修士論文

小学校段階のプログラミング的思考を操作として展開・評価する学習環境の構築

小林　祐太

主指導教員　　長谷川　忍

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学）

令和　3 年 2 月 3 日

Abstract

In Japan, programming education has become compulsory in primary education from 2020. The background of this effort is that the government promotes the growth of "programming thinking" acquired in the process of learning programming. Programming thinking is "the ability to think logically in order to realize a series of activities intended by oneself" and is expected as a problem-solving ability in an unpredictable society in the future.

However, one of the problems is that the evaluation method of programming thinking is not clear. This is because programming thinking is implicit, and it is difficult to observe its level and growth from the outside. Therefore, discussions have continued to this day regarding the development goals and evaluation criteria for programming thinking. In fact, Koizumi organized programming thinking into six components based on computational thinking practiced in the United Kingdom in order to formulate evaluation criteria for programming thinking. Furthermore, Benesse positioned the six elements proposed by Koizumi as learning goals for programming education.

In the conventional evaluation method, since the invisible concept of programming thinking is captured, the activities based on the programming thinking mentioned above are evaluated according to the achievement level set by the researcher. For example, Nishino created and operated a rubric that performs an objective evaluation in three stages in order to capture thoughts during programming learning based on Koizumi's evaluation criteria. Saito proposed the rubric "Rubric Pro EEs" based on a four-grade evaluation as an evaluation index for the entire programming education. However, in each case, the position is limited to the standard of achievement, and the level of programming thinking has not been quantitatively evaluated.

On the other hand, there is also a problem with training in programming education. According to the "Results of Survey on Efforts for Elementary School Programming Education by the Municipal Board of Education" published by the government in January 2020, the training of programming education for teachers nationwide was not wholly conducted. The cause is that the teaching methods and teaching materials are not sufficient. The programming education introduced in elementary school does not focus on acquiring programming abilities using programming languages such as the so-called C language but seeks to develop logical thinking in various subjects. Since it is clearly distinguished from programming ability, it is difficult to apply the technical education provided in high school and above as it is. In addition, although there are places where

programming education is researched in primary schools, it is difficult to collect information because the number is limited.

The purpose of this study is to build a learning environment in which the ability of programming thinking is evaluated by "formative evaluation" in order to grasp the degree of programming thinking during learning of elementary school children. Formative evaluation evaluates the learner's current achievement level and is one of the policies to switch the learning method depending on the evaluation. Based on this formative assessment, it is necessary to extract the process of thinking in order to grasp the degree of programming thinking. Therefore, we externalized thinking as an operation and tried to quantify the content of the operation during the exercise. Specifically, we have developed a system for extracting activities based on thoughts. As a teaching material that can be used even by teachers who do not have a clear understanding of programming education and learning methods, we aimed to create an environment where learners can surely demonstrate programming thinking by constructing the actual programming learning process in the system.

This research aims to capture the thinking when constructing the control structure in programming and capture the activities based on the elements of programming thinking that appear every time the problem is solved throughout the programming learning. It deals with learning from the design stage to looking back on the coding results and records the operations performed during the learning during that time.

In order to develop, we thought about how to acquire operations from learning based on programming thinking.

First, in order to capture the thoughts that occur during programming learning in detail, we defined six "thinking activities" that are the operations of the objects to be observed from Benesse's evaluation criteria. The defined thinking activities are "division", "abstraction", "generalization", "ordering", and "analysis".

Next, we set the learning phase assuming that these activities occur in actual programming learning. The learning phase was created in three stages: "design," "development," and "evaluation," with reference to the learning process that incorporates the concept of software design.

Finally, we designed a learning task called "thinking task" with four patterns to acquire the operation. Thinking tasks are "inference tasks" that capture abstraction and ordering, "division tasks" that capture division, "control tasks" that capture control, and "analysis tasks" that capture generalization and analysis. Then, the number of simple operations in the thinking task was defined as "quantity data," and the data calculated from the answer

results were defined as "quality data," which was positioned as an index of the ability of programming thinking in Thinkron.

Based on these definitions and designs, we created a web application called "Thinkron" that can acquire learner operation logs. It is based on JavaScript and has a UI that only clicks and drags and drops the mouse to use it easily. The task execution screen is a procedural maze search game in which the procedure is easily reflected as a result.

In order to evaluate Thinkron, we used "Rubric ProEEs" created for programming education and experimented to see if there was a correlation between Thinkron's score and rubric evaluation.

As a result, a correlation was found in all thinking activities except "control" of thinking activities. This suggests that thinking activities with correlation are effective as an index for estimating the ability of programming thinking. No significant difference was observed only in the "control" of thinking activity because it was not possible to capture deductive thinking such as conditional branching during coding. Therefore, it will be necessary to consider how to extract the target code and how to score it in the future.

In addition, the effect of improving the division ability was recognized through learning using Thinkron. There are few examples of learning materials that are conscious of the design side, and the learning effect of Thinkron in this design phase will be positioned as one of the learning examples that capture the design stage that is effective for programming education. After this experiment, we also conducted a sensitivity evaluation questionnaire about the usability of Thinkron. As a result of investigating the fun of Thinkron, the difficulty of asking questions, and the difficulty of operation, it was found that the operability was easy for children to accept.

目次

図目次

表目次

第1章 はじめに

1.1 研究背景

1.1.1 プログラミング的思考と思考評価

平成 29 年に発表された新学習指導要領により，令和 2 年度から初等教育においてプログラミング教育が必修化された[1]．この取り組みの背景には，プログラミングを学習する過程で獲得される「プログラミング的思考」の育成を政府が推進していることが挙げられる．プログラミング的思考とは「自分が意図する一連の活動を実現するために，どのような動きの組み合わせが必要であり，一つ一つの動きに対応した記号を，どのように組み合わせたらいいのか，記号の組み合わせをどのように改善していけば，より意図した活動に近づくのか，といったことを論理的に考えていく力」のことであり，今後の予測不能な社会における問題解決能力として期待されている[2].

しかし，必修化における問題の一つにプログラミング的思考の評価方法が明確でないことが挙げられる．これはプログラミング的思考が暗黙的であり，そのレベルや成長を外から観測することが難しいためである．そのため，文部科学省の提唱するプログラミング的思考と，その原点とされる「コンピュテーショナル・シンキング（計算論的思考）」のそれぞれの観点から，日本の初等教育で取り扱うプログラミング的思考の育成目標や評価規準について議論が重ねられてきた．実際に小泉らは，プログラミング的思考の評価規準を策定するため，英国で実施されていたコンピュテーショナル・シンキングに基づき，文部科学省の提唱するプログラミング的思考を 6 つの構成要素で整理した[3]．さらに，Benesse は文部科学省が定めたプログラミング教育における「思考力・判断力・表現力等」において，小泉らが提唱する 6 要素を「プログラミング教育を通じて目指す育成すべき資質・能力」の目標として位置付けた[4].

従来の評価手法においては，プログラミング的思考という目に見えない概念を捉えることから，前述したプログラミング的思考を根幹とした活動について，研究者が定めた達成度による評価が行われている．例えば西野らは，小泉らの評価規準を基にプログラミング学習中の思考を捉えるべく，3 段階の客観的評価を行うルーブリックを作成し，運用した[5]．また，齋藤らはプログラミング教育全体における評価指標として 4 段階評価によるルーブリック「Rubric ProEEs」

を提案している[6]．ただし，いずれにおいても達成度の基準による位置づけにとどまっており，プログラミング的思考のレベルを定量的に評価するまでには至っていない．

1.1.2 プログラミング教育における学習環境問題

プログラミング学習を推進するための教材は，現在も研究開発が進められており，文部科学省からはプログラミング的思考の育成について触れた，プログラミング学習を導入するための研修教材が提供されている[7]．しかし，令和 2 年 1 月に文部科学省より公表された「市町村教育委員会における小学校プログラミング教育に関する取組状況等調査の結果について」では，全国の小学校教員に対するプログラミング教育の研修は，当時の段階で完全には行われていないことが報告されており，少なくとも各校に一人以上の研修済み・研修予定の教員がいる市町村の割合は，すべての都道府県の中でおよそ 6 割程度の実施状況にとどまった[8]．

公式的な研修教材が提示されているにもかかわらず，研修に対する実施状況が完全でないことについて，大きく 2 点の要因が考えられる．

一つはプログラミング教育の必修化と同時に英語，道徳の授業科目化が行われた点である[1]．従来のカリキュラムにおける「外国語活動」,「道徳の時間」の内容が科目化することで，新たに評価の対象が増えた．しかし，従来のカリキュラムの置き換えとして導入されることから，プログラミング教育という新しい学習概念を取り入れることは，実践的な研修だけでなく知識や理解にも時間がかかると考えられる．その点から，英語，道徳に比べプログラミング教育の導入のハードルを高く捉えている可能性がある．

もう一点として，小学校段階でのプログラミング教育における指導例，教具・教材例が十分でないことが挙げられる．小学校において導入されるプログラミング教育は，いわゆる C 言語などのプログラミング言語を使ったプログラミング能力の習得を主とするわけではなく，前項で述べたプログラミング的思考を育む内容を教科横断的に実施しようとしている[2]．あくまでプログラミング的思考を育むことに注力し，明確にプログラミング能力とは区別しているため，高等学校以上で行われる技術的な教育をそのまま応用することは困難である．また，先行実施校で研究的にプログラミング教育が実施されている所もあるが，数は限られており，多くの現場の教員においては研修外での情報収集は難しいと

考えられる．令和 2 年 3 月，一般財団法人 LINE みらい財団の全国の小学校教員 618 人を対象にしたプログラミング教育におけるアンケート調査(図 1.1)では，「プログラミング教育の授業を通じた評価の仕方がわからない」が全体の半数以上を占め，次いで実践的な授業の進め方や教材の選出等が定かでない教員がいずれも 4 割以上いることが判明している[9].

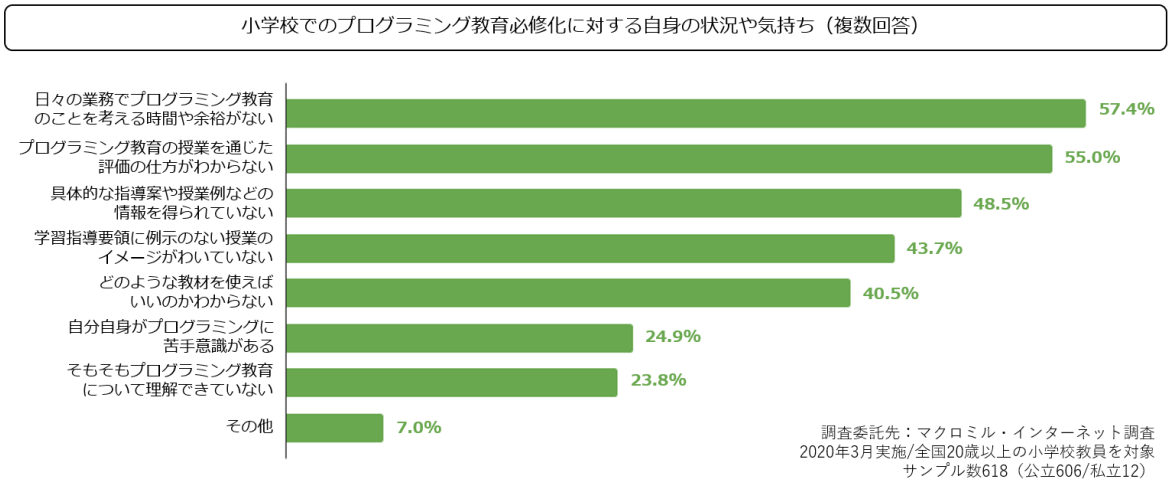


図 1.1：プログラミング教育に関するアンケート（LINE みらい財団）

加えて 2020 年に日本国内で流行拡大した COVIT-19 の影響もあり，令和 2 年 4 月以降から全国の多くの教育機関が臨時休校の措置をとる事態となった[10]. このことから小学校においても新体制の適応に遅れが生じたのではないかと考えられる．このような環境変化や本項で述べた問題より，プログラミング的思考の概念を捉えた実践的な学習環境については，少なくともプログラミング教育の開始直前では整っていない状況であることが伺える．

1.2 本研究の目的

本研究では，小学校の児童の学習中のプログラミング的思考の程度を捉えるため，プログラミング的思考による能力を形成的に評価する学習環境の構築を目的とする．形成的評価とは学習者の現時点における到達度や能力の程度について評価するものであり，その良し悪しは，学習方法を切り替える一つの指針となる．そのため，形成的評価を繰り返し行うことで学習の過程を評価することが可能となり，最終的な総括的評価の得点を裏付けるものとなる．この形成的評価に基づき，プログラミング的思考の程度を捉えるためには，思考の経過を抽出す

る必要がある．そこで，思考を操作として外化させ，演習中の操作の内容に対して定量化を試みる．具体的には思考に基づく活動を抽出するためのシステム開発を行う．

また 1.1.2 項より，プログラミング教育に対する理解や学習法が明確でない教員であっても利用可能な教材として，システム内に実際のプログラミング学習のプロセスを構成し，学習者がプログラミング的思考を確実に発揮できる環境を構成する．

1.3 本論文の構成

本論文の構成は以下のとおりである．

- 第 1 章　はじめに
  本研究の背景，目的および本論の構成について記述する．
- 第 2 章　関連研究
  本研究の関連研究および本研究の位置づけについて記述する．
- 第 3 章　提案手法
  本研究の目的を達成するための提案手法について記述する．
- 第 4 章　システムデザイン
  本研究の目的であるシステムの設計指針，開発方針について記述する．
- 第 5 章　開発
  本システムの目的であるシステムの開発経過について記述する．
- 第 6 章　実験
  本システムが出力するデータにおける評価実験について記述する．
- 第 7 章　おわりに
  本研究のまとめと展望，課題について記述する．

第2章 関連研究

2.1 プログラミング学習の流れ

文部科学省の「小学校プログラミング教育の手引き」によれば，問題点を見出して，問題解決に至る間のステップとしてプログラミング的思考が位置付けられており，問題の解決プロセスが例として挙げられている[11]．ただし，定型的なプログラミング学習の流れは明示されておらず，各小学校の方針に応じて指導計画が行われている状態である．

一方，中等教育ではあるが，大村はソフトウェア設計の考え方を取り入れたプログラミング学習のプロセスを考案している[12]．大きな特徴としては，目的のプログラムを構築するにあたり，設計のプロセスも評価した学習体系にするという点である．そもそもコーディングを行うには，設計の段階で仕様やUIなど，多くの取り決めを定め，導線を敷く必要があるため，設計の過程をスキップすると致命的なエラーやバグの原因となる．このように設計のプロセスは，プログラムへの具体化に必要な工程であり，設計段階から論理的な思考が働くことは容易に想像できる．

しかしながら，プログラミング学習支援システムの多くは開発と評価の部分にフォーカスが当てられており，設計部分を意識させることが少ないのが現状である．例えばロボットの動作を出力とするプログラミング教材では，メインの活動としてコーディングを扱うものの，直接ロボットの動きを確認しながらコーディングを行うトライ＆エラーを主体としているため，これらの教材のみでの設計部分の意識づけを行うことは難しい．

2.2 プログラミング学習支援システム

プログラミング学習の支援システムの現状として，これまでは学習塾や家庭学習向けのユーザ向けコンテンツとして開発されてきたものが多かったが，近年は教師が教具として用いることを想定したシステムへとシフトしてきている．例えば，大日本印刷の「SWITCHED ON Computing 日本版」では，プログラミング的思考を小学校低学年から段階的に定着させることを目的に，発達段階を考慮したプログラミング学習教材をユーザに提供しており，教員に対してはプログラミングの経験のない教師であっても対応できるよう，学習到達目標など

を細かに設定した指導書・指導案を提供している[13].

また, プログラミング学習下においては, 学習中の理解度を捉える補助システムも検討されている. Matayoshi らは学習者のプログラミング中の制御構造の抽象的な理解度を把握, 支援するため, 自然言語で記述できるエディタ内のコメント機能をツリー構造として再現し, 操作可能にしたプログラミング学習支援システムを制作している[14].

2.3 プログラミングの自動評価システム

プログラミングを行った操作の結果より, 学習中の思考を評価しようと試みるシステムも存在している. 太田らは, プログラミング的思考の原点とされるコンピュテーショナル・シンキングの概念について, 実際に学習者の構成したプログラムから発現された操作から自動評価システムを開発している[15].

また, 國宗らはプログラムを作成する上での操作や制御構造を自動評価するシステムを作成しており, 教員の作成したテストケースより, 学習者が組み上げたコードの達成度を可視化させている[16].

2.4 本研究の位置づけ

本研究の位置づけとしては, プログラミングにおける制御構造の構築時の思考を捉えるだけでなく, プログラミング学習全体を通して, 課題解決の度に発現するプログラミング的思考の要素に基づく活動を捉えようとしている. つまり, 設計の段階からコーディングの結果に対する振り返りまでの学習について扱い, その間の学習中に発揮された操作を記録していく. これを実現するには, プログラミング的思考の要素に基づく学習中の活動を定義し, 各要素に基づく活動が観測できるような場面を形作る必要がある. このプログラミング的思考の要素に基づき学習者が取り組む活動を, 本論では「プログラミング的思考の外化」とする.

本研究ではプログラミング的思考の外化を実現するシステムを開発するため, 要素に基づく活動（思考活動）の定義, プログラミング学習の流れ（学習フェーズ）の設定, 活動を捉える課題（思考課題）のデザインを行った. なお, 思考課題との混同を避けるため, 学習フェーズを通して行われる問題解決のための題材を「問い」と定義する.

最終的な学習支援システムとしては, 教師がプログラミング学習を進める上

での，一つのテンプレートとして学習環境を展開し，中でもプログラミング的思考における思考能力の程度を把握するための支援を行う．先行研究にならい，設計と開発のそれぞれに根差したアプリケーションをデザインし，学習者に対するプログラミングの補助として，制御構造を視覚的に理解できるような自然言語を用いたエディタを提供する．

第3章 提案手法

3.1 学習環境構築の全体構想

関連研究を基に，プログラミング的思考を観測するための学習環境の構築を試みる．全体的な構想を図 3.1 に示す．

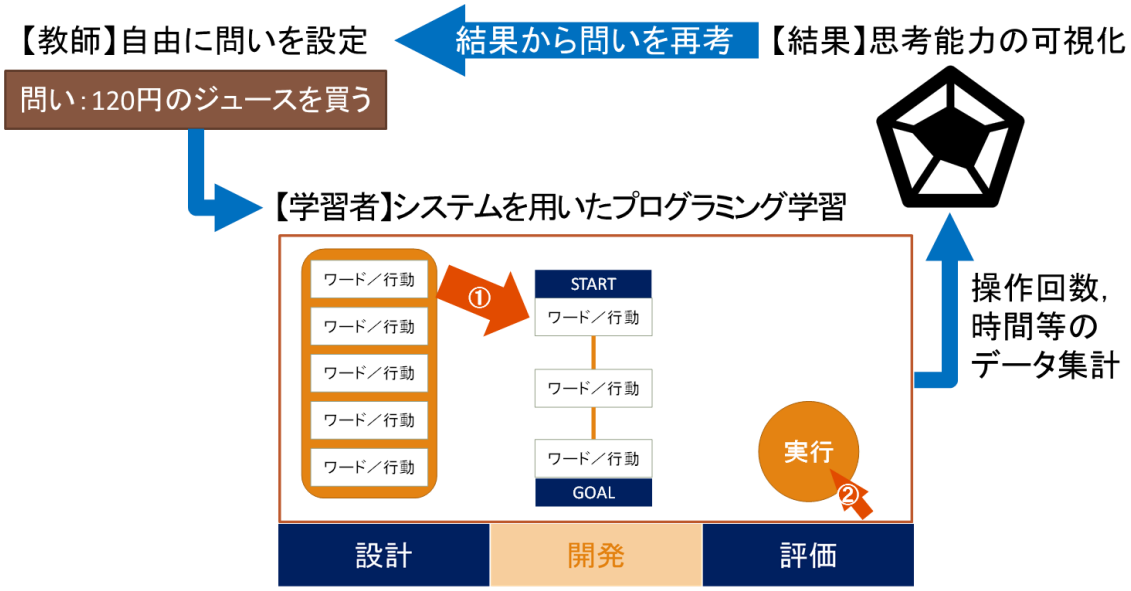


図 3.1：学習環境の全体構想

流れとしては，教師が対象となる問いを設定し，学習者はプログラム作成の体系に沿った学習支援システムを通じて，プログラミング学習を行う．この学習中の操作を記録し，量データ（操作回数など）や質データ（正答数など）からプログラミング的思考の能力評価を行う．得られた評価から教師は問いを再考し，足りない思考の要素を伸ばしていくことで，学習の循環を構成する．

本論においては，この全体構想における思考の観測部分，いわゆる学習支援システムの開発を主体として取り上げていく．

3.2 プログラミング的思考の外化プロセス

プログラミング的思考に基づく活動を観測する思考観測基盤の構築のため，プログラミング的思考の外化プロセスを考えていく．まず，プログラミング学習中に発生する思考を詳細に捉えるため，表 3.1 の Benesse の定めたプログラミン

グ的思考の教育目標より，6 つの構成要素に基づく学習中の活動を「思考活動」として定義し，表 3.2 にまとめた．思考活動は，後述のシステムを利用した学習を通じて発生する操作を想定しており，この活動における頻度を児童のプログラミング的思考に対する形成的評価の指標として捉える．

表 3.1：プログラミング的思考の評価規準

構成要素	教育目標
論理的に考えを深める	コンピュータの動きを自らの問題解決で使うために論理的推論を行う
動きに分ける	大きな事象を解決可能な小さな事象に分割する
記号にする	分割した事象から適切な側面・性質を抜き出す
一連の活動にする	記号（動き）の類似部分を特定し，別の場面でも利用できる内容にする
組み合わせる	目的に合わせてよりよい手順を創る
振り返る	目的に対する評価の観点を考え，結果が意図した活動に近づいたか評価する

表 3.2：本研究における思考活動の定義

思考活動	本研究における定義	Benesseの評価規準
分割	大きな事象を解決可能な事象に分割する．	動きに分ける
抽象化	事象・概念から必要な側面や性質，要点を抜き出す．	記号にする
一般化	事象から類似パターンを見つけ規則化する．	一連の活動にする
順序立て	動作や組み合わせの順番を考える．	組み合わせる
制御	コードによる動作を理解し，論理的推論の下でコーディングする．	論理的に考えを深める
分析	実行結果が，意図した活動に近づいたかどうか判断する．	振り返る

表 3.2 における「分割」,「抽象化」,「一般化」,「順序立て」,「分析」は，開発するシステムにおける学習中の操作を想定しているため，多少の文脈の違いは

あるが，文意は Benesse の教育目標と同義である．「制御」についても同様であるが，今回は実際のコーディングを通じて観測することを想定して定義している．

3.3 学習フェーズの設定

次に「学習フェーズ」を設定し，最も観測されやすい思考活動を各フェーズ内に位置付けた．学習フェーズは前述した大村のソフトウェア設計の考え方を取り入れた学習プロセスを参考に,「設計」,「開発」,「評価」の 3 段階で構成した．また評価フェーズは，学習者の設計，開発のそれぞれのフェーズに対して，振り返りを行うことを目的としており，学習フローとしては，設計，評価，開発，評価の順となる（第 4 章参照）．

表 3.3 は学習フェーズの定義と思考活動の位置づけをまとめたものである．フェーズを設定することにより，初等教育での授業設計において，本研究で提案する学習環境を導入する際に適応しやすくなると考える．

表 3.3：学習フェーズの定義と思考活動の位置づけ

学習フェーズ	観測する思考活動	定義
設計	抽象化 順序立て 分割	目的のプログラムを実現するための仕様を策定するフェーズ
開発	制御	設計に基づいた仕様をプログラムとして具体化するフェーズ
評価	一般化 分析	設計，開発フェーズでの振り返りや，結果の概念整理を行うフェーズ

設計フェーズはプログラムを作成するための方針を固めることを目的とし，出題された問いを解決するための必要な行動を抽出したり手順を推定したりすることを主体とした学習フェーズである．そのため，抽象化や順序立てといった思考活動が主に観測される．また，直接的にコードの制御を行う前段階として位置づけられるため，推測した手順をコードとして分解する手立て（分割）も設計フェーズとして含む．

開発フェーズは抽象化された方法を詳細なコードに変換してプログラムを構

築するフェーズであり，制御が観測する思考活動の対象となる．

評価フェーズにおいては，設計や開発の結果に対する振り返りを実施し，取り組んだ内容の整理を行うフェーズであり，分析や一般化を主体として行う．

3.4 思考課題のデザイン

学習フェーズに対応するように，思考活動を捉えるための 4 つのパターンの「思考課題」を設計した．思考課題はシステム上で取り組まれる課題であり，操作の時間や回数など，学習者の思考活動を操作として検出する．図 3.2 は思考課題における設計イメージである．

