験条件

2021 年度に実施された信州大学工学部電子情報システム工学科 2 年生向けの C 言語プログラミング演習を対象とした．70 分間の期末試験に出題された 5 題のうち 3 課題（以下，解析課題問 2，問 3-1，問 3-2）を用いた．解析課題は，テンプレートに元からあるコードの修正・削除は禁止されており，コードの追加のみ行うようになっている．解析課題を解いた学習者の学習者コードのうち AST 生成が成功したコードをスナップショットとした．表 1 に各解析課題の学習者コード数とスナップショット数を示す．なお，これらのスナップショットは学習者のコーディングが止まると 2 秒後に自動保存される学習者コードである．

5.2 結果

学習者ごとにスナップショットの枚数が異なる．解析における個々の学習者の影響を等しくするために，期末試験時間内にスナップショットが 20 枚以上保存されている学習者を解析対象とした．学習者が解析課題に取り組んだ時間をそれぞれ 10 等分し，学習者コードが保存された時間が等分時間に近いスナップショット 10 枚を個人データとした．表 2 に各課題で解析対

表 1: 学習者コード数とスナップショット数

解析課題	学習者コード	スナップショット
問 2	20,622	7,705
問 3-1	17,463	10130
問 3-2	5,975	4,151

表 2: 学習者とコンパイルの成否数

解析課題	学習者数	s_ok	s_ng
問 2	105	653	397
問 3-1	120	702	498
問 3-2	51	287	223

象とした学習者数と，スナップショットにおけるコンパイル成否数を示す．表中，s_ok はコンパイルが成功したスナップショット数，s_ng はコンパイルが失敗したスナップショット数である．それらのスナップショットに対して 5 つの指標を計算し，相互の相関を求めた．問 2 の Jaccard 係数，類似度，相違度，CC，行数の相関を表 3 に示す．表 3 の薄灰色部分は s の相関係数である. 黒色部分はコンパイルが成功した s_ok と失敗した s_ng との相関係数である．

問 2 の結果（表 3）より，Jaccard 係数と類似度は強い正の相関があり， Jaccard 係数と相違度，そして類似度と相違度ではほとんど相関がなかった. 一方で，問 3-1 と問 3-2 では Jaccard 係数と類似度，そして類似度と相違度ではかなり正の相関があり，Jaccard 係数と相違度ではやや負の相関があった．また，問 2 の類似度と CC，そして類似度と行数では強い正の相関があった．これらは，問 3-1 と問 3-2 でも同様の結果となった．また，問 2 では，いずれの指標でもコンパイルの正否による影響はないが, 問 3-1 と問 3-2 では，特に相違度に関係する相関係数に影響が確認された．

これらの結果から，Jaccard 係数や類似度，相違度は，課題の内容や難易度により，各メトリックスとの相関が異なることがわかった．

5.3 考察

5.2 の結果から，課題の内容や難易度がメトリックス値に影響することを推察された．各解析課題ではコーディング量が異なる．そのため，学習者のコーディング量（追加したトークン数）の影響を除いた場合の Jaccard 係数，類似度，相違度の偏相関係数を計算した．結果を表 4 に示す．

それぞれの解析課題において，Jaccard 係数と類似度の偏相関係数は，0.99，0.99 ，1.0 で，強い正の相関があった. 類似度と相違度では, -0.51, -0.35, -0.35, Jaccard 係数と相違度では，-0.51，-0.37，-0.38 となり，かなり負の相関があるという結果となった．これらの結果から，Jaccard 係数と類似度は同等の指標であると考えられる．また，Jaccard 係数と対の関係にある Jaccard 距離は相関係数が[Jaccard 係数の相関係数－1]となるが, 類似度と相違度ではそうはならない. そのため，類似度と相違度により Jaccard 係数とは異なる質的評価が可能になると考える．

