JSiSE Research Report
vol.33,no.6(2019-3)

画像比較による俯瞰的プログラミング能力評価のための可視化コンテンツ組生成

清光 英成 *1, マルティネス ディック *2, 孫 一 *3, 大月 一弘 *1
*1 神戸大学大学院国際文化学研究科, *2 株式会社 トラフィックス,
*3 神戸情報大学院大学情報技術研究科

Combination of Samples for Evaluating the Panoramic Understanding of Programming by Programmed Visual Contents Comparison

Hidenari Kiyomitsu*1, Dick Martinez Calderon*2, Yi Sun*3, Kazuhiro Ohtsuki*1
*1 Graduate School of Intercultural Studies, Kobe University, *2 Traffics Co. Ltd.,
*3 Graduate School of Information Technology, Kobe Institute of Computing

Abstract: This paper discusses the combination of visual samples used in the Programmed Visual Contents Comparison Method (PVCC). The PVCC is an aptitude and ability test to evaluate ability related to the Panoramic Understanding of Programming (PUP). This method is based on the comparison of two displayed images or interactive animations produced by programming. If a question is showed to a user taking the test, he/she is requested to decide which one of the samples is the more difficult to build with programming than the other, or, if the difficulty is similar for both of them. To clarify the evaluated ability of a user, we will organize the difference between visual samples in the inclusion, by difficulty and by perceivability of programming concept.

キーワード：テスト理論, 質的評価, グループワーク評価, プログラミング教育，プログラミングの俯瞰的理解

1. はじめに

ソフトウエア開発者のようなプログラミングに関する深い知識を有していなくても，いくつかの外部資源を利用して効率よく「動くものを作る」プログラマが増えつつある．近年のプログラミング環境の変化は，その特徴として，

(1) Web 上に多数のサンプルプログラムや解説が掲載されており，プログラム開発を業務とするデベロッパーもそれをコピーペーストするなどしてプログラムを作成
(2) 様々なプログラムが関数化され提供されるようになり，プログラム開発者はライブラリから最も適したものを探すという作業もプログラム作成において重要
(3) ゲーム・３Ｄ作成などの様々なアプリケーションに特化したソフトウエア開発ツールが提供されるようになり，それらのツールを利用することにより一部分のみソースコードを自分で記述するだけでプログラムが完成

といった点があげられる．ソフトウェア開発においてプログラムの一部あるいは，大枠部分をブラックボックスと捉え，それらをうまくつなぎ合わせるのみでプログラムを完成させていると考えることができる．ここでは，このような状況を「コピペ時代のプログミング環境」と呼ぶ．

開発環境の変化に加えて情報技術を利活用する業種の多様化により，ソフトウエア作成に従事するプログラム開発者の経歴やプログラミングの学習形態も多様化している．例えば，専門学校のデザイン系・ゲーム系のコースにおいては，初期の段階でグラフィック作成ツールやゲーム作成ツールの利用を教え，そのツール上でソースコードを書くというカリキュラムを採用していることが多い．また，情報工学系のコースにおいても，その延長線上でプログラミングを行う場合が増えている．大学の情報工学系の学科においても，伝統的なボトムアップ型のプログラミング学習方法である「スクラッチからプログラムを書く」という方法以外に，サンプルプログラムを渡してそれを書き直していく方法，グラフィカルなライブラリ（オブジェクト）が既に多数準備されているようなプログラミング言語による学習方法を採用している場合も増えている．

このようなプログラミング開発環境・学習経験の変化にともない，実際のソフトウエア開発現場においてもプログラム開発従事者のプログラミング能力が変化している．文献 (1) は，マイクロソフトのような大手ソフトウエア開発企業に従事する開発者においても「他人の作成したソースコードが読めない」者が多数いることを指摘している．このことは、開発者のプログラミング能力が低下していることも一因である可能性もあるが，現在のプログラム開発に

おいては必ずしもソースコードが完全に読めなくてもプログラム開発が可能であることも示している．実際に，ソフトウェア開発以外の分野を専業とする多くのプロフェッショナルがプログラムを書いて自らの問題解決に利用するようになった．著者らは,「プログラミング言語の文法などは完全には覚えていなくとも，作成すべきプログラムを俯瞰的に理解できており，他人のプログラムを参照したりすることにより高度なプログラムを作成できる」人材も増加しているのではないかとの仮説を立てている．