推論課題は「抽象化」と「順序立て」を捉える設計フェーズの思考課題であり，与えられた問いを解決するために，いくつかの行動が書かれたオブジェクトの中から必要な行動を抽象化し，それらを並び替えることで目的の順序の組み立てを行う．分割課題は「分割」を捉える設計フェーズの思考課題であり，推論課題で抽出した各行動をより詳細度の高い“動作”に分解する．ここでの“動作”は，次点の制御課題で扱われるコード（最小単位の動き）を組み合わせることで，再現可能な事象とする．制御課題は「制御」を捉える開発フェーズの思考課題であり，関数等を用いながら制御構造を捉えた開発を行う．分析課題は「分析」と「一般化」を捉える思考課題であり，設計と開発の各フェーズにおいてデバッグを実施するほか，開発フェーズでのコードの整理などを行う．ただし，操作の内容は直前の課題と同様である．これらの思考課題と学習フェーズ，思考活動の位置づけを図 3.3 に示す．

課題中に扱う題材については，文部科学省が定める「プログラミング教育を通じて目指す育成すべき資質・能力」について，知識・技能として「身近な生活でコンピュータが活用されていることや、問題の解決には必要な手順があることに気付くこと」[2]と定められていることから，本研究では手順が結果として反映されやすい手続き型の探索ゲームのような結果表示を行う．

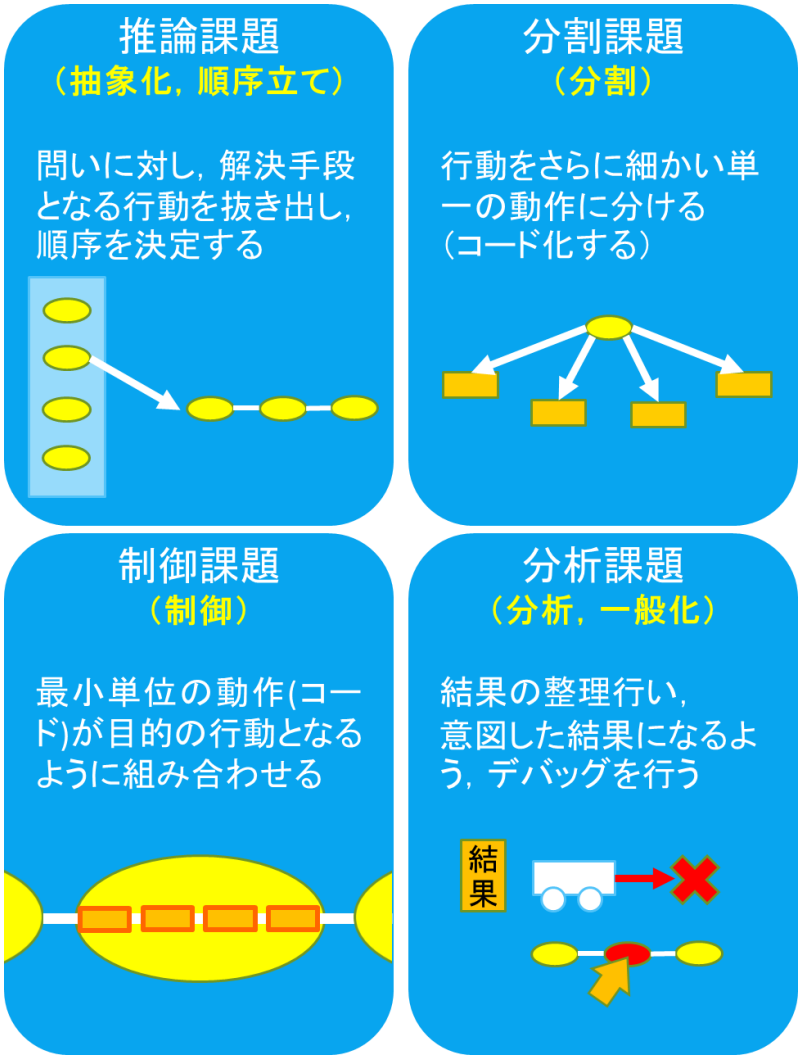


図 3.2：思考課題イメージ

	プログラミング学習		
フェーズ	設計	開発	評価
思考課題	推論課題 分割課題	制御課題	分析課題
思考活動	抽象化，順序立て 分割	制御	分析，一般化

図 3.3：思考課題の位置づけ

3.5 思考課題中の操作の習得と評価方法

3.5.1 評価の方針

はじめに思考課題中の評価対象について述べる．思考課題における遷移の構図としては，設計または開発フェーズ後に，評価フェーズを行うことから，設計，開発フェーズに基づく思考課題後に分析課題を行う流れとなる．前節で述べたように，分析課題は直前の課題と同様の操作を行うことから，評価フェーズにおいても抽象化や分割といった思考活動が観測される．そのため，分析課題以外の思考活動の観測については，評価フェーズが終了するまでを評価の対象とする（図 3.4）．

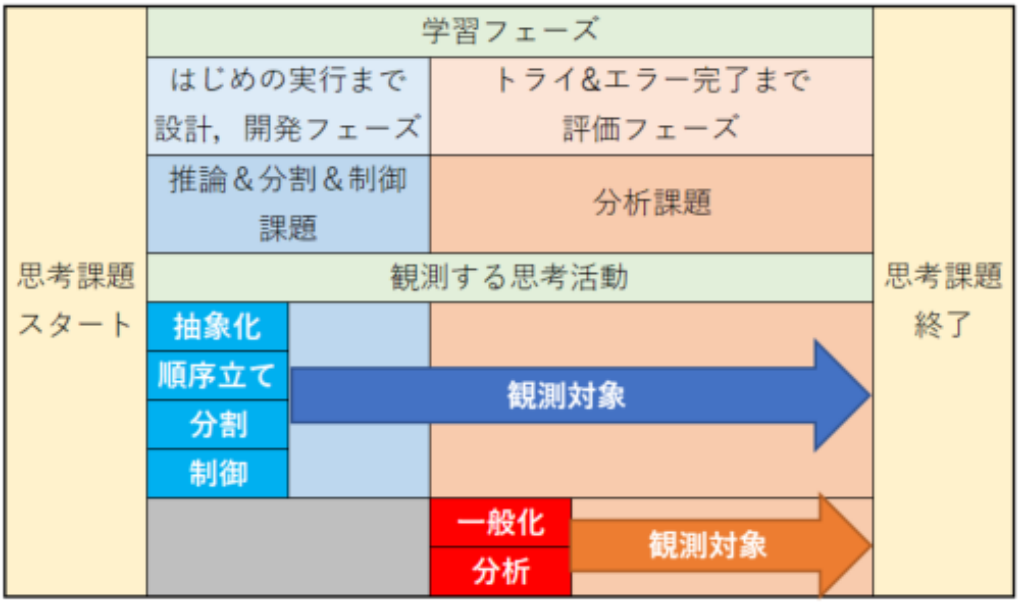


図 3.4：観測する思考活動

次に思考課題の定量化について述べる．定量化に至っては，思考課題中のインターフェースに対する操作回数を「量データ」として取得し，結果と操作回数から算出される「質データ」より思考活動の評価とする．量データは課題中にはたらく思考の発現頻度を表し，問いの難度に呼応するものと考えられる．質データは課題中に発現された能力の質を表し，問いを解決する際のパフォーマンスを捉えようとしている．

質データについては，学習者の解答の結果に対する操作傾向より数値化を行う．解答の結果については教師側があらかじめ設定した模範解答のデータと比

較することで算出を行う．学習者の操作傾向については図 3.5 のようにいくつかのパターンが考えられ，必ずしも操作数が多いことがパフォーマンスの高評価につながるとは言えない．

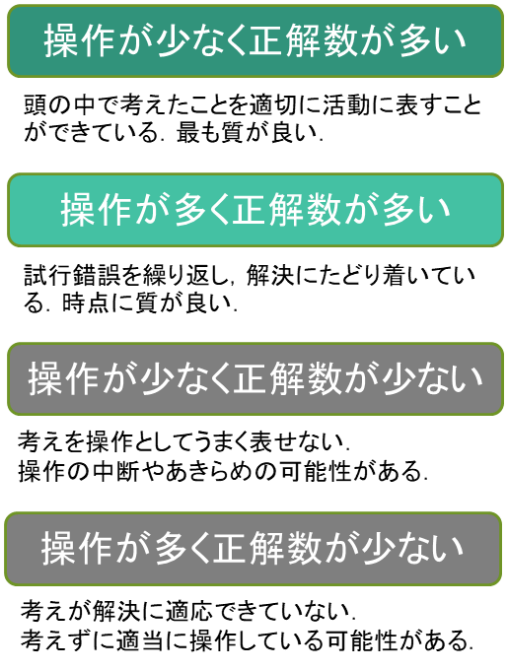


図 3.5：想定し得る学習者の操作パターン

そこで正解数を主体に，かつ，既定の操作回数に到達した時に最大のパフォーマンスであると判断し，その操作数を超えることで評価が下がるように質データを算出する．正解数を c，正解総数を C，規定値（操作結果の最適値）を s，操作回数を n とし，質データ q について(1)式に示す．

$$q = \frac{cs}{C(|s-n|+s)} \qquad (1)$$

3.5.2 推論課題中の操作

推論課題の構成イメージを図 3.6 に示す．

推論課題における具体的な操作は，あらかじめ用意した行動の名前が書かれたラベル（行動ラベル）のリストから，任意のラベルを抽出，並び替えを行うといったものである．抽象化の観測は，抽出する行動ラベルを受け取り部分であるレシーバへのセットと除去の操作，レシーバ自体の増減の操作を捉える．順序立てに関してはレシーバへのセットと除去の操作，レシーバにセットされた行動

ラベル同士の入れ替え操作を捉える．最後に「実行」を選択することで，操作に対する結果を返す．

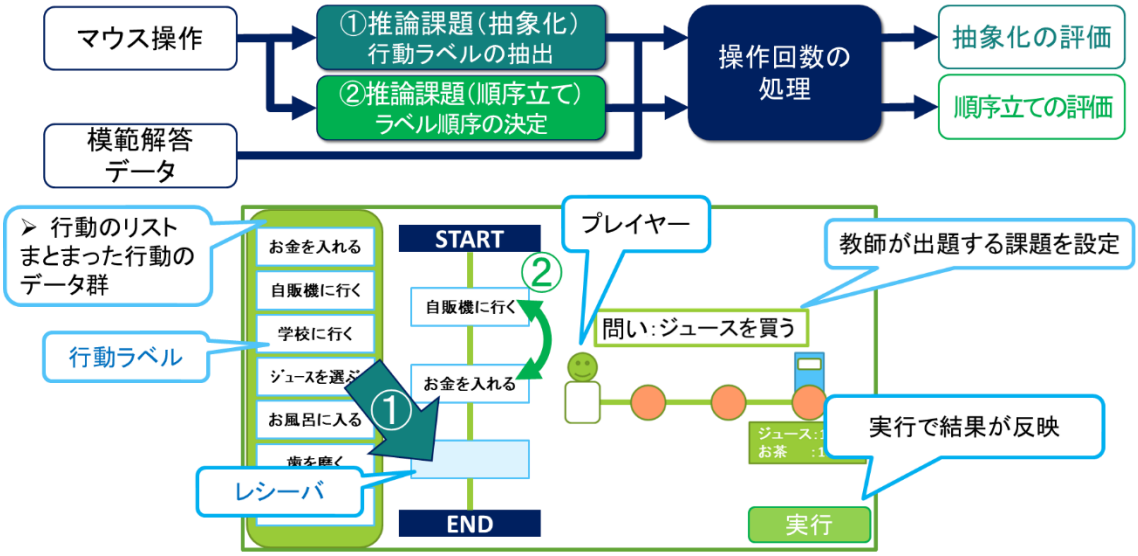


図 3.6：推論課題構成イメージ

3.5.3 分割課題中の操作

分割課題の構成イメージを図 3.7 に示す．

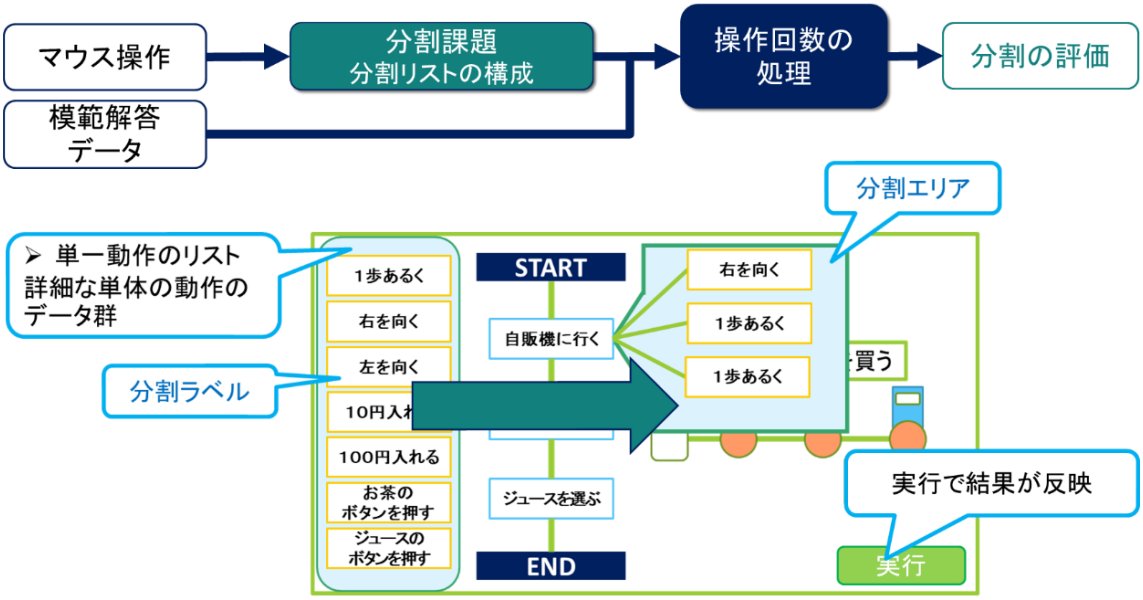


図 3.7：分割課題構成イメージ

分割課題では，推論課題で構成した行動ラベルの内容を引き継ぎ，示された行動ラベルそれぞれに対して，細かな事象に分割する操作を行う．具体的には，推論課題よりも詳細な動作の名前が書かれたラベル（分割ラベル）のリストから，任意のラベルを抽出し，行動ラベルに対応する分割エリアに，順序を含めて動作を構成するといったものである．分割の観測は，分割エリアへのセットと除去の操作，分割ラベル同士の入れ替え操作を捉える．最後に「実行」を選択することで，操作に対する結果を返す．

3.5.4 設計フェーズにおける分析課題

分析課題は各課題後に実施する振り返りの活動として行われるため，設計フェーズでは，推論課題，分析課題後，開発フェーズでは制御課題後にそれぞれ実施される．推論課題後に行う分析課題の構成イメージを図 3.8 に示す．

設計フェーズ後の分析課題における具体的な操作は，元となる思考課題の操作と同様である．ただし，思考活動の目的として，結果に対するエラー原因の考察や，目標の動作に近づいているかどうかを確かめるといった思考へとシフトするため，操作が同様な別課題として位置付けた．分析課題へシフトするタイミングは，元となる課題の「実行」を選択した直後とし，課題が完全に正答するまでの間は分析課題を継続する．分析の観測は，元となる思考課題のすべての操作を捉える．

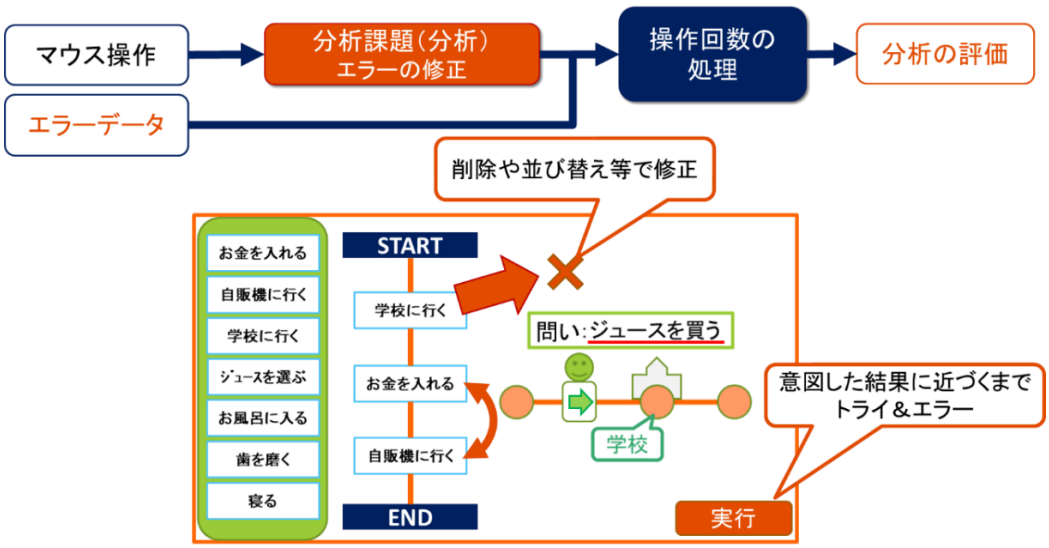


図 3.8：分析課題（推論後）構成イメージ

3.5.5 制御課題中の操作

制御課題の構成イメージを図 3.9 に示す.

制御課題では，分割課題での結果を引き継ぎ，分割した内容を基にプログラムを構成していく．具体的な操作としては，自然言語で書かれたコード（条件分岐や関数をかみ砕いた最小の動作）のリストから，任意のコードを抽出，並び替えを行うといったものである．制御の観測は，エディタへのコードのセット，除去，入れ替え操作を捉える．「実行」を選択することで，操作に対する結果を返す.

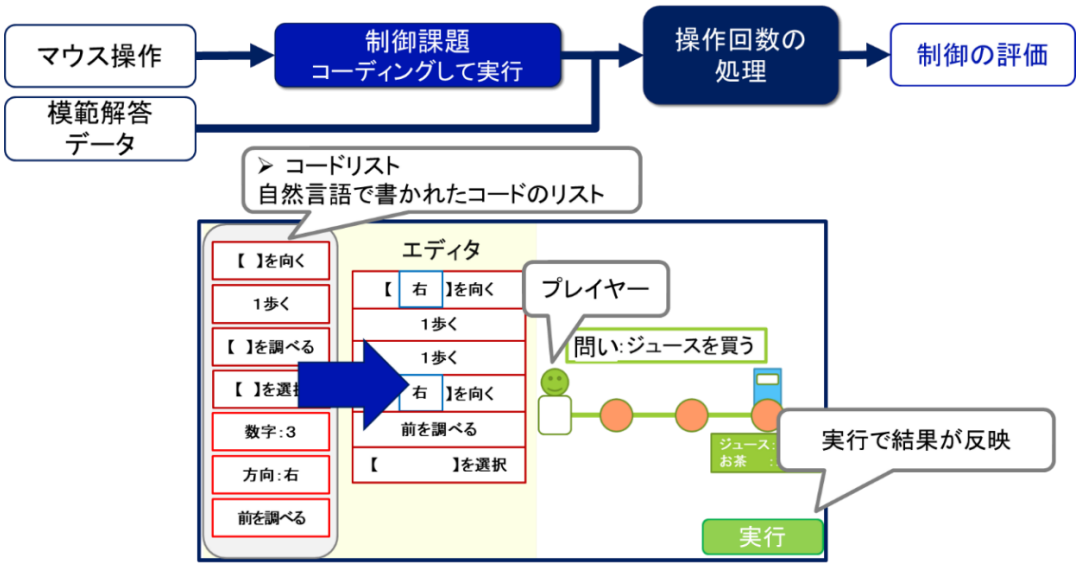


図 3.9：制御課題構成イメージ

3.5.6 開発フェーズにおける分析課題

開発フェーズにおける制御課題後に行う分析課題の構成イメージを図 3.10 に示す.

開発フェーズ後の分析課題における具体的な操作は，制御課題の操作と同様であるが，設計フェーズ後の分析と異なる点として，一般化の思考活動を捉える操作が加わっている．具体的には，正解のコードを組み終えた後に関数化を促すファンクションコードを使用させ，関数の構築およびコードに適応させるといったものである．一般化の観測は，エディタ内へのファンクションコードのセット，除去，入れ替え操作を捉える.

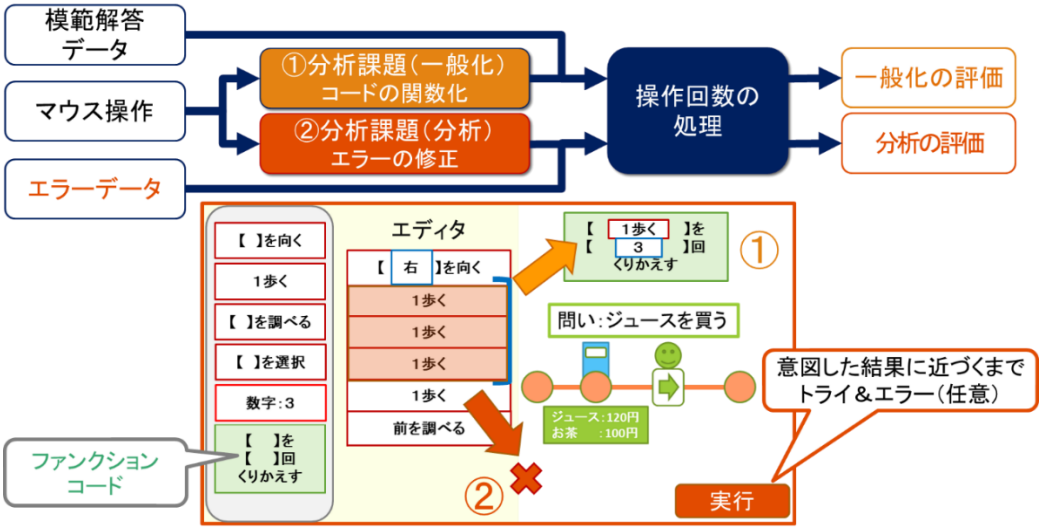


図 3.10：分析課題（制御後）構成イメージ

第4章 システムデザイン

4.1 Thinkron（シンクロン）の設計指針

プログラミング的思考の外化プロセスを反映した, 思考の育成支援および学習評価を行うシステム「Thinkron（シンクロン)」の開発を行った.

まず, 外化プロセスより実際のプログラミング学習中の操作を捉えることから, パソコンやタブレットにおける入力装置から思考活動の抽出を行う．デバイスについては各教育現場における環境の違いから複数種類に対応する必要があり, アプリケーションの導入については, 児童が操作することを考慮し, 手動によるインストールの手間を省けることが好ましい．そこで，システム自体は Web アプリケーションとして開発し，学習フェーズに則ったプログラミング学習ができるものとした.

次に要件定義として, 学習者の取り組む問いの設定を行う. 形成的評価を行うことができる学習環境の構築にあたっては，教師が学習者のプログラミング的思考の能力に対して, 問いの内容をコントロールができることが望ましい. 今回の研究においては，問いのデータを直接作成した（第 5 章）が，展望としては教師用フォームを作成し，問いの投稿や評価の閲覧を可能とする機能を付与する.

最後に UI について，タイピングによる入力操作を避け，すべてマウスのクリック, ドラック＆ドロップのみで対応できるものにした. 画面のタッチ操作が可能なデバイスについては，タップ操作およびドラッグ操作についても実装を行った.

外観は, 記号や画像を多用し, オブジェクトが具体的にイメージできるようにし，学年問わずシステムを理解できるように記載文字は学習済みの常用漢字を充てた．なお，今回は小学 3 年生以上の学習者に対応させている.