表 3: 問 2 の相関係数

相関係数		Jaccard係数	類似度	相違度	CC	行数
Jaccard 係数	s_ok		0.94	-0.06	0.54	0.78
	s_ng					
類似度	s_ok	0.94		0.13	0.73	0.91
	s_ng	0.94				
相違度	s_ok	0.01	0.21		0.34	0.26
	s_ng	-0.21	-0.02			
CC	s_ok	0.57	0.76	0.4		0.65
	s_ng	0.49	0.67	0.23		
行数	s_ok	0.78	0.91	0.33	0.69	
	s_ng	0.77	0.91	0.13	0.6	

表 4: 偏相関係数

解析課題	Jaccard 係数 類似度	類似度 相違度	Jaccard 係数 相違度
問 2	0.99	-0.51	-0.51
問 3-1	0.99	-0.35	-0.37
問 3-2	1.0	-0.35	-0.38

6. 学習者の進捗状況のパターンについて

リサーチクエスチョン 2)「学習指導が必要な学習者の進捗状況を特定のパターンに分けることができるのか」について，表 2 の学習者のスナップショットを用いて考察する．提案指標である類似度と相違度の 2 つの指標を用いてクラスタリングとプロセスマイニングを行った. クラスタリングの手法は k-means++法[10]である．k-means++法では，k-means 法が有する初期値依存性が解消されている．また，プロセスマイニングには Disco[11]を用いた．

6.1 クラスタリング

個々の学習者のコーディングを 10 のスナップショットで抽象化した結果に対して，類似度と相違度の 2 つの値によりクラスタリングした. エルボー法により，クラスタ数を 5 とし，それぞれのクラスタにラベル付を行った（図 5）．ここで，スナップショットがテンプレートと一致する場合の類似度と相違度の値はどちらも 0, 模範解答と一致する場合の類似度は 1, 相違度は 0 となる．青点は課題に取り組み始めた段階の Beginning クラスタ（以下，B クラスタ），緑点は課題を進めている途中で，状態が不確定な Indefinite クラスタ（I クラスタ），橙点は比較的順調に課題を解いている Favorable クラスタ（F クラスタ），紫点は学習指導が必要な可能性が高い Warning クラスタ（W クラ