伝統的に，プログラミング能力を評価するために筆記試験による評価，作成した作品（プログラム）の評価などを行ってきた．しかし，他人のプログラムやプログラミング・マニュアル等をうまく参照しながらプログラムを作成する人はペーパーテストが必ずしも好成績とはならないし，作品を評価する場合も出来上がった作品やソースコード見ただけでは，どのような方法でどの程度の理解の上に作成したのかの判断が難しいという課題が生じている．そこで，本研究においては，プログラミング能力を測るための新しい評価手法を確立することを目的として，可視化コンテンツ比較法の提案と評価システムを構築してきた [(2)(3)]．デザイナーのコンピュータ利用などに関しては，グラフィックツールとプログラミングにおけるギャップを埋めようとするシステムの提案 [(4)(5)] はあるが，ソフトウェア開発を専業としないプログラマの俯瞰的理解を確認しようとする研究は他に存在していない．プログラミングスキルを評価する多数のオンラインプログラミングテストサイトがあり，プログラミング能力の証明書を発行するものもある [(6)(7)]．これらのサイトでは，コードを書くことを必要とするテスト形式か，ペーパーテストと同様なプログラミング知識についての問題に回答させている．

本稿では，可視化コンテンツ比較法に用いる可視化コンテンツの組合せの性質ついて，可視化コンテンツ組の差異の元となるプログラミング技術や技法（コンセプト）を，

- 包含するプログラミングコンセプトの有無
- プログラミングコンセプト間の難易比較
- 難易度の差の大きさ

に基づいて整理し，可視化コンテンツ比較法を応用したユーザ個別能力評価と協調学習に利用できる可視化コンテンツの組合せについて議論する．

2. 可視化コンテンツ比較法

可視化コンテンツ比較法は，プログラムによって描画された 2 つの出力画像（静止画，アニメーション，対話的画面遷移，グラフ，テキスト等）を利用者に比較させ，どちらの画像を描画するプログラムを書く方が難しいかを回答させることにより，プログラミングに関する俯瞰的な理解を測定する方法である．比較対象となる画像対の差異に気づき，コーディングテクニックの難易を正しく判断できるプログラミング能力を持つ利用者は，正しく回答することができる．図 1 は，左右のキャンバス上をそれぞれで 3 回クリックした結果である．便宜上，左右のキャンバス上の同様な座標をクリックした例を示している．

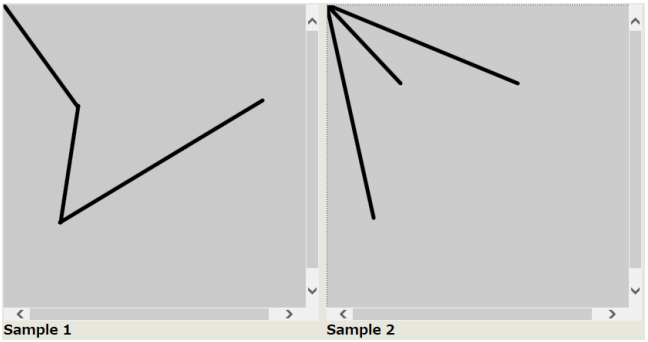

図 1: 可視化コンテンツ比較のためのコンテンツ組

可視化コンテンツ比較法のテストシステムは，コンテンツ比較部・ガイダンス部・サーベイ部の三部からなる．コンテンツ比較部以外は実験の目的などにより使い分けられる．

コンテンツ比較部は可視化コンテンツ組を表示し，四肢選択（「左」,「右」,「同じ（ほぼ同じ）」,「わからない」）で回答する応答システムである．

ガイダンス部は，ヒントと操作説明の二種類の説明文からなり，可視化コンテンツ組と同時に提示する．ヒントは，被験者が問題の分析を始めるために適切な糸口を示し，正解を連想させない文章である．

サーベイ部は，正解とその解説文を被験者に見せた後に，解説文に同意あるいは理解できたか，解説文を読む前に解説内容を把握出来ていたかという，問題に対する理解と気づきについて「はい」,「いいえ」で回答を求める．さらに，可視化コンテンツ比較に使用されている効果や動作について，オフィスソフトや画像処理ツールなどプログラミングコードを書かない方法での経験，解説文内で使用されているプログラミングテクニックに関する経験についての概要と詳細を頻度（一度もない，一回か二回，たまにする，よくする）での回答を求める．

実施した実験 [(2)(3)(8)] では，被験者個別能力測定において問題 1 問あたりの所要時間は，コンテンツ比較部の回答のみであれば 10 秒～30 秒程度，サーベイ部が 1 分程度であった．協調学習観測においては，複数人で可視化コンテンツの難易を議論するため所要時間はまちまちであるが，実験後に行ったフィードバックアンケートに「楽しく，負

