UIの違いによる考え方の違い: ユーザ操作の測定・分析のためのスクリーンリーダーUIの設計
Different UI, different way of thinking: UI design for screen reader to measure and analyze users' operations

山口 琢[†], 大場 みち子[‡]
Taku Yamaguchi, Michiko Oba
[†] フリー, [‡] 公立はこだて未来大学システム情報科学部
Independent Researcher, Faculty of Systems Information Science, Future University Hakodate

概要

「考え方」の研究において、Webアプリケーションとして実装した文章やプログラム・コードのジグソー・パズルを題材に、見て並べ替えるUIと聞いて並べ替えるUIとでプレイヤーの考え方が異なるかどうかを比較するとき、2つのUIが備えるべき要件を論ずる。考え方・解き方の違いを比較するためには、測定できるデータから比較したい事象を検出できるようにUIを設計・実装する必要がある。

キーワード: 思考力, 考え方, 並べ替え作文, 並べ替えコーディング, プロセスの測定と分析, スクリーンリーダー, Voice User Interface, 試行錯誤, 定石

1. はじめに

教育では思考力が重要視されてきている。Computer-based testing (CBT) の導入が進み、従来の紙ベース試験とは異なる手法で「思考力・判断力・表現力」を評価できると期待され、研究されている[1]。同じ問題データに対して、複数のUser Interface (UI) を用意して、アクセシビリティを高められることもCBTのメリットである。

スマートスピーカーなど音声ユーザインタフェース (VUI) やオーディオブックが普及し始めている。スマートスピーカーによるアプリは、音声のみによる対話で操作できる。スマートフォンにはスクリーンリーダーが用意されていて、音声による画面の読み上げとタッチ操作で、画面を見ずにアプリケーションを操作できる。

国連のSDGs (Sustainable Development Goals) では教育などですべての人に等しい機会を保証するという目標を掲げている。障害者差別解消法が施行され「社会的障壁の除去の実施についての必要かつ合理的な配慮」が義務・努力義務とされた。

1.1 着想

思考力を論じるとき「考え方」ということばが使われる。このことは「考えることに方法がある」と認識されている現れであろう。考えることに方法があるなら、考えながら何かを操作するとき、考える方法が操作のパターンに表れると期待できる。野球や囲碁・将棋の放送番組で、実況・解説の楽しみ方の1つはこれであろう。

ITを利用したパズル・アプリケーションであれば、アプリの操作を記録 (測定) する仕組みを適切に設計することで、解く考え方がパズル操作の測定データに反映されると期待できる。この手法で「考え方」を研究するとき、パズル・アプリの設計、測定データの設計、測定データの分析手法、パズル問題そのものの設計の、それぞれに工夫が必要となり研究テーマとなりえる。

この手法を適用することで、見て読む場合と、見ずに聞いて読む場合とで、考え方に違いがあるか比較できると期待する。コンピューター・システム的に言えば、全く同じ試験問題データを、見るUIと聞くUIとの2種類の試験UIで提示したとき、それら試験問題を解く考え方は同じなのか?

1.2 本稿の目的とアウトライン

本稿では、パズル・アプリの設計を取り上げる。題材は、後述する文章やプログラム・コードのジグソー・パズルである。このパズルを、目で読んで並べ替える場合と、見ずに聞いて並べ替える場合とで、並べ替え方が異なるのか比較する研究が対象である。そして、後者の聞いて並べ替えるUIの要件を検討するのが本稿の目的である。

とはいえ、本研究のような手法が広く知られているとは言いがたい。この手法は、近年のITの進歩と普

及に負うところが大きいため歴史が浅い。そこで、既発表の内容から、本研究の手法を説明する。

まず、文章やプログラムのジグソー・パズルを題材として、パズルのピースをドラッグ&ドロップで並べ替えて解くアプリ (2.1 節)、パズルを解く操作 (プロセス) の測定データ (2.2 節)、測定データを分析する手法の例 (2.3 節、2.4 節) を説明する。

次に、考え方の研究手法としてプロセスを測定・分析する手法の特徴を、予期せぬ考え方の発見や試行錯誤を例に説明する (2.5 節、2.6 節)。

最後に、パズルをスクリーン・リーダーで解く UI を 2 例あげて、本稿の主題である UI の設計を論じる 3. 章)。

2. プロセスを測定・分析する手法

われわれは、IT によるデジタル・パズルをうまく設計して、パズルを解く操作を測定・分析することで、解く過程の思考を捉えて、学習・指導などに役立てる研究に取り組んでいる。このコンセプトを前回の第 35 回大会でもポスター発表した [2]。