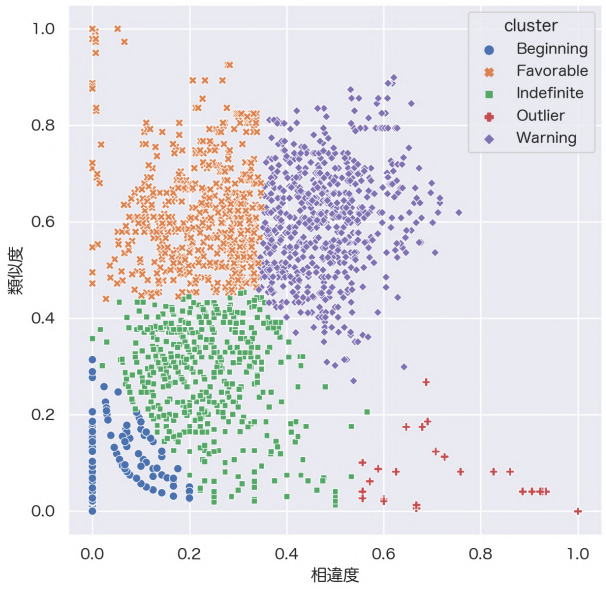

図 5: 類似度と相違度の散布図
（クラスタごとに色分け）

スタ），赤点は極端に類似度が低く，相違度が高い Outlier クラスタ（O クラスタ）とした．

図 5 の結果から，各クラスタについての考察を行う．B クラスタは，テンプレートの値（原点）に近い場所に分布しているため，課題に取り組み始めた学習者，もしくは，あまり手が動いていない学習者であると考える．つぎに，I クラスタは，B クラスタの右上付近に分布しており，およそ類似度は 0~0.4，相違度は 0~0.6 の範囲にある．このクラスタは，B クラスタにいた学習者がコーディングを行い，課題を解き進めている段階であると考える．F クラスタは，I クラスタの上側に分布しており，およそ類似度は 0.4~1.0，相違度は 0~0.35 の範囲にある．このクラスタには，模範解答と同じく，類似度が 1 で相違度が 0 であるスナップショットが含まれており，相違度が比較的低いため，順調に課題を進め，模範解答に近づいている学習者が多いと考える．W クラスタは，類似度が分布している範囲は F クラスタと似ているが，相違度の分布がおよそ 0.35 以上と高い値となっている．このクラスタに含まれる学習者は，課題を解き進めているが，模範解答に含まれない無駄なコードを多く記述している可能性が高いため，学習指導が必要である可能性が高いと考える．O クラスタは，類似度が 0.3 以下で相違度が 0.5 以上のあたりに分布している．このクラスタには，禁止事項であるテンプレートの修正，または削除をしてしまっているスナップショットも含まれており，W クラスタと同様に，学習指導が必要と考える．

表 5: 最後のスナップショットの平均値と標準偏差

	類似度	相違度
問 2	0.64±0.18	0.27±0.15
問 3-1	0.56±0.15	0.44±0.14
問 3-2	0.59±0.12	0.40±0.15

表 6: 解析課題ごとのスナップショットの割合

クラスタ	問 2 (1050)	問 3-1 (1200)	問 3-2 (510)	全て (2760)
Beginning	18.5%	14.0%	15.5%	16.0%
Favorable	46.3%	12.2%	19.0%	26.4%
Indefinite	19.0%	25.5%	25.5%	23.0%
Outlier	3.0%	4.2%	1.0%	3.1%
Warning	13.1%	44.2%	39.0%	31.4%

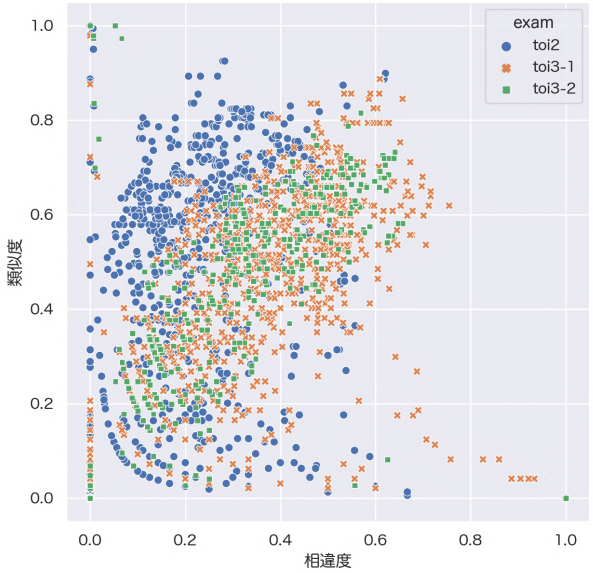

図 6: 類似度と相違度の散布図
（課題ごとに色分け）

ここで，表 5 にそれぞれの解析課題の最後のスナップショットの類似度と相違度の平均値と標準偏差の値，表 6 に解析課題ごとのスナップショットの存在するクラスタの割合を示す．解析課題の難易度は問 2，問 3-1，問 3-2 の順に高くなる．表 5 を見ると，解析課題 3 つの中で問 2 の類似度の値が一番高く，相違度の値が一番低い．類似度と相違度には課題の難易度が反映していると考えられる．また，問 3-1，問 3-2 は類似問題であり，類似度と相違度の値も近い結果となった．問 3-1 よりも問 3-2 の類似度が高く，相違度が低いのは，問 2 や問 3-1 を解き終えた学習者のみが問 3-2 を解いているためと考える．

また，図 6 に図 5 の散布図を解析課題ごとに色分け

したものを示す．この図より，解析課題 3 つの中で最も難易度が低い問 2 の分布と，難易度が高い問 3-1，問 3-2 の分布を比較すると，問 2 の分布が左上側によっており，問 3-1，問 3-2 の分布は比較的似ていることがわかる．

