プログラミング教育を受ける小学生を対象とした論理的思考の育成手法

松浦雅子[†1]　平山秀昭[†1]

概要：2020 年に小学校におけるプログラミング教育が必修化された．その狙いはプログラミングの技能の習得にあるのではなく，論理的思考を育むことにあるとされている．小学校でプログラミングに触れ，段階的に論理的思考を育んでいくことを考えた時，その中心的な役割を担うのは当然ながら小学校の教員であるが，それだけでは不十分と考える．我々は小学生以下の子供を持つ保護者等にプログラミング教育を施し，保護者等が補完的に子供に対してプログラミングを教えるという方向を考えている．その際，保護者等に論理的思考の意味は教えるが，子供には論理的思考について説明させたりはせず，子供が小学校教育を受ける長いスパンの中で，論理的思考の意味が子供に伝わる時期や機会を意識して貰い，適切なタイミングでアドバイスしていくようにする手法について提案する．

キーワード：小学校プログラミング教育，プログラミング教育必修化，論理的思考

Method for Developing logical Thinking of Primary Schoolchildren who Receive Programming Education

MASAKO MATSUURA[†1]　HIDEAKI HIRAYAMA[†1]

1. はじめに

2020 年に小学校におけるプログラミング教育が必修化された．その狙いはプログラミングの技能の習得にあるのではなく，論理的思考を育むことにあるとされている．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合せが必要であり，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，記号の組合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」とされている[1]．

小学校におけるプログラミング教育必修化の狙いにある論理的思考とは何か？ここでは，システム開発の上流工程に位置する要件定義が行える能力に繋がる思考力と捉えることとする．当然ながら小学生が論理的思考を身に付け，要件定義ができる様になる訳ではなく，将来できる様になるためのきっかけ作りとなることと捉える．要件定義とは，利用者のシステムに対する要求（「利用者はそのシステムで何をしたいのか？」）に基づき，それを実現するために必要な機能等を明確に文書化することである．プログラミング教育を通してコンピュータはプログラムを正確に記述しなければ正しく動作しないことを理解しながら，何かを実現するためには必要な機能を明確化する必要があることを段階的に身に付けていく必要がある．我々は論理的思考をこの様に捉えて議論を進めていく．

2020 年を迎え，小学校では準備を進めてきたが，その意図が十分に理解されているかというと疑問がある．例えば，実践事例とされる題材[2]には,SCRATCH[3]等のビジュアルプログラミング言語により図形の作図を行うという極めて単純で小さなプログラムが示されている．複雑あるいは規模が大きくなるとプログラムそのものの理解が難しくなるという理由なのだろうが，それで情報技術を専門としない小学校の教員に十分な意図が伝わるのだろうかという疑問が湧く．勿論，ソートやサーチの様な基本アルゴリズムであれば，小さなプログラムであっても論理的思考を育むという意図は通じる．

一般に，単純で小さなプログラムを作成するだけでは，論理的思考というほど大げさなものは不要である．プログラムが複雑になったり，大規模になったりする時に論理的思考の必要性が生じる．複雑なプログラムや，大規模なプログラムを開発する際には必ず設計が必要になる．論理的思考を発揮するのは設計フェーズである．プログラミングフェーズでは，設計フェーズのアウトプットをプログラミング言語で実装していく，すなわち，プログラミング言語に置き換えていくだけなので，論理的思考を発揮する訳ではない．図形の作図を行うというような極めて単純で小さなプログラムをもって論理的思考について説明しても十分な理解が得られるか疑問がある．

ただし，小学校におけるプログラミング教育必修化の狙いは論理的思考を育むことのみではなく,「プログラムの働きやよさ，情報社会がコンピュータをはじめとする情報技術によって支えられていることなどに気付く」,「身近な問題の解決に主体的に取り組む態度やコンピュータ等を上手に活用してよりよい社会を築いていこうとする態度などを

†1 目白大学メディア学部メディア学科
Department of Media Studies, Faculty of Media Studies, Mejiro University

育む」ともされている[1]．よって，小学校ではプログラミングという情報技術に触れ，中学校，高等学校と情報技術の教育を進める中で段階的に論理的思考を育んでいくのだと捉えればよいと考える．

小学校でプログラミングに触れ，段階的に論理的思考を育んでいくことを考えた時，その中心的な役割を担うのは当然ながら小学校の教員であるが，それだけでは不十分と考える．少し前の状況ではあるが平成 27 年に総務省が公表した報告書[4]によると,学校以外で行われているプログラミングに関する教育を行っているのは非営利目的の組織（NPO 法人，法人化されていない組織等）によるものが半数を超えている．一方，営利目的の組織（プログラミング教室等）によるものは 3 分の 1 程度である．しかもプログラミング教室の多くは関東，それも都内に集中している．また，プログラミング教室の開校は 2013 年以降に急増していて，講師はプログラミング経験のない者が多く，基本的な指導のみを行い，理系のプログラミング経験のある大学学部生・院生のアルバイト講師がエラー発生時の対応等を行っている．このような講師不足の問題から，シニアのプログラミング経験者の活用や，家庭の主婦が講師になってちょっと開けるような広がり方を目指す意見も上がっている．