2.1 ジグソー・テキスト

図 1 は、分割されランダムに並べられた文章のピースを、並べ替えて完成させる文章のジグソー・パズル、並べ替え作文である。「ジグソー・テキスト」と呼んでいる。スマホなどで操作する Web アプリケーションである。「ひとつは…金銭を要求する。」などの部分を、パズルのピースと呼ぶ。パズルを解くユーザーはプレイヤーである。

ゼロから文章を組み立てる作文と比べると、ジグソー・テキストの特徴は「何を」組み立てたのかがプレイヤー間で共通なことである。プレイヤーはピースを読解して、文章全体を組み立て直す、限定された作文と見なすこともできる。誤字脱字や表現の推敲・修正は並べ替え操作 (編集操作) に含まれない。文章の順番にのみ絞って、推敲や修正の様子を測定・分析できる。

2.2 測定データ

ジグソー・テキストはプレイヤーの並べ替え操作を測定している (図 2)。まず、パズルを解き始めたときと完成したときに、その時刻と、そのときのピースの並び順が記録される。ピースはドラッグ&ドロップで並べ替えられる。ドラッグし始めたときとドロップしたときに、その時刻と、ドラッグやドロップされたピース、それぞれのときに前後にあったピースや、ピース全体の並び順が記録される。図 2 は、ID が s3 のピースをドラッグし始めたときと、ドロップしたときに記録されるデータを示している。