担と感じない」などの肯定的な回答が多くあり，簡便でストレスの少ないプログラミング能力測定方式であるといえる．

3. 可視化コンテンツ組

可視化コンテンツ比較法のコンテンツ比較部では，例えば，図 1 のような描線結果が異なる画像を見せ，各動作を実現するプログラミングの難易を回答させることにより，コーディングに必要な俯瞰的理解を確認する．図 1 左の動作を実現するためには，直前にクリックされた座標を保持する必要があるが，図 1 右の動作は，初期設定座標からクリックされた座標までの描線で実現できるため，直前にクリックされた座標を保持する必要がない．このようなプログラミングに関する俯瞰的理解力を持つ利用者は，図 1 左のプログラムを書くほうが難しいと答えることができる．

利用者のプログラミングに関する俯瞰的理解力を測る手法の 1 つである可視化コンテンツ比較法においては，可視化コンテンツ組が測りたい能力の有無を顕在化させる対である必要がある．図 1 は左右のキャンバス上で同様な座標をクリックした例であるが，図 2 は，図 1 の対話的動作を無作為に行った例である．

情報系専門学校の学生を対象とした実験 [3] では，対話的動作を行う方式での回答において，全 16 問の成績中央までの上位の不正解は 20.6%(14/68)，下位の不正解は 53.0%(35/66) であった．

IT 関連企業の従業員 9 名を対象とした実験 [8] 中の個別学習においては，ガイダンス部に

ヒント：描かれた線それぞれの末端に注目してください．

操作：

1. マウスポインタをいずれかのキャンバスの上において
2. 右クリックして
3. キャンバス上で移動してもう一度右クリックして様子を見てください．

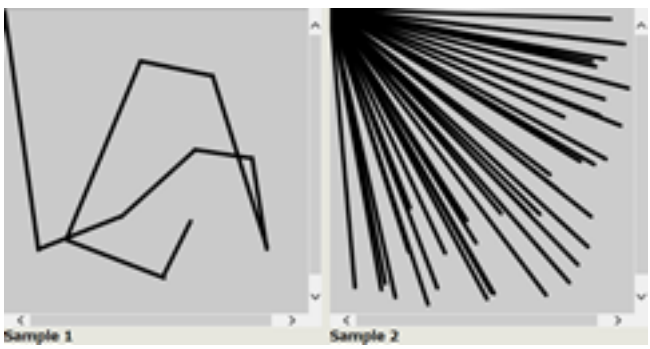

図 2: 図 1 の無作為操作の例

表 1: 回答の場合分け

ケース	正誤	理解	気づき
ケース 1	○	はい	はい
ケース 2	○	はい	いいえ
ケース 3	○	いいえ	はい
ケース 4	○	いいえ	いいえ
ケース 5	×	はい	はい
ケース 6	×	はい	いいえ
ケース 7	×	いいえ	はい
ケース 8	×	いいえ	いいえ

表 2: 被験者の回答ケースと正解数

被験者	A	B	C	D	E	F	G	H	I
ケース	4	1	1	1	1	5	1	1	8
正解数	3	7	9	5	6	3	7	7	1

を設定した．サーベイ部の「理解」「気づき」と回答の正誤を表 1 に場合分けした．ケース 1 とケース 8 であれば，この問題に対してプログラミングに関する俯瞰的な理解の有無を合理的に測定できている．ケース 4 であれば，偶然に正解したと考えるのが妥当である．9 名の被験者の回答ケースと出題全 10 問の正解数を表 2 に示す．正解数の多い被験者はケース 1 で回答しており，この可視化コンテンツ組が「直前座標の記録」というコンセプトの俯瞰的理解の評価に適していることがわかる．

しかし，被験者の環境によっては，対話的動作を実装できないことも考えられる．そのような場合，可視化コンテンツ比較法では静止画対をコンテンツ比較部に表示するが，動作をイメージしやすくする工夫がある．図 3 は，図 1 の対話的動作の動作順を丸数字で表示した例である．図 1 では操作結果の推移が，図 2 を静止画として表示した場合にはどの座標をどのような順番でクリックしたのかが曖昧であるが，図 3 のように表示すれば，幾分かは曖昧性を排除できると考えられる．

