プログラミングを用いた授業づくりに向けて：「小学校からのプログラミング教育について考える」シンポジウム実施を通じて

研究ノート

プログラミングを用いた授業づくりに向けて —「小学校からのプログラミング教育について考える」シンポジウム実施を通じて—

Designing programming lessons From discussion of “Symposium on programming education in elementary schools”

遠山紗矢香
Sayaka TOHYAMA
静岡大学情報学部

論文概要：2016 年 10 月 23 日に実施された「小学校からのプログラミング教育について考えるシンポジウム」で得られた知見を，2020 年から開始される予定の小学校でのプログラミング教育に向けて整理した．シンポジウムでの基調講演では，小学校でのプログラミング教育は「プログラミング的思考」の育成が目指されることが示された．一般講演では，手続きを書き出すことを通じて抽象的な概念を操作することがその育成に資する可能性が示された．実践報告では，小学校教員の立場から，子供向けプログラミング教育の先駆者の立場から，プログラミング教育の研究者の立場から，それぞれプログラミング的思考を育成するためのプログラミング活動の具体的な実践例が示された．これら実践例の共通点は「前向きアプローチ」，「モデルの構築と再吟味」，「協調学習」の 3 点に集約された．最後に，これら 3 点を実現するために「機能的学習環境」を構築することの意義について考察した．

キーワード：小学校でのプログラミング教育，コンピュテーショナル・シンキング，
プログラミング的思考，アクティブ・ラーニング，機能的学習環境

Abstract: This paper reviews the discussion of symposium on programming education in elementary school, which was held in October 23, 2016. Programming education will be carried out in Japanese elementary school from 2020 in order to make students possess an ability of “thinking as programming,” which is based on the policy of the Japanese Ministry of Education. In the symposium, the keynote speaker indicated the importance of development of computational thinking in every elementary education. Furthermore, the invited speaker showed some cases to make students learn the programming skills, and he demonstrated the possibility that the description of process by program contributes comprehension of conceptual operation. Additional three speakers provided each practical report based on their career such as an elementary school teacher, a pioneer of programming education for children, and a researcher respectively. They commonly pointed out the significance for programming education in elementary school, which is insisted the following three elements; forward approaching, structuring model and its verification, and collaborative learning. For the purpose of achieving these elements, it is strongly important to implement “Functional Learning Environment” is discussed.

Keyword: programming education in elementary schools, computational thinking, thinking as programming, active learning, functional learning environment

1　目的

本稿の目的は，2016 年 10 月 23 日に実施された，静岡大学情報学部主催「小学校からのプログラミング教育について考えるシンポジウム」の報告を行い，今後のプログラミング教育に向けて知見を整理することである．そのため

に本稿では，2 章にてシンポジウムを企画するに至った背景の整理やこれまでの議論の整理を行い，3 章にてシンポジウムの構成と概要について報告したうえで，4 章では講演について筆者の観点から学習・教育の先行研究を踏まえた解釈を行い，さらに 5 章で今後のプログラミングを用いた授業に向けた整理を行う．

本稿では，「プログラミング」や「プログラミング教育」という用語が様々な文脈で，多様な対象に用いられている現状を踏まえて，これらの用語を次のように用いる．まず，プログラミング言語を用いて ICT に対する命令を「プログラム」として記述し，そのプログラムを実行することによって，自分が期待した結果を得ようとする行為を広く「プログラミング」と呼ぶ．また，授業やワークショップといった学びの場における学習活動としてプログラミングを用いること全てを「プログラミング教育」と呼ぶ．この意味で，本稿で用いる「プログラミング教育」という言葉は，特定の教育目的や方法を規定しないことに留意されたい．

2　背景

2.1. 学校とプログラミング教育

これまで，日本でプログラミングに触れる学習者の多くは，情報系の専門高校や高等専門学校，大学を選択した者に限られていた．義務教育段階でプログラミングに触れる機会は，2012 年度以降の中学校の技術科「計測・制御」のみだった．一方で 2020 年からは，小学校の各教科にもプログラミングが取り入れられる方針が示され，中学校以上でも生徒がプログラミングに触れる機会が増加することが決定している (1)．これらの動きは，プログラミングが私たちにとってこれまで以上に身近なものになっていくことを含意していると考えられる．

海外では，小学校を含む学校教育課程でプログラミング教育が日本に先駆けて実施されている (2)(3)．こうした海外のプログラミング教育のカリキュラムで目にする機会が多いのは，「computational thinking」(4)(5)[1] を育成するねらいである．”compute” は「コンピュータ」の語源であり，「計算」を意味する単語である．しかし，加減乗除に代表される狭義の計算を指すだけではなく，コンピュータがそうであるように，情報を操作する処理全般を指すと捉えるのが妥当だろう．提唱者である Wing は，computational thinking はこれからの社会を生きるすべての人が獲得すべきスキルだと訴えている．人の創造的な活動を，計算機（コンピュータ）も活用しながらよりよく発展させるうえで有効だからだという．それは，計算機科学の領域で用いられる問題解決の考え方—問題をどのように加工すれば計算可能な状態にできるか，という観点に立って，複数の抽象レベルで解決方法を検討すること—が，様々な問題解決場面で効果を発揮すると期待されるからだという．これが実現されれば，再帰的に考えること，並列に処理すること，一連の手続きをパッケージ化して手軽に呼び出せるようにすることなどを用いて，多様なものごとを効率的に解決することが可能になると期待される．

computational thinking はプログラミングに留まるものではない．ではなぜプログラミングが computational thinking を育成すると期待されているかを推測すると，プログラミングは物理世界の制約を受けずに仮想世界を自由に構築できる手段だからだと考えられる．仮想世界だからこそ，試行錯誤を通じて問題そのものや問題解決方法のモデル化を検討できるという強みがある．

2020 年から日本の小学校で必修化されるプログラミング教育の目的が整理された「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）」(1) を見ると，computational thinking との共通点が少なくないように見受けられる．例えば，プログラミング教育は「将来どのような職業に就くとしても、時代を超えて普遍的に求められる力として

[1] computational thinking と類似する考え方は 1960 年代にすでに示されているが，本稿では近年のプログラミング教育と関連の強い論考のみに触れる．