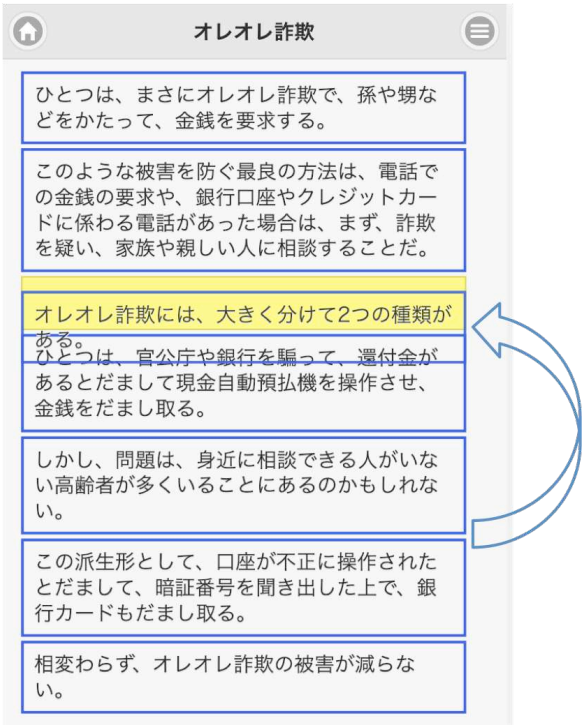

図 1 ドラッグ&ドロップで並べ替える Web アプリケーション「ジグソー・テキスト」

ドラッグ&ドロップはセットで一瞬の操作と思えるが、それでもドラッグとドロップを区別して記録する理由は 3.2 節で明らかになる。

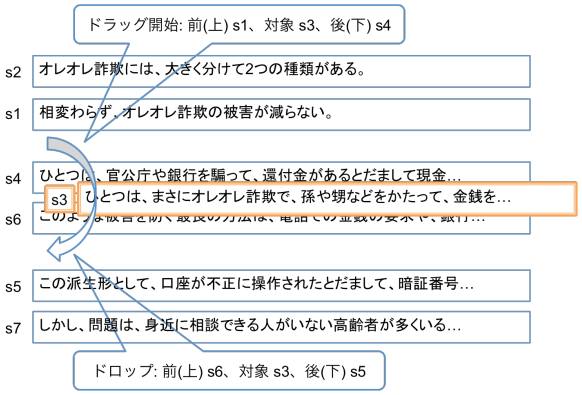

図 2 並べ替え操作の測定

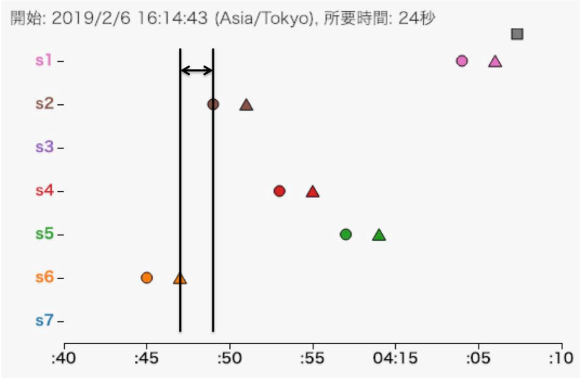

図 3　対象と移動先を順に選択する UI での並べ替え操作の時系列散布図

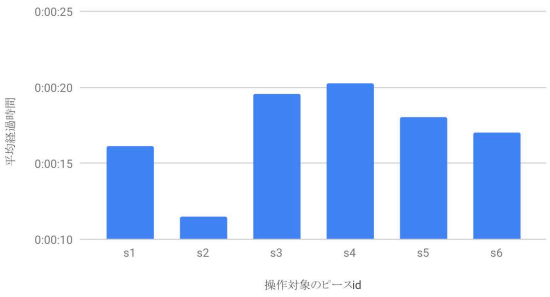

図 4　横軸がピース、縦軸がピースを動かすまでの経過時間の平均。s2 ピースは、動かす判断が早い。

2.3　パズル操作の分析例

図 1 の UI で、あるパズルをプレイヤーが解く様子を、2.2 節や図 2 のように測定したデータを、時系列散布図で示したのが図 3 である。縦軸の s1〜s7 が並べ替え対象のピース、横軸が経過時間、丸○がドラッグ開始すなわち移動開始、三角△がドロップすなわち移動完了を示す。前のドラッグ&ドロップの移動完了 (三角△) と、次のドラッグ&ドロップのドラッグ移動開始 (丸○) との間の時間は、次に何をどこへ動かすか考えている時間といえよう。

このようなデータに基づいて、ピースを動かすために考えた時間、すなわち、前のピースを動かした後でこのピースを動かすまでの平均経過時間を図示したのが図 4 である。図 3 で、三角△と丸○の間の時間を集計したものである。この例では、s2 のピースは、移動開始するまでの時間が他のピースよりも短い。プレイヤーたちは「s2 ピースの何らかの特徴を捉えて、s2 の移動先を素早く決める」という考え方・解き方をしていたと推定できる。これが、このアプリ UI の、あるいはプレイヤーたちの、あるいはパズル問題の、あるいはこれらの複合した特徴である。

```
s1  // 以下は1つ以上の要素を持つdouble型配列を引数dataとして受け取り、その平均値を計算して返すメソッドの中身である。
s2  // 正しく計算できるものになるように、四角で囲まれた部分の順番を適切な順にせよ。
s3  double sum = 0.0;
s4  for (int i = 0; i < data.length; i++) {
s5      sum += data[i];
s6  }
s7  return sum / data.length;
```

図 5　ジグソー・コードの例題、データの平均を求める

n \ n+1	s3	s4	s5	s6	s7
s3	7	1	4	1	5
s4	4	5	3	**10**	5
s5	2	1	3	0	3
s6	**9**	0	2	4	4
s7	5	3	1	4	3

図 6　並べ替え操作の時間的な共起行列

2.4　ジグソー・コードと分析例

われわれは、ジグソー・テキストと同様に、コンピューター・プログラムのソース・コードを並べ替える並べ替えコーディング「ジグソー・コード」でも研究に取り組んでいる [3]。

図 5 はジグソー・コードの問題例である。図 1 と違って、プレイする画面でなく、簡易分析アプリの画面である。この例はプログラミング言語 Java の問題である。数値の配列を受け取って、それら数値の平均値を計算して返すメソッド (関数) のプログラム・コードを完成するように求めている。図では、コードは正しい順序に並んでいる。

図 6 は、図 5 を解いた 32(人) のプレイを集計した表である。行列の見出しにある s4 などは、パズルのピースの ID である。表は、行見出しにある s3〜s7 のピースを動かした直後に、列見出しのピースを動かした回数を、交差するセルに数えたものである。図では、s4 の後に s6 を動かした回数が 32 プレイの全体で 10 回あった。この表を、並べ替え操作の時間的な、または手順的な共起行列と呼んでいる。