6.2 プロセスマイニング

6.1 のクラスタリングの結果をもとに，プロセスマイニングを行い，スナップショットが属するクラスタの時系列の変化を考察する．

図 7 にプロセスマイニングの結果を示す．図 7 は，すべての学習者の 10 のスナップショットの試験開始時から終了時までの遷移 (以下, プロセス) の中から，出現頻度が上位 30%のプロセスに含まれる学習者を抽出したものであり，クラスタのラベルとその経路を辿った学習者の割合が示されている．O クラスタは，上位 30%のプロセスには含まれなかったため，図 7 には 4 つのクラスタのみが示されている．また，図上部にある▽が開始地点であり，下部にある□が終了地点である．開始地点からプロセスを追っていくと，すべての学習者がまず B クラスタに遷移したことがわかる．B クラスタは，テンプレートに近いクラスタであるため，学習者がテンプレートをもとに課題に取り組んでいる様子が表れている．つぎに，約 7 割の学習者が I クラスタに遷移した．これは，コーディングを行い，順調に課題を解き進めていることを示している．残りの 3 割の学習者はそれぞれ，1 割が F クラスタ，2 割が W クラスタに遷移している. また, I クラスタでは，クラスタ内にしばらく停滞する学習者と，すぐに F クラスタ, もしくは W クラスタに遷移する学習者が見られた．このことから，W クラスタに含まれる学習者に加えて, I クラスタに長時間停滞し, F クラスタに進まない学習者も学習指導の必要があると考える．

7. 進捗状況可視化システム

リサーチクエスチョン 3)「学習指導が必要な学習者の進捗状況を演習の過程で検出できるのか」について，プログラミング演習中に使用することを想定した進捗状況可視化システムのプロトタイプを開発した．そして，学習進捗状況の可視化方法のシミュレーションを行った．

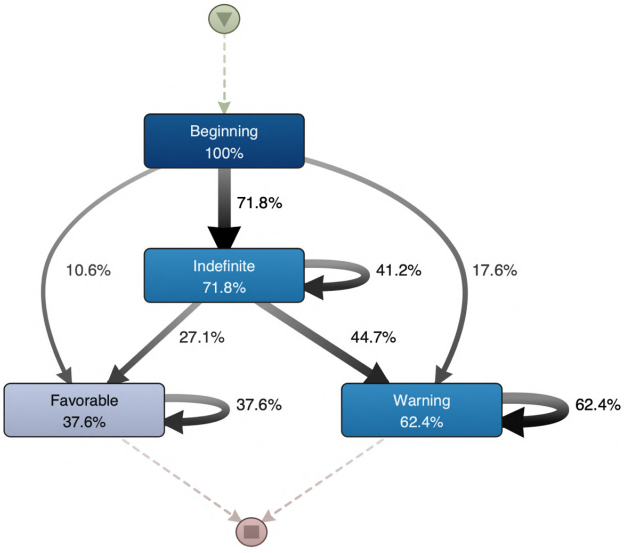

図 7:プロセスマイニングの結果

プログラミング演習での使用にあたり，最低限以下の 4 点の機能が必要である．

① 学習者のスナップショットの回収
② スナップショットの解析
③ 類似度，相違度の計算
④ 類似度，相違度の可視化

今回は，プロトタイプの開発であるため，①については，解析課題の学習者コードを使用することとし，②～④の機能の開発を行った．

7.1 システムの構成

本システムでは Docker Compose[12,13]を使用し，web アプリケーションの開発を行った．フロントエンドでは，フレームワークに React[14]（TypeScript[15]）を使用，バックエンドでは，Express[16]（Node.js[17]）を使用している．解析用サーバーには，Express と Gum Tree の環境を作成し，データベースは MongoDB[18]を使用した. 主に, 解析用サーバーに機能②, ③を実装し, フロントエンドに機能④を実装した.