2. 研究目的と研究方法

前述のような状況を鑑み，子供が小学校でプログラミングに触れ，段階的に論理的思考を育んでいけるようにすることを目的に，我々は小学生以下の子供を持つ保護者等にプログラミング教育を施し，保護者等が補完的に自分の子供に対してプログラミングを教え，論理的思考を育んでいくという手法を提案している[6][7]．

その際，保護者等に論理的思考の意味は教えるが，子供には論理的思考について説明させたりはせず，子供が小学校教育を受ける長いスパンの中で，論理的思考の意味が子供に伝わる時期や機会を意識して貰い，適切なタイミングでアドバイスしていくようにして貰う．本稿では，このような考えの下で，プログラミング経験のない保護者等を対象に実施した実験教育の結果について報告する．

3. 保護者等向けプログラミング教育の内容

プログラミング言語は SCRATCH を選んだ．SCRATCH を選んだ理由は，以下の通りである．

- SCRATCH は，小学生にも理解し易いビジュアルプログラミング言語である．
- ビジュアルプログラミング言語は多数あるが，基本的には，どれも SCRATCH をベースとしている．
- SCRATCH は，ビジュアルプログラミング言語の中で最も普及している．
- SCRATCH はブラウザ上で動作し，家庭に普通にある PC でプログラミングができ，特別な機器を購入する必要がない．
- SCRATCH は，作ったプログラムをインターネット上で共有する仕組を提供している．

＜実験内容＞

実験教育 1

教育期間は 90 分×3 回とした．体験的な教育は 60 分×1 回であることが多いが，それでは子供に教えられるレベルに到達することは困難であると考えた．自分がなんとなく理解するレベルではなく，自分が理解し，必要に応じて調べながら，子供に教えられるレベルに教育できる最小時間として 90 分×3 回とした．対象は，プログラミング経験のない，あるいは，ほぼない保護者等である．以下に示す内容は，プログラミング経験のない，あるいは，ほぼない保護者等には高度な内容かとも考えたが，我々が目指す「保護者等が補完的に自分の子供に対してプログラミングを教える」というレベルを実現するには必要なレベルと考えた．また，90 分×3 回という教育期間は，実効性のある範囲で考えた．3 回の教育の内容は，以下のようにした．

（1）第 1 回

まず，簡単に小学校でプログラミング教育が必修化された狙いが論理的思考を育むことにあることを伝える．ただし，あまり深く考えるのではなく，とにかくプログラミングをやってみることを伝える．そして，世界で最も普及しているビジュアルプログラミング言語 SCRATCH を使うことを伝える．

次に SCRATCH にユーザ登録する．そして，SCTRATCH でスプライト（オブジェクト）をステージで動かすプログラミングを丁寧に教える．SCRATCH の操作に慣れることも含める．ここで作るプログラムは 10 ブロック以内の一塊のプログラムである．できたところで，プログラミングしたというより，言われた通りに操作して作ったプログラムの意味を相手に説明させる．説明させることでプログラムの意味を理解させる．

作成した SCRATCH のプログラムが，Java 等の非ブロック型のプログラムならどのようになるのかを示す．非ブロック型のプログラムでは,「{」と「}」のバランスが取れている必要があるとか,条件文は「if」で,繰り返し文は「while」を使うとか，事前に文法というものを習う必要があり，小学生には少し難しいことを説明し，概略を理解して貰う．図 1 は，第 1 回のテキストの一部である．

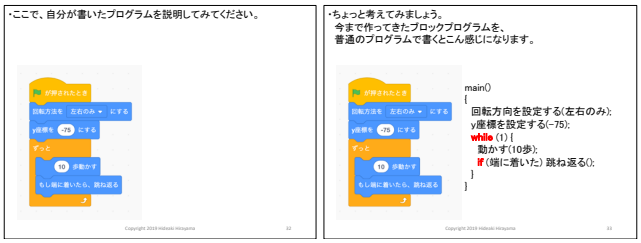

図 1 第 1 回のテキストの一部

また，コンピュータはプログラミングされた通りに処理を行う．間違ったプログラミングをすれば間違った処理をする．正しいプログラミングをすれば正しい処理をする．人間は曖昧な指示をしても，その意図を解釈してなんとかすることもあるが，コンピュータはそうはいかない．プログラミングした通りにしか動かない．だから正しくプログラミングする，正しく指示しなければならない．これが論理的思考の基本であることを説明する．わずか 10 ブロック程度であるが，プログラミングを経験した保護者等は論理的思考の意味を概略理解できるようになる．ただし，子供に対しては，その説明は不要であることを伝える．

最後に，これまでとは異なるスプライトとステージを新たに選んで貰い，これまでやったことと同じプログラムを自力で最初から作成して貰う．これにより，SCRATCH で基本的な動作をさせるプログラムを自力で作れるようになる[5]．

(2) 第 2 回

第 2 回では，少し複雑なプログラムに取り組む．10 ブロック程度の塊からなるプログラムを 2 個作る．変数，タイマー，音も使ってみる．スプライトをクリックされた時のイベントも扱う．プログラミングは前回同様に丁寧に教える．できたところで，やはりまだ，プログラミングしたというより，言われた通りに操作して作ったプログラムの意味を相手に説明させる．説明させることでプログラムの意味を理解させる．また，今回も，作成した SCRATCH のプログラムが，Java 等の非ブロック型のプログラムならどのようになるのかを示す．