の『プログラミング的思考』などを育むこと」であり，「コーディング[2]を覚えることが目的ではない」と説明されている．プログラミングという文脈に閉じたスキルではなく，プログラミングを通じて多様な文脈で生きて働くスキルを育成すること，つまり「プログラミング的思考」を育成することが目指されていると捉えられる．

また，議論の取りまとめにおいて「プログラミング的思考」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」と説明されている．これは，自分が期待することを他者に実行してもらうために，依頼先の相手が誤解しないように指示を出すことと同義である．この「誤解しない指示」が，「記号」という言葉に対応する．例えば，タクシーを拾った地点から自宅まで最短ルートで帰宅するには，タクシー運転手に対して「○○の交差点を△の方向へ進んでください」といったまぎれのない指示が必要になる．しかも，選んだ道路が渋滞していれば，次善のルートを再考し運転手へ指示をし直す必要がある．つまり，プログラミング的思考は，自分が実現したいことの計画を立て，その計画を自分以外の者が実行できるようまぎれのない指示を形成し，指示の実行結果を踏まえてより良い計画を考え直して指示をし直す，という一連の活動に表れると考えられる．

以上を踏まえると，プログラミング的思考は小学校段階が念頭に置かれていることもあり，computational thinking と比べてより基本的な概念だと捉えられる．プログラミング的思考に依ってものごとを処理する際に有用な知識，例えばデータ構造やアルゴリズムといった計算機科学の知見を学ぶことが，computational thinking を育成することへつながると期待できる．

[2] プログラミング言語を用いてプログラムを記述する行為

2.2. 次期学習指導要領とプログラミング教育

小学校からのプログラミング教育の実施が盛り込まれた 2020 年から実施される次期学習指導要領は，教師が「何を教えるのか」よりも，子供たちは「何ができるようになるか」に注目した指針が示される見込みである (6)．これに伴い，子供たちが知識をただ暗記するのではなく，生きて働く知識を獲得できるよう促すことが求められる．

そのために次期学習指導要領では，各学校おける「カリキュラム・マネジメント」の重要性が改めて示されている．各学校の状況に合わせて，各学校の教育目標を実現するために，子供達の学びの過程をより良いものにすべく，授業や課外活動等の編成を工夫することがこれまで以上に求められる見込みである．また，「社会に開かれた教育課程」というキーワードで示されているように，地域社会との連携を深めることでより良いカリキュラム編成を実現することも期待されている．プログラミング教育の文脈でも，「各小学校において，各学校における子供の姿や学校教育目標，環境整備や指導体制の実情等に応じて，教育課程全体を見渡し，プログラミング教育を行う単元を位置付けていく学年や教科等を決め，地域等との連携体制を整えながら指導内容を計画・実施していくことが求められる」(1) という説明がみられる．小学校でのプログラミング教育を実現するためには，地域等といかに連携するかを検討することが重要であると考えられる．

また，次期学習指導要領では，子供たちの「学び方」の質をこれまで以上に向上させるために，これまでの学習指導要領には明記されてこなかった「どのように学ぶか」も記載される見込みである．それが「主体的・対話的で深い学び（アクティブ･ラーニング）」というキーワードである．これは，子供一人ひとりが学びに対して主体的に向き合い，仲間同士，教職員や地

域の人，先哲の考え方を手掛かりに考えること等の「対話」を通じて，各教科等についてすでに知っていることと新しく学んだことを関連付けたり新しい考えを形成したりする「深い学び」を引き起こす重要性を示している．プログラミングにおいても，「一人で黙々とコンピュータに向かっているだけで授業が終わったり，子供自身の生活や体験と切り離された抽象的な内容に終始したりすることがないよう，留意が必要である」[(1)]という記載がみられる．このことは，子供一人ひとりが主体となって，自身の日常経験とプログラミングの間を往還しながら，対話的な学びを通じて事象や概念への深い学びへ至るような学習活動の必要性を示唆しているように受け止められる．

2.3. 静岡大学情報学部とプログラミング教育

静岡大学情報学部は，「情報学」を専門として設立された，日本の国立大学の中で最も歴史のある学部である．文系と工学系を融合させることによって新しい情報学をうちたてることを目指して，1995 年に設立された．本学部においてプログラミングは「融合」を実現するための手段である．文系も含む多様な研究対象—例えば言語，文化，産業，社会構造など—は，元来互いに異なるものである．しかし，データの形式を整えたり，手続きや事象の変化の過程を客観的に書き出したりといった共通的な記述方法に落とし込むこともできる．つまり，先述した computational thinking の観点でものごとを処理することで，文系や理系といった領域の違いや個別具体的なものごとの違いを意識する必要がなくなるのである．その利点を享受するために，情報学部では理系と文系のどちらの学生も，computational thinking を身につけることが期待される．だからこそ，情報学部生は全員が，アルゴリズムやデータ構造を含む広義のプログラミングを学ぶのである．

computational thinking が，プログラミング的思考に関わる知識を計算機科学の見地から豊富化したものである可能性を 2 章で述べた．また，小学校でのプログラミング教育がプログラミング的思考を育成するためのものであることも触れた．これらを踏まえれば，静岡大学情報学部での教育の少なくとも一部は，小学校でのプログラミング教育にも転用できるはずである．しかし，小学校の子供たちにとってどの程度の目標設定をすべきであり，その目標に向かっていかに小学校の子供たちに合った学習活動を提供するかといった，実践上の課題が残る．そこで静岡大学情報学部では，初等教育関係者も含む幅広い講演者を招いて，市民とともに小学校段階からのプログラミング教育の在り方について考える機会を設けることとした．それが，2016 年 10 月 23 日に実施されたシンポジウムである．学部内において本シンポジウムは，地域貢献の一環として位置づけられた．

3. シンポジウム概要

本章では，「小学校からのプログラミング教育について考えるシンポジウム」の概要について述べる．

3.1. 企画・構成

シンポジウムの主催者は静岡大学情報学部であり，実働は情報学部内に設置されたプログラミングシンポジウム WG が行った．本 WG のメンバは，シンポジウムの構想を打ち出した発起人である竹内勇剛教授に加えて，地域連携推進室の構成員，および筆者であった．