セルの値は 0 から 10 まであり一様ではない。一番多いのが s4 の次に s6 を動かした 10 回、次に s6 の次

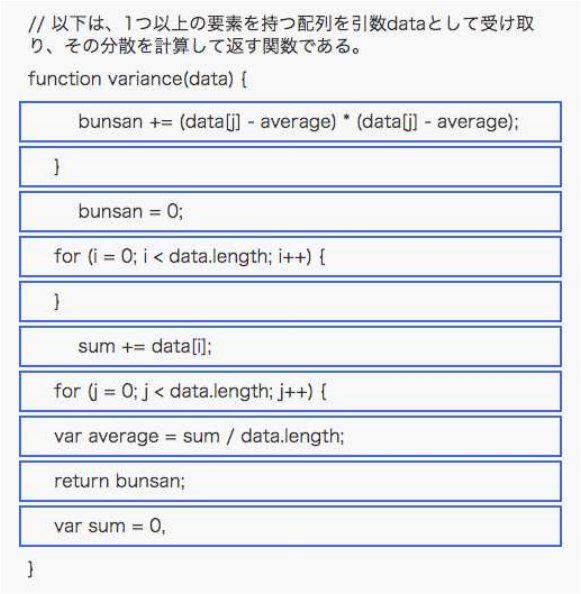

図 7　分散を求めるパズルの開始直後

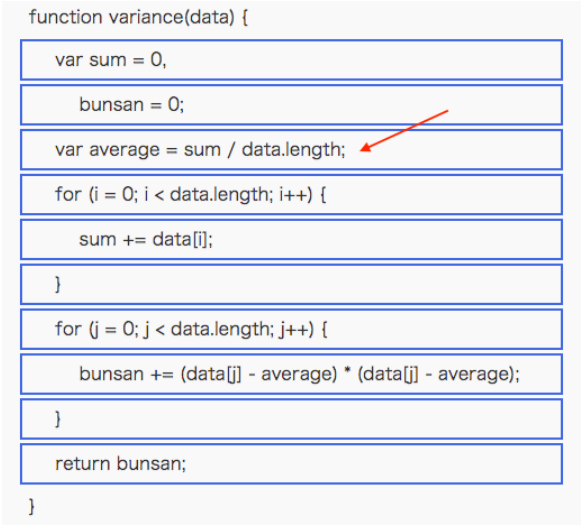

図 8　分散を求めるパズルを解いている途中

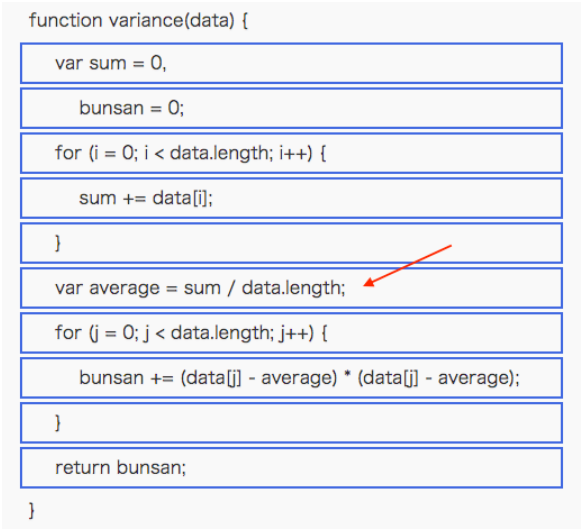

図 9　分散を求めるパズルを完成したところ

に s3 を動かしたのが 9 回で多い。s6 の次に s4 を動かしたのと、s5 の次に s6 を動かしたのは 0 回であった。この偏りは、プレイヤーたちにとっての、特定のピース間の特別な関係を反映していると解釈できる。

図では、s4 は for 文の開始の開きカッコ”{”であり、s6 は閉じカッコ”}”である。開きカッコと閉じカッコの対をまとめて動かしていると考えられる。このように、共起関係というパズル操作のパターンから、カッコの対を完成させようとするプレイヤーの考え方を推定できる。

2.5　未知の考え方の発見 - 手法の特徴、その 1

以上のようなプロセスを測定・分析する「考え方」研究の手法の特徴を述べる。

前節までは、図 4 や図 6 のように、先ずデータに基づくパターンの発見があり、それに基づいて「構文を整えている」といった解釈をあてた。この順序で研究や指導を進められることが、この手法の特徴の 1 つである。このため、未知・無名の「考え方」、予期せぬ考え方を漏らしにくいと期待できる。予め期待した「考え方」が発現したか否かを判定するのとは、異なるアプローチを取れる。