7.2 可視化シミュレーション

指導者がプログラミング演習中に，学習者の進捗状況を把握するための機能についてシミュレーションを行った．図 9 はそれぞれ，解析課題問 2 の試験開始から 10 分後，30 分後，試験終了の各時点での，類似度と相違度の散布図である．プロットされている点 1 つが 1 人の学習者であり，点の色は，その時点での学習者の累積スナップショット数を相対的に表している．

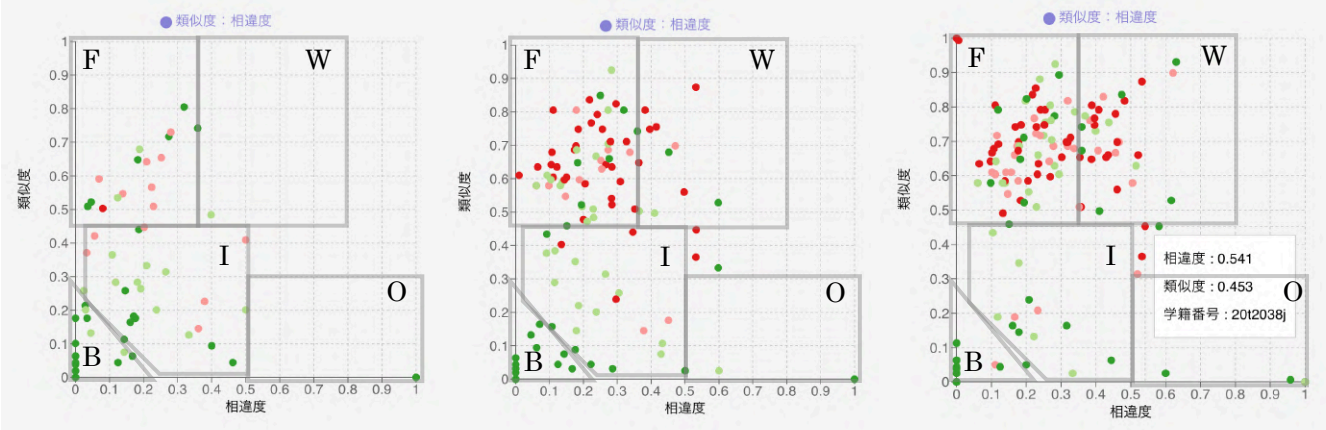

図 9: 類似度と相違度の散布図　（左から，試験開始 10 分後，30 分後，試験終了時）

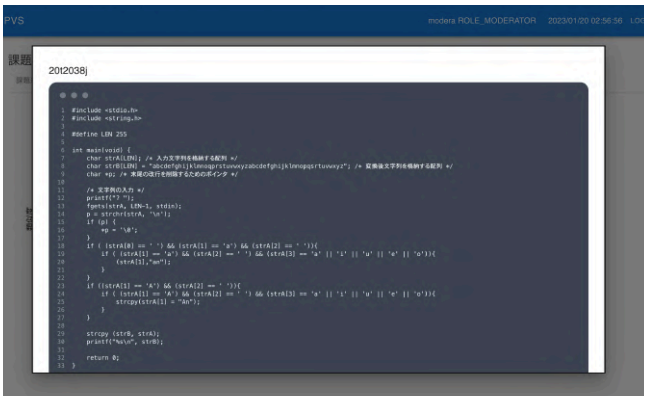

図 10:ソースコード確認画面

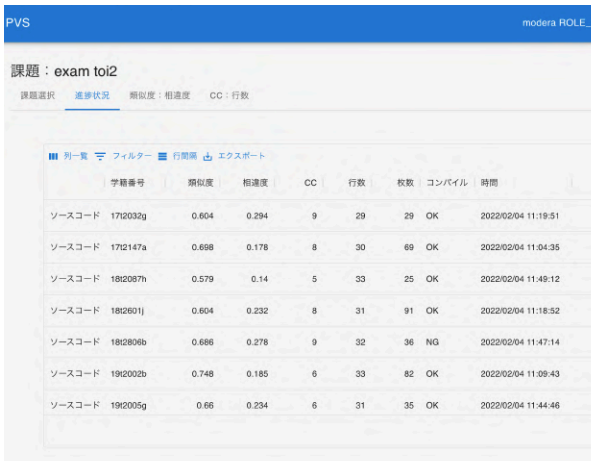

図 11:学習者進捗状況一覧

参照時点でのスナップショットの数が少ない方から，全体の 0~25%を緑色，25~50%を黄緑色，50~75%を桃色，75~100%を赤色とし，4 段階で表示している．試験開始から 10 分後では，B クラスタや I クラスタ，F クラスタに含まれる学習者が多く，W クラスタには学習者が存在していない．試験開始から 30 分後では，F クラスタの学習者が増えている．また，W クラスタに含まれる学習者も現れてきており，学習指導を行うポイントであると考える．一方で，B クラスタや，I クラスタでスナップショットの数が少なく，停滞していそうな学習者も見られるため，これらのクラスタに対しても，学習指導が必要であると考える．試験終了時では，F クラスタ，W クラスタの両方で学習者が増加している．

7.3　システムの機能