シンポジウムは，2016 年 10 月 23 日に「えんてつホール」（JR 浜松駅から徒歩 1 分，最大収容人数 約 500 名）にて開催した．幅広い市民の参加を促すために，参加費無料，事前参加登録不要とした．

シンポジウムの告知は，静岡大学情報学部 web サイト，浜松市教育委員会を通じた小中学校や公民館等でのチラシの配布，記者クラブへの周知を主な手段とした．また，関係者が SNS 等を通じて発信したり，メール等で知人へ周知

表１　シンポジウムの構成

時間	内容
13:00 〜 13:05	開会の挨拶： 酒井 三四郎（静岡大学情報学部長）
13:05 〜 13:15	シンポジウムの趣旨と進行の説明： 遠山 紗矢香（静岡大学情報学部）
13:15 〜 13:50	基調講演：「小学校段階で期待されるプログラミング教育の方向性」堀田 龍也（東北大学大学院 情報科学研究科 ; 文部科学省「小学校段階における論理的思考力や創造性，問題解決能力等の育成とプログラミング教育に関する有識者会議」主査）
13:50 〜 14:25	一般講演:「プログラミングを通して何を学んでいるのか〜情報系大学生の場合〜」 太田 剛（静岡大学情報学部）
14:25 〜 14:40	実践報告１：「小学校教員が考えるプログラミング教育」 冨永 浩司（静岡大学教育学部附属浜松小学校）
14:40 〜 14:55	実践報告２：「作ることで学ぶ　〜構築主義によるプログラミング学習の目的とその可能性〜」 阿部 和広（青山学院大学社会情報学部）
14:55 〜 15:10	実践報告３：「リテラシーとしてのプログラミング教育：日本のプログラミング教育のルーツ再訪を通して」 松澤 芳昭（青山学院大学社会情報学部）
15:10 〜 15:30	休憩（質問用紙の回収）
15:30 〜 16:10	パネルディスカッション： 田村 学（文部科学省初等中等教育局 視学官）及び上記発表者 5 名 司会：遊橋 裕泰（静岡大学情報学部）
16:30 〜 16:40	フロアからの質疑応答と議論
16:40 〜 16:50	まとめと閉会の挨拶： 竹内 勇剛（静岡大学情報学部）

したりするなどの手段も適宜用いられた．浜松市，浜松市教育委員会，静岡県教育委員会の各会より後援を頂き，静岡大学工学振興基金の協賛，浜松 IT キッズプロジェクト推進会議の協力を得た．また，当日は静岡大学テレビジョンの取材を受けた．

シンポジウムは表 1 の構成に示すように，小学校段階でのプログラミング教育の意義の確認と実践報告を踏まえたうえで，参加者からの疑問を広く受け付ける構成とした．基調講演は，小学校からのプログラミング教育の必修化を決定した組織である「小学校段階における論理的思考力や創造性、問題解決能力等の育成とプログラミング教育に関する有識者会議」主査の堀田氏に依頼し，万人に向けたプログラミング教育が必要とされる背景や今後の見通しをつかも

うとした．一般講演は，静岡大学情報学部の文系学科生も含む大学生にプログラミング教育を実施してきた太田剛教授に依頼することで，万人がプログラミングを学ぶ意義を考えるためのヒントを得ようとした．実践報告の 3 件は，学習者が学びを深めるための手段としてプログラミングを位置づけたうえで，小学校でプログラミング教育をそれぞれ異なる立場から実施した経験をお持ちの阿部氏，冨永氏，松澤氏に依頼した．さらに，パネルディスカッションでは田村視学官にもご登壇いただき,「生活科」や「総合的な学習の時間」の実践やカリキュラム研究を踏まえたプログラミングによる探究活動への示唆を得ようとした．

参加者は，開会前に配布された質問用紙に質問等を記入することで，休憩後に上記 6 名から回答を得ることができた．質疑応答では，参加者から挙手制で質問を受け付けた．

3.2. シンポジウムに対する筆者のかかわり

筆者は，シンポジウム発起人である静岡大学情報学部情報科学科教授 竹内氏らとともに，小学生向けのプログラミングワークショップを複数回行い，その効果を評価してきた [7][8]．このワークショップは，プログラミング教育に関する先行研究の知見 [9]，および認知科学や学習科学の領域で示されてきた協調学習の知見 [10][11] に基づいたものである．ワークショップの実施・評価で得られた知見を用いて，筆者は，実践報告で登壇した 3 氏の人選および調整，シンポジウム当日の全体の構想説明を担当した．

シンポジウムの冒頭説明では，(1) プログラミングそのものを学ぶこと，(2) プログラミングによって構築されたシステムを活用することを通じて学ぶこと，(3) プログラミングという行為を通じてプログラムで表象した対象についての理解を深めたり学び方について学んだりすることではプログラミング教育の目的が異なることを整理した．また，小学校へ導入されることを踏まえて，2020 年より開始される予定の次期学習指導要領のキーワードである「主体的・対話的で深い学び」，および「社会に開かれた教育課程」の 2 点を前提としてプログラミング教育の在り方を考える必要があることを述べた．さらに，プログラミング教育の必修化は「プログラミング」という教科の設置ではなく，既存の各教科にプログラミングを導入することが示されている点についても確認した．

3.3. シンポジウム当日の概要

本節では各登壇者の講演内容について，小学校でのプログラミング教育を実現するために特に重要だと考えられるポイントを中心に，筆者の要約を示す．質疑応答で登壇者が回答された内容についても，以下の要約に統合して示す．なお，シンポジウム当日の来場者は，浜松市近郊に在住する一般市民を中心とした 90 名弱（静岡大学情報学部関係者を除く）だった．当日の運営詳細については静岡大学情報学部地域連携推進室の報告書を参照されたい．

3.3.1. 基調講演：小学校段階で期待されるプログラミング教育の方向性（堀田龍也氏）

現行学習指導要領では，高等学校の共通教科「情報」のうち科目「情報の科学」でプログラミングを体験する機会がある．しかし，本科目は全国の 2 割程度の高等学校での実施に留まっている．中学校の「技術・家庭科」の技術分野では，プログラミングを体験するとしても時数としては数時間である．小学校ではプログラミングに触れる教科等はない．つまり，小学校から高等学校までの間でプログラミングを一切体験する機会がない子供もいると考えられる．