2.6　試行錯誤 - 手法の特徴、その 2

試行錯誤について、正誤のはっきりするジグソー・コードの例を題材に説明する。図 7、図 8、図 9 は、平均を計算する図 5 から発展させて、分散を計算するパズルである (ただし、プログラミング言語が、Java から JavaScript に変わっている)。図 7 がパズル開始直後、図 8 が途中、図 9 が完成形で正解である。

図 8 と図 9 の違いは矢印で示した行で、変数 average の宣言かつ代入するところである。

これはいわば「ひっかけ問題」であり、このプレイヤーは途中で引っかかったが、最終的に正解に達した。文法としては変数はどこでも宣言できるし、宣言せずに使うこともできる。しかし、変数を、関数の最初でまとめて宣言するという作法やコーディング規則を採

用する場合がある。図 8 のプレイヤーは、その作法を踏まえて変数 average の宣言を関数の最初に配置したと考えられる。これでも文法的に正しいし、実行してもエラーにはならないが、計算結果は間違っている。変数 average を計算するには、データの和をデータ数で割る必要があり、データの和の計算の後に置かなくてはならない。図 9 の位置に置くのが正しい。

プレイヤーがひっかかるのは、変数の宣言場所について作法、いわば定石を身につけているからである。このこと自体はスキルの一部としてプラス評価できるし、引っかかったことはそのスキルの現れである。また、ひっかかった後で修正できたのは、自分の答えを見直して誤りに気づけたからであり、見直せること自体もスキルの一部としてプラス評価できる。

このように、プロセスを測定・分析する手法では、試行錯誤の内訳を検出できる。既知の考え方を元に引っかけ問題を作って出題し、解答がひっかかった結果になっているかどうかを評価する手法では、これらを検出するのは困難である。

もちろん、ひっかからずに正解に直進するプレイヤーもいるであろう。試行錯誤の検出を日々の指導に活かすことは想像できるが、これらを試験や成績に反映するためには多くの研究が必要であろう。

3. スクリーン・リーダーで操作する UI

われわれは、スクリーンリーダーに対応した UI を開発した (図 10、図 11)[4]。ドラッグ＆ドロップを使う Web アプリケーションは、そのままではスクリーンリーダーで操作できないが、代替 UI を用意すればスクリーンリーダーによって画面を見ずに操作できる。

この UI は、iOS(iPhone や iPad) のスクリーンリーダー VoiceOver に対応している [5]。スクリーンリーダーには、Web 画面を端から読ませることもできるし、タップした部分を読ませることもできる。タッチ・スクリーンの画面なのに、タップした部分を読ませられるとはどういうことか？VoiceOver が起動すると、1 回のタップで読み上げ、ダブルタップで実行という具合に操作の仕方が変わる。タップしながら画面上を探して、リンクなどを見つけたらダブルタップで実行するのである。

図 10 はピースの移動を開始する前の状態、図 11 は移動中である。前図で「オレオレ詐欺には、大きく分けて 2 つの種類がある。」をタップして選んだ直後が、後図の画面である。「オレオレ詐欺には…」というピースの前に「移動するピース:」というテキストが挿入されていて、読み上げると、これが移動中のピースであることが分かる。ピースの間には「移動先 先頭」や「移動先 1」といった疑似ピースが挿入されていて、これらが移動先に該当する。移動先の疑似ピースをタップすることで、先に選んだピースがそこへ移動する。これを繰り返すことでピースを並べ替える。