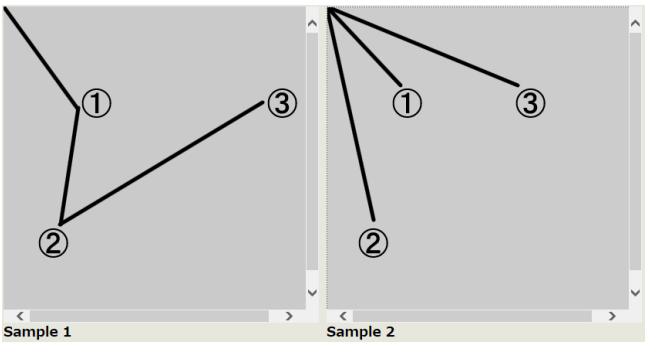

図 3: 図 1 に動作順を表示した例

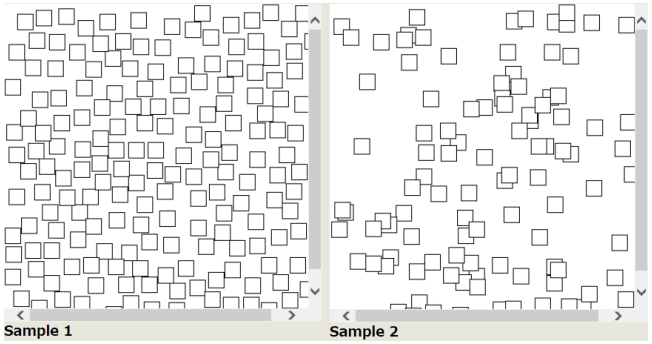

図 4: 重畳禁止コンセプトの有無によるコンテンツ組

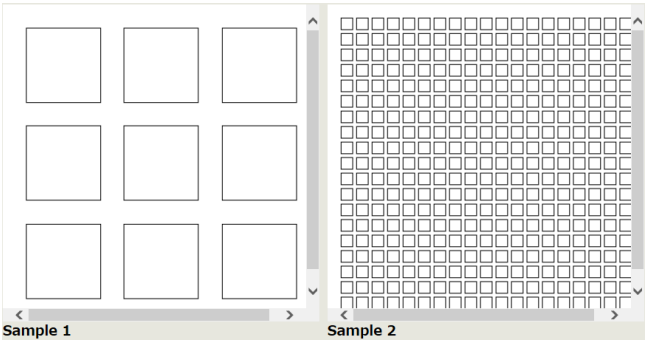

図 5: 静止画像の難易を比較するコンテンツ組

4. 可視化コンテンツ組の性質

可視化コンテンツ比較の回答は，四肢選択（「左」,「右」,「同じ（ほぼ同じ）」,「わからない」）である．本節では，可視化コンテンツ組の差異の元となるプログラミング技術や技法（コンセプト）の組合せを整理する．

4.1 コンセプトの有無

包含するプログラミングコンセプトの有無を可視化コンテンツ組の比較から見出し，他方にないプログラミングコンセプトを含むコンテンツの方が難しいという回答を正解とする問題である．図 1 が対話的動作に関するこのタイプの可視化コンテンツ組である．

図 4 は，乱数により描画する座標を決定し，正方形を描画していく可視化コンテンツ組である．図 4 左のみに重畳禁止の制約を課している．各コンテンツの動作を観察することにより，包含するプログラミングコンセプトを見つける必要がある．図 4 左は重畳禁止であるから，キャンバスが正方形で埋めつくされ易く，動作が収束するかのように見える．図 4 右は重畳が禁止されていないため，描画がいつまでも続く．図 4 左は初期の画像変化が多く，図 4 右は初期の画像変化が少ないため，図 4 右の方が重そうに見える可視化コンテンツ組である．

4.2 同一コンセプト

異なる形状や効果を包含する可視化コンテンツ組の比較から，同一のプログラミングコンセプトによるが，入力値（パラメータ）などのみの違いによりコンテンツ間の差異を実現していることを見出し,「同じ（ほぼ同じ）」という回答を正解とする問題である．

図 5 左は縦横 3 個ずつ，図 5 右は縦横 19 個ずつの正方形が規則的に描画されている.「入れ子になった繰り返し」が共通に包含するプログラミングコンセプトであるようなコンテンツ組である．これら 2 つのプログラミングによって生成された画像は，正方形の大きさと個数が異なるが，正方形を縦横同数描画するという点で同じである．プログラムを構成する事項の違いは，正方形の個数と大きさ，つまり，入力は正方形の個数または大きさと考えられる．コーディングにおける差異は入力値のみであり，処理は同様で出力の違いは入力の差に依存する．