第 1 回，第 2 回で，言われた通りに操作してプログラムを作れるようになった．しかし，これではプログラミングができるようになったとは言えない．自分で考えて，自分で調べてプログラミングができるようになる必要がある．それができないと，子供に聞かれても解答してあげることができない．それでは，子供にプログラミングを教えてあげることはできない．

そのために，第 2 回では，作ったプログラムを色々と改造してみる．動きを変えたり，速度を変えたり，色々なことを試みる．そして，ブロックパレットに並んでいる様々な機能を見ながら，それを使ってプログラムを改造してみる．これにより，プログラムを自分で改造すること，SCRATCH が持つ機能を自分で調べてみることを経験してみる[5]．図 2 は，第 2 回のテキストの一部である．

図 2 第 2 回のテキストの一部

(3) 第 3 回

第 3 回では，先に非ブロック型のプログラムを見てから，それが SCRATCH だとどのようなプログラムになるのかを見ていく．Java 等の非ブロック型のプログラムでの“Hello World”，変数宣言，代入文，演算といった操作が，SCRATCH だとどうなるのかを見る．コンピュータが元々は軍事目的の弾道計算のために作られ，ビジネスで使われるようになっても売上計算等のために使われ始めたこと．つまり，コンピュータは，元々は計算を高速に行う機械として開発されたことを理解する．これは，その後，コンピュータが様々な分野で使われるようになったこと，人工知能のようにいわゆる計算目的以外で使われるようになったことを理解し，様々な応用を考える頭を育むきっかけになることを理解して貰う．

最後は，3 つのスプライト（フットボーラー，ボール，コウモリ）からなるシューティングゲームを自力でプログラミングして貰う．そのためにシューティングゲームの設計を示す．最初に基本設計を，次に詳細設計を示す．その上でプログラミングを行なって貰う．かなり難しいが，自分で考えながらやって貰う．最後にできたところまでのプログラムの説明をして貰い，わからなかったところ，うまくいかなかったところを皆で考えながら修正する．図 3 は，第 3 回のテキストの一部である．

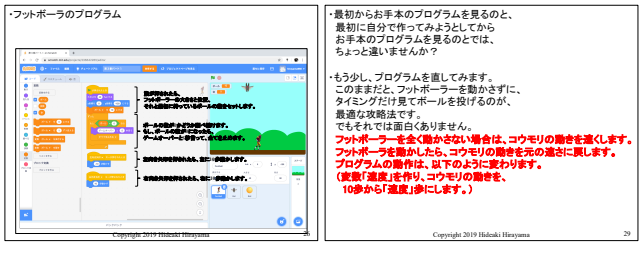

図 3 第 3 回のテキストの一部

最後まで完璧にできるとは限らないが，自分で考え，自分で調べ，プログラミングのやり方を教える．また，いきなり解答（プログラミング例）を見るのと，自分で悪戦苦闘してから解答を見るのでは，効果が全く異なる．あとは，

持ち帰って，自分で復習をして貰う．これで，ようやく，子供に聞かれて解答してあげられるようになる．これで初めて，子供にプログラミングを教えられるようになる．

また，あわせて，最初に示した設計にこそ論理的思考があるということを，概略理解して貰う．すっきりとは理解できないだろうが，後々，理解できるようになるだろうと考える[5]．

実験教育 2

実験教育 2 は，実験教育 1 のフォローアップの位置付けで，90 分×2 回の教育とした．実験教育 2 は，最初から予定していたものではなく，実験教育 1 の参加者からの要望で実施した．両回とも，これまで教えていなかった機能を教えた後，課題プログラムの仕様を提示し，自分で考えてプログラミングして貰う．いきなりプログラミングするのではなく，まずどのように作るか設計を考えて貰うこととした．

(1) フォローアップ第 1 回

最初にリストとメッセージの機能について説明する．続いて，リストとメッセージの機能を使ってプログラミングする課題としてクイズプログラムを提示した．出題者のスプライトとステージのみからなる．

最初にステージのプログラムで，準備としてクイズの問題と解答をリストで作成する．それが終わったら，ステージのプログラムが出題者のスプライトにメッセージを送り，問題リストを使って出題を行っていく．ユーザから解答を受けたら，解答リストと比較して正解判定をするものである[5]． 図 4 は，フォローアップ第 1 回のテキストの一部である．

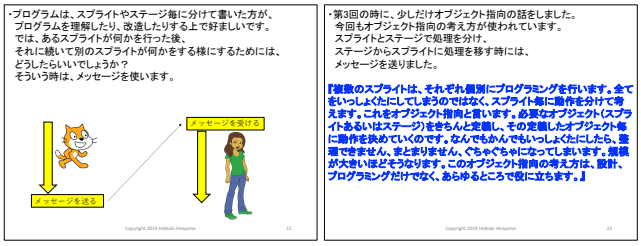

図 4 フォローアップ第 1 回のテキストの一部

(2) フォローアップ第 2 回

最初にクローンの機能について説明する．続いて，クローンとメッセージの機能を使ってプログラミングする課題としてブロック崩しを提示した．ブロック，ラケット，ボールのスプライトとステージからなる．