学習指導要領は 10 年に一度改訂される．次期の学習指導要領は，小学校では 2020 年度より実施予定であり，2016 年末には幼稚園から高等学校までのすべての新学習指導要領の方針がほぼ決まる予定である．改訂頻度と世界的な動向を踏まえると，プログラミング教育は今回の改訂に必ず含めるべきだと判断された．その

ために，小学校へのプログラミング教育の導入について専門的に検討するための組織が設置され，3 回の会議によって迅速に議論が取りまとめられた．

これほどの急展開となったのは，プログラミング教育がこれからの教育に不可欠だと考えられたためである．次期学習指導要領では，先生が「教える」だけではない，新しいモデルの教育への期待も示されている．こうした背景があるからこそ，今回の改訂にプログラミングを含めることができたとも言える．これから教員となる学生を育成する教員養成機関のカリキュラムや，現職教員に対する研修は，現時点では十分に対応が完了しているわけではない部分もあるが，新しいモデルの教育観に立脚して，企業や NPO といった学校外からの協力も得ることで，プログラミング教育を実施できる教員の養成・研修は不可能ではなくなると考えられる．

有識者会議での議論のとりまとめに示された「プログラミング的思考」という言葉は，専門的には computational thinking を指す．また，プログラミング的思考の説明にある「記号」とは，専門的なプログラミングの世界では一般的に「命令」と呼ばれているものを指す．これらの言い換えは，プログラミング教育のねらいを可能な限り平易な表現を用いて説明するために行った．

既存の教科では代替し難い，プログラミングでなければ学べないことは何かと言えば，「プログラムを作らなければわからないこと」だと考える．つまり，プログラミングを体験することを通じて，情報技術を見る見方・考え方を養うということである．例えば，日常生活で ICT が便利に動いていることに対して「うまく命令を出してくれた人がいる」と実感するだけでなく，どうすればより良く命令を出すことができるか，自分が命令するならばどのようにするか，といったことまで考えられるようになることが期待される．

プログラミング的思考として捉えることができる事象は，世の中に様々ある．例えば，算数で学ぶ筆算は「アルゴリズム」であり，音楽で学ぶリピートは「繰り返し」処理である．このような教科の中で，実際にプログラミングを体験させながらプログラミング的思考を育てていく．学校教育の各教科での学びに向かって，プログラミングに関する知見が集約されていくことで，学校の外からも手助けが可能になると期待される．これは「社会に開かれた教育課程」の実現にもつながる．

小学校でのプログラミング教育の実現のためには，プログラミングを行うための基盤となる ICT の整備も進めていく必要がある．プログラミング以前に ICT を道具として活用して学習することに慣れさせておく必要もある．大学や産業界とのさらなる連携も重視されていくと考えられる．

3.3.2. 一般講演：「プログラミングを通して何を学んでいるのか～情報系大学生の場合～」（太田剛氏）

高校までにプログラミングを全く学習する機会がないまま，文系入試を通り，さらに，一旦は大学におけるプログラミングの授業からドロップアウトしそうになってしまった学生に対する「自主ゼミ」を主催した経験から，今回はお話させて頂く．

プログラミングは，「コンピュータ」という他者に対して，自分の意図していることをしてもらうよう，指示書をつくる作業である．これは，ブロック遊び，作業全体の見通しの計画（プランニング），対話，の 3 つと類似点が多い．決められた部品を組み合わせて使うのがブロック遊びである．どの段階までに何が必要で，何を完了させておく必要があるのか，自分以外の者にもわかるように記述するのがプランニングである．背景や立場，考え方等が異なる相手であっても，相手の気持ちを想像して話をするのが対話である．プログラミングにおいては，指示書を作り，それを修正する過程を通じて，実

際には手で触ることができない仮想的なものを頭の中に作り上げる概念の操作を行う．子供にとってプランニングや，相手のことを想定して話をすること，抽象的な概念を操作することは，いずれも容易ではない．

プログラムは一般的に，想定通りに動かないことが多々ある．近年のプログラミング環境では，図形の配置と組み合わせでプログラムを記述するものが現れてきており，文法間違いで苦労することはほとんどないものの，想定通りに動かないことも少なくない．しかし，想定通りにいかない場合，間違っているのは必ず自分である．コンピュータは，言われたことを言われた通りに実施している．つまり，プログラムの誤りを修正してきちんと動くようにするには，「自分が間違っていること」を認めた上で，PDCA（Plan, Do, Check, Act）のサイクルを回すことが必須である．

間違いの原因は主に 3 つある．1 つ目は，問題に対する誤解である．これは，問題そのものに対する理解に不十分な点や誤りが含まれていたことを意味する．2 つ目は，指示の書き方の誤りである．これはプログラムの記述に誤りがあることを意味する．3 つ目は，想定外の使われ方がなされた場合である．これは，プログラム（システム）が人々にどのように使われるかについての想定が不十分だったことを意味する．こうしたいくつかの間違いの中から，自分が間違えた原因を特定するためには，自分自身が理想としている状態を認識し，現実にプログラムがどのように動いているかをしっかりと観察した上で，そのギャップが生じてしまった理由を論理的に分析して取り除くことの繰り返しが求められる．そこにプログラミングの難しさがある．

計画すること（プランニング）や，他者に適切な指示をして意図通りに仕事をさせることを学んだり，実際に計画を実施してうまくいかなかったときに何が起こっていて，それは何が原因かを追求したりする経験は，学校教育の中では組織的・系統的に行われていないように思われる．もちろん，学校の授業では上記のような活動が意図されていると想定されるが，これらを直接的に行いやすいのはプログラミングだと言える．プログラムを作ったり，間違いに気付いたり，その間違いを修正したりする一連の過程には時間がかかる．しかし，少しずつ成功経験を積み重ねていくことで，徐々にできるようになっていくと考えられる．重要なのは，このようにして PDCA サイクルを回し，自分の意図したことが徐々にできるようになっていくという経験が，子供の成長，自己実現のためにとても大事な活動であるという点であろう．