4.2 Thinkron の遷移モデル

Thinkron を具現化するために，遷移モデルについて整理した．図 4.1 に示すように, 初めに教師側が問いを設定し, 学習者は学習フェーズにしたがって思考課題に取り組む. 設計と開発のそれぞれのフェーズ後に, 目的の操作ができたかどうか振り返る評価フェーズを設け，ここでは実行結果に対するトライ＆エラーを繰り返し行わせる．完全に正答することで次のフェーズに進む完全解答形式

を採用しているため，エラーの原因やコードの整理などの振り返り学習をスキップせずに確実に行わせることができる．

全体像から見た Thinkron の強みとしては，これらの複数の思考課題を統合した UI で連続的なプログラミング学習が行えることにあり，この学習フェーズの流れから，プログラミング的思考を操作の一つ一つとして観測・評価し，学習者の伸ばすべき思考要素を取り上げた学習内容を組み込んだプログラミング学習を再度展開することができる．

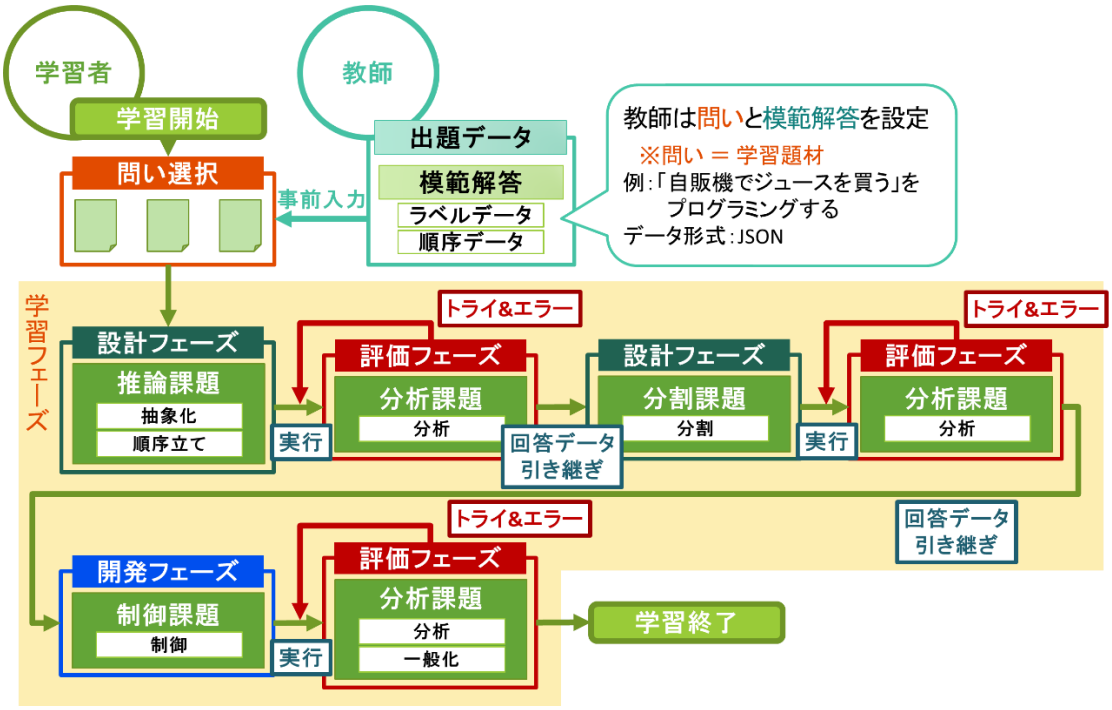


図 4.1：Thinkron の遷移モデル

4.3 思考課題の実装デザイン

思考課題の実装において，分析課題は他の思考課題の操作内容を引き継ぐため，表 4.1 のように 3 つの課題として改めた．また，実装のデザインを図 4.2 に示す．

実装の基本方針としては，ラベルやコードの一覧である「リスト」やそれらを直接配置する「エディタ画面」をブラウザの左側，実行結果を表示する「実行画面」をブラウザの右側，問題文はブラウザ上部に配置するようにしている．「実行」（以降，実行ボタン）やその他画面上の制御に関するフォームは，エディタ上部に配置した．

表 4.1：実装した課題一覧

課題名	概要
手順	推論課題を実施し，「実行」選択後に分析課題を実施する．
動きに分ける	分割課題を実施し，「実行」選択後に分析課題を実施する．
コーディング	制御課題を実施し，「実行」選択後に分析課題を実施する．

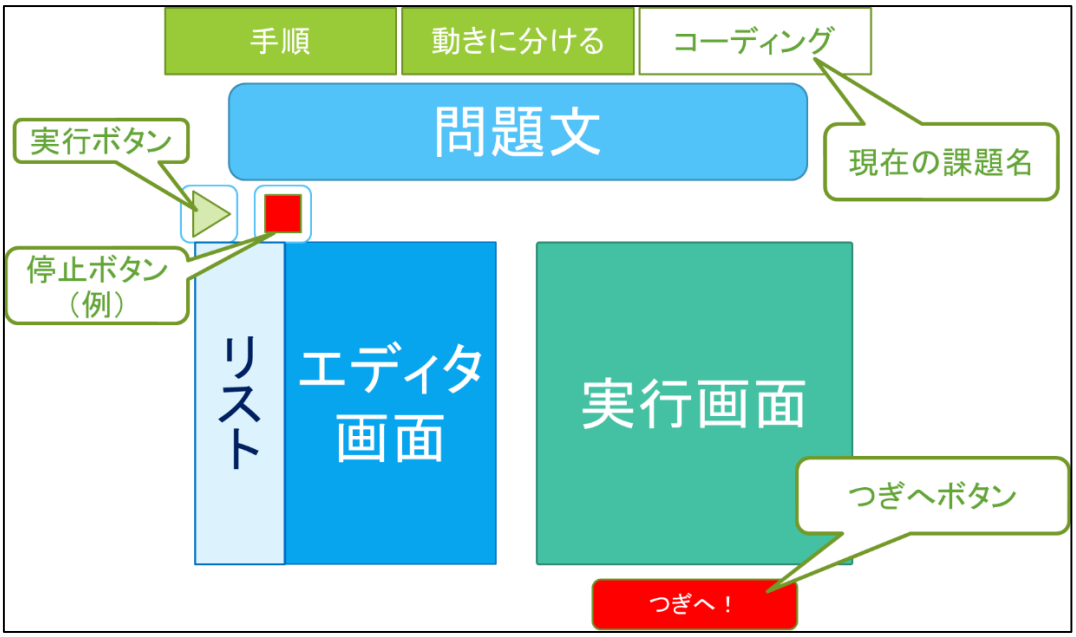


図 4.2：思考課題画面の基本構成

各課題の基本的な操作としては，リストやエディタ画面内の対象のオブジェクトを直接ドラッグ＆ドロップし，実行ボタンを押す．これにより，実行画面へ操作の結果が反映される．正答することで次へボタンが出現し，これを押すことで次の課題に遷移する．課題間の遷移は，分析課題が終了した後に次の課題に移るための「つぎへボタン」を出現させ，それをクリックすることで遷移を実現する．

また，実行画面においては，エディタの内容をただ反映するのではなく，設計フェーズにおいては目的地への道のりを各課題で表示させるようにし，開発フェーズにおいては目的地に対して，プレイアブルキャラクターをコード（命令）によって操作するような表示にする．

4.4 使用する JavaScript ライブラリ

今回は操作性の高い動的なアプリケーションを再現するため,「p5.js」という JavaScript のライブラリを用いる[17]. p5.js とはビジュアルデザイン用に設計された言語「Processing」のコーディング環境を JavaScript 上で再現したもので, Processing 特有のアニメーション処理を JavaScript の DOM 操作と併用して実装することができる. 特徴的な仕様として, html のタグ id に関連付けて独自のキャンバスを展開し, フレーム単位で描画する点である. これにより高度な動作表現を可能としている. また, スプライトと呼ばれるオブジェクトをインスタンス化することができる. スプライトには生成時点でオブジェクトサイズ大のコライダー（接触判定面積）が設定されており, オブジェクト同士の当たり判定を容易に検出することができる. 任意のフォルダに画像をまとめることで, スプライトに直接アニメーションを付与することも可能である.

プログラムの生成については, ビジュアルプログラミングによるコーディング環境を提供する「Blockly」を用いる[18]. Blockly は Google が開発した JavaScript ライブラリであり, ブロックタイプのコードを組み合わせることでプログラムを表現することができる. また, 作成したプログラムは JavaScript のコードとして変換することが可能であり, eval 関数（JavaScript の標準ビルトインオブジェクト）と組み合わせることで, 作成したコードをそのままブラウザ上で実行することができる.

4.5 開発方針

Thinkron のアクセスフローを図 4.3 に示す.

開発方針として, Thinkron の構成は html, php で行い, 視覚的な動作表現は JavaScript の DOM や前節のライブラリを用いる. はじめにアクセス時のタイトル画面である Title にアクセスし, 利用者情報の取得を行う. その後, 問いの選択画面である Question List に遷移し, あらかじめ教師が設定したデータ（以降, 教師データ）を取得する（表 4.2）. Question List では出題する問いの内容が保存されているロードデータを取得する. そして, 問いを選択することで思考課題を提供する Task に遷移する. 思考課題におけるリストには, 手順課題にアクトデータ, 動きに分ける課題にムーブデータ, コーディング課題にコードデータと, それぞれ専用のデータを用いる. 実行画面には Thinkron の題材である「手順が結果として反映されやすい手続き型の探索ゲーム」（3.4 節）を再現するマップデ

ータを適応し，生成されたマップ上の課題解決に取り組んでいく．結果の表示においては Question List の模範解答を参照し比較を行う．すべての課題が終了したら，Result に遷移し結果を返す．この時，各思考課題における操作回数を Log Data に保存する．

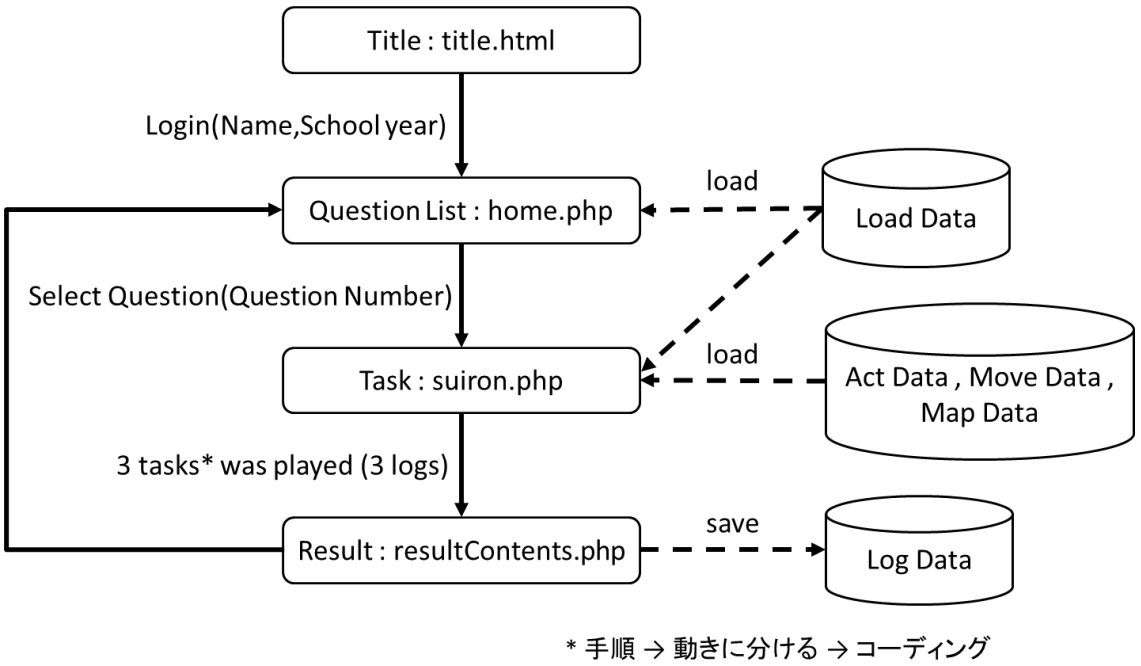


図 4.3：Thinkron のアクセスフロー

表 4.2：教師データ一覧

データ名	ファイル形式	用途
ロードデータ	JSON	問いの内容，模範解答を問題番号別に管理する．
マップデータ	CSV	各思考課題の実行ボタンを押した後に結果を反映する画面を構成する．オブジェクトの配置をテーブルで管理する．
アクトデータ	JSON	推論課題で用いるラベルデータを管理する．
ムーブデータ	JSON	分割課題で用いるラベルデータを管理する．
コードデータ	XML	制御課題で用いるラベルデータを管理する．

第5章 Thinkron の開発

5.1 開発環境

Thinkron の開発環境は図 5.1 のとおりである．なお，動作チェックには本学の貸与 PC（Surface）を使用した．サーバについては， JAIST の個人サーバ（https://www.jaist.ac.jp/~s1910095）および，仮想 Web サーバを構築する XAMPP[19]を用いて動作の確認を行った．動作確認環境については図 5.2 のとおりである．

デバイス	ThinkPad X1Carbon
OS	Windows 10 Pro(x64)
CPU	Intel(R) Core™ i7-8650U
RAM	16.0 GB
エディタ	Visual Studio Code(1.50.1)

図 5.1：Thinkron 開発環境

デバイス	Surface(本学の貸与PC)
OS	Windows 10 Pro(x64)
CPU	Intel(R) Core™ m36Y30
RAM	4.0 GB
ブラウザ	Firefox 84.0.2(x64)
PHP(JAIST)	5.3.27
PHP(XAMPP)	7.3.3

図 5.2：Thinkron 動作確認環境

5.2 教師データの開発

5.2.1 生成した教師データ

図 4.1，4.3 のとおり，教師が問いのデータを作成し設定しておく必要があるため，該当データの作成を行った．

ロードデータは JSON 形式で作成し，問題番号，出題文言，模範解答を一括して保存している（図 5.3）．

マップデータは，実行画面上のオブジェクトを記号で配置しており，これをメインプログラムが読み込むことで任意のマップを生成する（図 5.4，図 5.5）．マップの生成には p5.js を使用し，あらかじめ用意したアイコンを記号と関連付けて配置している．

アクトデータ，ムーブデータは JSON 形式で作成し，画像の情報とそのラベルが保持するオプションデータを保持している（5.2.2 項参照）．

コードデータは Blockly の仕様に従い，今回は html 上に埋め込まれた xml タグで作成している（5.2.3 項参照）．定義した XML は，コーディング課題のリスト上でブロック形状のオブジェクトとして生成される．なおコードの形状，内容の定義は，JavaScript によって別ファイルに定義される．

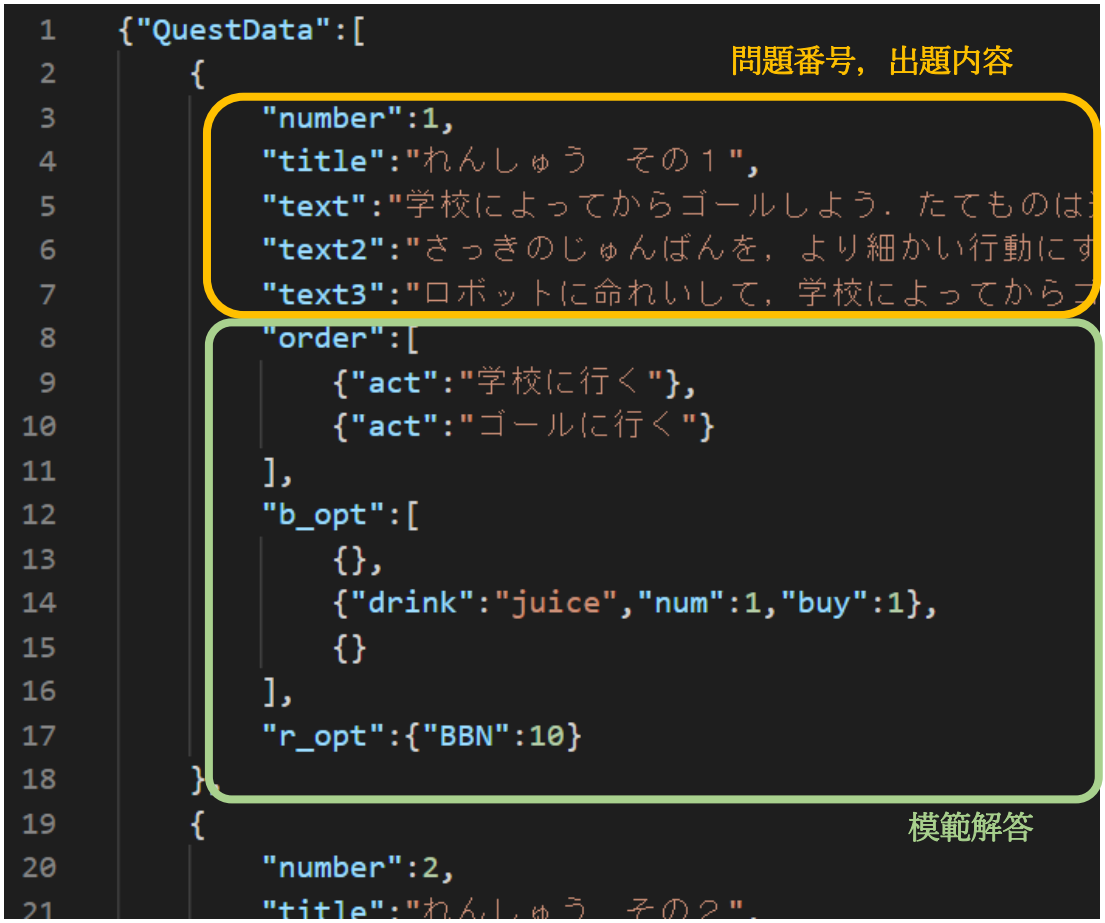


図 5.3：ロードデータの作成

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		黒枠には配置できません													
2		B	B	B	B	B	B	B	B	B	B	B		S・・・スタート地点	
3		B										B		G・・・ゴール地点	
4		B		B	O	B	O	B		B		B		E・・・イベント	
5		B		G				O				B		O・・・障害物	
6		B		B	O	B		B		B		B		P1・・・学校	
7		B		O	S	O		O				B		P2・・・自販機	
8		B		B		B		B		B		B		P3・・・公園	
9		B		O				P1				B		P4・・・神社	
10		B		B	P4	B	O	B		B		B			
11		B										B			
12		B	B	B	B	B	B	B	B	B	B	B			
13		黒枠には配置できません													
14															

図 5.4：マップデータの作成

図 5.5：マップデータの生成

5.2.2 ラベルデータの作成

アクトデータを用いたラベルの作成において，今回はマップデータで構成された目的のオブジェクトに向かって移動するという行動（以降，移動行動）と，目的のオブジェクトに対してイベントを発生させるという行動（以降，イベント行動）の 2 つのバリエーションを設定することにした．そこで，目的のオブジェクトへの行動を「トリガー」とし，ラベル名やトリガー先の情報，トリガーの発生条件などを付与したデータを作成することにした．

ムーブデータを用いたラベルの作成においては，分割された内容一つ一つをマップ上での動きとして直接反映するために，実行画面に働きかける専用の関数を作成し，ラベルに対応する関数のコードをテキストで付与したデータを作成することにした．

これらのデータの作成を円滑にするため，簡易的なラベル生成システムの開発を行った（図 5.6，図 5.7）．

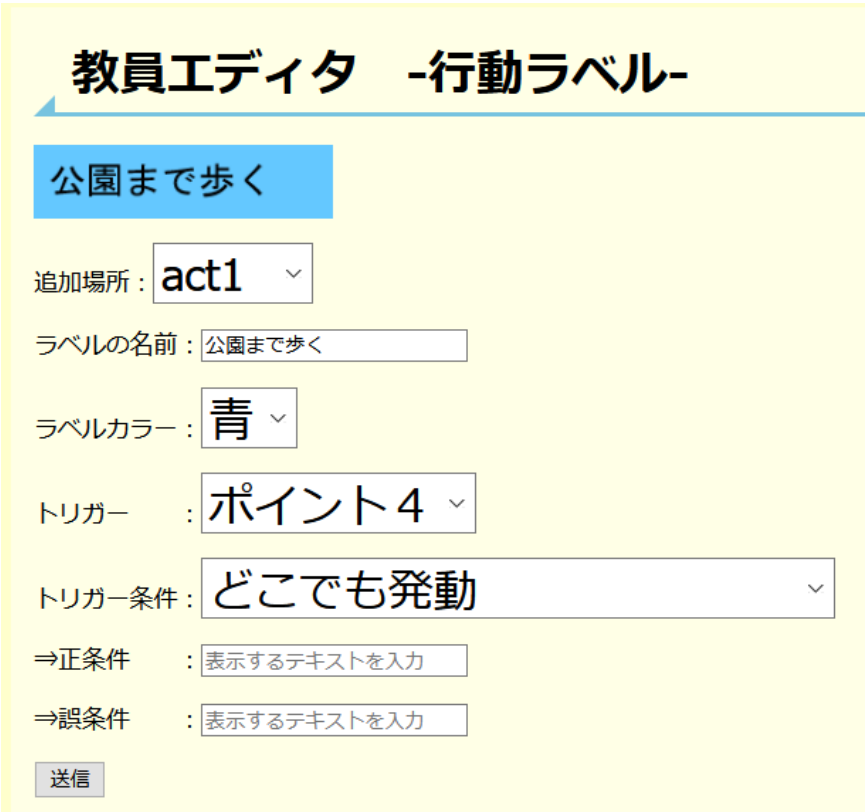


図 5.6：行動ラベルの生成フォーム

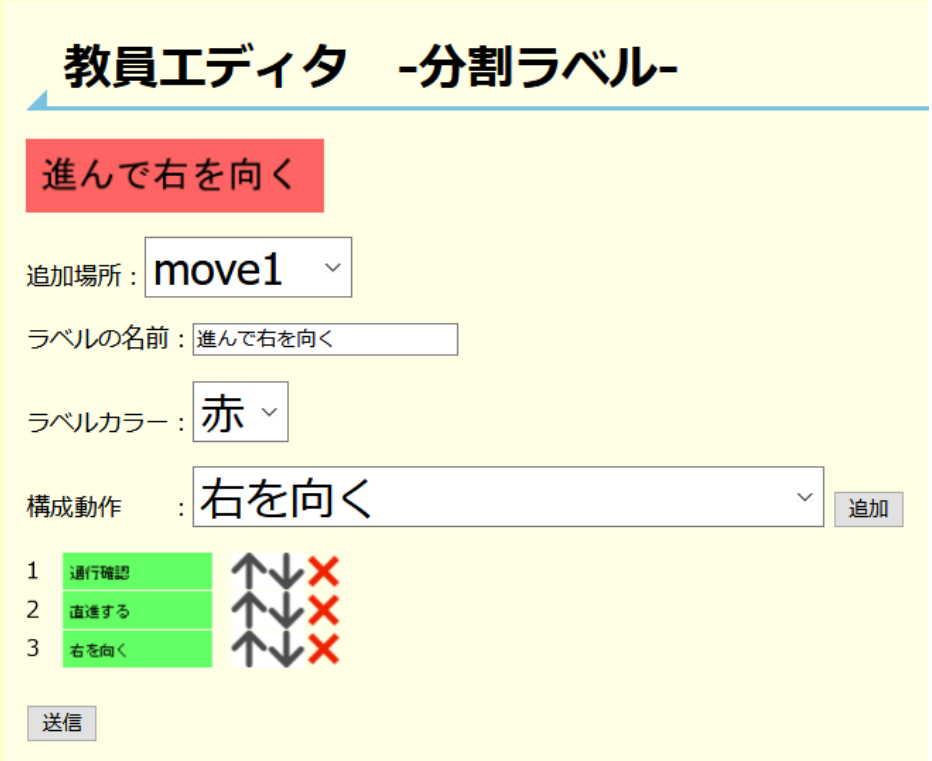


図 5.7：分割ラベルの生成フォーム

具体的には，ラベルに対する情報をプルダウンメニュー等で設定し，書き出す際には画面上部に反映されているラベルの描画を画像データとして保存する．ラベルデータはリストとして一覧で大量に表示し，更に直接ドラッグ操作するため，ブラウザの読み込み速度の低下を招く可能性がある．そこで Data URI スキームを用いて画像を文字列化し，フォームの設定内容と共に JSON ファイルに保存している．図 5.8，5.9 に作成したアクトデータおよびムーブデータの一例を示す．

作成されたアクトデータは，ラベル名，ラベルの画像 uri，トリガー先であるマップデータ上の記号，トリガー条件，トリガー条件を満たす場合の表示テキスト，トリガー条件を満たさない場合の表示テキストの 6 要素で構成される．前述した 2 つのバリエーションの判断については，トリガー条件（図 5.8 の”opt”に記述される（”in”の場合は，イベント行動，”out”の場合は移動行動）．

作成されたムーブデータは，ラベル名，ラベルの画像 uri，動作の関数テキストの 3 要素で構成される．

図 5.8：ラベルデータ例（アクトデータ）

図 5.9：ラベルデータ例（ムーブデータ）

5.2.3 ラベルデータの作成

コードの作成においては，Google の提供する Blockly Developer Tools[20]を用いてコードを作成した（図 5.10）.

Blockly Developer Tools では，コードのビジュアルと組み合わせるコードの制約条件を付与したオリジナルコードを作成することができる．コードをエクスポートする際には，ブロックの形状，制約の情報が書かれたファイルと，実行内

容の定義が書かれたファイルがダウンロードされる．ただし，実行内容については，直接ファイルに書き込む必要がある．作成されたコードは一つの XML フ
XML の構成としては，コードを種類別に格納するカテゴリを category タグ，具現化するコードを block タグ，プルダウンオプションを field で囲む（図 5.11）．

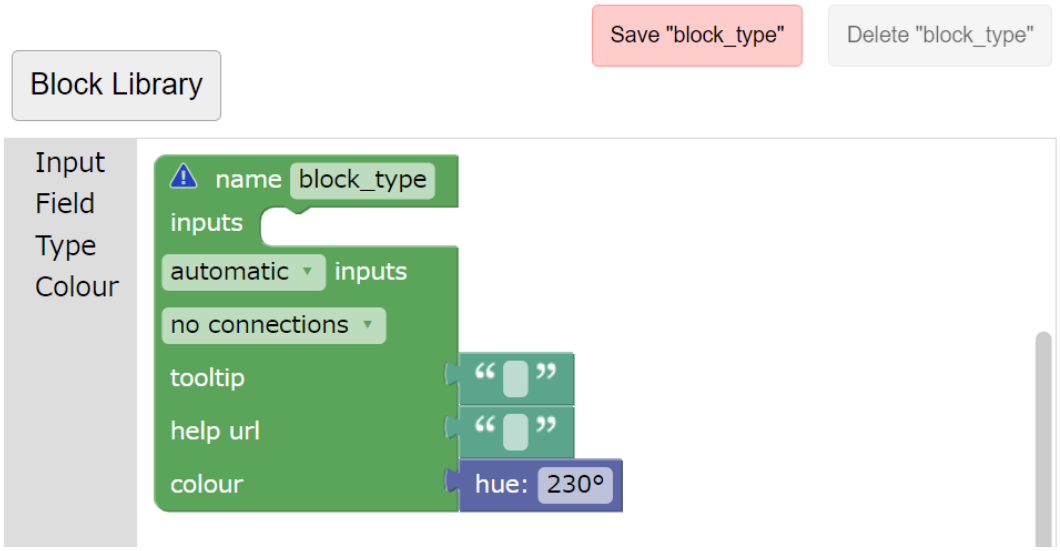


図 5.10：Blockly Developer Tools

```
<xml id='toolbox' style='display: none' >
    <category name='動き' colour='#FF0000'>
    <block type='turn90'>
    <field name='LR'>right</field>
    <field name='NAME'>90</field>
    </block>
        <block type='monney'>
        <field name='NAME'>100円玉</field>
        </block>
        <block type='drink'>
        <field name='NAME'>お茶</field>
        </block>
        <block type='pick'></block>
        <block type='walk'></block>
        <block type='pick_m'></block>
    </category>
```

図 5.11：コードリストの作成

5.3 タイトル・問い選択画面の開発

タイトル画面を図 5.12，問いの選択画面を図 5.13 に示す.