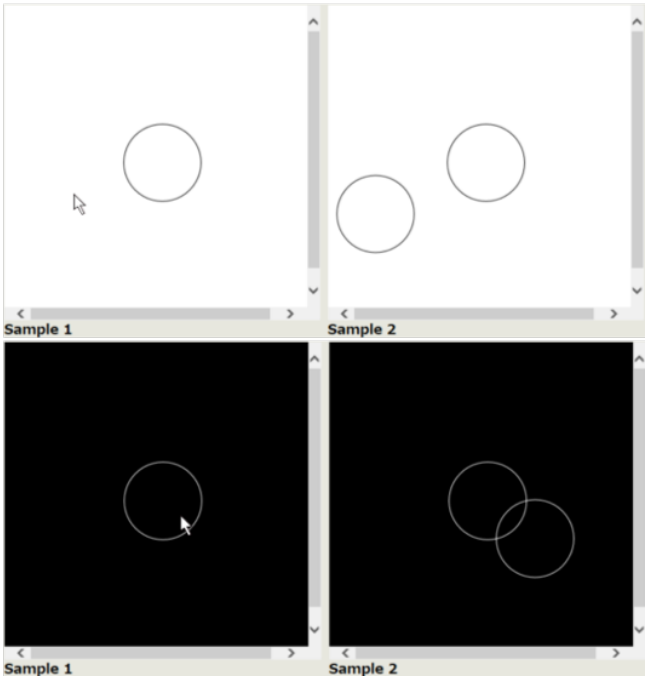

図 6: 対話的動作の難易を比較するコンテンツ組

図 6 は対話的動作に関するこのタイプの可視化コンテンツ組である．図 6 左の可視化コンテンツは，マウスポインタと中央の円とが重なると背景が黒くなり，図 6 右の可視化コンテンツではマウスポインタの矢印と置き換えた円が中央の円に触れると背景が黒くなる．一見，衝突判定のように見え，そのように実装も可能である．しかしながら，検出されたマウス座標と中央の円の中心との距離が，図 6 左の可視化コンテンツは中央の円の中心からの距離が円の半径，図 6 右の可視化コンテンツは中央の円の中心からの距離が円の直径より近くになると背景を黒くすることでも同じ動作を実現できる．