3.3.3. 小学校教員が考えるプログラミング教育（冨永浩司氏）

「ヘリウムガスを入れた風船を給食のお盆の上に乗せて，お盆から手を放すと，風船はどうなるでしょうか？」（筆者註：ヘリウムガス風船はお盆とくっついた状態で地面に落下した）これまでの授業では，こうした子供達の意表をつく演示実験を先生が行い，なぜ演示した結果のようになるかを先生が説明し，子供たちは先生の話を聞くことで学ぶスタイルが多かったと考えられる．

一方で，これからの授業では，子供が自分なりに「なぜ風船がお盆と一緒に落ちたのか」という問いに対して，納得できる説明を作りあげることが求められる．この活動を通じて，子供達の科学的な見方・考え方を養うことも求められると考えられる．こうした学びは必ずしも一人で行うだけでなく，集団での学びも用いられる．

自分自身は小学校にて理科の授業を担当することが多いが，これまでにプログラミングを行ったことはなかった．今回はシンポジウムに合わせて，自分が担当している 5 年生の図画工作にて，プログラミングを取り入れた単元を計画した．テーマとして，「大人になるまでにできるといいなと思うもの」を子供たちに考えて

もらい，そのイメージをプログラミングで表現する 12 時間扱いの単元とした．

授業は，1 時間を子供達が構想図を描く時間，3 時間を「Scratch[3]」や「Google Blockly[4]」，「プログラミン[5]」等を用いてプログラミングに親しむ時間，8 時間で子供達それぞれがプログラムを作る時間とした．作品づくりで用いるプログラミング環境は，子供達に自由に選ばせた．子供達は，プログラミング経験のない子から経験のある子までさまざまだった．子供達が構想した作品も，どこでもピントがあう眼鏡，ほしいものを出してくれる箱，人間も乗れるシャボン玉，障害物を見つけると自動的に停止するなどの機能が搭載された車など，一人ひとり異なっていた．

プログラミングの過程では，教師が「話し合いなさい」と声を掛けなくても自然と話し合いが始まっており，互いに指摘しあう様子が見られた．学習者自身が自分に合ったゴールを設定することができるのも利点だった．子供たちは自分なりに仮説を立てて検証し，プログラムを作っていた．期待したことができるようになると，さらに次にやりたいことを見つけて挑戦する様子も少なからず見られた．

これらの子供達の様子を見ていると，プログラミングは，これまでは技能のある一部の教師にしか実現できなかった授業を，さまざまな教師ができるようにする可能性があると考えられる．プログラミングは，子供達の主体性を最大限に引き出すことができる手段だからである．また，「もうちょっと頑張ってみよう」という子供達の態度も引き出せる．子供が主体的に体験する過程を通じて，様々なことが学習されていると感じる．子供達が想像したものを描くだけでなく，それを動かすことができる手段はプログラミングの他にはない．

制度や授業時数など，乗り越えなければならない課題はあるものの，子供たちと一緒に教師が取り組み，学んでいくことが，結果として児童にも良い学習機会を提供することにつながるのではないか．そのためには，教師も楽しみながらプログラミングを取り入れることが重要だと考える．

プログラミングの評価は，長い目で検討することが重要だと考える．例えば，道徳の授業で 1 時間学べばすぐに道徳的に素晴らしい子供になるかと言えば，必ずしもそのようになるわけではないことは，多くの大人が知る通りである．「何を学んでいるか」については，究極的には子供を信じるしかない側面もある．一人ひとりの子供がプログラミングを経験したその先に，よりよい学びが生まれていくことに役立つような評価をしていくことが重要だと考える．

3.3.4. 作ることで学ぶ　～構築主義によるプログラミング学習の目的とその可能性～（阿部和広氏）[6]

現代の子供たちは，生まれたときからパソコンやスマートフォン等の ICT に囲まれていたことから「デジタルネイティブ」と呼ばれる．彼らは大人が驚くほどの速さでハードウェアやアプリケーションを使えるようになる．しかし，与えられたものを流暢に使うことができるだけの状態は，スキナー箱に入れられたネズミと何が異なるのか？「ICT を使いこなす」と表現できるレベルに到達するには，何が必要か？

この問いに対して，子供たち自身が興味のあるものを作る過程で学びが引き起こされるという考え方，すなわち「構築主義」（Constructionism）(12) の重要性を主張したい．プログラミングの文脈における構築主義は，Papert の言葉を借りれば，「コンピュータに子供をプログラムさせる」のではなく，「子供にコンピュータをプログラムさせる」ことである．子供がプログラミングを行う過程で，結果として学びが引き起こされると考えるのである．他人から教えられたことが子供たちの興味関心から遠ければ，子供達の身には付きにくい，とい

[3] https://scratch.mit.edu/
[4] https://blockly-games.appspot.com/
[5] http://www.mext.go.jp/programin/
[6] 本講演資料は http://www.slideshare.net/KazuhiroAbe2/ss-67580067 でご覧いただける．

う考え方になる．

例えば大学生が，高等学校までの間に学んだ知識で解けるはずの応用問題を解けないことが少なくない．受験やテストで出題されたことがある問題の形式でないときに，より困難になるようである．これらは，子供が大学生になるまでの間に，自分にとって興味関心があること以外は流している可能性を示している．それならば，子供たちの興味関心にまず寄り添うことで学びを引き出せるのではないか．

「Scratch」というプログラミング環境では，命令を出してネコを動かし，そのネコの軌跡で図形を描くことができる．正方形を描くには，ネコに「100 歩動く」「90 度回る」という 2 つの命令を「4 回繰り返す」よう命令すればよい．では正三角形の場合はどうすればよいか？子供だけでなく大人も，正三角形の性質を知っていれば「100 歩動く」「60 度回る」という 2 つの命令を「3 回繰り返す」ことで実現できると考えるだろう．しかし，実行すると誤りに気付く．ネコが平面上を回りこむためには外角である 120 度を指定しなければならないのである．

私たちは，正三角形の内角の大きさ（60 度）を，経験から切り離された知識として持っているために，自分が自転車を運転しているときに「60 度回る」とどうなったかという経験より優先してしまう．ものを作る，つまりプログラムを作る経験を通じて子供達が自分で法則を見出すことが，構築主義のやり方である．