ブロックはクローンの機能を使って 8 個作成する．ブロックにボールが当たったら，ブロックのプログラムはボールに「バウンド」のメッセージを送る．ラケットにボールが当たったら，ラケットのプログラムはボールに「バウンド」のメッセージを送る．ボールは「バウンド」のメッセージを受けたら跳ね返るようにする[5]．図 5 は，フォローアップ第 2 回のテキストの一部である．

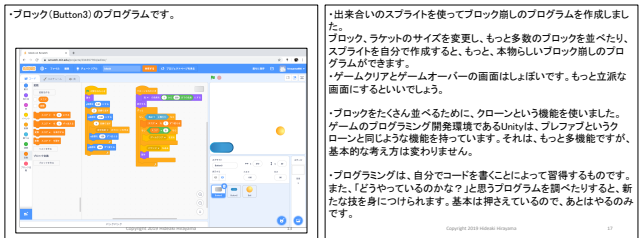

図 5 フォローアップ第 2 回のテキストの一部

実験教育 3

実験教育 3 では，対象者を変えて，小学生の子供を持つ父親，現役の小学校教員，中学生の子供を持つ母親として，実験教育 1 と同じ内容のプログラミング教育を実施した．この意図は，小学生以下の子供を持つ保護者等との違いを見るためで，実験教育 3 は実験教育 1 の対照実験としての位置付けである．

今回は，90 分×3 回分を 1 日で実施した，これは，現役の小学校教員，小学生の子供を持つ父親のスケジュールを考慮したためである．

4. 実験教育の結果

●実験教育 1 の結果

2 章に示した内容のプログラミング教育を 2019 年 7 月 2 日，4 日，9 日の 3 日間で実施した．ただし，第 3 回の内容であるシューティングゲームのプログラミングは，第 2 回の最後で第 3 回までの課題として提示し，第 3 回において課題プログラムのレビューを行なった．対象となった小学生以下の子供を持つ母親は 4 名（A，B，C，D）である．

実験教育の対象者を募集したが十分な人数が集められなかったため，4 名という少人数の実験結果となった．ただし，1 回の実験教育の対象者として 4 名は適切な人数である．小学生の子供を持つ保護者等向けに 50 名規模の講習会を実施することがあるが，それは教育の場というより説明の場と言える．我々が目指す「保護者等が補完的に自分の子供に対してプログラミングを教える」というレベルの教育を行うには，4〜6 名程度が適切である．4〜6 名を対象とする実験教育を複数回実施できることが望ましかった．なお，実験教育の対象者 4 名は，全て東京 23 区在住で，うち 3 名は公立小学校の小学生の子供を持つ母親，1 名は小学校入学前の子供を持つ母親である．

後日（3 週間後），4 名に対してアンケートを実施した．アンケートの内容及び結果を図 6 に示す．アンケートの結果にあるように対象者は全員，これまでプログラム経験が

ない，または，ほぼないという想定通りの被験者だった．自由記述の主な内容は，図 7 のようになった．

(1)教育の期間は？		やや短い (2:C, D)	丁度いい (2:A, B)		
(2)教育の難度は?		やや難しい (1:D)	丁度いい (3:A, B, C)		
(3)プログラミングの経験は？	ない (1:B)	ほぼない (3:A, C, D)			
(4)SCRATCHを知っていたか？	知らなかった (1:B)	名前を聞いた程度 (2:A, C)			知っていた (1:D)
(5)プログラミングを理解できたか？			何とも言えない (2:A, D)	まあ理解できた (2:B, C)	
(6)復習すればできるか？				まあできる (3:A, B, D)	できる (1:C)
(7)プログラミングは楽しかったか？			何とも言えない (1:A)	まあ楽しい (1:D)	楽しい (2:B, C)
(8)興味・関心は増したか？				まあ増した (3:A, B, D)	増した (1:B)
(9)子供に教えられるか？			何とも言えない (2:A, C)	まあ教えられる (1:D)	教えられる (1:B)
(10)子供と一緒にやりたいか？				まあやりたい (2:A, C)	やりたい (2:B, D)
(11)プログラミングと設計の違いを理解できたか？	理解できなかった (1:B)	やや理解できなかった (1:D)		まあ理解できた (1:C)	理解できた (1:A)
(12)論理的思考を理解できたか？				まあ理解できた (4:A, B, C, D)	
(13)必修化の意味を理解できたか？			何とも言えない (2:A, D)	まあ理解できた (2:B, C)	
(14)必修化に賛成か？			何とも言えない (3:A, C, D)	まあ賛成 (1:B)	
(15)プログラミングは重要か？			何とも言えない (2:A, D)	まあ重要 (2:B, C)	
(16)楽しく学べる教科になるか？			何とも言えない (1:A)	まあなる (3:B, C, D)	
(17)習い事に取り入れたいか？			何とも言えない (2:A, C)	まあ入れたい (2:B, D)	