図 5.12：Thinkron タイトル画面

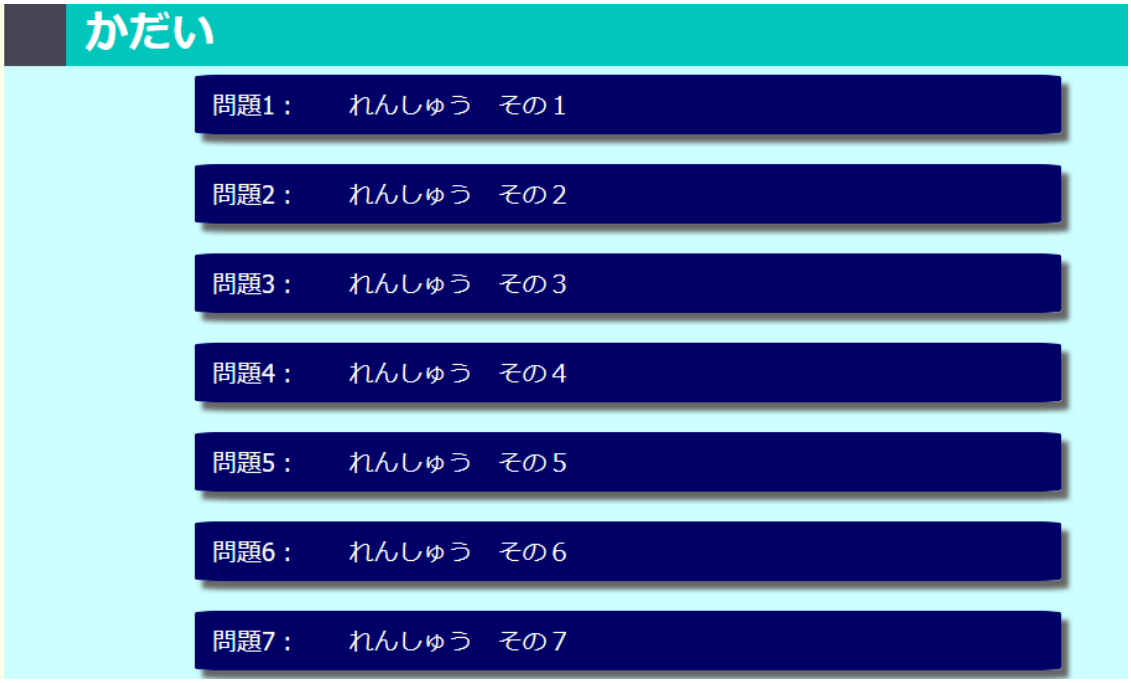


図 5.13：問い選択画面

タイトル画面は後述（第 6 章）の実験に合わせ，学習者（匿名）と学年を把握するためのフォームを設けている．はじめは中央部の「スタート」は消えている状態であり，「名前をえらぼう！」,「学年をえらぼう！」のプルダウンメニューを選択することによって,「スタート」が出現する．「スタート」を押すことで問いの選択画面に遷移する．

問いの選択画面については，教師が設定したロードデータの内容が反映され，図 5.13 のように問いが一覧表示される．各問いはボタン形状で生成され，ボタンを押すことで指定の思考課題の画面へ遷移する．

5.4 手順課題の開発

5.4.1 リスト，エディタ画面の開発

手順課題のリスト，エディタ画面の一例を図 5.14 に示す．

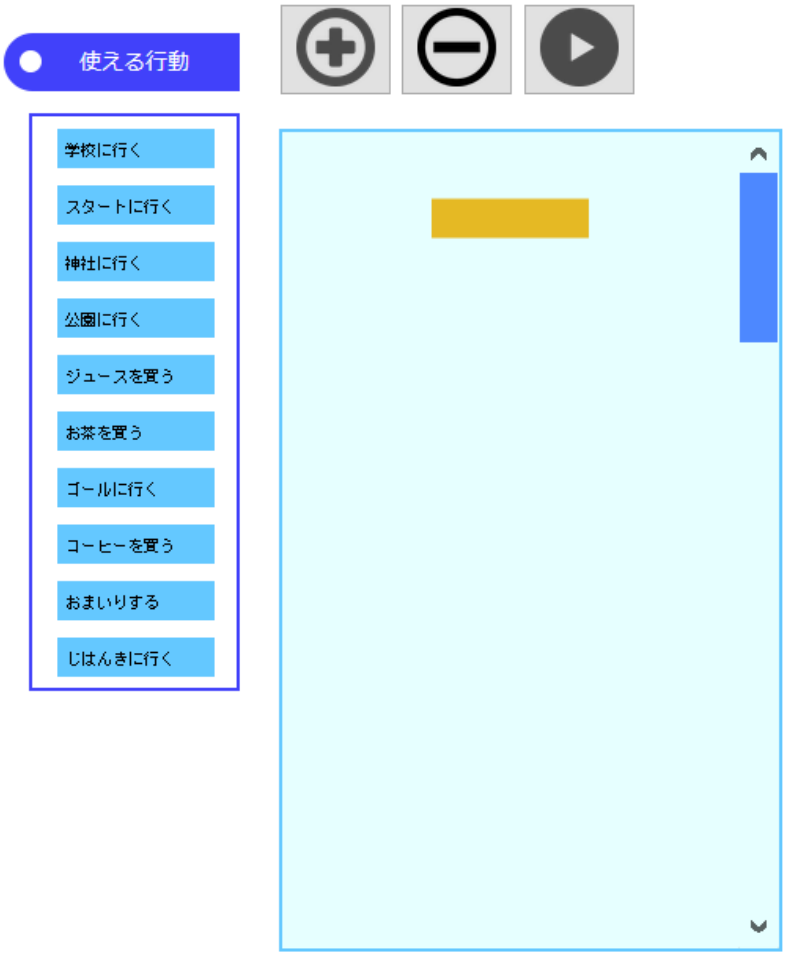


図 5.14：手順のリスト，エディタ

リストは，JSON ファイルより読み込んだ url をブラウザ上で画像を構成して表示している（5.2.2 項参照）．リスト内のラベルをドラッグすると，DOM により html 要素ごと座標を操作し，マウスカーソル，もしくはタッチ操作に合わせて追従する．

エディタ画面においては，抽出してきたラベルの受け取り部分であるレシーバを p5.js のスプライトで生成している．レシーバにラベルがセットされていない場合は，図 5.14 のとおりオレンジ色の長方形が配置され，エディタ上部のレシーバを増減させるボタン（＋，—）を操作し，レシーバの量をコントロールすることができる．

レシーバへのラベルのセット時には，p5.js のキャンバス外から html の要素をそのまま抽出してくることができないため，次の手順で実現した．まず，マウスに追従する透明なスプライトを用意しておき，レシーバとの衝突判定を検出できるようにしておく．次に，画像の uri を WebAPI のマウスクリックイベント等で取得（ドラッグ）する．最後にドロップ先のレシーバと，追従しているスプライトとの衝突を判定し，衝突していたら取得した uri をレシーバに貼り付ける．これを応用し，ラベルのスワップ操作を実現する．ドラッグの際に追従する透明のスプライトに取得したラベル画像を貼り付けておき，ドロップ時に接触しているスワップ先のレシーバへラベル画像を貼り付ける．レシーバに接触していない場合は削除の操作と判断し，ドラッグ元のレシーバの画像と追従するスプライトを削除して，カーソルに追従する透明なスプライトを再生成する．ラベルセット後のエディタを図 5.15 に示す．

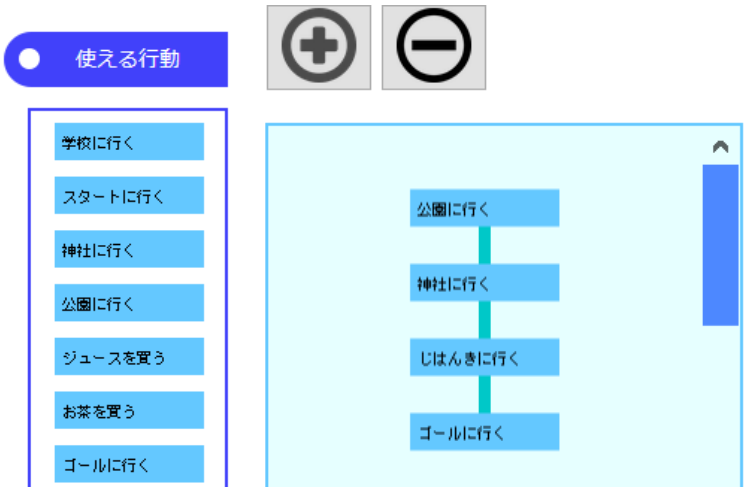


図 5.15：ラベルセット後のエディタ画面（手順）

エディタの上部には実行ボタン（▶）を配置し，押すことで実行画面に結果を表示する．実行中は図 5.15 のように一時的にボタンが消えるようになっている．これは，ボタンの連打による意図しない操作や実行結果への影響を考慮したためである．また同様の理由で，手順課題をクリアした後は，画面内の操作が行われないように p5.js の描画停止および，実行ボタンの非表示の対応を行っている．

5.4.2 実行画面の開発

手順課題の実行画面の一例を図 5.16，5.17，5.18 に示す．

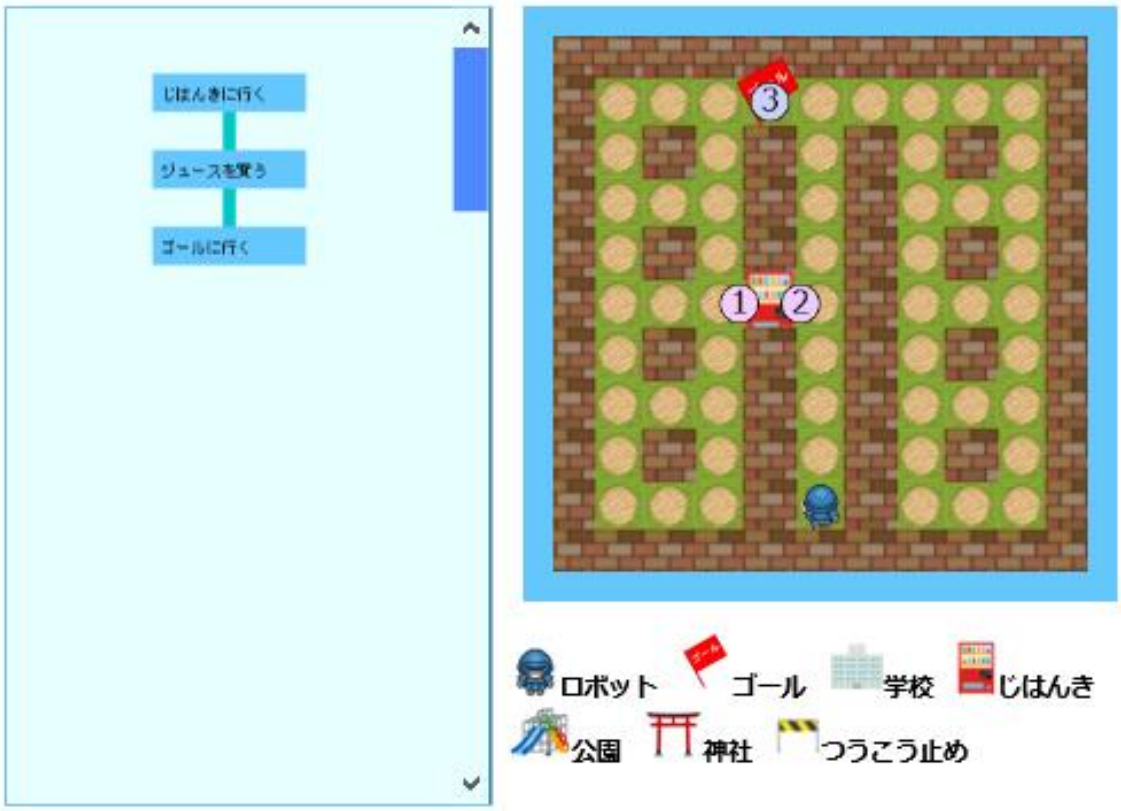


図 5.16：ラベルセット時の実行画面（手順）

ラベルをレシーバにセットした際は，図 5.16 のようにラベルの配置順に合わせて実行画面に番号が振られる．これは，ラベル（アクトデータ）に付与されているトリガー先の情報を読み取って自動的に付与している（5.2.2 項参照）．もし，トリガー先が連続する場合は図 5.16 の①，②のように表示される．実行ボタン

を押すと，移動行動の場合はトリガー先までの道のりが三角形のオブジェクトで表示され，イベント行動の場合はオブジェクト上にテキストが表示される（図 5.17，図 5.18）．道のりの表示にはダイクストラ法を用いており，トリガー先までの最短経路が表示される．

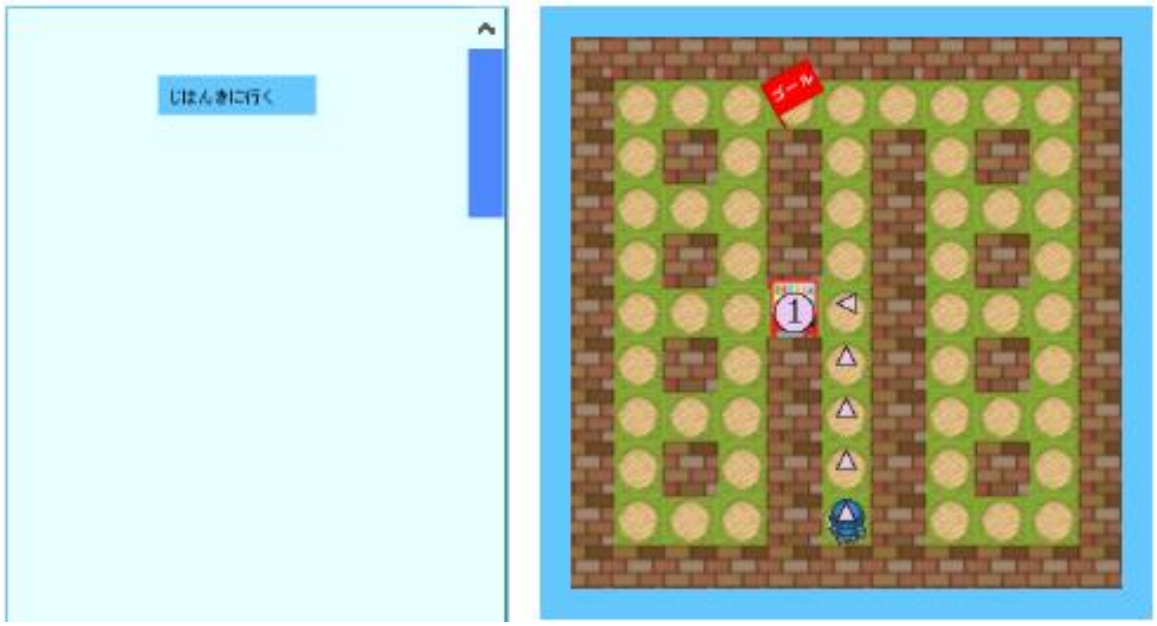


図 5.17：移動行動の実行

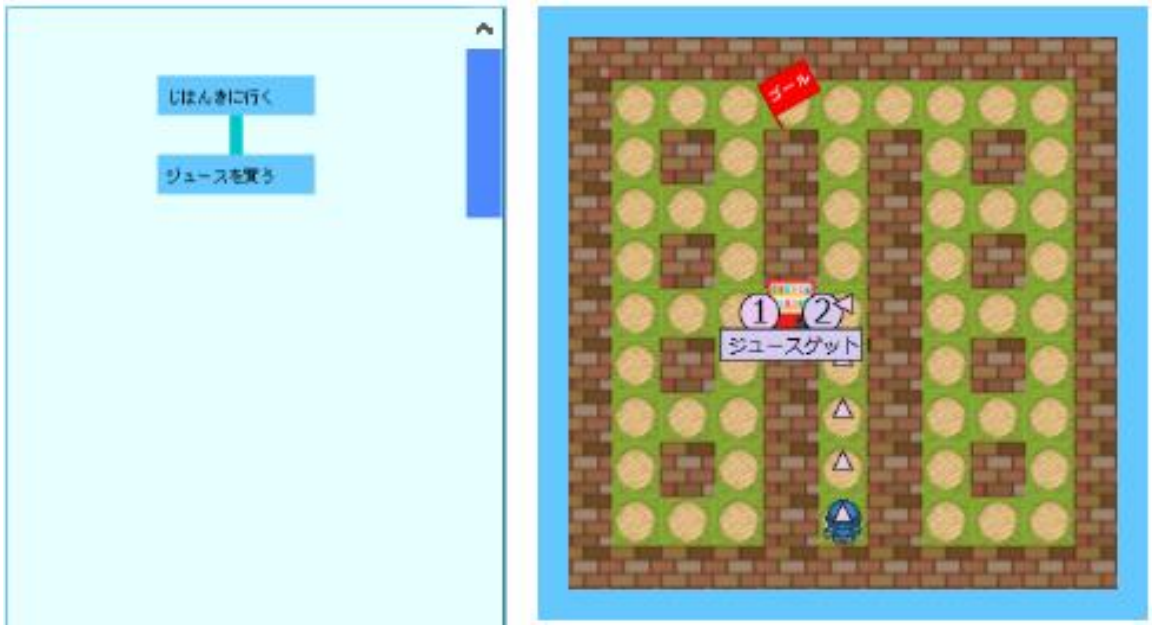


図 5.18：イベント行動の実行

5.4.3 全体構成

図 5.19 に手順課題の全体画面を示す.

図 5.19：全体画面一例（手順）

構成として，画面上部に今挑戦している課題名と，問題の番号，取り組む内容が表示される．画面右上にはタイマーが設けられており，問いを解き始めてからの経過時間を表示している．

正解の判定には，模範解答に設定されているラベルの名前と順番より，組み立てたレシーバ上のラベルと一致するかどうか比較を行う．実行画面の下部には実行画面上のオブジェクトの詳細が表示されているが，問いに正解した場合には，図 5.20 のようにアラートを表示し，アラートを閉じると実行画面の下部が「つぎへボタン」に変更される（図 5.21）．なお，不正解の場合は図 5.22 のようなアラートが表示される．

図 5.20：実行後の正解アラート

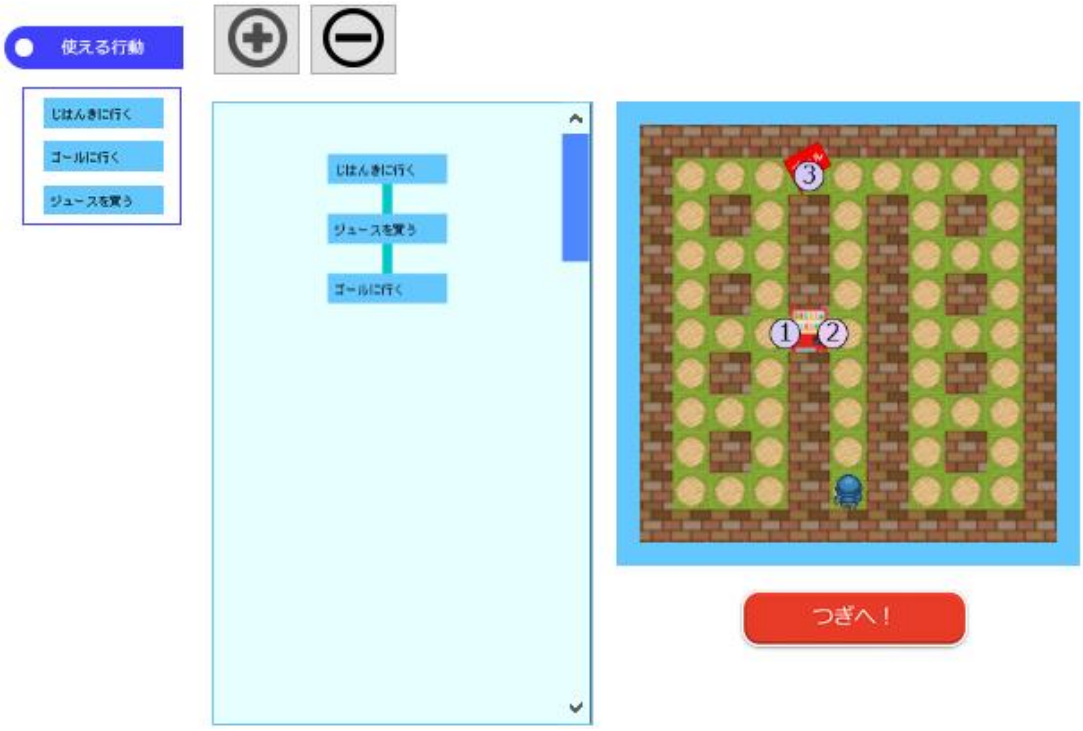


図 5.21：つぎへボタン表示時

図 5.22：実行後の不正解アラート

5.5 動きに分ける課題の開発

5.5.1 リスト，エディタ画面の開発

動きに分ける課題のリスト，エディタ画面の一例を図 5.23 に示す.

リスト，エディタ共に使用している技術は 5.4.1 項と同様であるが，手順課題で作成した内容を引き継いだ行動ラベル（図 5.23 中の青いラベル）がレシーバとして機能している．ドロップされたラベルはレシーバに対応した分割エリア（分かつエリア）に蓄積される（図 5.24）．分割エリア内のスワップは，手順課題と同様であるが，削除については右横に表示されるごみ箱マークをクリックすることで対象ラベルが消える．なお，分割エリアのラベルを他の分割エリアに移動することはできない.

実行ボタンの仕様についても手順課題同様である.

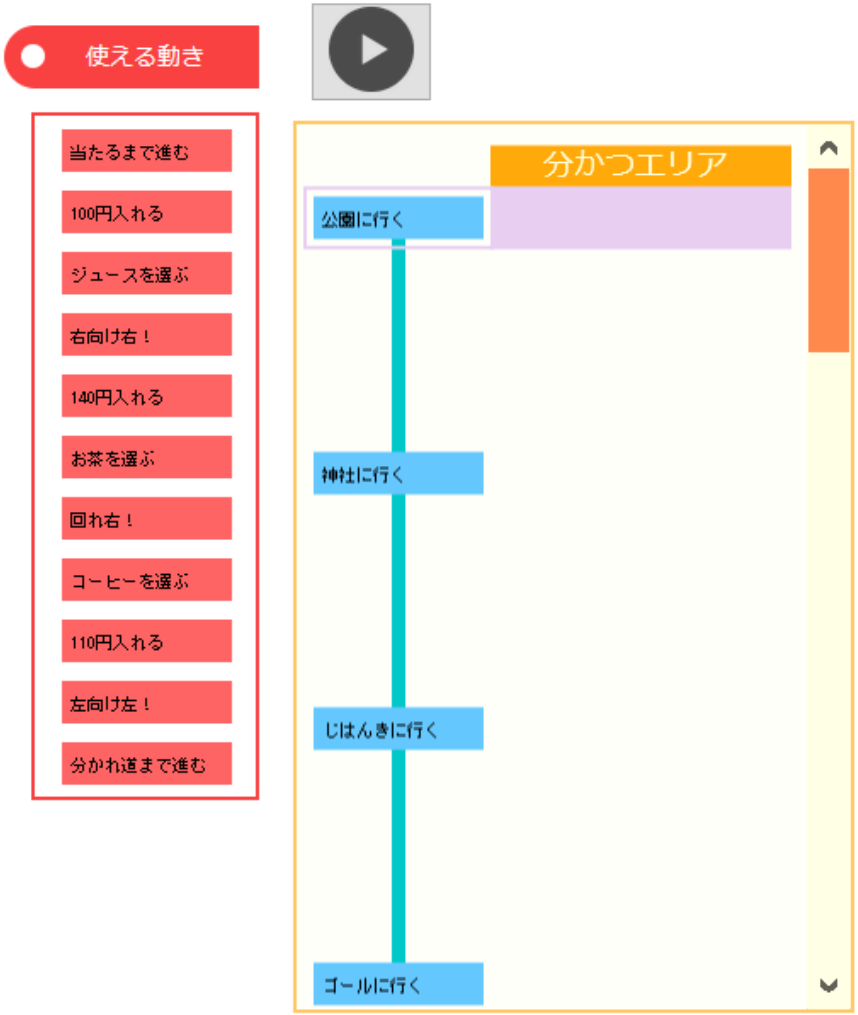


図 5.23：動きに分けるのラベルリスト

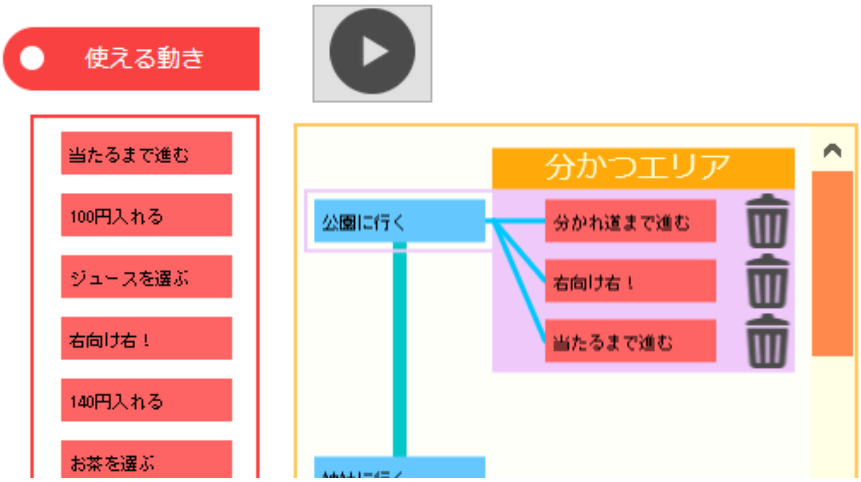


図 5.24：分割ラベル設置後

5.5.2 実行画面の開発

手順課題の実行画面の一例を図 5.25，5.26，5.27 に示す.