構築主義の観点に立って子供達の遊びを観察すると，ブロック遊びでもクレヨン遊びでも，「想像，作成，遊び，共有，振り返り，想像…」という螺旋構造（creative learning spiral）[(13)] がみられる．学習にもこの螺旋構造を活かすことができるのではないか．そう考えて，小学生に対して Scratch を用いたワークショップや授業を手掛けてきた．三鷹市立中原小学校「中原はちのすけクラブ」では，子供同士で「どうやってやったか教えて！」と会話が生まれる例が数多く見られた．仙台市立荒町小学校では，先生の教示を守らずに，ネコよりネズミの方が大きかったり，キャラクターを必要以上に数多く作ったりすることで楽しむ子供の姿が見られた．品川区立京陽小学校では，学年・教科を問わずにプログラミングを用いた授業作りを行っている．中でも，3 年生理科「風やゴムのはたらき」の単元では，輪ゴムを引っ張る長さを変えたときのゴム車の動き方について仮説を立て，Scratch でシミュレーションを行う授業が実施された．仮説検証的な考え方は小学生でも可能なことが示された．プログラミングを経験した子供たちは，試行錯誤したり，自ら進んで考えるようになったりしたという．

コンピュータは子供達の多様な興味関心に寄り添うことができるツールである．プログラム次第で何にでもなり，反応がすぐに得られる．何度でも間違うことができ，時間や空間の制約を受けない．生活空間や住んでいる場所の違いも乗り越えることができる．子供達が外で遊ぶのと同じように，プログラミングを経験する機会も作りたい．そのために，本を出版したり [(14)][(15)][(16)] テレビ番組「Why? プログラミング」を制作したりしてきた．これからさらに事例を増やしていくことが望まれる．

楽しいことには本質的な価値がある．先生方にもプログラミングを楽しんでいただきたい．教員向け研修会でも，まず先生方自身に楽しんでいただくことを重視している．子供に主導権を渡す授業作りの仕掛けの一つとしてプログラミングを活用していただくことで，楽しむことから学びを引き起こすためのお手伝いができればと思っている．

3.3.5. リテラシーとしてのプログラミング教育：日本のプログラミング教育のルーツ再訪を通して（松澤芳昭氏）

プログラミング教育の目的は様々だが [(17)]，万人に向けてのプログラミング教育として最も意義深いのは「プログラミングで学ぶ」ことだと考える．その理由は，「概念が意味をなす状

況」をプログラミングによって作ることができるためである．概念とは，整数，分数，円などの数学的な概念はもちろんのこと，すべての教科で学ぶ概念的な知識を指す．「概念が意味をなす」とは，概念的な知識が私たちの生活や体験といった具体的な世界で実際に機能することを指す．例えば，店で買い物をするときには買い物の合計金額が手持ちの合計金額に収まるかを計算できる必要がある．こうした状況は，子供は加減算を学ぼうとする動機づけとなる．一方で，分数同士の割り算をするために，割る数の逆数をとって掛け算する手続きを教えることは，概念が状況から独立してしまっていると言える．

戸塚滝登氏が 1980 年代に小学校で実施されたプログラミング教育では，まさに概念的知識が意味をなす状況が作り上げられていた．正多角形の各外角の大きさを求める戸塚氏の算数科の授業にて，「正 n 角形」を描くには外角 1 つがいくつになるかを子供たちが考える実践が行われた．授業では，戸塚氏が日本の子供たちのために用意した「LOGO」が用いられた．LOGO は，子供がプログラミングを通じて学習するために構築されたソフトウェアである [17]．この授業では，画面の二次元平面上を自由に動ける「タートル」に対して，学習者が角度や長さを使ってプログラムを作り，動き方を指示することで，タートルの動いた軌跡を描く LOGO の機能が用いられた．

戸塚氏は，正 11 角形を描くためのプログラムを子供たちに考えさせた．正五角形や正六角形の図形を学んできた子供たちは，既習事項を活用して正 11 角形の外角の大きさの仮説を立て，プログラムを作った．子供たちが作ったプログラムが実行された結果，固唾をのんで見守る子供たちの目の前でタートルは見事な正 11 角形を描いた．子供たちはその瞬間，跳ね上がるように立ち上がり，手を叩いたり叫んだりして思い思いに歓喜をあらわにした（筆者註：子供達の興奮度は授業時間中とは思えないほどだった）．

上述の授業は，子供だけでできることよりも少し上の課題を教師（戸塚氏）が与えることで，子供たちを学びへ向かわせていたと捉えることができる．子供自身が作った仮説を検証するためにプログラミングを位置づけることで，多角形の性質，つまり算数科の内容に対する子供たちの理解を深めることが意図されている．子供たちにとってプログラミングが自分の考えを評価する鏡になっていた．子供自身が本気で考えたこと（プログラム）を実行するからこそ，子供は実行結果を楽しみに待ち，自分たちの考えが正しかったことを目の当たりにしたことで歓喜したと考えられる．一方で，もし，子供たちが解を簡単に自力発見できるような課題だったり，場当たり的な試行錯誤が動機づけられるだけの課題だったりした場合，上記のような興奮は起こらなかったのではないだろうか．

学習評価や教師の役割についても，戸塚氏の実践は貴重な示唆を与える．上記のような実践であれば，プログラミング能力そのものを評価する必要はなく，プログラミングが扱う概念に対する子供の理解の深まりを評価すればよい．教師には，子供たちが試行錯誤するのを支援し，子供自身が自分の考えを評価するためにプログラミングを使えるよう促す役割が期待される．幼児教育の文脈で言えば，子供が様々な道具で遊ぶことで概念的な理解を深めるモンテッソーリ教育 [18] にも通じるものがある．子供たちがプログラミングを通じて学ぶことを愛することができるようになる（Love of learning by programming）ことを願う．

なお，戸塚氏の実践は戸塚氏自身の書籍 [19][20] をはじめ，複数の文献 [21][22][23] に紹介がある．

4. 講演に対する筆者の解釈

以上の実践報告では，プログラミングを通じて，いかにプログラミング的思考を育成するかについての具体例が示されていたと考えられる．そこで本章では，今後のプログラミング教