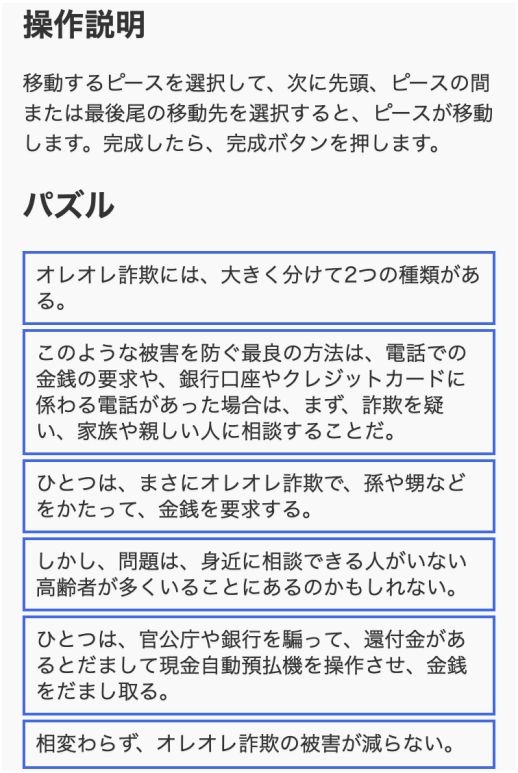

図 10　スクリーンリーダー向け UI: 移動するピースを選択し次に移動先を選択して並べ替える

「移動するピース: 」などを含めて、スクリーンリーダーが読み上げるので、画面を見なくても何が表示されているかが分かる。この UI によって、画面を見ずにパズルを解くことができる。

図 10 の画面でピースをタップして選択することが、図 1 でのドラッグ開始、すなわちピースの移動開始に該当する。図 11 で「移動先 5」などをタップすることがドロップ、すなわち移動完了に該当する。これに応じて、2.2 節や図 2 と同様に操作が測定される。そして、2.3 節や 2.4 節と同様に分析できる。

図 1 と、図 10 および図 11 とは、操作の仕方、すなわち UI が違うだけで、並べ替えるピースは同じである。アプリの実装としても、全く同じパズル問題データに、異なる UI をあてている。

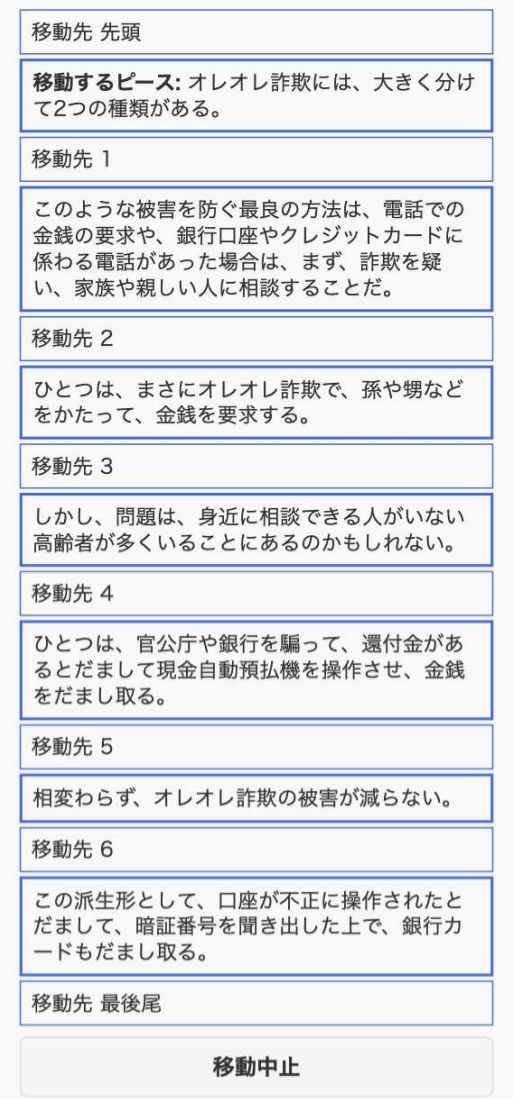

図 11　スクリーンリーダー向け UI: 移動するピースを選択した状態

3.1　別の UI

スクリーンリーダーで操作できる、図 11 とは別の UI も考えられる。実装はしていないが、この別 UI による並べ替え操作の測定・分析を考察することで、UI の要件を検討する。

図 12 の UI では、プレイヤーは移動するピースを選択して、画面下部の「ひとつ上に移動」ボタンを押して 1 つ前 (上) へ、「ひとつ下に移動」ボタンを押して後ろ (下) への移動させる。この 1 つずつの移動を、意図した位置に達するまで何回か繰り返せば、所望に位置へピースを移動できる。