図 5.25：1 番目のレシーバ選択時の実行画面（動きに分ける）

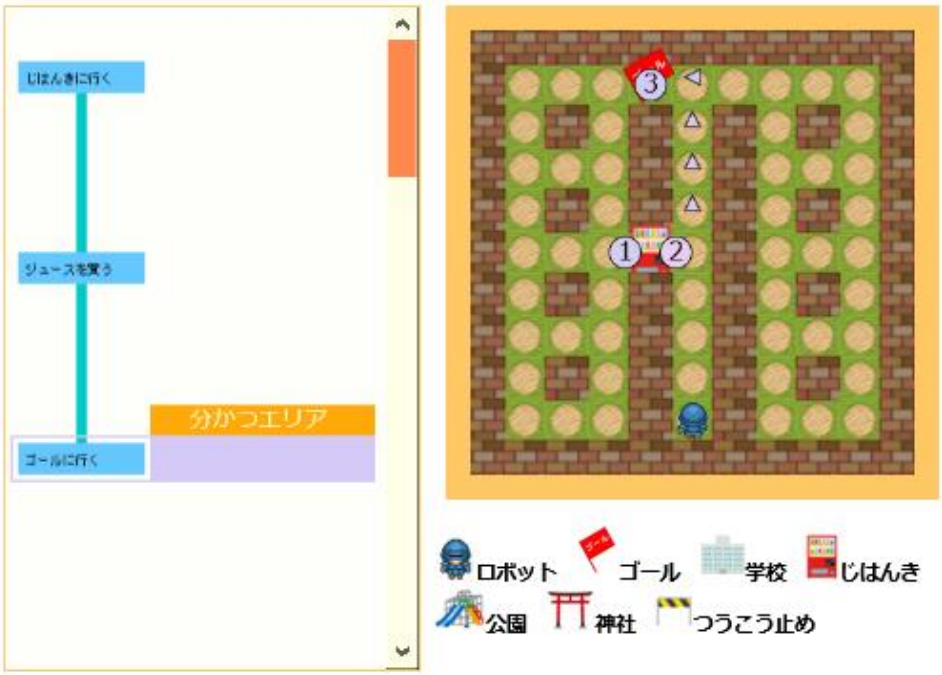


図 5.26：3 番目のレシーバ選択時の実行画面（動きに分ける）

図 5.27：動きに分ける課題の実行時

実行画面には，手順課題で作成したラベルの手順がレシーバとして表示され，このレシーバにカーソルを合わせると，実行画面上に手順課題の各レシーバに対応する結果が表示される（図 5.25，5.26）．実行ボタンを押すと，ラベルに紐づけられた関数が実行され，青色の三角形が実行画面上に分割エリア単位で表示される（図 5.27）．

5.5.3 全体構成

図 5.28 に分割課題の全体画面を示す．画面上部の構成は手順課題同様である．

正解の判定には，元となる行動それぞれに合わせて別の判定が行われる．移動行動の場合は，ダイクストラ法で作成された手順と同じ道に沿って進んでいるかどうかを調べる．イベント行動においては，分割された諸動作の手順で，ターゲットがクリアされているかどうか（今回は，自動販売機で目的の飲み物を買えたかどうか）を調べる．アラートやつぎへボタンの表示は手順課題同様である．

図 5.28：全体画面一例（動きに分ける）

5.6 コーディング課題の開発

5.6.1 リスト，エディタ画面の開発

図 5.29 にコーディング課題におけるリスト，エディタ画面の一例を示す．

構成については，Blockly ライブラリのツールボックス（リスト）および，ワークスペース（エディタ画面）の生成コードを用いている．

リストにおいて，各コードは指定のカテゴリ内にストックされており，カテゴリをクリックすることでコードの一覧を展開，省略することができる（図 5.30）．カテゴリ内のコードはドラッグ＆ドロップすることで，エディタ画面内に配置することができ，コードの形状にもよるが，凹凸部同士を組み合わせるようにして一つの大きなコードを作成していく（図 5.31）．コードを削除する場合は，画面右下のゴミ箱マークにコードをドロップするか，左横のグレーの領域にドロップする．ゴミ箱マークをクリックすると，削除したコードの一覧を展開するこ

とができ，再度エディタ上に配置することができる．

エディタ上部には実行ボタンの他に停止ボタン（■）を設けている．これは，実行中のプログラムを強制的に停止することができる．無限ループなど，永続的な実行コードを停止するために使用する．実行ボタンの仕様は他課題と同様である．なお，エディタ画面の下部には，現在使っているブロック（コード）の数と，キャパシティの情報が表示される．

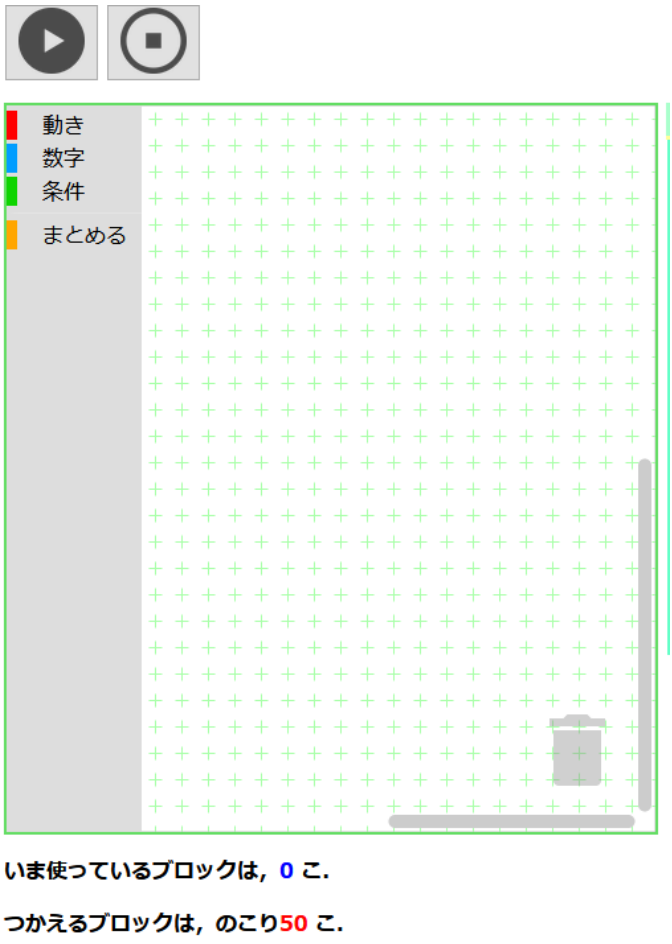


図 5.29：コーディングのエディタ画面

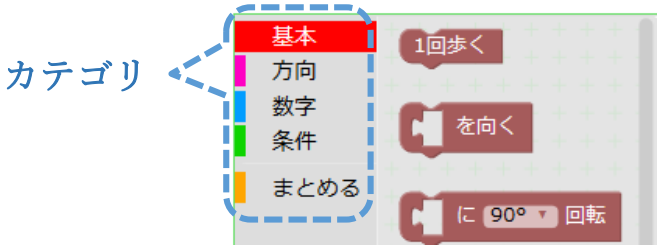


図 5.30：コーディングのコードリスト（カテゴリ展開時）

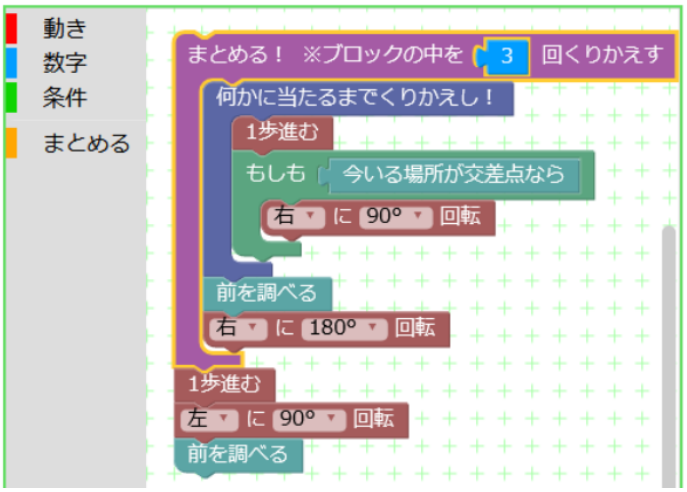


図 5.31：コードの配置後

5.6.2 実行画面の開発

コーディング課題の実行画面の一例を図 5.31，図 5.32 に示す．

コーディング課題の実行画面は，タブ切り替えによって「分割メモ」という画面に切り替えることが可能である．分割メモは，動きに分ける課題で構成した内容を表示する機能であり，設計内容を見返しながらのコーディングが可能である．

図 5.32：分割メモ

図 5.33：実行画面（コーディング）

実行ボタンを押すと，コードに紐づけられた関数が実行されるが，動きに分ける課題と異なり，実行画面上の「ロボット」がコードに合わせて動作する（図 5.34）．正解の判定は動きに分ける課題同様である．

図 5.34：実行後の表示（コーディング）

5.6.3 全体構成

図 5.35 に制御課題の全体画面を示す．画面上部の構成は手順課題同様である．

実行画面の上部においては，イベント行動の再現の際に用いる情報が表示される（今回は，自動販売機で目的の飲み物を買うため，所持金の情報が表示されている）．

思考活動における制御，分析の操作は，図 5.34 のようにコードを組み立てる操作から評価を行うが，一般化においては確実に目的の活動を捉えるため，所定のコードを使用させることで観測できるようににした．具体的には，「まとめる」という一般化専用の関数（以降，ファンクションコード）を作成し，問いの解決時まで使用不能に設定しておく（図 5.36）．そして，解決と同時にファンクションコードの使用制限を外し，アラートでファンクションコードの使用を促すと

いうものである（図 5.37，5.38，5.39）．あらかじめ，問いを解決する最適なコード数をロードデータ上に設定しておき，エディタ内で使用しているブロック数の情報と比較することで，問いの完答の判断とする．こうすることで，ファンクションコードを使ってブロック数（コード数）を減らすという活動目的が生まれる．問いの設定と最適コード数の設定に依存するが，組み立てたコードにパターンチックな要素が含まれていれば，パターンを抜き出し，複数個所に適応させてコード数を減らすといった一般化が再現できる（図 5.40，5.41）．

正解，不正解時のアラートは手順課題同様であるが，最後の思考課題であるため，つぎへボタンの表記を「けっかはっぴょう！」とし，次節の結果ページに遷移する（図 5.42）．

図 5.35：全体画面一例（コーディング）

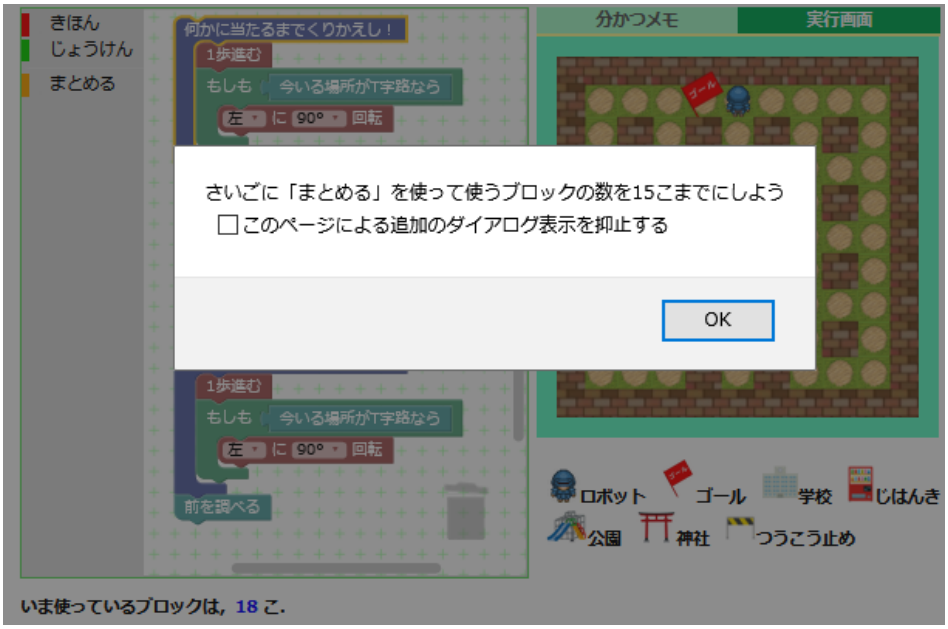


図 5.36：一般化アラート

図 5.37：ファンクション（使用不可）

図 5.38：ファンクション（使用可）

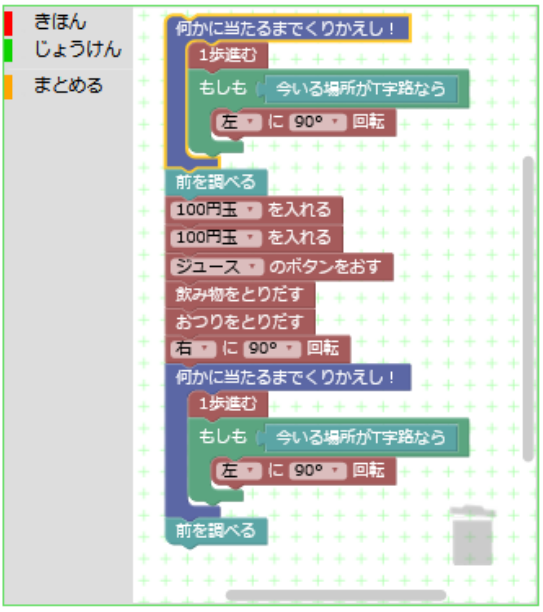


図 5.39：パターンチックなコード例

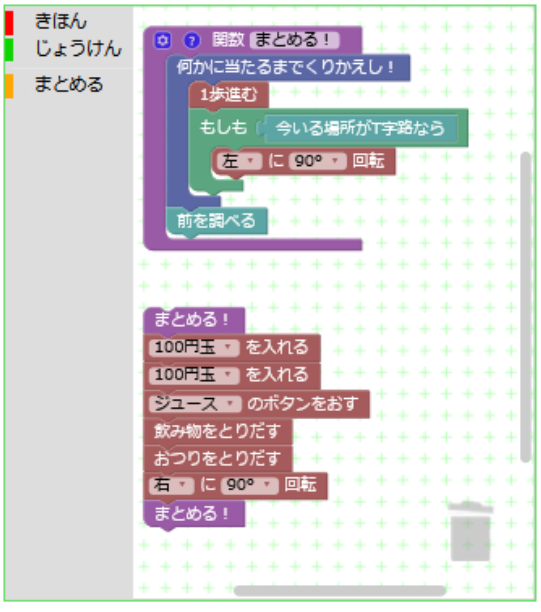


図 5.40：ファンクションコード適用後

図 5.41：コーディング課題完答後

5.7 結果ページの開発

図 5.42 に Thinkron の結果ページの画面を示す.

図 5.42：結果ページの一例

構成としては, 各課題のクリア時間が記載され, クリア時間に応じて獲得される星の数が変化する. 現時点では, 手順課題を 5 分, 動きに分ける課題を 10 分, コーディング課題を 15 分と設定している．展望としては，各課題の操作回数に基づき, 量データ, 質データをチャートとして表示させることなどが考えられる.

「もどる」を押すと，問い選択画面に遷移する.

第6章 実験

6.1 Thinkron の性能評価実験

Thinkron によって取得された定量データが，既存の評価指標と比較してどの程度信頼できるものであるのかを確認するため，評価実験を行った．本実験では問いにおける難易度の違いにより，ルーブリックを用いて学習者の達成度の違いを示すとともに，定量データがルーブリック評価の指標に従うかどうかを調査するものである．

6.2 評価方法

評価実験の全体方針として，実際に Thinkron を用いて問いを解き，学習結果をルーブリックで段階評価する．その値と Thinkron から得られた量データ，質データを比較し傾向（一致度）を分析する．

本研究では難度の異なる問いに対する評価の変化を確認するために低難度と高難度の二種類の問いを準備した．3.5.1 項に示したように，これらの各問いにおける量データ（操作回数）を取得すると共に，質データを算出する．なお，評価実験の際には時間に限りがあるため，思考課題ごとの制限時間を設ける．そのため，制限時間内でクリアできた場合の質的な差を検討する必要がある．そこで，クリアタイム t，制限時間 T とすると，質データ q は(2)式で表すことができる．

$$q = \frac{T-t}{T} + \frac{cs}{C(|s-n|+s)} \qquad (0 \le t \le T) \qquad (2)$$

また，一般化の評価については，正解数 c の代わりに全体のコード数の減少数から質的評価を行う．目標のコード数を B，タイムアップ時のコード数を b とし，正解総数 C の倍率に合わせると一般化における質データ q は(3)式で表すことができる．

$$q = \frac{T-t}{T} + \frac{\frac{(2B-b)}{B}Cs}{C(|s-n|+s)} = \frac{T-t}{T} + \frac{Bs}{(2B-b)(|s-n|+s)} \qquad (3)$$

これらの質データについて，思考活動の定量化データとして有効であるかどうかルーブリック評価との相関によって示す．ルーブリックにおいては，齋藤ら

が提案した Rubric ProEEs を参考にする[21]. Rubric ProEEs は，小学生を対象としたプログラミング教育における学習到達度を 4 段階で評価するルーブリックであり，特定の学習方法やツールに依存しない全般的なプログラミング学習に適応できる．また，プログラミング学習における活動を厳密に捉えた評価が多く，Thinkron の操作に対応できる点も多い．そこで，Thinkron の Log Data に操作回数とは別に各インターフェースの操作履歴を保存しておき，その内容に Rubric ProEEs を適用する．

評価の項目は Benesse の評価規準の概念および，思考活動の段階評価として対応するものを選出した（表 6.1）．ただし，Thinkron の仕様により段階の分別として一致しない部分も存在するため，Rubric ProEEs の評価観点に沿うような形で，思考課題上の操作に適応する文言に一部差し替えて用いる（表 6.2）．

なお，学習者によって各思考課題を一度の実行でクリアするケースも考えられる．この場合，分析課題が行われないため，ルーブリックの評価ができない．そこで，表 6.1 の「事象の分析」の項目に基づき，原因と結果の関係を考えた上でクリアしたとみなし，評価 4 を付けることにする．

表 6.1：Rubric ProEEs 該当項目

思考活動	RubricProEEs該当項目	4	3	2	1
順序立て	動作の構築・関数化	目的を実現するため，複数の手順の最適な組み合わせを考え，汎用性，再現性のある手順を創作できる	目的を実現するため，複数の手順を，順次処理，繰り返し処理，条件分岐処理などを利用して組み合わせることができる	手続きは複数の手順で構成されていることに気づき、与えられた手順を目的に合わせて並び替えられる	逐次実行を理解していない
分割	問題の細分化	大きな問題をこれ以上分割できない小さな問題に正しく分割できる	大きな問題を複数の小さな問題に正しく分割できる	大きな問題を二つ以上の小さな問題に分割できる	問題を分割できない
分析	事象の分析	事象の原因と結果の関係を考え、具体的な関係性から抽象的なルールや原則を導き、筋道立てて書き出すことができる	事象の原因と結果の関係を考え、具体的な関係性に気づき、筋道立てて書き出すことができる	事象の原因と結果に関係性があることに気づくことができる	事象の中にある関係性に気づけない
抽象化	動作の抽出	目的に合わせて、必要最低限の動作だけを取り出すことができる	目的に合わせて、必要な動作を自分で考えて取り出すことができる	目的に合わせて、必要な動作を選択肢から選ぶことができる	抽出ができない
制御	各要素を用いたプログラムの作成	問題を解決したり創造的な表現をしたりするプログラムを作成できる(含：逐次実行、イベント、ループ、条件分岐、 並列性、変数)	問題を解決したり創造的な表現をしたりするプログラムを作成できる(含：逐次実行、イベント、ループ)	問題を解決したり創造的な表現をしたりするプログラムを作成できる(含：逐次実行、単純なループ)	プログラミングできない
一般化	一般化	過去の複数の解決済の問題から、解決策の類似性や関係性を見出し、共通する規則や原則を一般化したルールを見つけ出し、他の問題に当てはめて解決に利用できる	目の前の問題を解決済の問題と比較し、類似性や関係性を適用して問題解決に利用できる	解決済の事象の中に、類似性や関係性がある事象があることに気付ける	他の事象との関連性を見つけられない

表 6.2：Thinkron 適応版ルーブリック

思考活動	RubricProEEs該当項目	4	3	2	1
順序立て	動作の構築・関数化	目的を実現するため，複数の行動ラベルの最適な組み合わせを考え，並び替えることなく再現性のある手順を創作できる	目的を実現するため，複数の行動ラベルを，順次処理を利用して組み合わせることができる	問題解決のプロセスが複数の行動で構成されていることに気づき，与えられた行動ラベルを目的に合わせて並び替えようとしている	逐次実行を理解していない
分割	問題の細分化	すべての行動ラベルについて，複数の小さな動作ラベルに正しく分割できる	一つの以上の行動ラベルについて，複数の小さな動作ラベルに正しく分割できる	行動ラベルを二つ以上の小さな分割ラベルに分割できる	行動ラベルを小さな分割ラベルに分割できない
分析	事象の分析	事象の原因と結果の関係を考え，具体的な関係性から抽象的なルールや原則を導き，筋道立ててラベルやコードを組み合わせることができる（エラー数の大きな減少が確認できる）	事象の原因と結果の関係を考え，具体的な関係性に気づき，筋道立ててラベルやコードを組み合わせることができる（エラー数の減少が確認できる）	事象の原因と結果に関係性があることに気づき，修正しようとしている	事象の中にある関係性に気づけない
抽象化	動作の抽出	目的に合わせて，必要最低限のレシーバを用意し，必要最低限の行動ラベルだけを取り出すことができる	目的に合わせて，必要な行動ラベルだけを自分で考えて取り出すことができる	目的に合わせて，必要な行動ラベルを選択肢から選ぶことができる	抽出ができない
制御	各要素を用いたプログラムの作成	問題を解決したり創造的な表現をしたりするプログラム作成できる(含：逐次実行，ループ，条件分岐)	問題を解決したり創造的な表現をしたりするプログラム作成できる(含：逐次実行，ループ)	問題を解決したり創造的な表現をしたりするプログラム作成できる(含：逐次実行)	プログラミングできない
一般化	一般化	結果から，共通する規則や原則を一般化したルールを見つけ出し，複数回（ループ内含む）に当てはめて解決に利用できる	結果に対して，類似性や関係性を適用して問題解決に利用できる	結果の中に，類似性や関係性がある事象があることに気付ける（まとめるブロック内部を構成している）	結果から関連性を見つけられない

6.3 実験概要

Thinkron を用いたプログラミング学習を通じて，学習者の操作履歴および，操作回数の取得を試みる．

出題する問いにおいては，異なる難易度に対する傾向の違いが現れるかどうかを確認するため，低難度，高難度で一種類ずつ用意した．ただし難度の高低差が極端すぎると，決まりきった結果になると予想される．そこで，内容に関しては Benesse の評価規準の知識・技能[4]に設定されている学年段階（低学年—高学年）の目標に沿うように作成している．なお，本実験における参加児童の学年が小学校 3 年生以上でありかつ，中学年の人数が多いことから，低難度を中学年向け，高難度を高学年向けの問いとして作成した．具体的には低難度の問いは順次処理を基本とし，高難度の問いでは条件分岐や帰納的な内容としている．また問いの難易度設定に合わせ，各思考活動における提示の仕様も変更している（表 6.3）．

表 6.3：難度別ギミック仕様

思考活動	低難度（中学年向け）	高難度（高学年向け）
●抽象化	ラベルをすべて使用	ダミーラベルも配置
●順序立て	順次的な問いを提示	帰納的な問いを提示
●分割	順序に沿った限定的ラベル	ダミーを含む帰納的ラベル
●制御	順序に沿った限定的コード	複数のカテゴリから選択
●一般化	関数ブロックでまとめる指示	ブロック総数を減らす指示
●分析	間違い箇所の指摘	間違い総数の指摘