実際の演習では，30 分後などの時点で学習指導を行うことで，終了時に W クラスタにいる学習者を減らすことができる可能性があると考える．また，B クラスタや I クラスタで停滞している学習者と，W クラスタにいる学習者では学習指導の内容が異なることが想定される．指導者は散布図の気になる点（学習者）をクリックすることで，その学習者のソースコードを確認することができ，学習指導の参考にすることが可能である（図 10）．この他の機能としては，学習者の進捗状況の一覧が確認できるテーブル（図 11）があり，ソースコードの表示や，類似度や相違度などの各指標の値，スナップショットの枚数や保存された時間などを一括で確認することができる．ソート機能やフィルタ機能もあるため，気になる学習者を絞り込むことが可能である．また，CC と行数の散布図（図 12）も表示することができる．この散布図には，テンプレートと模範解答の CC と行数の値の位置にそれぞれ，緑色と赤色で境界線が引かれている．指導者はこの図を確認することで，明らかに行数が多い学習者や CC の値が高すぎる学習者を発見することができるため，類似度と相違度の散布図と合わせて，学習指導に活用できると考える．

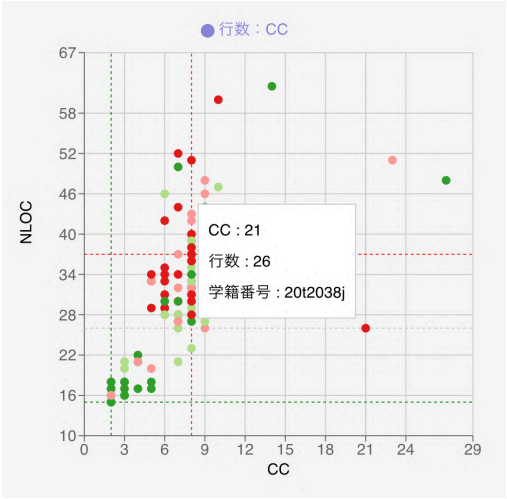

図 12: CC と行数の散布図

7.4 システムの評価

7.2 の結果をもとに，プログラム演習担当教員による提案システムの試用を行い，つぎの 2 点の意見を得た．1 点目は，「Warning クラスタの学習者のうち，スナップショット数が多い学習者（赤色）への指導の優先度が高いと考えられる．スナップショット数による色分けは，指導の優先度という点で有用性があると思われる」ということである．2 点目は，「クラスタごとに色分け，スナップショット数を色の濃さとすることで，クラスタとスナップショット数の両方の情報が得られるのではないか」ということである．

1 点目の意見から，提案システムの有用性が示唆されたと考える．2 点目については，クラスタ境界を背景画像として示し，現行のスナップショット数に基づく色表示のドットを重ねることで対応できると考える．