図 6 アンケートの内容及び結果

回答者A	(1)日頃の先生の忙しさを考えると，現場がさらに疲弊するのではないか．実際の学校の状況とは大きなギャップがある．
	(2)先生に任せるというより，先生をフォローする人，もしくは先生に代わるプログラミングの教育に長けた人材が必要．
	(3)子供の頃の経験は大人になるまでに，何がどう影響するか分からない．そういう「出会い」の機会という意味で期待する．
	(4)保護者がプログラミングできるか否かで差ができ，辛い思いをする子供が出るかもしれない．
回答者B	(2)まったくの初心者だったので，x座標とy座標が，どっちがどっちかすら記憶がおぼろげな状態だった．
	(3)1進むはドット1つ分だからほんの少ししか進まないという説明があればもっと分かり易かった．
	(4)課題をする時間が取れないので，自分で考えてみる回を1回設けた上で最終回の発表だと，もっと理解が深まった．
	(8)プログラミングと設計の違いの説明があったか，はっきり記憶していない．
回答者C	(1)子供に教えられそうかということではなく，子供と教え合うという関係性になるのではないか．
	(2)子供とは，それぞれが自分の時間に作って，一緒にシェアし合うという感じになるのではないか．
	(3)未知の世界だったプログラミングが，「やればできるかも」と思える機会となったが，時間を捻出することが最大の課題．
回答者D	(1)3日目の内容は高度だったが，より踏み込んで取り組みたいという意欲につながった．
	(2)3日間の講習と捉えたときに，難度の変化が激しい．1日目の単元は，もっと時間を短縮しても良い．
	(3)初回は，することがきっちり決まっていて後半になると自由度が増すが，子供の中には「決められたこと」をさせられると興味を失う特性の子が一定の割合で存在する．

図 7 自由記述の主な内容

●実験教育 2 の結果

2 章に示したプログラミング教育のフォローアップを，2019 年 10 月 10 日，24 日の 2 日間で実施した．対象者は，実験教育 1 の対象となった小学生以下の子供を持つ母親 4 名（A，B，C，D）のうちの 3 名である．実施後にアンケートを実施した．アンケートの内容及び結果を図 8 に示す．アンケートは，前回の内容の一部（5，6，9，10，11，12，13，14，15）とした．これは，フォローアップ後の変化を見るためである．自由記述の主な内容は，図 9 のようになった．

(5)プログラミングを理解できたか？				まあ理解できた (3:A, B, C)	
(6)復習すればできるか？				まあできる (2:A, B)	できる (1:C)
(9)子供に教えられるか？		やや教えられない (1:A)	何とも言えない (1:C)	まあ教えられる (1:B)	
(10)子供と一緒にやりたいか？			何とも言えない (2:B, C)		やりたい (1:B)
(11)プログラミングと設計の違いを理解できたか？			何とも言えない (2:B, C)	まあ理解できた (1:A)	
(12)論理的思考を理解できたか？			何とも言えない (1:C)	まあ理解できた (2:A, B)	
(13)必修化の意味を理解できたか？		やや理解できなかった (1:C)	何とも言えない (1:A)	まあ理解できた (1:B)	
(14)必修化に賛成か？		やや賛成しない (1:C)	何とも言えない (2:A, B)		
(15)プログラミングは重要か？			何とも言えない (2:A, C)	まあ重要 (1:B)	

図 8 アンケートの内容及び結果

回答者B	(1)ローマ字入力ができるようになる小学校高学年からでないとプログラミング作成を楽しむのは厳しいと感じた．

図 9 自由記述の主な内容

●実験教育 3 の結果

実験教育 3 では，対象者を小学生の子供を持つ父親(E)，現役の小学校教員（F)，中学生の子供を持つ母親（G）とした．実験教育 1 と同じ内容のプログラミング教育を 2019 年 11 月 24 日に，90 分×3 回分を 1 日で実施した．アンケート内容（実験教育 2 と同じ）を図 10 に示す．自由記述の主な内容は，図 11 のようになった．

(5)プログラミングを理解できたか？				まあ理解できた (3:E, F, G)	
(6)復習すればできるか？				まあできる (3:E, F, G)	
(9)子供に教えられるか？				まあ教えられる (3:E, F, G)	
(10)子供と一緒にやりたいか？				まあやりたい (1:F)	やりたい (2:E, G)
(11)プログラミングと設計の違いを理解できたか？			何とも言えない (1:G)	まあ理解できた (2:E, G)	
(12)論理的思考を理解できたか？			何とも言えない (1:G)	まあ理解できた (2:E, F)	
(13)必修化の意味を理解できたか？			何とも言えない (1:F)		理解できた (2:E, G)
(14)必修化に賛成か？			何とも言えない (1:F)		賛成する (2:E, G)
(15)プログラミングは重要か？				まあ重要 (1:F)	重要 (2:E, G)

図 10 アンケートの内容及び結果

回答者F	(5)小学校で教える場合は，ほぼマスターしてからでないと厳しい．分からないことが出てきた場合，授業が中断してしまう．また，自分より理解している子供がいて，逆に教えてくれるような場面が出てくると，ダメ先生というレッテルを貼られる恐れもある．
	(7)必修としないとプログラミングの授業をしない学校も出てきてしまうので仕方がない．ただし英語や道徳の教科化や授業時間増など，現場の先生の負担は増す．そのあたりの対策も必要．

図 11 自由記述の主な内容

5. 考察

実験教育の結果を分析していく．アンケート(1)～(17)は，アンケート(3)～(4)を除き選択肢が 5 個ある．アンケート(5)～(17)においては，前半 2 個の選択肢は DISAGREE（NEGATIVE），後半 2 個の選択肢は AGREE（POSITIVE）の傾向にある．