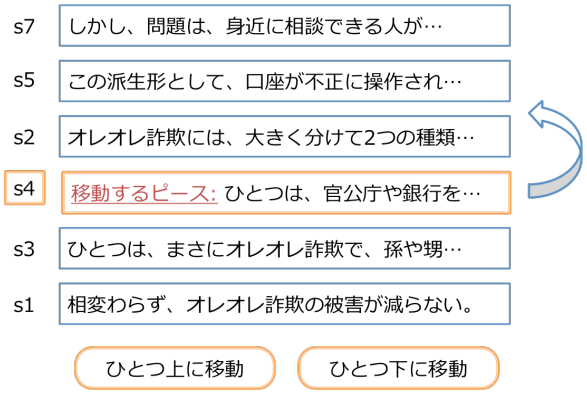

図 12　別の並べ替え UI: ピースを選んで、ボタンを押してひとつ上/下に移動する

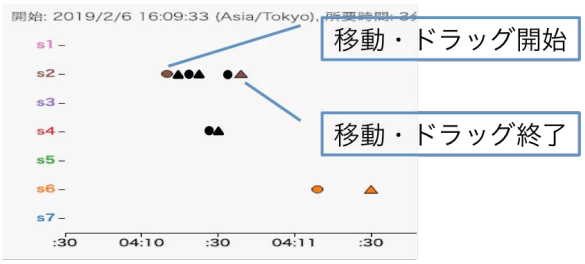

図 13　別の並べ替え UI による並べ替え操作の時系列散布図

3.2　要件検討

代替 UI では、図 1 や図 11 と異なり、「移動するピースを選んで、移動先を選択する」の 1 回で移動開始から移動完了が完結しない。また、s4 を 1 つ上に移動するには、s4 を選んで上に移動する方法と、s2 を選んで下に移動する方法の 2 つがある。

この「別の UI」でプレイヤーが解く様子は、図 13 の時系列散布図ようになるであろう。ピースを動かす判断の違いを図 4 の場合と比較するためには、どの丸○がユーザの移動開始の判断に該当し、どの三角△が移動完了に該当するのか、不確かさを含んで推定しなくてはならない。図 1 および図 11 の UI では、図 3 のように、移動開始と移動完了の時点は明確である。

すなわち、考え方・解き方の違いを比較するためには、測定できるデータから比較したい事象を検出できるように UI を設計・実装する必要がある。この考察で比較したように、移動する判断の早さを比較するためには、移動開始と移動完了を明確に判定できる UI が求められる。ドラッグ&ドロップは一瞬の操作に思えるが、2.2 節「測定データ」で、ドラッグとドロップを区別して記録している理由もこれである。

また、この UI 設計の要件は、ドラッグ&ドロップ

UIでは「パズルの解き方・パズルを解く考え方において、ドラッグ途中の軌跡には意味がない」という判断を前提としているとも言える。

4. まとめ

「考え方」の研究において、Webアプリケーションとして実装した文章やプログラム・コードのジグソー・パズルを題材に、見て並べ替えるUIと聞いて並べ替えるUIとでプレイヤーの考え方が異なるかどうかを比較するとき、2つのUIが備えるべき要件を検討した。考え方・解き方の違いを比較するためには、測定できるデータから比較したい事象を検出できるようにUIを設計・実装する必要がある。

今後は、本稿のように問題を解く過程を測定・分析することで、UIの違いによって考え方が異なるかどうかを検証する。同じ問題データで、解答や正答率が同じでも、UIによって考え方が異なる場合があると判明するかもしれない。解答に達するために、異なるスキルが発揮されたとも考えられる。その場合でも「同じ問題を解いた」と言えるか、検討することになるであろう。

また、本稿の手法で検出した「考え方」の評価も、将来は課題となるだろう。試行錯誤の結果ではなく、試行錯誤するプロセスそのものをどのように評価すべきか、多くの研究が必要となるだろう。

謝辞

本研究はJSPS科研費17K01085の助成を受けたものである。

文献

[1] 久野靖, 思考力・判断力・表現力を評価する試験問題の作成手順, 情報処理学会 情報教育シンポジウム論文集,2018(1),1-8, 2018
[2] 山口琢, 小林龍生, 高橋慈子, 大場みち子, パズル操作の測定・分析による思考の推定, 日本認知科学会第35回大会, sP2-17, 2018
[3] 川北紘正, 大場みち子, 山口琢, プログラミング思考過程に基づくプログラミング時の行動分析と傾向, 情報処理学会 第81回全国大会, 2019
[4] 山口琢, 大場みち子, スクリーンリーダーで操作するジグソー・テキスト - アクセシブルな学習分析とComputer Based Testing, 情報処理学会 研究報告コンピュータと教育（CE）,2019-CE-149(15),1-8, 2019
[5] VoiceOver, iPhone ユーザガイド, https://support.apple.com/ja-jp/guide/iphone/iph3e2e415f/12.0/ios/12.0