8. おわりに

本稿では，プログラミング演習で，学習指導が必要である学習者を同定し指導者の学習指導をサポートすることを目的として，3 つのリサーチクエスチョンに基づき研究成果を述べた．

本研究では，学習者の進捗状況を 5 つのクラスタに分け，学習指導が必要な学習者について考察した．進捗状況可視化システムを使用することで，解答の方向性が間違っている可能性がある W クラスタの学習者には解答のヒントを与え，B クラスタや I クラスタで長時間停滞している学習者にはアルゴリズムや課題内容についての初歩的な説明を行うなど，クラスタごとに異なる学習指導を行うサポートができると考える．

今後は，可視化システムの機能追加や実際の授業での使用について検討していく．

参 考 文 献

(1) 槙原絵里奈，藤原賢二，井垣宏ほか：“初学者向けプログラミング演習のための探索的プログラミング支援環境 Pockets の提案”，情報処理学会論文誌，Vol.57，No.1，pp.236-247（2016）
(2) 堀口悟史，井垣宏，井上亮文ほか：“講義資料閲覧ログを用いたプログラミング講義進 捗管理手法の提案”，情報処理学会論文誌，Vol.53, No.1, pp.61–71 (2012)
(3) 浦上理，長島和平，並木美太郎ほか：“プログラミング学習者のつまずきの自動検出”，情報処理学会論文誌，Vol.2020-CE-154，No.4，pp.1-8（2020）
(4) 市村哲，梶並知記，平野洋行：“プログラミング演習授業における学習状況把握支援の試み”，情報処理学会論文誌, Vol.54，No.12，pp.2519-2526（2013）
(5) 井垣宏，斎藤俊，井上亮文ほか：“プログラミング演習における進捗状況把握のためのコーディング可視化システム C3PV の提案”, 情報処理学会論文誌, Vol.54, No.1, pp.330-339（2013）
(6) 堀口諒人，筒井善規，井垣宏：“プログラミング演習における学生のプログラミング行動推定のための授業環境と実験環境の比較”，第 7 回実践的 IT 教育シンポジウム（rePiT2020）論文集，pp.114-120（2020）
(7) 秋山直人，新村正明：“プログラミング課題における進捗状況可視化手法の提案”，教育システム情報学会研究会講演論文集，Vol.34，No.6，pp.51-55（2020）
(8) J.-R.Falleri，F.Morandat，X.Martinez，et al.,“Fine-grained and accurate source differencing”，Proc. of the 29th ACM/ IEEE International Conference on Automated Software Engineering, pp.313-324 (2014)
(9) GumTreeDiff/gumtree：A neat code differencing tool – GitHub，https://github.com/GumTreeDiff/gumtree（2023 年 1 月 19 日 確認）
(10) David Arthur, Sergei Vassilvitskii，“K-means++: The Advantages of Careful Seeding”，Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete algorithms，pp.1027-1035（2007）
(11) Process Mining and Automated Process Discovery Software for Professionals - Fluxicon Disco, https://fluxicon.com/disco/,（2023 年 1 月 19 日 確認）
(12) Docker: Accelerated, Container Application Development, https://www.docker.com,（2023 年 1 月 24 日 確認）
(13) Docker Compose – Docker-docs-ja 20.10 ドキュメント，https://docs.docker.jp/compose/toc.html（2023 年 1 月 24 日 確認）
(14) Meta Platforms, Inc. React. https://ja.reactjs.org/（2023 年 1 月 24 日 確認）
(15) Microsoft. TypeScript.https://www.typescriptlang.org/,（2023 年 1 月 24 日 確認）
(16) OpenJS Foundation.Express,https://expressjs.com/ja/,（2023 年 1 月 24 日 確認）
(17) OpenJS Foundation. Node.js，https://nodejs.org/ja/,（2023 年 1 月 24 日 確認）
(18) MongoDB,Inc.MongoDB,https://www.mongodb.com/ja-jp,（2023 年 1 月 24 日 確認）