4.3 異なるコンセプト

異なる形状や効果を包含する可視化コンテンツ組の比較から，それぞれの可視化コンテンツのプログラミングコン

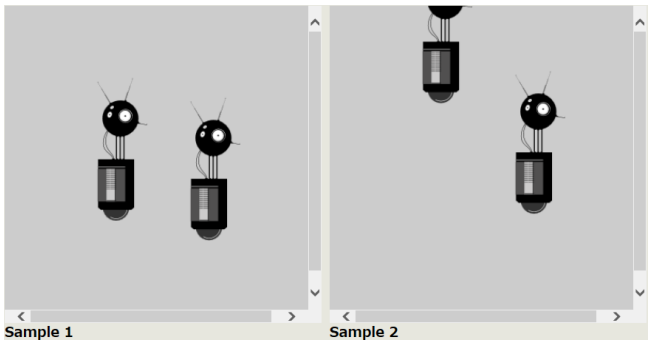

図 7: 異なるコンセプトの難易を比較するコンテンツ組 1

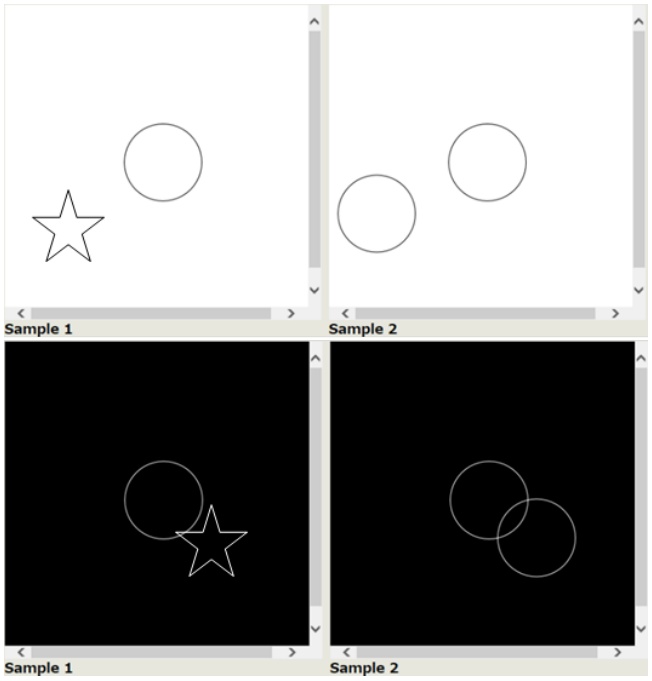

図 8: 異なるコンセプトの難易を比較するコンテンツ組 2

セプトを見出し，プログラミングコンセプト間の難易差から回答させる問題である．

図 7 は，周期的に移動するオブジェクトを描画するアニメーションの比較である．それぞれ 2 つのオブジェクトが $\pi/4$ の位相差で図 7 左は正弦関数，図 7 右は正接関数により移動する．可視化コンテンツの差異の元となるプログラミングコンセプトはサインとタンジェントである．コーディングにおいて，2 つのプログラムの違いは該当箇所を sin と書くか tan と書くかだけであるから，「同じ（ほぼ同じ）」という回答を正解とする問題である．しかし，図 7 左はキャンバス内での単調な移動であるのに対して，図 7 右はキャンバスを上下に通り抜けるため，情報系専門学校の学生を対象とした実験 [3] では図 7 右の方が難しいとの回答が 53.7%(72/134) であった．成績上位者とそれ以外の回答分布に差はなかった．

図 8 は図 6 左のマウスポインタの矢印を右側の移動円に内接する大きさの星型に変更した可視化コンテンツである．図 8 左の実装に衝突判定が必要かどうかは，星型の凸

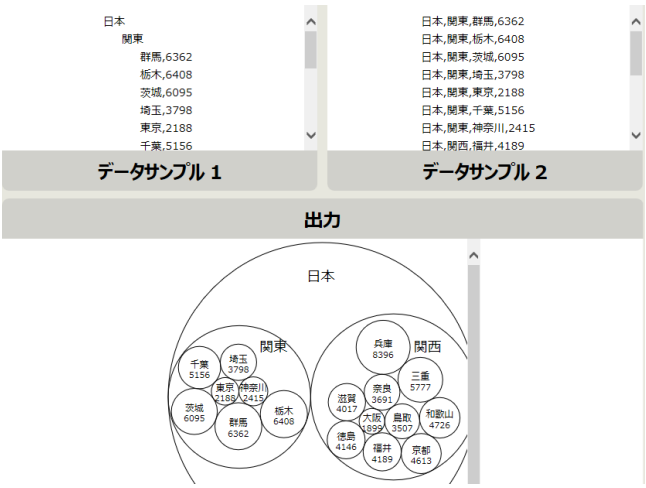

図 9: 入力データの可視化

部の接触だけでは判別できず，マウス検出座標から中央の円の中心までの距離が直径未満で星型のいずれの部分も中央の円に接触していない時に背景が黒くなるかどうかで判別する必要がある．背景が黒くなればマウスポインタとしての図形が星型か円かの違いであり，プログラミングコンセプトは同一である．この場合，「同じ（ほぼ同じ）」という回答を正解とする問題である．他方，マウス検出座標から中央の円の中心までの距離が直径未満となっただけでは背景が黒くならず，星型が接触した時のみ背景が黒くなれば，図 8 左のプログラミングコンセプトは衝突判定である．衝突判定には距離の大小比較というプログラミングコンセプトを含むため，図 8 左のプログラミングコンセプトが衝突判定であれば，図 8 右との間にプログラミングコンセプト間の難易差が存在することになる．この場合，「左」の方が難しいという回答を正解とする問題である．

4.4 入力データ

同一のデータを異なる手法で可視化する問題と，異なる形式で同じ意味を持つデータによる可視化の問題を作成している．図 9 は，図 9 左の JSON のような形式のデータと図 9 右の CSV 形式データを入力データとした時にどちらがコーディングし難いかを回答する問題である．一般に，ファイル中のデータは行毎に読み込み，データの構造を配列やリストにより模倣することで可視化フェーズへと渡される．つまり，行毎に書式やデータの表現方法が異なれば，その違いを吸収するためのコードを書く必要がある．そのため，図 9 は「左」の方が難しいという回答を正解とする問題である．プログラミングコンセプトに関する理解については上述のとおりであるが，Web サービス構築や Web API が身近なプログラマ，また d3.js を知る利用者は，図 9 右には JSON 形式にするための一手間があるため「右」の方が難しいという回答するかもしれない．他方，データ

ベースや事務系情報処理を業務とする利用者はデータの表現方法が一様であるかどうかだけで，換言すれば，プログラミングコンセプトとは関係なく正解の「左」の方が難しいという回答をするかもしれない．これは，プログラミングに関する俯瞰的な理解に加えて，利用者のプログラミングに関わる経歴の推定に利用できる可能性を示唆している．