アンケート(1)～(4)の結果を見ると，対象者は想定通りプログラミングの経験は,ない(75%)/ほぼない(25%)であった．SCRATCH に関しては，知らなかった(25%)/名前を聞いたことがある(50%)/知っていた(25%)であった.知っていたと回答した 1 名は，SCRATCH に触れたことがあった．この回答者(D)の自由記述(1)(2)(3)に見られるように,SCRATCH に触れたことがある対象者の場合は，第 1 回の内容である利用者登録や操作説明は，もう少し短くしてもよいと考える.教育の期間は,やや短い(50%)/丁度いい(50%)であった.期間がやや短いと感じたことに関しては，回答者 B の自由記述(4)，回答者 C の自由記述(3)に見られるように，第 2 回の終了時に，第 3 回までの課題を提示したが，その課題に取り組む時間の捻出が難しかったことによる．この結果から，課題の提示及び課題への取り組みは，講義時間の中で実施するのが適切だと考える（改善点 1）．難度は，丁度いい(75%)/やや難しい(25%)であった.どのような点をやや難しいと感じたかというと，回答者 B の自由記述(2)(3)に見られるように，座標やスプライトの歩数に関する部分の説明をしなかったことによる．この結果から，コンピュータの基礎的な部分の説明は，省略せずに実施すべきであると考える（改善点 2）.

アンケート(5)～(8)の結果を見ると，プログラミングの理解度は,まあ理解できた(50%)/なんとも言えない(50%)であった．復習すればできるかという質問には，まあできる(75%)/できる(25%)の回答だった．また，プログラミングが楽しいかという質問には，楽しい(50%)/まあ楽しい(25%)/なんとも言えない(25%)の回答だった．興味・関心が増したかという質問には,まあ増した(75%)/増した(25%)の回答だった．この結果から，もう少し講義を追加し，プログラミング経験を増やせば，プログラミングができるようになるという感覚が強くなると考える（改善点 3）．また，実際に受講者からは追加講義の要望があったため，実験教育 2 としてフォローアップ教育を実施した.

実験教育 2（フォローアップ教育）のアンケート結果では，実験教育 1 より更に結果が良くなっている．プログラミングの理解度は，まあ理解できた(100%)に増えた．復習すればできるかという質問では，まあできる(66%)/できる(33%)で変化はなかったが,フォローアップ教育の結果から,今回の実験教育でプログラミングがほぼ理解でき，復習すればプログラミングができるようになる可能性が高いと考える（結論 1）．なお，小学生以下の子供を持つ母親以外を対象とした対照実験である実験教育 3 でも，ほぼ同様の結果が得られている.

アンケート(9)～(10)の結果を見ると，子供にプログラミングを教えられそうかという質問では，なんとも言えない(50%)/まあ教えられる(25%)/教えられる(25%)の回答だった．子供と一緒にプログラミングをしたいかという質問では,やりたい(50%)/まあやりたい(50%)という回答だった.どのように子供と一緒にプログラミングをしたいのかというと，回答者 C の自由記述(1)(2)に見られるように，一緒に取り組むというより，それぞれの取り組みを教え合うような形だということであった．この結果から，今回の実験教育でプログラミングを理解し，復習すればプログラミングもできるようになり，子供と一緒に教え合うようにプログラミングに取り組んでいけるようになると考える（結論 2）.

アンケート(11)～(12)の結果を見ると，プログラミングと設計の違いを理解できたかという質問では，バラバラの回答だった（理解できなかった(25%)/やや理解できなかった(25%)/まあ理解できた(25%)/理解できた(25%)）．それに対し，論理的思考を理解できたかという質問に関しては，まあ理解できた(100%)という回答だった．プログラミングと設計の違いに関しては，回答者 B の自由記述(8)に見られるように，説明された印象がないという受講者も見られた．確認したところ,「そういえばそういう説明を聞いた」という返事だった．この結果から，プログラミングと設計の違いについては，もう少し丁寧な説明が必要だと考える（改善点 4）.

アンケート(13)～(14)の結果を見ると，プログラミング教育の必修化の意味に関しては，まあ理解できた(50%)/なんとも言えない(50%)であった．それに対し，必修化に賛成か否かに関しては,なんとも言えない(75%)/まあ賛成(25%)であった．プログラミング教育の必修化に賛成しない理由は，回答者 A の自由記述(1)(2)に見られるように，小学校の教員が多忙であるという現実を考慮したことによる．同様の内容は，実験教育 3（小学生以下の子供を持つ母親以外を対象とした対照実験）の回答者 F（現役小学校教員）の自由記述(7)にも見られる.