育に資する知見を得るために，堀田氏と太田氏の講演内容に依拠しながら，これらの実践報告に共通するポイントを整理する．まず，ポイントを 4.1 節から 4.3 節にて整理する．続く 4.4 節にて，各ポイントと実践報告とを対応づけて解釈する．

4.1. 前向きアプローチ

堀田氏の講演からは，プログラミングが，「新しいモデルの教育」の在り方を具体化する可能性が感じられた．筆者は「新しいモデル」とは，教師が知識をわかりやすく伝達し学習者はそれを受け止めるという講義型の授業だけでなく，学習者が主体となって知識を創り上げ，学習目標を達成したらその先にさらに次のゴールを探すことができるような授業を指すと考える．

こうした実践は，「前向きアプローチ」（あるいは創発的アプローチ）と呼ばれる実践づくりのアプローチ [(24)] と捉えられる．Scardamalia らによれば，前向きアプローチとは，既存の教育目標に対して子供が「今できること」，「わかること」を出発点に，教育現場がより良い教え方を探し，必要な場合は目標自体も随時修正する．子供が目標を超えて学ぶ姿を見せれば，それに合わせて目標を高く設定し直すことも行う，と定義されている（解説資料 [(25)] 参照）．これと対比的なのは，「明確に定義された教育の大目標から逆算して，そこに向かうための下位目標を設定し，発達段階に応じて学年ごとに割り振る．教育現場は，設定された目標から見て，子供のレベルの不足を把握し，差を埋めるように教育する」という「後ろ向きアプローチ」とされている．

前向きアプローチの実践例として，小学校 1 年生の子供たちの疑問「秋になると葉っぱが赤くなるのはなぜだろう」を契機に，子供達同士が協力して調査をしたり仮説検証をしたりしながら自分たちでその答えを創り上げたり，さらにその先にある「知りたいこと」を見つけたりしていくトロント大学の実験校での実践が示されている．教師は，電子掲示板システム「Knowledge Forum」を提供したり，子供達の進度に合わせて掲示板に新しい資料を提供したり，子供達に掲示板の書き込み内容を集計するよう促したりする支援（足場掛け）[(26)] を通じて，子供達の主体的な学びを促す．子供自身がもともと学ぶ力を持っている [(27)] からこそ，能力を子供達が発揮できるように支援することが重要になる．

一方で，子供達に疑問の発見を委ねると，教師が子供たちに学んでもらいたいと願っている学びが起こり難いのではないか，という懸念もある．これに対しては，東京大学 CoREF[(28)] にて蓄積されている授業例が参考になる．CoREF に集められた小学校から高等学校までの協調学習型の授業例では，授業の始めに教師が示した問いを超えて，授業終了時には子供達が自分なりに新しく知りたいことや疑問を見つけることが示されている．松澤氏の見解「子供だけでできることよりも少し上の課題を教師が与えること」が子供達の学びを引き出すうえで有効だという裏付けになっていると考えられる．

4.2. モデルの構築，再吟味

プログラミングの特長を知ることは教科の特性に合わせてプログラミングを活用するための近道になると考えられる．太田氏の講演はその手掛かりを提供しているように思われる．太田氏は，プログラミングが具体的な文脈から手続きを取り出して抽象的な概念を作ることや，PDCA のサイクルを回すことでその概念を自分の期待する方向へと作り変えていくことができる点を指摘した．これらは，プログラミングが抽象的な概念，つまりモデルを形成することを促すことや，そのモデルをより良く作り変えていく活動が PDCA サイクルで後押しされることだと捉えられる．

大人は日常生活の中で，複数の事例から一般的な規則を見つける帰納的推論を行っている [(29)]．現実の問題が抽象的な規則として書き出さ

れることは，帰納的推論はモデル的な知識を構築するための過程とも捉えられる．

子供も，眼前の現象に対して自分なりに規則性を見出す能力を持っていることが，計算問題を解く過程で見られる規則的な間違いから示されている[30]．つまり，子供の計算間違いは表面的な試行錯誤の結果と言うよりは，子供なりのモデル的な知識に基づいた，系統立てられた操作の現れだと考えられる．

これらより，子供の学びを促すためには，子供達がモデルを作り変えていく再吟味の機会を設けることが重要だと考えられる．そのためには，子供たちがモデルを再吟味するための方法そのものを学ぶことが重要なように思われる．しかし，特定の文脈から切り離した，汎用的な「モデルを再吟味するための方法」を教え込んでも，文脈に依存しないモデルを再吟味するスキルを子供に獲得させることは困難である[31]．そこで，子供がその「方法」を繰り返し使う経験ができるように学習活動を整えるアプローチ[32]をとることになる．

このアプローチは，学校教育場面では，様々な授業で繰り返しある方法を経験できるようにカリキュラムを編成することで実現できる．例えば，アメリカの幼稚園から高等学校までの期間を対象とした科学教育の標準カリキュラムでは，モデルを構築したりモデルを活用したりする学び方（developing and using models）[33]が，様々な題材を学習する場面に，何度も繰り返し登場する．近年表されている computational thinking の特徴的な側面の中にも「モデリング」が示されている[34]．太田氏が，プログラムの作成や修正には時間がかかるが，少しずつ成功経験を積み重ねていくことで，徐々に自分の意図したことをプログラミングで実現できるようになる可能性を指摘しているように，プログラミングを通じて，繰り返し学び方を経験することが重要だと考えられる．

4.3. 協調学習

阿部氏の示した creative learning spiral[13]は，太田氏が示した PDCA サイクルを学習者中心の学びの文脈でさらに具体化したものと解釈することもできる．一方で creative learning spiral に特徴的だと考えられるのは，「share」が含まれている点である．

一人ひとりが自分なりの考えを持ったうえで，自分とは異なる他者と話し合うことで，自分の考えを見直したり自分なりの理解をさらに深めたりすることは，Miyake[35]が「建設的相互作用」と説明した現象である．建設的相互作用は，互いの考えが見えやすく，互いの考えの正誤が容易には判断し難いときに促されるという[36]．筆者が運営したプログラミングワークショップを振り返れば，子供同士が互いに自分のプログラムを見せ合いながら話し合うことでバグを取り除いていく建設的な対話が起こっていた[7]．