入力データを用いる可視化コンテンツ比較は，サーベイ部を用いた気づきと理解の確認による回答のケース分けとともに，利用者が既に持っている知識とプログラミングコンセプトとの関わりを利用者が学習できるように組み合わせて出題する．

5. 学習方法に応じたコンテンツ組

個別学習やテストシステムを用いたプログラミング能力評価においては，利用者が一人でコンテンツ比較を行う．協調学習では，複数の利用者が同じ可視化コンテンツ組を同時に見ながら正解を議論する．このように，学習方法によって利用の形態が異なる．そこで，学習方法の違いから可視化コンテンツ比較のためのコンテンツの組合せについて考察する．

5.1 個別学習

利用者が一人でコンテンツ比較を行う場合，可視化コンテンツ比較に用いられる可視化コンテンツが包含するプログラミングコンセプトに関しての俯瞰的理解能力をその利用者が持っていれば，可視化コンテンツ間の差異に気づくことができなければならない．能力を持つ利用者が認知可能な可視化コンテンツ組の判別に関して，統計的手法を用いた研究 [2][3] と質的調査による研究 [8] を進めている．問題の正解についてどちらかが難しいとほぼ同じを用意したが，個別学習においては，どちらかが難しい場合は片方に使われて他方に使われていないプログラミングコンセプトが存在し，ほぼ同じ場合は同じソースコードで実現されるような可視化コンテンツ組としている．可視化コンテンツ間の差異が著しい場合，例えば，図 6 左と図 8 左とをコンテンツ組とした場合，可視化コンテンツが包含するプログラミングコンセプトに関しての俯瞰的理解能力をその利用者が持っていなくとも可視化コンテンツ間の差異に気づくことができてしまうため，適切なコンテンツの組合せとは言えない．可視化コンテンツが包含するプログラミングコンセプトの難しさの比較ができなくても正解を選ぶことができるからである．

可視化コンテンツの作成にあたっては，様々なプロブラミングレベルに対応した問題集 [9][10][11][12] を参考にし，それぞれの可視化コンテンツが包含するプログラミングコンセプトに関して認知可能性に注意しながら組み合わせている．また，プログラミング言語や技法の違いに起因する難易度の逆転の可能性が否定できない問題は避けるようにしている．

5.2 協調学習

複数の利用者が同じ可視化コンテンツ組を同時に見ながら正解を議論する協調学習においては，個別学習の場合と同様に可視化コンテンツ比較に用いられる可視化コンテンツが包含するプログラミングコンセプトに関しての俯瞰的理解能力をその利用者が持っていれば，可視化コンテンツ間の差異に気づくことができなければならないが，協調学習に参加する利用者の少なくともひとりが気付けば良い．

IT 関連企業の様々な部署に所属する 7 名を参加者とした協調学習の観察では，問題が簡単であれば，上級エンジニアは初心者に回答させ，回答の理由を尋ねていた．回答が不正解であれば，上級エンジニアが初心者に正解とその解説をしていた． 例えば，図 1 にした問題に対して，事務部門の参加者が「右が難しい」と回答した．その理由は「画像作成ツールを使用して線を描画するときはマウスで次の場所をクリックするだけでよいのですが，右の図の場合は毎回マウスを開始点に移動する必要があります．特に左上の始点をクリックするのは面倒なので右の方が難しいと思いました」であった．他の多くの参加者は，図 1 右は初期値と入力値による描線，図 1 左には直前の描線の終点を新しい描線の視点にする座標値交換が必要であることが理解できていて，プログラミング初心者の参加者達に代わる代わる説明をしていた．他の問題でも同様な知識の承継を観察することができた．

プログラミングの熟練者の間でも図 6 の問題に関して議論があり，参加者の一人がこの問題の特殊性（衝突判定を用いない実装方の存在）に気づいた．参加者全員が問題の正解と理由に同意したとしても，各参加者の理解度は異なると考えることができる．

他者間の議論を傍聴することによって，すでに持ち合わせている知識の再構成による知識の洗練やスキルの獲得が期待できる．

6. まとめ

可視化コンテンツ比較法におけるコンテンツの組合せについて，可視化コンテンツ組の性質と学習方法などの視点から議論した．可視化コンテンツ組の性質を，

(1) 可視化コンテンツが包含するプログラミングコンセプトの有無

(2) 同一のプログラミングコンセプトによる異なる可視化表現

(3) 異なるプログラミングコンセプトの難易の差

(4) 入力データ付き可視化コンテンツ組

に分けて整理した．

また，学習方法に応じた可視化コンテンツ組の生成について，利用者が一人でコンテンツ比較を行う場合と複数の利用者が同じ可視化コンテンツ組を同時に見ながら正解を議論する場合に分けて，可視化コンテンツ間の差異に関する認知可能性の扱いの違いを示し，それぞれの場合における，適切な可視化コンテンツ組について議論した．