また，実験教育 3 において回答者 F（現役小学校教員）の自由記述(5)には,以下のような記述が見られたことに注意を払う必要がある.「小学校で教える場合は，ほぼマスターしてからでないと厳しい．分からないことが出てきた場合，授業が中断してしまう．また，自分より理解している子供がいて，逆に教えてくれるような場面が出てくると，ダメ先生というレッテルを貼られる恐れもある.」

アンケート(15)～(17)の結果を見ると，プログラミングは重要かという質問では，まあ重要(50%)/なんとも言えない(50%)の回答だった.楽しく学べる教科になるかという質問では,まあなる(75%)/なんとも言えない(25%)の回答だっ

た．この結果から，プログラミングは小学生にとって，まあ重要で，まあ楽しく学べる科目になるだろうという程度の肯定性で保護者達は考えている．ただし実験教育 2 の回答者 B の自由記述(1)に見られるように，ローマ字を学んだ後であることが前提となるだろうと考える．習い事に取り入れたいかという質問では，まあ入れたい(50%)/なんとも言えない(50%)の回答だった.回答者 A の自由記述(3)(4)を見ると,プログラミングに触れる機会に期待するものの，算数や国語といった教科と違い，家庭でプログラミングを見られる保護者等がどの程度いるのかということを危惧している．更に記述 (2) では，プログラミングの教育に長けた外部の人材の必要性について述べている.この結果から，小学校の教員を補完する形でプログラミングを教え，論理的思考を育んでいくには，保護者等だけでは難しいと考える（改善点 5)．なお，受講者には説明しなかったが，プログラミングの教育に長けた外部の人材に関しては,[1]にも記述されている．

以上の結果から，子供が小学校でプログラミングに触れ,段階的に論理的思考を育んでいけるようにすることを目的とし，小学生以下の子供を持つ保護者等にプログラミング教育を施し，保護者等が補完的に自分の子供に対してプログラミングを教え，論理的思考を育んでいくという我々の提案手法は，(結論 1)，(結論 2) に示すように適切であると考える．ただし，(改善点 1) 〜 (改善点 5) に示すように，改善の余地がある．

なお，今回の実験教育の対象者である 4 名は，いずれもプログラミングの講義への参加を希望してきた前向きな受講者である．しかし，全ての保護者等がそのように前向きであるとは限らない.あるいは,前向きであっても時間的に困難な場合もある．それを考えると，保護者等だけで我々が考えている方向を実現するのは難しいだろうと推測される．そのため，(改善点 5) にあるように，多忙な小学校の教員が行うプログラミング教育を補完するには，保護者等だけでなく，保護者等を中心とした地域コミュニティが支援を行い，子供が小学校教育を受ける長いスパンの中で，論理的思考を育んでいくことも必要であると考える．

6. 関連研究

参考文献[9]では，プログラミングがどのような思考力を育むことができるのかという「プログラミングがもたらす教育的効果」を明らかにすることを目的に，独自に作成した学習指導案をもとに,小学 2 年生に向けて授業実践し，授業前後における児童の論理的思考力・読解力・問題解決力の変化を検証している． 参考文献[10]では，小学生向けにプログラミング的思考を促す目的で,対話的/スクリプト的実行，順次・分岐・反復を盛り込んだアンプラグド授業を実践している．参考文献[11]では，暗記型のプログラミング教育では，小学生がプログラミング的思考を習得することは困難とされているため，小学生にとって身近な日常の活動を題材としたプログラミング的思考育成ツールを開発している． 参考文献[12]では，近隣地域の学生を活用したプログラミング指導者（メンター）の育成と，大阪「ものづくり」DNA を継承する人材育成のきっかけとするための効果的・効率的な小学生向けロボットプログラミング教育を実施し,「地域完結型プログラミング教育モデル」の実証を進めている．参考文献[13]では，現在において情報系人材が不足していることによってプログラミング教育が促進されている背景から，小・中学生を対象としたプログラミング学習支援のアプリケーション開発を行っている．参考文献[14]では，デジタルテクノロジーの書き手を育て，豊かな言語能力の育成を目指すためには，パソコンを特別視せず，子供が誰から指図されることもなく，自然に使うような環境や雰囲気を醸成する必要があると述べている．

7. おわりに

小学校でプログラミングに触れ，段階的に論理的思考を育んでいくことを考えた時，その中心的な役割を担うのは当然ながら小学校の教員であるが，それだけでは不十分である．我々は小学生以下の子供を持つ保護者等にプログラミング教育を施し，保護者等が補完的に子供にプログラミングを教えるという方向を考えている．子供が小学校教育を受ける長いスパンの中で，論理的思考の意味が子供に伝わる時期や機会を意識して貰い，適切なタイミングでアドバイスしていくようにして貰う．この考えの下で，小学生以下の子供を持つ母親 4 名に対して実験教育を行なったところ，我々の提案手法は，概ね適切であった．

ただし，今回の実験教育の対象者は 4 名と少なかった．そのため,研究に協力してくれる小学校,自治体等を探し，実験の数を増やすことが今後の課題と言える．

また，提案した手法を実施していく場合には，保護者等を支援してくれる地域コミュニティが重要な役割を果たすので，プログラム経験のある人材に実験教育への協力を得ることも今後の課題と言える．

あわせて，以下に示すような補完的なツールの検討も考えられる．

（提案手法の補完ツール）

子供が小学校教育を受ける長いスパンの中で，論理的思考を育んでいくためのツールによる支援も有効かもしれないと考え，子供に論理的思考の意味を伝えるタイミングを教えるツールを試作した[8]．図 12 に示すこのツールは，家庭内や地域コミュニティの中でも，論理的思考を学ぶ機会を見出していくことを狙いとし，子供と保護者等との日常会話の中から，「会話の言葉」をピックアップし，“学

ぶタイミング”を教えてくれるツールである．また，論理的思考に段階的に取り組むことをサポートできると考える．

具体例として， ある 1 つのプロジェクトを,子供と保護者等が一緒になって考えていくプロセスを想定する．例えば,「野球大会」や旅行の計画などである．その際,

① 順序立てて考える
② 矛盾なく考える
③ 正確に理解する/表現する

といった「何故そうするのか」「どうしてそうするのか」といった「正確に考え，理解する」プロセスを考えさせるきっかけとする．家庭内や地域コミュニティで話し合う時,子供側から， 例えば， 「どうすれば？」「何をすれば？」「どうやって？」「無理」「できない」「わからない」等の言葉が出た時に，それを音声認識し，コンピューターが“声をかけるタイミングです”と発する機能を持つ.