実際に作成した問いは，逐次処理を基本とした探索問題であるが，どちらの難度にも共通して，算数的要素を加えている．これは，第一章でも述べたように，プログラミング的思考を教科横断的に育むという観点から取り入れた．低難度については，スタートからマップ上の自動販売機に向かい，そこでジュースを買った後にゴールする内容とした．算数的な要素として，コーディング課題において買い物算を行う．高難度については，目的地までの最短経路を選択してゴールする内容とした（ただし，一度寄った目的地には行けない）．算数的な要素として，手順課題において道のりの比較を行う．

作成した刺激の一部分を図 6.1，6.2 に示す．図は手順課題であるが，抽象化を行うリストの中身や順序立てを考える際の問題の提示の仕方が異なる．

図 6.1：低難度の問い（手順課題）

図 6.2：高難度の問い（手順課題）

6.4 実験手順・実験計画

最初に低難度，二回目に高難度の問いに取り組む形で実験を行う．ただし Thinkron の操作性に支障をきたさないように，事前に練習課題を設ける形で執り行った．また，各思考課題の行う時間について，今回は全行程 30 分とし，手順課題 5 分，動きに分ける課題 10 分，コーディング課題 15 分で行った．

以上を踏まえ，学童保育所において 4 日間のワークショップを開催した．小学校 3 年生～6 年生 12 名を対象とし，1 日当たり 1 時間（準備時間や答え合わせ等含む）の日程で本実験を実施した．全日程の内，はじめの 2 日間は Thinkron に対する操作方法を学習する時間にあてている．日程の内訳は，表 6.4 のとおりである．

表 6.4：実験全日程

日程	内容
1日目	基本操作の学習
2日目	
3日目	低難度問い実施
4日目	高難度問い実施

本実験の前後においては，学習者の設計段階の能力を見るプレテスト，ポスト

テストを実施した．内容としては，架空のマップにおいて，ロボットに指定の食べ物を買ってきてもらい，自宅に戻ってくるまでの手順を指定のテスト用紙に書き出すものである．各テストはマップや設定が違うものの，自宅から買い物を行う地点までの往路手順数は変わらないものとした（図 6.3，6.4）．

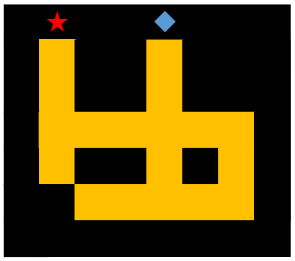


図 6.3：プレテストマップ

図 6.4：ポストテストマップ

本実験後には，Thinkron に対する感性評価アンケートを実施した．Thinkron に対する面白さ，問いの難しさ，操作の難しさについてリッカート尺度による質問紙を作成して回答させた．

6.5 実験結果

6.5.1 実施状況

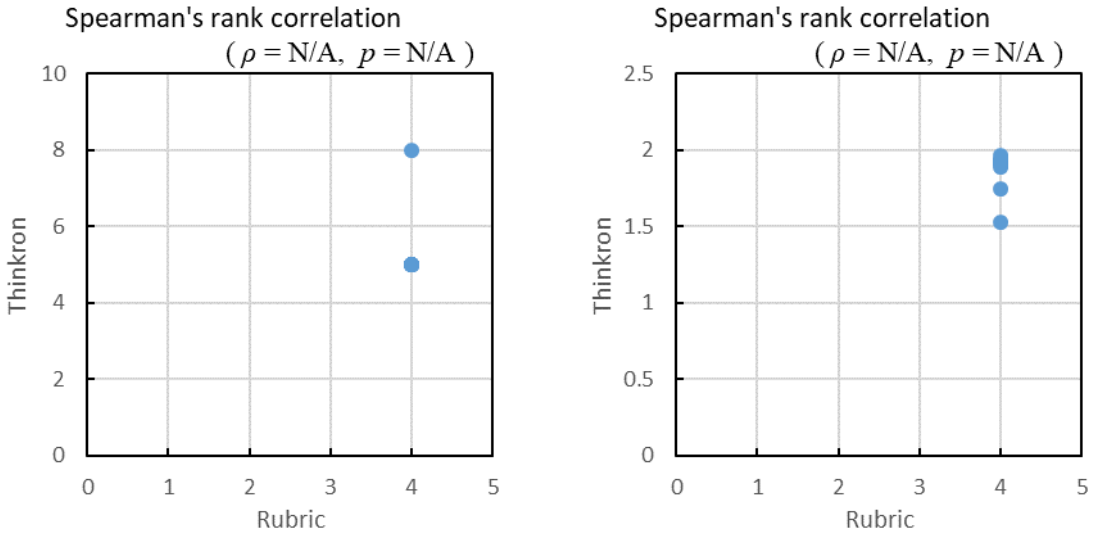
本節では，学童保育所の小学生 12 名（内，プログラミング経験者 3 名）に対して行った Thinkron を用いたプログラミング学習の結果について記載する．なお，実施時の学習状況（早退，欠席等）により，低難度の問いでは 8 名，高難度の問いでは 9 名における学習結果について述べる．ルーブリックの評価については，本学長谷川研究室の他の学生に依頼した．

6.5.2 手順課題

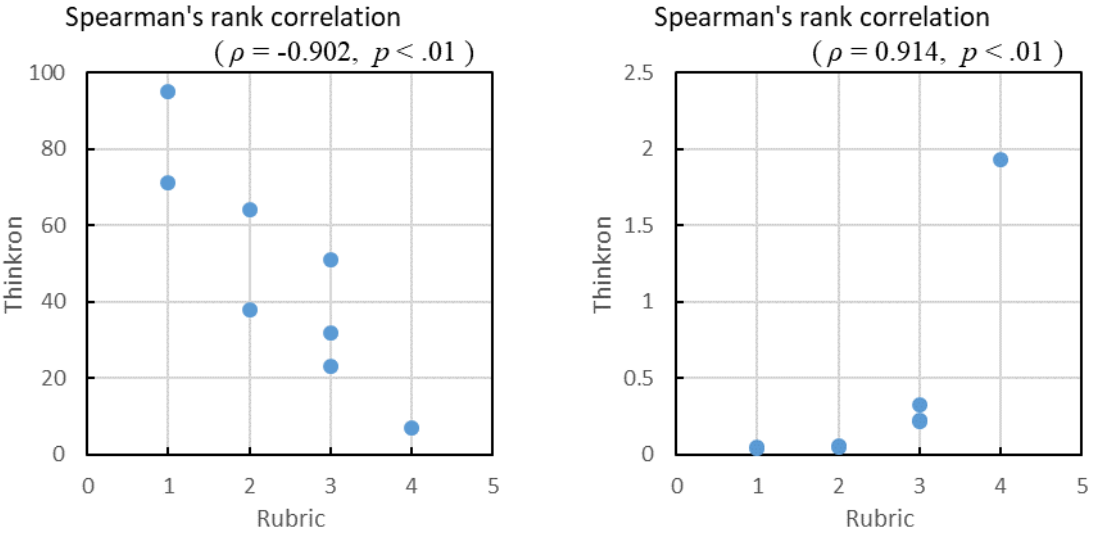
低難度と高難度の問いにおける抽象化の分析結果を図 6.5 に示す．

低難度と高難度を比較すると，低難度のルーブリック評価が最大値の 4 に集中している．これは，全員がこの問いに必要なラベルを理解し，適切に抽出できていることを示す．また，低難度では多くが一度の実行までに正解していることから，量データ（操作数）に対するばらつきがあまり見られない．その点，質データにおいてはクリアタイムによる得点のばらつきが見られる（量，質データ間の分散比 $F = 51.4$, $p < .01$）．このことから，同じルーブリック評価内でも解答

の差を検出することが可能である．なお，相関係数はルーブリックによる得点の差がないことから求められなかった．



(a) 低難度量データ　　(b) 低難度質データ



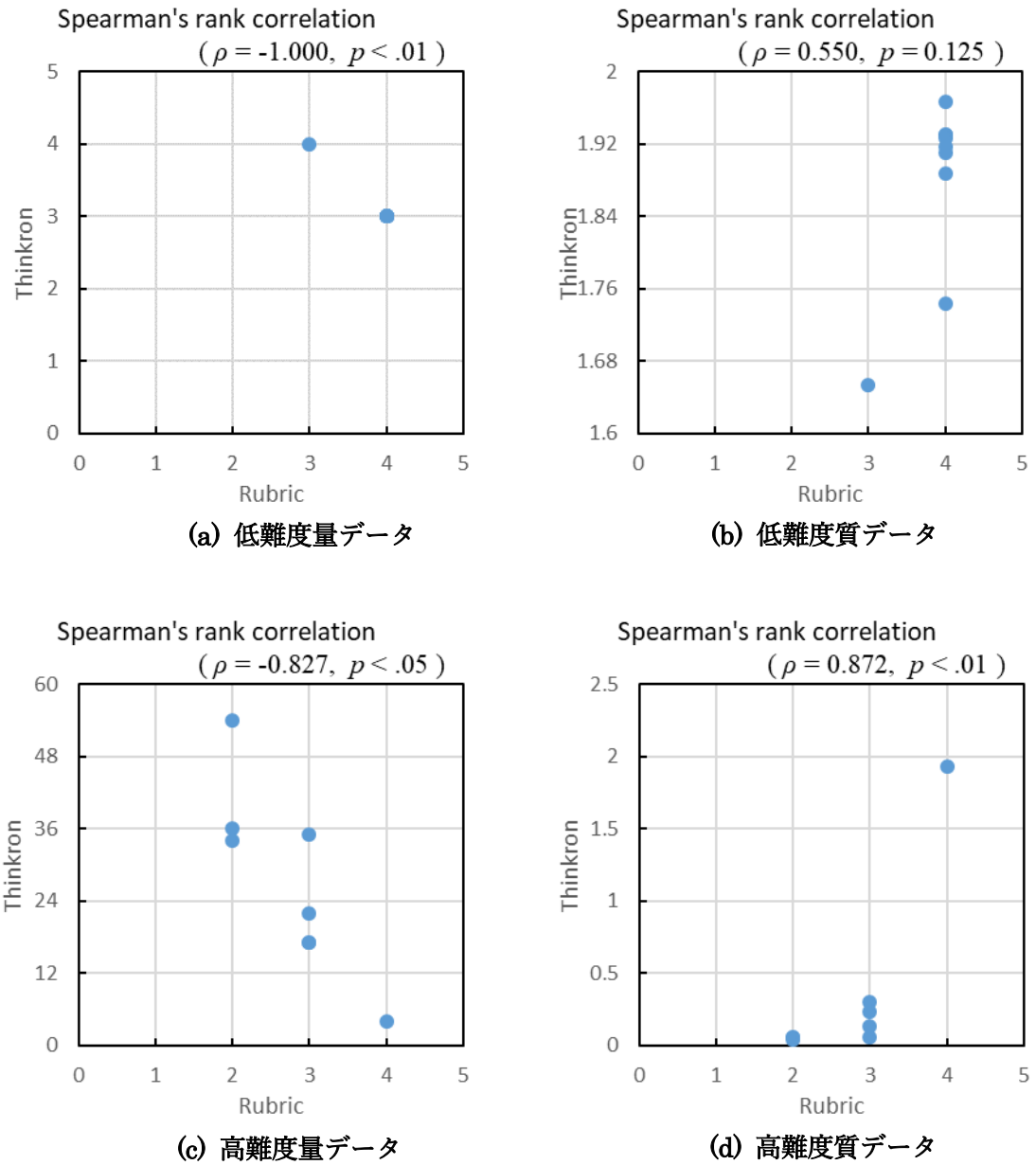
(c) 高難度量データ　　(d) 高難度質データ

図 6.5：抽象化の散布図

高難度の解答においては，ルーブリック評価に差が見られた．量データにおいては操作回数が少ないほどルーブリックによる評価が有意に高く，出題内容の対応力や難度への思考力の差がうかがえる（ρ = -0.902, p = 0.002）．質データ

についてもスピアマンの順位相関係数より高い相関が認められた（ρ = 0.914, p = 0.001）．このことから，高難度の問いにおける抽象化の定量化においてはRubric ProEEsの評価の概念に沿う結果となった．

次に，低難度，高難度の問いにおける順序立ての分析結果を図 6.6 に示す．



(a) 低難度量データ　(b) 低難度質データ

(c) 高難度量データ　(d) 高難度質データ

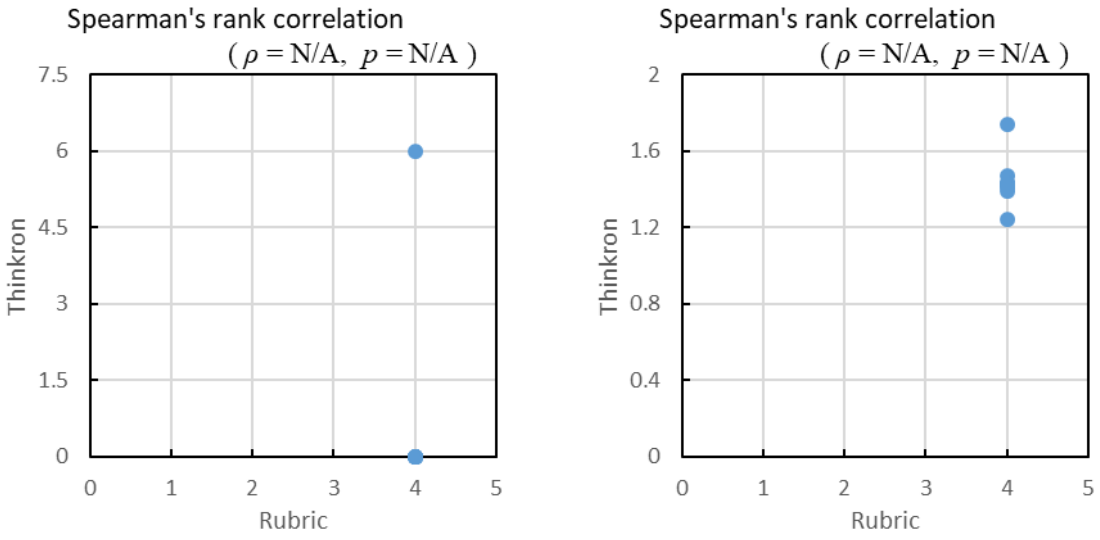
図 6.6：順序立ての散布図

低難度については．量データに相関がみられるがルーブリック評価が 3 点の者は一人しかいなかった為，疑似相関である．質データにおいても相関係数よ

り中程度の相関が認められるが，ルーブリック評価の同点者が多いため，質データに対する詳細な評価は得られなかった．

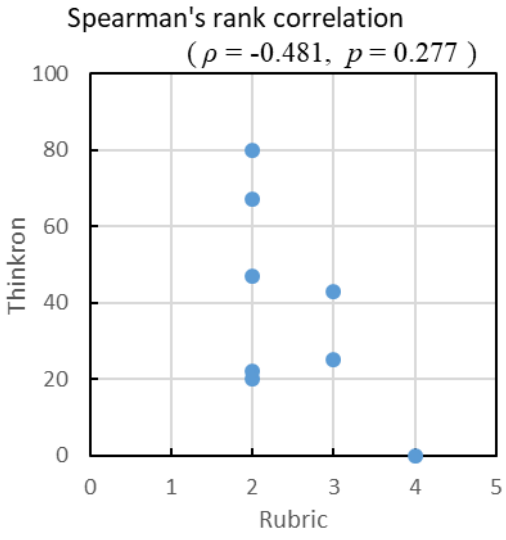
一方，高難度についてはルーブリックの評価が分散し，量，質データ共に高い相関が認められた（量データ：ρ = -0.827，p = 0.011，質データ：ρ = 0.872，p = 0.005）．このことから，高難度の問いにおける順序立ての定量化においてはRubric ProEEs の評価の概念に沿う結果となった．

最後に低難度，高難度の問いにおける分析（手順）の分析結果を図 6.7 に示す．

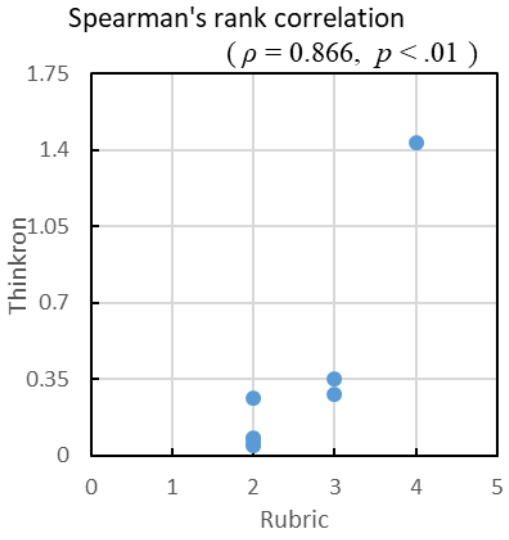


(a) 低難度量データ

(b) 低難度質データ



(c) 高難度量データ



(d) 高難度質データ

図 6.7：分析（手順）の散布図

低難度については，抽象化の結果と同様，ルーブリック評価の偏りにより，量データでのばらつきがあまり見られない．質データ上では，クリアタイムによる得点の差が見られる（量，質データ間の分散比 $F = 242.3$，$p < .01$）．

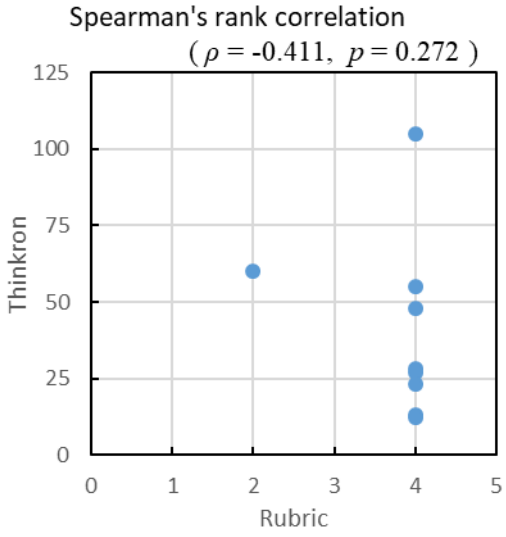
高難度については，質データにおいて高い相関が認められた（$\rho = 0.866$，$p = 0.005$）．このことから，高難度の問いにおける分析（手順）の定量化においては Rubric ProEEs の評価の概念に沿う結果となった．

6.5.3 動きに分ける課題

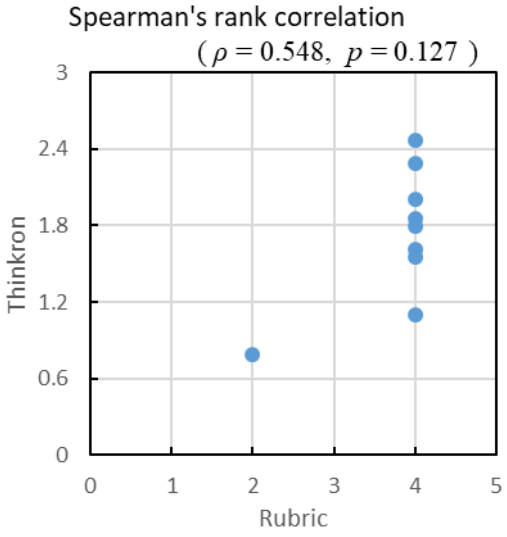
低難度，高難度の問いにおける分割の分析結果を図 6.8 に示す．

低難度においては，量データにおけるルーブリック評価の差異はほとんどないものの，問いを一度でクリアした被験者は少なく，操作回数に差が出ている．質データにおいては相関係数より若干の相関が認められるが，ルーブリック評価の同点者が多いため，順序立て同様，質データに対する詳細な評価は得られなかった．

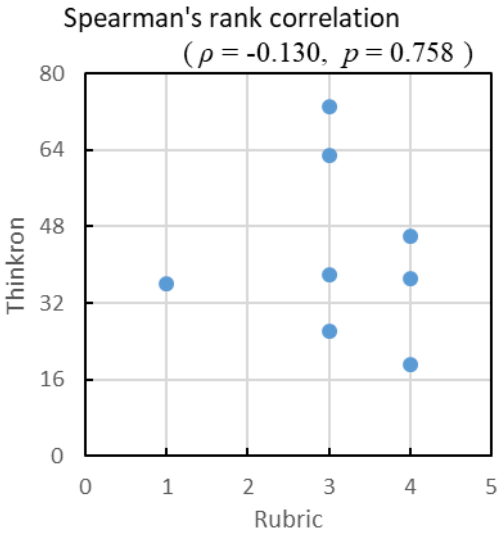
高難度においては，質データにおいて高い相関が認められた（$\rho = 0.913$，$p = 0.002$）．このことから，高難度の問いにおける分割の定量化においては Rubric ProEEs の評価の概念に沿う結果となった．



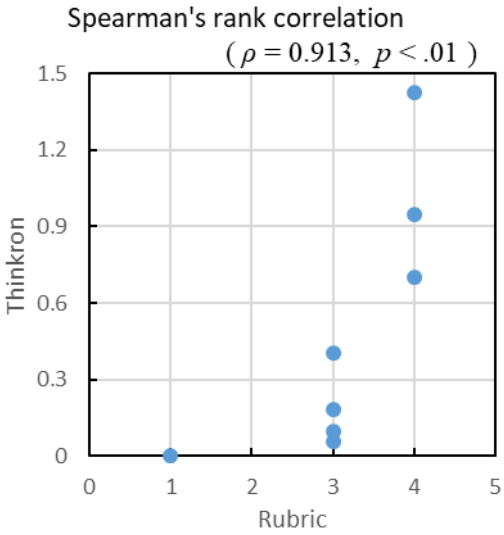
(a) 低難度量データ



(b) 低難度質データ



(c) 高難度量データ

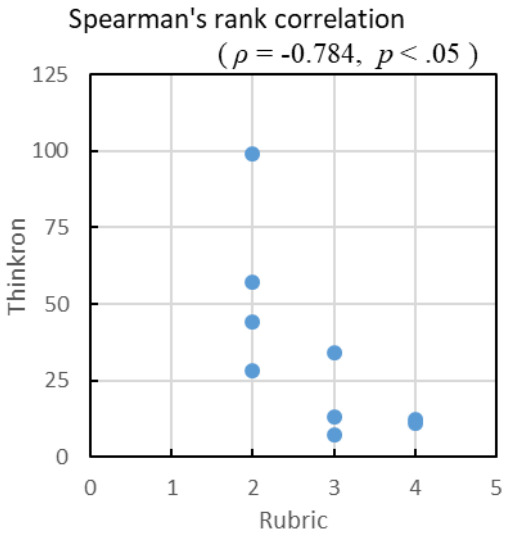


(d) 高難度質データ

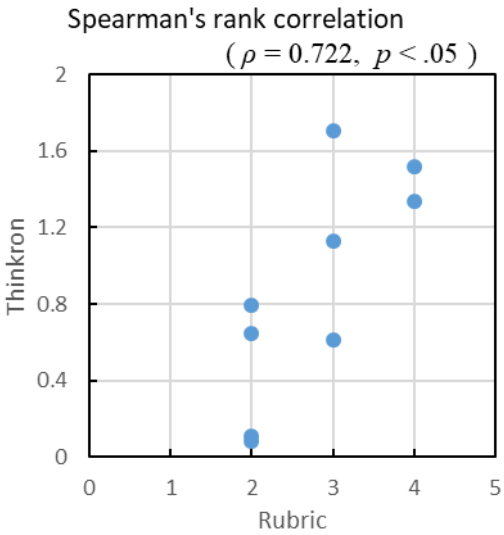
図 6.8：分割の散布図

次に，低難度，高難度の問いにおける分析（動きに分ける）の分析結果を図 6.9 に示す．

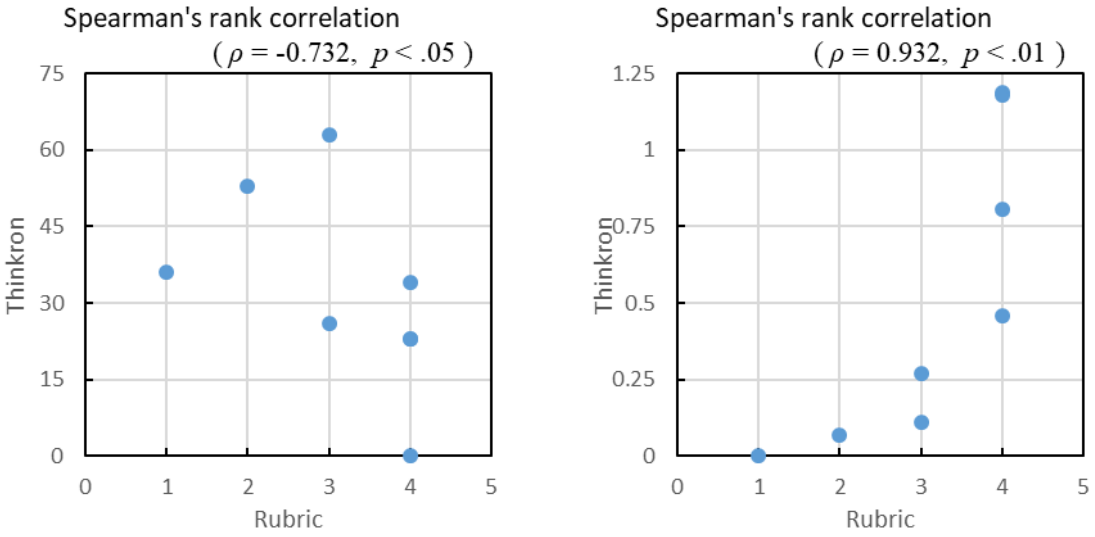
低難度の量データにおいて，操作回数が減るほどルーブリック評価の値が有意に高くなる傾向が見られた（ρ = -0.784, p = 0.012）．低難度での分割における思考の能力差はあまり見られなかったが，同じ分割能力の中でも目的の動作に近づけていく思考（分析）の質には差があることが操作回数から見てとれる．質データにおいても高い相関が認められた（ρ = 0.722, p = 0.028）．