個別学習とテストシステムを用いたプログラミング能力評価においては，可視化コンテンツ比較に用いられる可視化コンテンツが包含するプログラミングコンセプトに関しての俯瞰的理解能力をその利用者が持っていれば，可視化コンテンツ間の差異に気づくことができる認知可能性を持たせるべきであることに言及した．

協調学習においては，プログラミング熟練者から初心者へのプログラミング技術ならびに知識の承継が観測できた．また，熟練者間でも所属部署や職種により問題の捉え方が異なり，議論や意見交換により建設的な知識流通を確認した．また，他者間の議論を傍聴することによって，すでに持ち合わせている知識の再構成による知識の洗練やスキルの獲得が期待できることもわかった．個別学習と協調学習の両方のフィードバックアンケートから被験者らは楽しい方法でテストを受けることができたとの回答を得ている．簡便でストレスの少ないプログラミング能力測定方式であるとともに，新しい学習法の提案といえる．

謝辞

本研究の一部は科研費 (15K01068)「コピペ時代のプログラミング能力評価手法の開発」の支援による．ここに記して謝意を表す．

参考文献

(1) Thomas D. LaToza and Brad A. Myers. Hard-to-answer questions about code. In *Evaluation and Usability of Programming Languages and Tools*, PLATEAU '10, pp. 8:1–8:6, New York, NY, USA, 2010. ACM.

(2) Dick Martinez Calderon, Kin Man, Hidenari Kiyomitsu, Kazuhiro Ohtsuki, Yukinobu Miyamoto, Yi Sun, and Masami Hirabayashi. Measurement range increment in a method for evaluating panoramic understanding of programming. In *2016 IEEE Frontiers in Education Conference, FIE 2015, Eire, PA, USA, October 12-15, 2016*, pp. 1–8, 2016.

(3) Dick Martinez Calderon, Kin Man, Hidenari Kiyomitsu, Kazuhiro Ohtsuki, Yukinobu Miyamoto, and Yi Sun. An evaluation method for panoramic understanding of programming by comparison with visual examples. In *2015 IEEE Frontiers in Education Conference, FIE 2015, El Paso, TX, USA, October 21-24, 2015*, pp. 1–8, 2015.

(4) Fatih Kursat Ozenc, Miso Kim, John Zimmerman, Stephen Oney, and Brad Myers. How to support designers in getting hold of the immaterial material of software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pp. 2513–2522, New York, NY, USA, 2010. ACM.

(5) B. Myers, S. Y. Park, Y. Nakano, G. Mueller, and A. Ko. How designers design and program interactive behaviors. In *2008 IEEE Symposium on Visual Languages and Human-Centric Computing*, pp. 177–184, Sep. 2008.

(6) Hackerrank. https://www.hackerrank.com/. Accessed on 2019-2-5.

(7) Microsoft. Microsoft Certification Exam List - Microsoft Learning. https://www.microsoft.com/en-us/learning/exam-list.aspx. Accessed on 2019-2-5.

(8) Hidenari Kiyomitsu, Dick Martinez Calderon, Kazuhiro Ohtsuki, Sun Yi, Toshiharu Samura, Yukinobu Miyamoto, and Masami Hirabayashi. An approach for evaluating it employees' programming ability using the programed visual contents comparison method. In *2018 IEEE 7th International Conference on Teaching, Assessment, and Learning for Engineering, TALE 2018, Wollongong, Australia, December 4-6, 2018*, pp. 747–754, 2018.

(9) Daniel Shiffman. *Learning Processing: A Beginner's Guide to Programming Images, Animation, and Interaction.* Morgan Kaufmann, Burlington, MA, 2008.

(10) Daniel Shiffman. The nature of code. Self-published, 2012.

(11) Kostas Terzidis. *Algorithms for Visual Design Using the Processing Language*. Wiley Publishing Inc., Indianapolis, IN, 2009.

(12) Bonedikt Gross Hartmut Bohnacker and Julia Laub. *Generative Design: Visualize, Program and Create with Processing*. Princeton Architectural Press, New York, NY, 2012.