実際に実験教育 2 の対象者に，機能を見てもらい感想を聞いてみた．今は，幼稚園でも，何かのプロジェクトを考えさせる時に，まずは自分で計画を考えるプロセスを踏んでいるので，家庭でできたら良いのではという意見が出た.

また，論理的思考能力をつけるには，「具体案を自分で考える」ということを繰り返す必要があるため，このタイミングを教えてくれる機能は面白く興味があるという感想を得た.学校での「プログラミング」教育の補完として，さらに様々な方法で，論理的思考について考えるきっかけを作り,日常生活の中から,保護者等と一緒に学ぶ機会を見出すことができる.

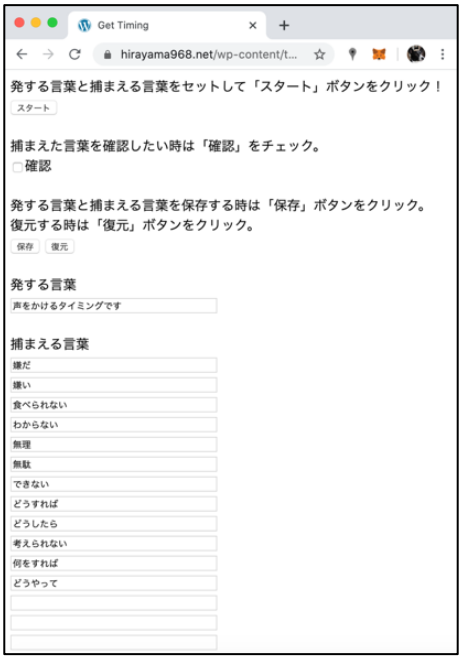

図 12 ツールのイメージ

参考文献

[1] “小学校プログラミング教育の趣旨と計画的な準備の必要性について（1)”．文部科学省初等中等教育局情報教育・外国語教育課情報教育振興室.
http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2019/05/21/1417047_001.pdf.

[2]“小学校プログラミング教育の趣旨と計画的な準備の必要性について（2)”．文部科学省初等中等教育局情報教育・外国語教育課情報教育振興室.
http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2019/05/21/1417047_002.pdf.

[3] John Maloney, Leo Burd, Yasmin Kafai, Natalie Rusk, Brian Silverman, Mitchel Resnick. Scratch: A Sneak Preview. Second International Conference on Creating, Connecting and Collaborating through Computing, Kyoto, Japan, 2004, p. 104-109.

[4]“プログラミング人材育成の在り方に関する調査研究報告書”．平成 27 年 6 月．総務省.
http://www.soumu.go.jp/main_content/000361430.pdf.

[5] “SCRATCH プログラミング”.
https://hirayama968.net/2019/10/24/programming-2/.

[6]松浦雅子，平山秀昭．小学校においてプログラミング教育が実施される状況を鑑みた母親向けプログラミング教育．情報処理学会研究報告．2019，Vol．2019-CE-151，No．8，p．1-9.

[7] 松浦雅子，平山秀昭．小学校においてプログラミング教育を受ける小学生を対象とした論理的思考の育成方法．情報処理学会研究報告．2019，Vol．2019-CE-154，No．19，p．1-8.

[8] https://hirayama968.net/2020/05/18/gettiming/.

[9]崔仁珠，上倉隼，佐藤連，高塚穂佳，保土田雪成，松延奈美，森田昌樹，杉本紀子，中村文宣，吉田典弘．小学校プログラミング教育の学習指導案の作成と教育的効果の評価-Programming Of the Kids By the Kids For the Kids.-．情報処理学会研究報告．2019，Vol．2019-CE-148，No．10，p．1-17.

[10] 倉橋農，越智徹，尾崎拓郎，島袋舞子．子ども向け授業にリンクした保護者・教師向けプログラミングコースの検討．情報処理学会研究報告．2019，Vol．2019-CE-150，No．9，p．1-5.

[11] 内田早紀子，松村敦．日常の活動を題材とした小学生向けプログラミング的思考育成ツールの開発．情報処理学会研究報告．2018，Vol．2018-CE-147，No．9，p．1-5.

[12] 吉田研一，伊藤寿晃，山脇智志，大森康正．小学生を対象としたプログラミング教育指導者育成方法とその評価．情報処理学会研究報告．2017，Vol．2017-CE-138，No．10，p．1-4.

[13] 尾花拓海，鈴木龍成，吉村明人，臼田莉菜，半澤魁士，佐久間拓也，川合康央，池辺正典．小・中学生を対象にしたプログラミング学習を支援するアプリケーション開発プロジェクト．情報教育シンポジウム．2017，p．202-205.

[14] 阿部和広．子どもの創造的活動と ICT 活用．情報処理．2015，Vol．56，No．4，p．350-354.