(a) 低難度量データ



(b) 低難度質データ



(c) 高難度量データ　　(d) 高難度質データ

図 6.9：分析（動きに分ける）の散布図

高難度においても量，質データ共に高い相関が認められた（量データ：ρ = -0.732，p = 0.039，質データ：ρ = 0.932，p = 0.001）．

これらのことから，低難度，高難度の問いにおける分析（動きに分ける）の定量化においては Rubric ProEEs の評価の概念に沿う結果となった．

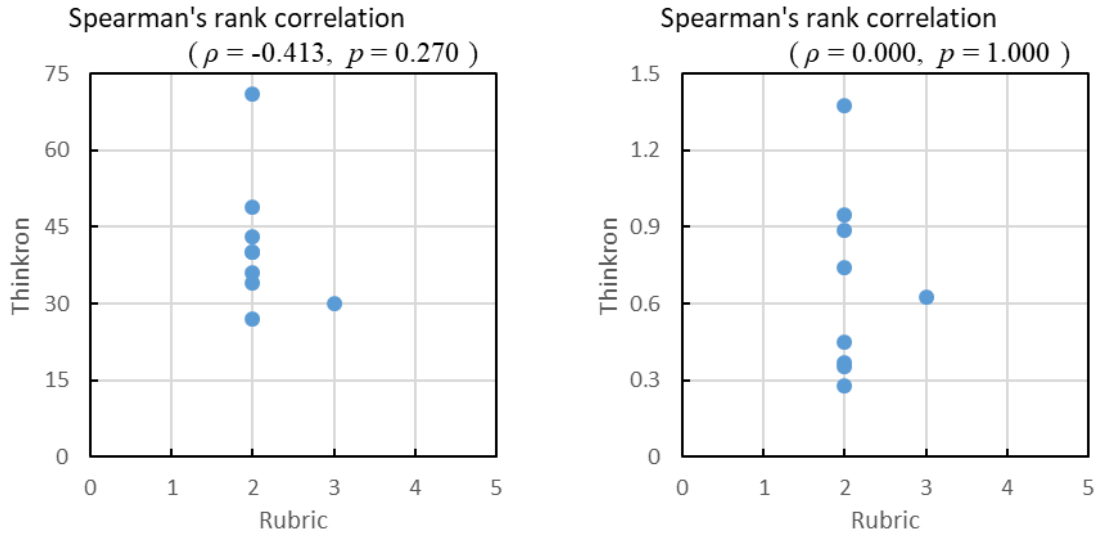
6.5.4 コーディング課題

低難度，高難度の問いにおける制御の分析結果を図 6.10 に示す．

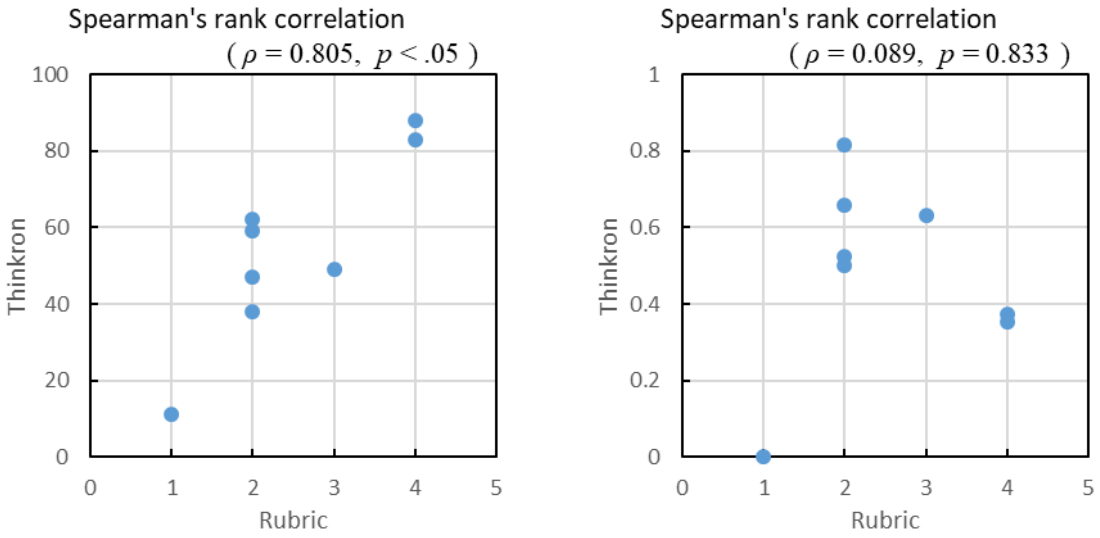
低難度においては，量，質のデータ共に相関が見られなかった．一方で高難度では，量データのみ正の高い相関が得られた（ρ = 0.805，p = 0.016）．

質データとルーブリック評価に相関が認められない原因として，ルーブリックの評価段階が技術的な達成度を見ようとしている点にある．今回，Thinkron 上の操作の評価としては，操作回数と正解数をベースとした質の評価に特化しており，表現技法の判定評価までは考慮できていなかった．なお，実験までの準備段階として事前に条件分岐やループ処理の方法をレクチャーし，問いの中ではこれらの制御パターンを意識したマップや逐次処理の組み合わせを提供している．さらに，高難度の問いでは完全解答までに必ずループ処理や条件分岐を使わなければ，解決できないようになっている．結果，高難度においては，演繹的な処理に気づいて適応した被験者が，ルーブリック評価より数名確認できる．また，

高難度の量データに相関が認められる要因としては，ルーブリック評価の高得点者が大量の操作による試行錯誤の末に，演繹的な処理にたどり着いた可能性がある．このことから，比較的プログラミング経験が浅くかつ，逐次処理以外の複雑な制御を推し量る上では，コードを組み上げようとする操作回数による評価も検討の余地がある．



(a) 低難度量データ　　(b) 低難度質データ



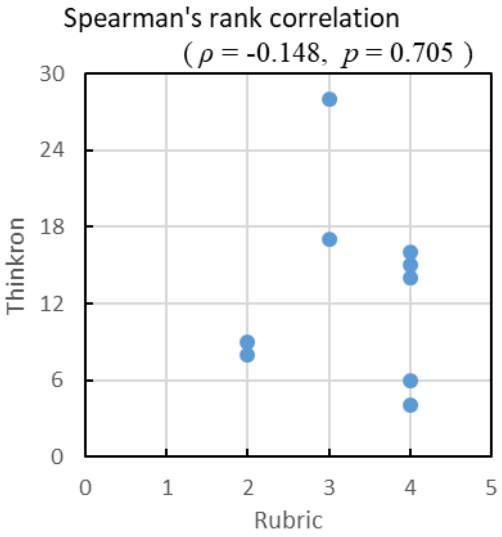
(c) 高難度量データ　　(d) 高難度質データ

図 6.10：制御の散布図

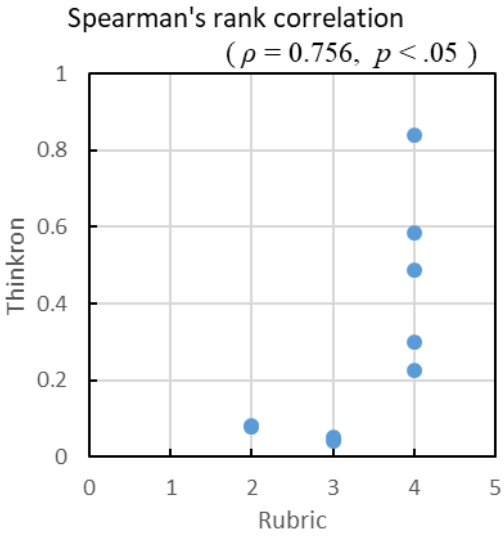
次に，低難度，高難度の問いにおける一般化の分析結果を図 6.10 に示す．

低難度，高難度共に質データにのみ高い相関が認められた（低難度：$\rho = 0.805$，$p = 0.016$，高難度：$\rho = 0.850$，$p < .01$）．

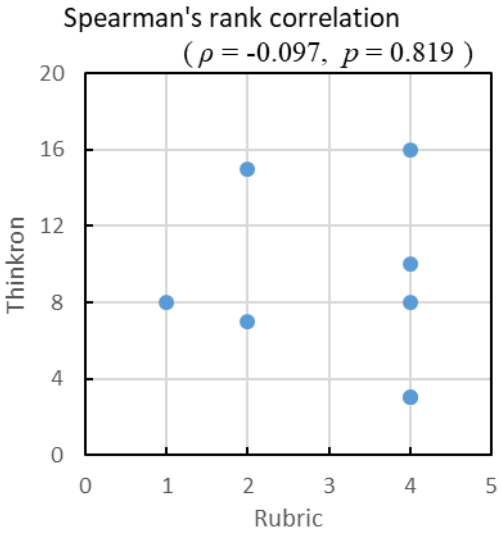
よって低難度，高難度の問いにおける一般化の定量化においては Rubric ProEEs の評価の概念に沿う結果となった．



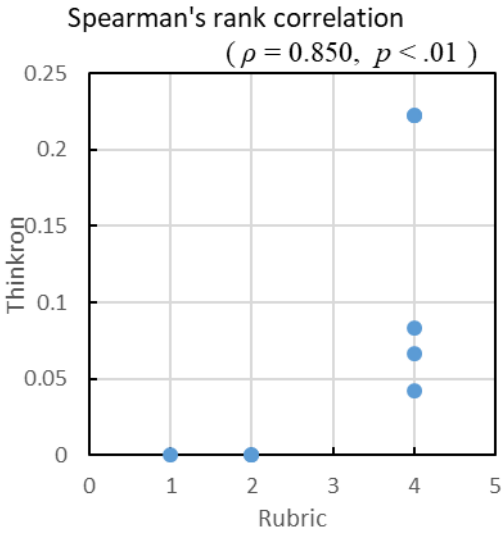
(a) 低難度量データ



(b) 低難度質データ



(c) 高難度量データ



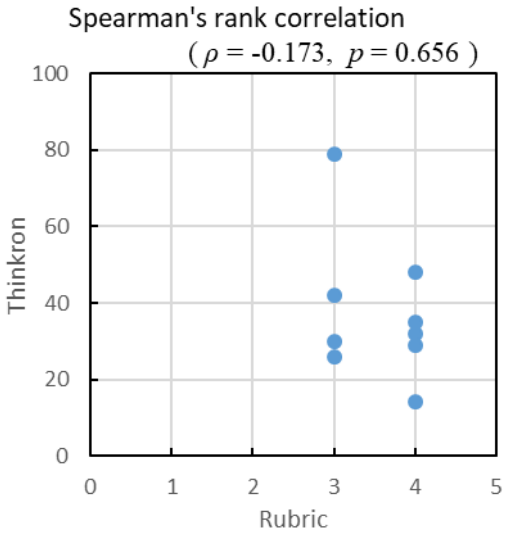
(d) 高難度質データ

図 6.11：一般化の散布図

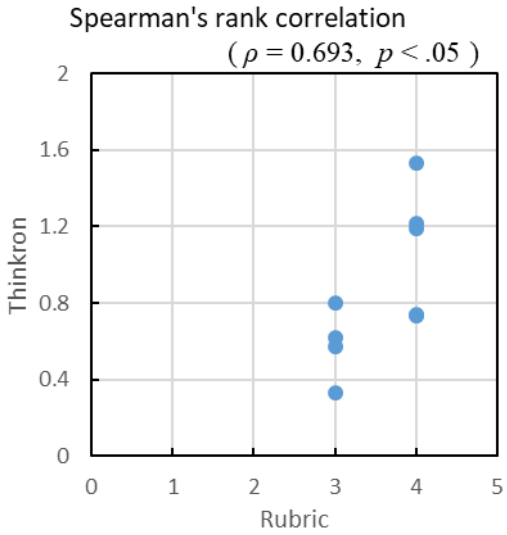
最後に低難度，高難度の問いにおける分析（コーディング）の分析結果を図 6.12 に示す．

低難度においては，質データにおいて高い相関が認められた（ρ = 0.693，p = 0.039）．高難度においても質データにおいて相関の傾向がある．

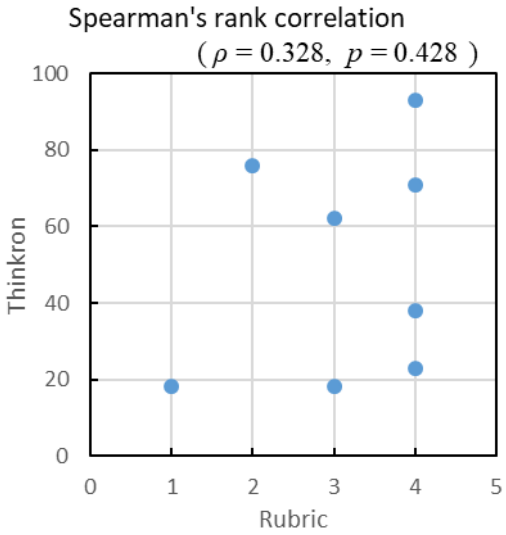
よって低難度，高難度の問いにおける分析（コーディング）の定量化においては概ね Rubric ProEEs の評価の概念に沿う結果となった．



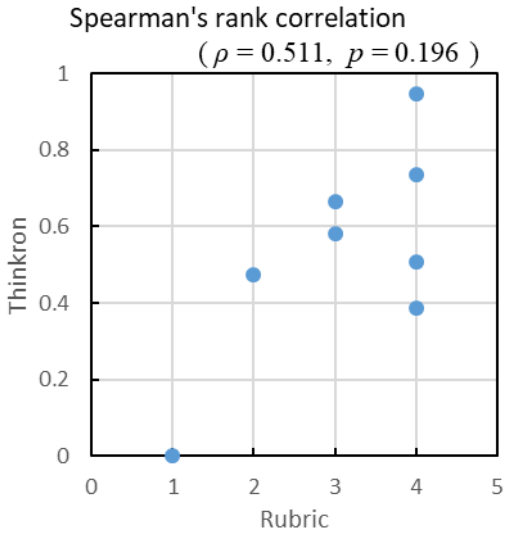
(a) 低難度量データ



(b) 低難度質データ



(c) 高難度量データ



(d) 高難度質データ

図 6.12：分析（コーディング）の散布図

6.6 性能評価実験のまとめ

プログラミング的思考の思考要素に基づく思考活動において，各思考活動の定量化指標として量データと質データを定義し，Rubric ProEEs の評価指標に則って各データとの相関を分析した．結果，質データについては思考活動「制御」を除き，概ねルーブリック評価に沿うことがわかった．これにより相関のある思考活動については，プログラミング的思考の能力の一指標として有効性が示された．ただし，問いの難度によってルーブリック評価が偏った場合の質データの得点差の指標が有意であるかどうかは判断できなかった．

6.7 プレテスト・ポストテストの結果

6.3 節で示したように，Thinkron による学習を通じることで設計段階の能力に変化があるかテストを行った．本実験の各難易度において，設計フェーズまでを正常に記録できた被験者 9 名におけるプレテスト，ポストテストの結果を図 6.13 に示す．図の結果は，テスト用紙に書き出された手順の総数の推移である．

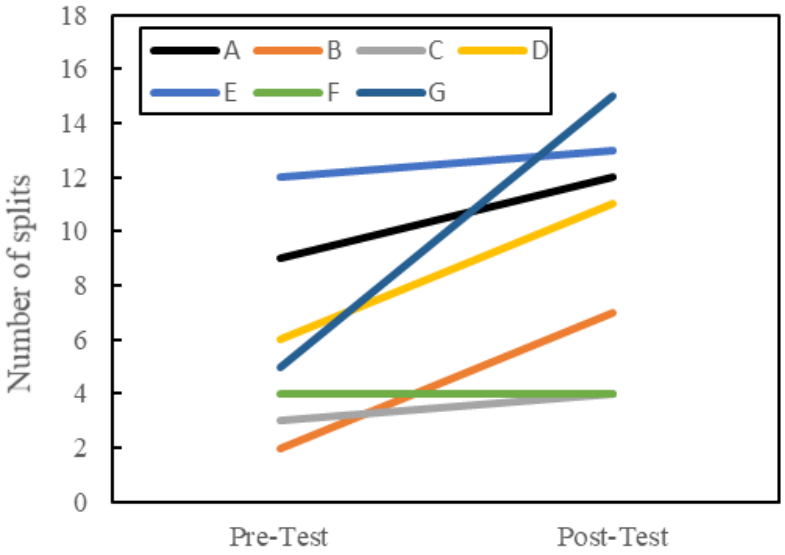


図 6.13：分解能力の結果

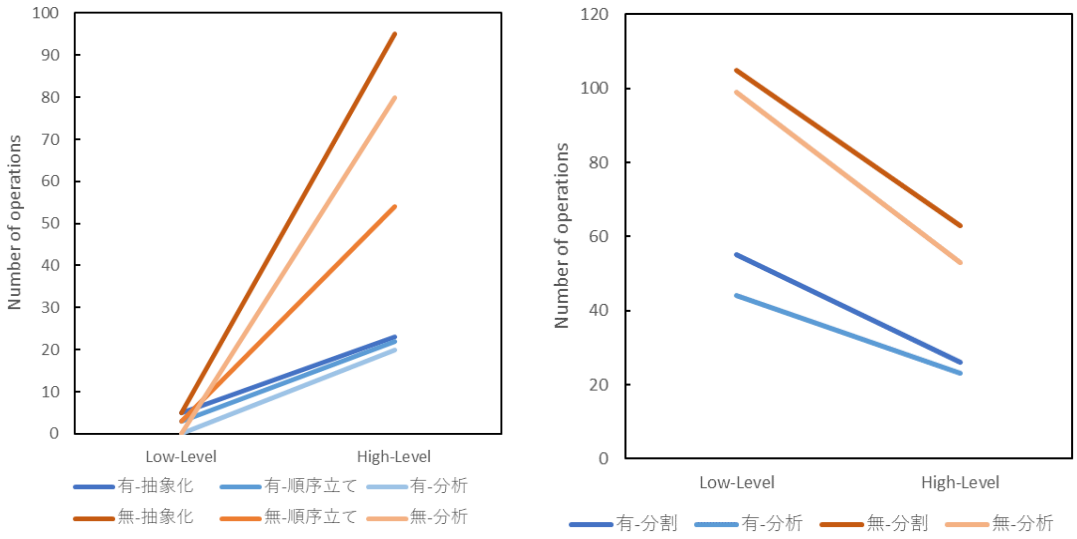
図 6.13 より，F のみ回答に変化が見られず，その他は書き出された手順の総数が向上している．この結果に対し Wilcoxon の符号付き順位検定を行ったとこ

ろ，有意水準 5%で有意差が認められた（$p = 0.027$）．このことから単純な分割能力は，総じて向上しているといえる．分割の内容についても，より詳細化されているものが多くあった（付録参照）．

この結果を受け，分割能力の伸びている被験者 G と変化のない被験者 F について着目し，量，質データの観点から 2 者間の特徴の違いについて比較を行った．しかし，F については実験時のモチベーションの落差が大きいこともあり，操作過程における明確な要因が得られなかった（結果としては F の方が，分割能力が高かった，付録参照）．そこで次点に伸びの低い被験者 C との比較結果について述べる．

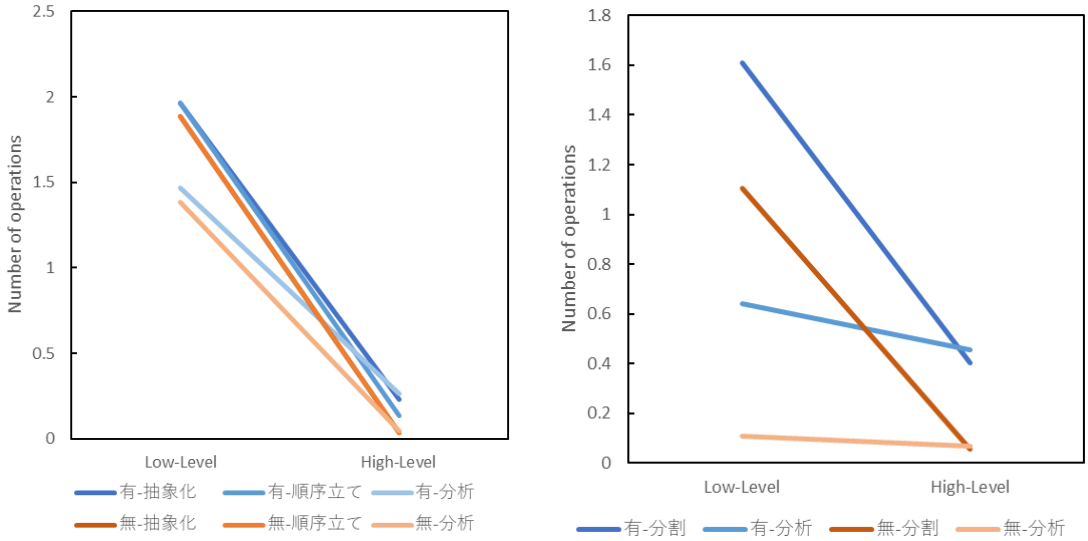
図 6.14，6.15 は被験者 G と被験者 C における難度間の得点推移である．図中，「有-」と示されているのが被験者 G，「無-」と示されているのが，被験者 C である．なお被験者 G はプログラミング経験者，C，F は未経験者である．

操作回数に関しては，被験者 C の回数が総じて高く挑戦的に取り組んでいることがうかがえる．一方，被験者 G に関しては操作回数が少なく，質においては C を上回っていることがわかる．なお動きに分ける課題では，両者とも低難度をクリアし，高難度はタイムアップしている．この背景を踏まえると G はすぐにはアウトプットせず，比較的推論しながら操作を行っていると考えられる．これについては，動きに分ける課題における分析の質の差に最も現れていることがわかる．



(a)手順課題　　(b)動きに分ける課題

図 6.14：得点推移（量）



(a)手順課題　　(b)動きに分ける課題

図 6.15：得点推移（質）

今回の 2 者間のケースにおけるテストの差の要因としては，視覚的な結果の表現がある Thinkron では，施行を繰り返すことで模索的に結果を導き出すことができるが，紙媒体でのテストではそれができないことが第一に考えられる．そのため，分割能力が引き出されなければ，紙面上でのアウトプットも難しいものとなる．被験者 G の結果や被験者像を踏まえると，元々ある程度の分割能力を有しており，数回にわたる Thinkron の操作を通じて分割操作の目的，意義を理解し，テスト下で分割能力を高度に引き出すことができたと考えられる．

6.8 感性評価の結果

6.8.1 実施内容

6.3 節で示したように，Thinkron に対する面白さ，問いの難しさ，操作の難しさについて，実験後に感性評価アンケートを実施した．対象者はプレテスト，ポストテストを行った 9 名である．表 6.5 に質問項目を示す．

表 6.5：質問項目

質問番号	内容	回答形式	備考
Q1	シンクロンのたのしさはどうですか	4件法	——
Q2	シンクロンのむずかしさはどうでしたか	5件法	難度別に調査
Q3	シンクロンのそうさはどうでしたか	5件法	——

6.8.2 Thinkron に対するモチベーション評価（Q1）

Q1 において，児童に対する Thinkron の受け入れやすさを調べるため，Thinkron を使用した際の楽しさについて回答させた. つまらない, すこしつまらない, すこしたのしい，たのしいの 4 件法で行い，中間回答が無いように構成した．回答の度数をネガティブ（Boring），ポジティブ（Funny）の 2 群に分けた結果を図 6.16 に示す.

回答度数を正確二項検定で分析したところ，p =0.09 で有意傾向となった．このことから，Thinkron を使用する際の受け入れやすさはある程度認められていることがわかる.

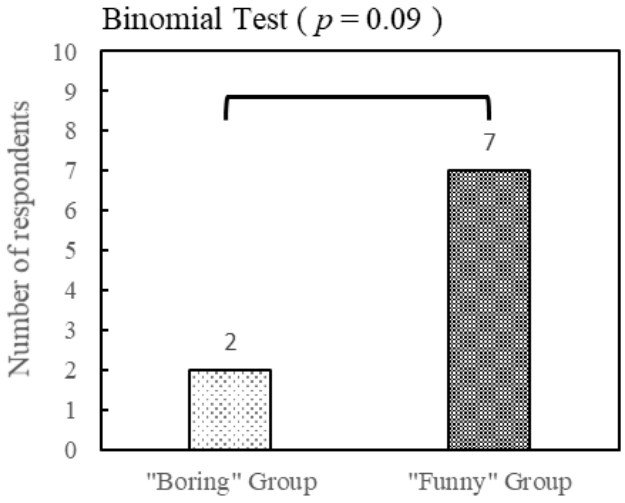


図 6.16：Thinkron の楽しさについての回答度数

6.8.3 問いの難度評価（Q2）

Q2 において，実験に用いた問いの難度に差があったかどうか調べるため，低難度, 高難度それぞれを解いた際の難しさについて回答させた．Thinkron を使用した際の楽しさについて回答させた. 選択肢として, かんたん, すこしかんたん, ちょうどいい，すこしむずかしい，むずかしいの 5 件法で行った．結果を図 6.17，6.18 に示す.

低難度と高難度に対し Wilcoxon の符号付き順位検定を行ったところ有意水準 5%で有意差が認められた（p = 0.02）．難易度の偏りはあるものの異なる難易度を提供できた.

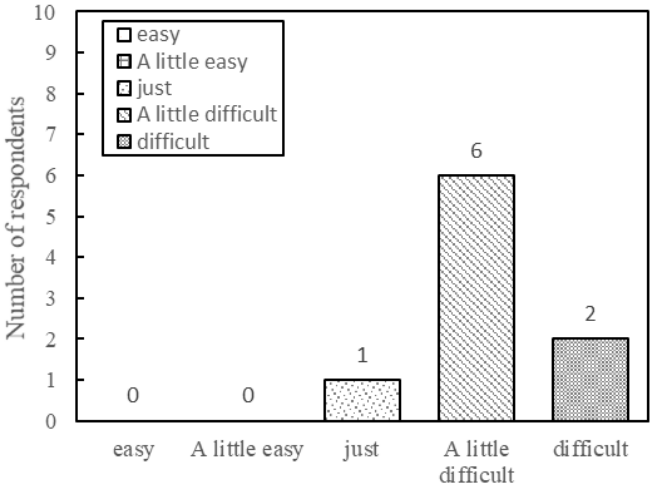


図 6.17：低難度の問いの難しさについての回答度

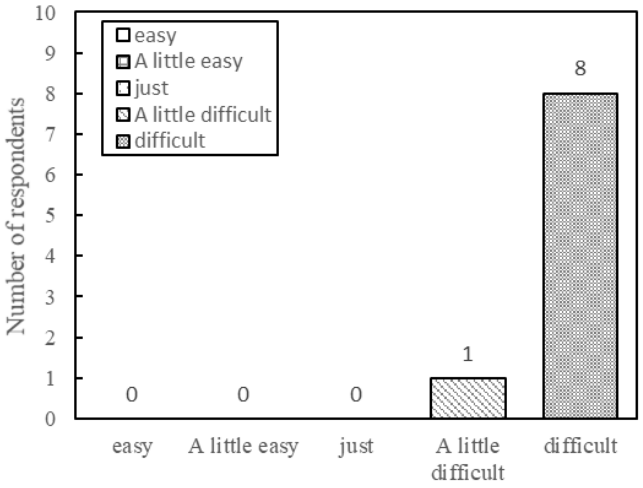


図 6.18：高難度の問いの難しさについての回答度

6.8.4 Thinkron の操作難度の評価

Q3 において，児童に対して Thinkron の操作性能が相応しいものであるか調べるため，Thinkron 全体の操作の難しさについて回答させた．選択肢として，かんたん，すこしかんたん，ちょうどいい，すこしむずかしい，むずかしいの 5 件法で行った．結果を図 6.19 に示す．

度数としては，“ちょうどいい”と“すこしむずかしい”が同数となり，“すこしかんたん”と感じた被験者は 1 名のみであった．ただし，“かんたん”,“むずかしい”に

回答したものはおらず，現段階における UI としては難しすぎず，簡単すぎない適切な操作性であったといえる．

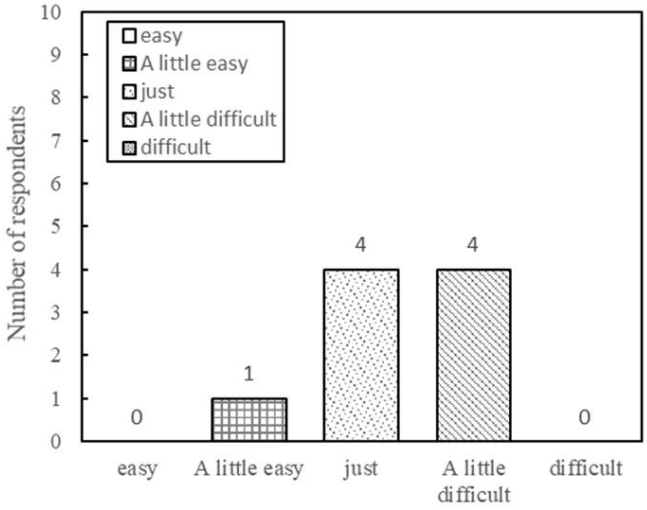


図 6.19：操作性に関する回答度数

第7章 おわりに

7.1 本稿のまとめ

本稿では小学校におけるプログラミング教育において，プログラミング学習中に行われる思考「プログラミング的思考」を観測可能な操作（思考活動）として展開し，定量的な評価を行う学習環境「Thinkron」の構築を行った．定量化に関しては，学習者の操作ログを取得し，単純な操作回数である量データと，解答結果より導き出される質データをそれぞれ定義し，これを Thinkron におけるプログラミング的思考の能力の指標として位置付けた．この指標の有効性を調べるべく，Rubric ProEEs の評価指標に則ったルーブリック評価と相関関係を調べたところ，思考活動「制御」を除くすべての思考活動において相関関係が認められた．これにより，相関関係が認められる思考活動についてはプログラミング的思考の能力を推し量る上での一つの指標として有効性が示唆された．また，Thinkron を用いた学習を通じることで，分割能力の向上効果が認められた．設計面を意識づけた学習教材の例は少なく，今回の設計フェーズにおける Thinkron の学習効果については，プログラミング的思考を育むシステムとしての有効性を持つと同時に，プログラミング教育に有効な設計段階を捉えた学習例の一つとして位置付けられる．本実験後には，Thinkron における使用感についての感性評価アンケートも行った．Thinkron に対する面白さ，問いの難しさ，操作の難しさについて調査した結果，児童に受け入れやすく適した操作性であることが分かった．問題の難しさについては，難しさのバイアスが高いものの，設定難易度としては優位に区別される出題であったことが示された．

7.2 本学習システムの在り方

Thinkron は学習者のプログラミング的思考の育成を行うため，個々の形成的な評価を基にプログラミング的思考の苦手部分を洗い出し，教師の提供する問いと Thinkron の思考課題によって，着実にプログラミング的思考を育むことを想定している．そのため，Thinkron はコーディング中の一部の思考を切り出すことはせず，あくまでも学習フェーズである設計段階やトライ&エラーも含んだプログラミング全体の流れから，思考活動の抽出を行っている．単純に学習者にプログラミングを行わせて思考を抽出する場合，必ずしも目的のプログラミング

的思考の構成要素が育まれるとは限らず，取り上げる題材によっては各構成要素が反映されたものを作成していくのは難しい．またそれらの要素が含まれているのかどうかを確認するのも困難である．だからこそ，一つ一つの構成要素に焦点を当てた操作を行う課題を持ち，それらについて定量的評価を行う Thinkron は，プログラミング的思考を育むことに特化した学習環境であると言えよう．

7.3 今後の課題

今回の実験において，思考活動の「制御」のみ，有意な差は認められなかった．6.5.4 項でも述べたが，コーディング課題中の条件分岐等の演繹的な思考を捉えることができていなかったことが原因である．そのため，今後は対象コードの抽出方法と，それに対する得点化について検討する必要がある．また，今回のルーブリックとの相関関係については，あくまで Rubric ProEEs に基づくものであり，他のルーブリックとの評価には適応できない恐れがある．検証人数も限られた中で行っているため，今後は本実験で得られたデータをベースとして，より指標の精度を上げられるように検証を繰り返していく必要がある．今回は質データをメインに取り上げているが，量データにおいても試行回数や時間，一定時間内の操作記録などを組み合わせることで，学習者の粘り強さや挑戦的思考を推し量る指標になり得ると考える．

学習環境の構想全体から現段階を捉えると，学習者側のシステムのみが開発できたに過ぎない．今後の展開としては量データ，質データを見やすい形で表現する学習結果のページの構成や，教師サイドのプラットフォームを構築していく．またコンテンツに関しては，本研究では迷路探索のタスクの実装を行ったが，今後教科横断的な対応をしていくには，より自由なタスク処理ができる必要がある．買い物算の他にも多彩なイベント処理を追加し，自由度のある実行画面の設計を行うことで，問いのバラエティに富んだ表現度の高いシステムを目指したい．

研究業績

・小林祐太, 長谷川忍：プログラミング的思考を操作として展開・評価する学習環境の提案, 信学技報, vol. 120, no. 167, ET2020-12, pp. 13-18, 2020 年 9 月.

謝辞

本研究を進めるにあたり主指導員教員である長谷川忍准教授には日頃より研究方針について御指導を賜りましたことを心より感謝申し上げます．また太田光一助教におかれましても研究に対する助言や激励頂きまして心より感謝申し上げます．

I22 ゼミ室の同期研究メンバーである堀口優氏，中川恵輔氏からは研究手法における助言をいただき，柿本氏，高橋氏からは実験において大変ご助力いただきましたことを深くお礼申し上げます．

お忙しい中実験にご協力いただきました，特定非営利活動法人学童会つるぎピノキオクラブ様，オンライン学習塾 MUSASHI-Pro 様におきましても心より感謝申し上げます

参考文献

[1] 文部科学省，小学校 学習指導要領（平成 29 年告示），https://www.mext.go.jp/content/1413522_001.pdf，参照 Aug,3.2020.

[2] 文部科学省，小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ），https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/053/siryo/__icsFiles/afieldfile/2016/07/08/1373901_12.pdf，参照 Aug.3,2020.

[3] 小泉力一，小田理代，後藤義雄，星千枝，永田衣代，小学校段階におけるプログラミングで育成する資質・能力の評価規準開発，教育システム情報学会,42 ,pp.435-436,(2017).

[4] Benesse，プログラミングで育成する資質・能力の評価規準（試行版），https://benesse.jp/programming/beneprog/wp-content/uploads/2018/08/ver2.0.0.pdf，参照 Aug.3,2020.

[5] 西野和典，田中太志朗，近藤秀樹，山口真之介，大西淑雅，プログラミング的思考を評価するルーブリックの作成とその試用，教育システム情報学会,43,pp.141-142,(2018).

[6] 齋藤大輔，佐々木綾奈，鷲崎弘宜，深澤良彰，武藤優介，田村麻里子，西澤利治，小学生を対象としたプログラミング教育のためのルーブリックの提案，STEM 教育研究，Vol.1，pp.41-51(2018).

[7] 文部科学省，小学校プログラミング教育に関する研修教材，https://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1416408.htm，参照 Aug.3,2020.

[8] 文部科学省，令和元年度　市町村教育委員会における小学校プログラミング教育に関する取組状況等調査の結果について，https://www.mext.go.jp/content/20200107-mxt_jogai02-000003715_002.pdf，参照 Feb.2,2021.

[9] LINE，LINE みらい財団、プログラミング教育必修化に関する調査を実施　不安を感じている教員が 7 割以上、特に 20-34 歳の若い世代は約 9 割，https://linecorp.com/ja/csr/newslist/ja/2020/259，参照 Feb.2,2021.

[10] 文部科学省，小中高等学校等の臨時休業の実施状況について（令和 2 年 4 月 22 日時点），https://www.mext.go.jp/content/20200424-mxt_kouhou01-000006590_1.pdf，参照 Feb.2,2021.

[11] 文部科学省，小学校プログラミング教育の手引（第三版），https://www.mext.go.jp/content/20200218-mxt_jogai02-100003171_002.pdf，参照 Aug.3,2020.

[12] 大村基将，紅林秀治，ソフトウェア設計に基づく初学者のためのプログラミング学習の過程に関する考察，教科開発学論集（4）,pp.151-159，(2016).

[13] 大日本印刷，小学校で求められるプログラミング的思考を育む教材ソフト DNP プログラミング教材ソフト「SWITCHED ON Computing 日本版」，https://www.dnp.co.jp/biz/solution/products/detail/1192363_1567.html，参照 Aug.3,2020.

[14] Matayoshi Y.,Nakamura S.,Abstract Thinking Description System for Programming Education Facilitation,HCII 2020,vol.12206,pp.76-92,(2020).

[15] 太田剛，森本容介，加藤浩，プログラミング能力の発達段階と要因に関する定量的分析，情報教育シンポジウム,pp.85-92,(2017).

[16] 國宗永佳，中林清，プログラミング導入演習に対して学習支援システムと自己調整学習が与える影響の分析，通信ソサエティマガジン,Vol.50,秋号,pp.100-109,(2019).

[17] p5.js，https://p5js.org/，参照 Aug.3,2020.

[18] Blockly，https://developers.google.com/blockly，参照 Aug.3,2020.

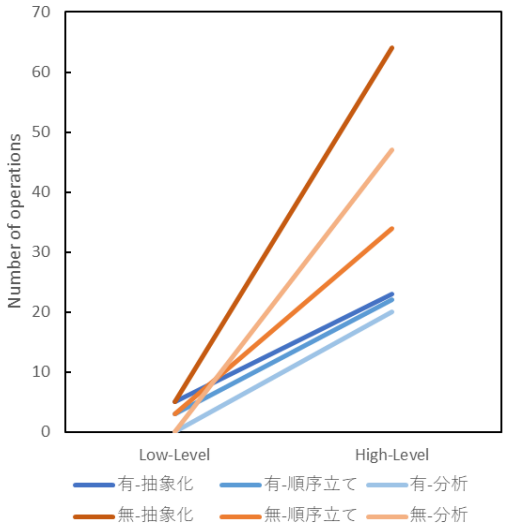
[19] XAMPP，https://www.apachefriends.org/jp/index.html，参照 Feb.2,2021.

[20] Blocly Developer Tools，https://blockly-demo.appspot.com/static/demos/blockfactory/index.html，参照 Feb.2,2021.

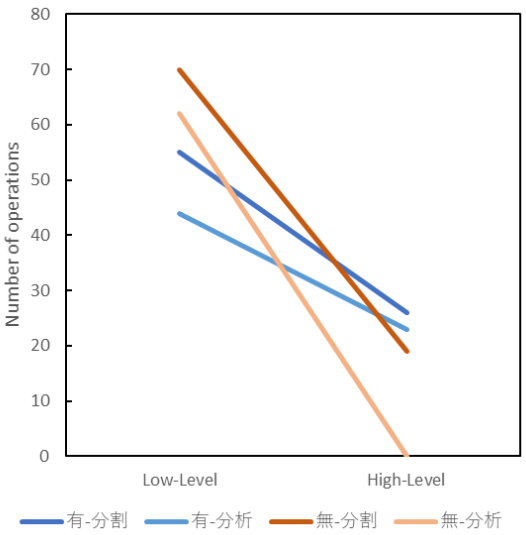
[21] Rubric ProEEs，http://g7programming.jp/wp-content/uploads/2018/02/RubricProEEs_171106.pdf，参照 Oct.20,2020.

付録

（１）被験者 F，G の実験結果における得点推移の比較

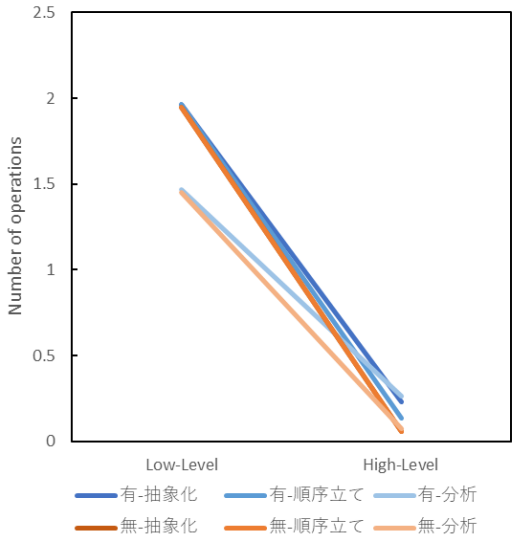


(a)手順課題

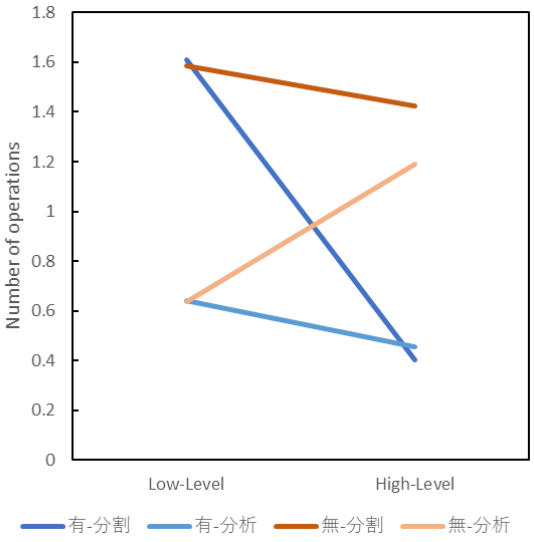


(b)動きに分ける課題

図 量データにおける被験者 G，F の得点推移（有：G 無：F）



(a)手順課題



(b)動きに分ける課題

図 質データにおける被験者 G，F の得点推移（有：G 無：F）

（2）プレテスト・ポストテストの回答例１

※塗りつぶしは実在の店名

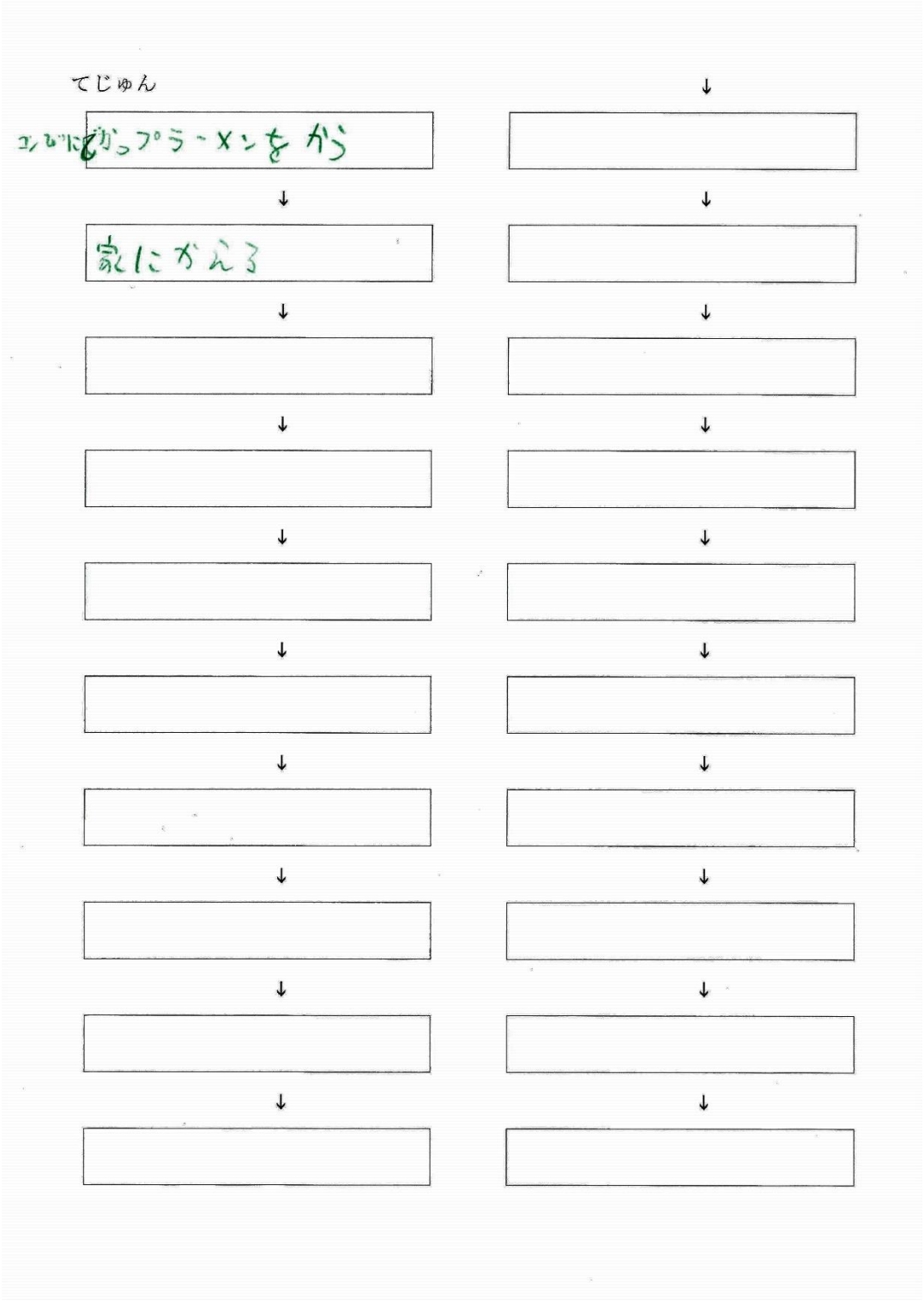


図 プレテスト１

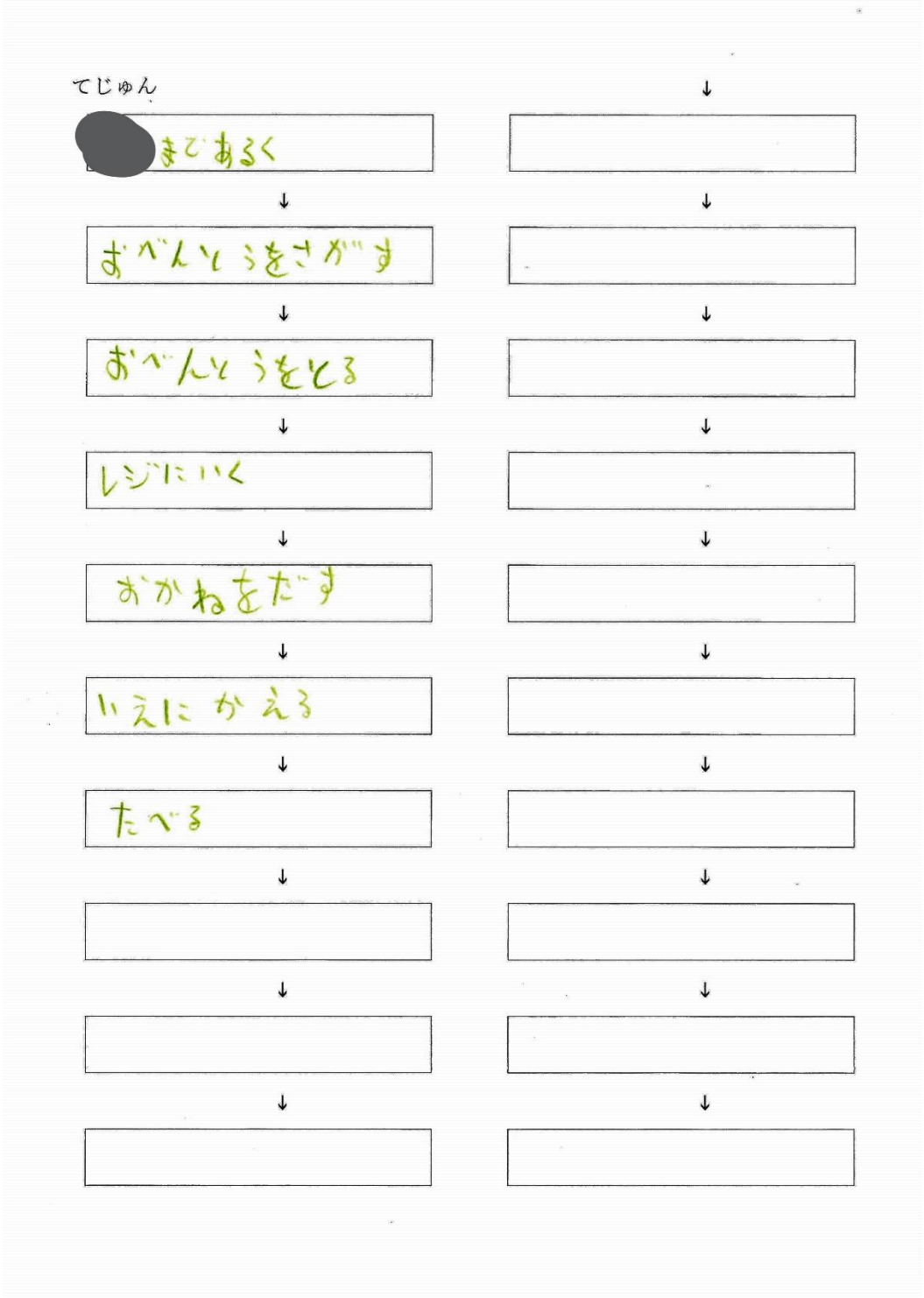

てじゅん
[redacted]まであるく
おべんとうをさがす
おべんとうをとる
レジにいく
おかねをだす
いえにかえる
たべる

図 ポストテスト１

（2）プレテスト・ポストテストの回答例 2

図 プレテスト 2


てじゅん
わかれ道まで進む。
右に90度回転
交差点まで進む
左に90度回転
あたるまで進む
おべんとうを買う
右に180°回転
交差点まで進む
右に90°回転
あたるまで進む
左に90°回転
あたるまで進む

図 ポストテスト 2