子供たちが自分たちの能力を最大限発揮しながら，自分の考えを見直し，より良いものへと作り変えようとする学習過程では，困難に直面することも少なくない．その困難を乗り越えるためには，協調学習が有効だと考えられる．Clement[37]は，子供たちが日常経験で身に着けた素朴なモデルと，学校で学ぶような理論や原則の間が乖離しているために，子供たちの知識は短期のうちに剥落しやすいことを指摘した．これに対して．三宅・三宅[38]が示した「知識の社会的構成モデル」は，子供たちの素朴モデルと，理論的知識の間を子供自身が結びつけていくことで深い理解に到達すると考えるモデルである．子供自身の素朴モデルと理論は矛盾することも少なくないため，この間を結びつけるために子供同士が対話をすることが有効だと考えられている．

プログラムは限られた種類の部品を用いて作るため，子供達が互いに見比べたときに差異が見えやすいことが期待される．外化物としてのプログラムを活用しながら協調学習を行うこと

で，子供達がプログラムの改変を通じて自分の考えをもより良く作り変える機会になる可能性がある．

4.4. 実践報告との関連性

4.4.1. プログラミングと前向きアプローチ

本シンポジウムの実践報告は全て，プログラミング教育の文脈における「前向きアプローチ」の具体例と捉えられる．これらの実践で子供達は，一人ひとりが主体となって作りたいものを作る過程で，必要となる知識や技術について試行錯誤したり仲間と相談したりしながら，広義の問題解決を進めていた．例えば冨永氏の講演では，ゴールの設定を子供達に委ねることで，子供達が自分の仮のゴールをさらに高みへと作り変えていく制作活動が実現されたことが示唆された．阿部氏の講演では，「ネコがネズミを追いかけるゲームを作る」という教師が与えた当初のゴールすら超えて，子供達がゲームを独自に「改良」して楽しんだ事例が報告された．戸塚氏の実践では，正三角形を LOGO で描いた経験を活かして，子供たちが正 n 角形を描くための外角の大きさの規則を見つけようとしていた．

これらは特別な子供たちだからできたわけではない．阿部氏や戸塚氏の実践は一般的な公立小学校で行われている．冨永氏の実践も，プログラミングを初めて経験する子供たちが大部分を占める学級で行われている．また，特別にプログラミングに詳しい教師だからできたわけでもない．冨永氏の報告は，自身初めてのプログラミングを用いた授業である．冨永氏の実践からは，プログラミングが子供たち一人ひとりの学びに寄り添うことができる性質を活かして，子供たちの主体的・対話的な学びを引き出す足場掛けを行うことの重要性が示唆される．また，講演で上映された戸塚氏の授業風景からは，初発の問いを教師が決めても，子供たちの前向きな学びは阻害されるどころか，さらに深まることが示されている．この点については次節でも触れる．

4.4.2. プログラミングとモデル構築，再吟味

阿部氏の講演にて紹介されたゴムの力のシミュレーションや，戸塚氏の n 角形の角度を求める授業は，子供が既知の複数の事例を統合して自分なりのモデルを構築することや，プログラムの実行結果やゴムの力で動く車の実行結果を観察してモデルを再吟味している例だと考えられる．教師は子供達がより良いモデルを作り上げられるように，問いを出したり，シミュレーションができる環境を整えたりすることで足場掛けをしていると捉えられる．

授業が子供達の興味や疑問感に寄り添いながら連続的に構成されているのも，足場掛けの一種だと考えられる．松澤氏の講演で示されたように，「自分の考えを評価するためにプログラミングを活用する」ことで，子供は自分で新しい考えを創り上げる支援を得たと考えられる．戸塚氏は子供の帰納的な推論を促すために，正三角形の次は正方形，正五角形，それなら正 n 角形は？といったように問いかけていくことで，子供たちだけでは困難な，モデル的な知識を構築する足場掛けをしていたと解釈できる．戸塚氏がプログラミングの特長と教科の本質の両方を結び付けることで，こうした高度な足場掛けが実現されたと考えられる．

4.4.3. プログラミングと協調学習

阿部氏の講演では，子供同士が「それどうやったの？」などと互いに話し合う様子が示されていた．冨永氏の実践や戸塚氏の実践でも，プログラミングを通じて，子供たちが自然と関わり合いながら学習を進める様子が示されていた．これらの過程で子供達は，自分なりのプログラムやアイディアを仲間同士で話し合うことで，何かしらの手掛かりを得ていたと推測される．しかし，子供達が対話を通じてどのように学びを深めたかについては，講演時間が限られてい

たこともあり，個々の事例について詳細に確認することは困難だった．

今後は，協調過程の詳しい記録を取ることで，子供一人ひとりが学びの主体となり対話に参加することで深い学びに到達したかを評価できる仕組みが必要になるだろう．また，子供たちのプログラムの改訂履歴を参考にすることで，対話のみでは容易でない分析も可能になると考えられる．それによって，子供たちにとってモデル化やモデルをより良く作り変える過程がこれまで以上に明らかになる可能性がある．

5. 筆者が考える「プログラミング教育」

以上を踏まえて筆者は，小学校の各教科におけるプログラミング教育では，上記 3 点を実現するための学習環境としての「機能的な学習環境」(21) を作り上げることが有効だと考えた．機能的学習環境とは，学習活動が子供達の目から見て目的や明確な働きを持っており，自然な形で学習活動が引き起こされるように学習環境全体を構成するという考え方である．

5.1. 機能的学習環境を構築するために

機能的学習環境は，30 年以上前に書かれた三宅氏の書籍にあらわされた考え方である．本稿で再度この考えをとりあげたのは，30 年以上が経過した現在も，この基本的な考え方を前提としたうえで議論を深めていきたいと考えたためである．この見地に立つと，各教科の本質的な問い（Big Idea）(39) について深める学びを，子供達自身が主体となって「前向き」に進める過程で，プログラミングをいかに効果的に活用するかを検討するという考え方が示される．

例えば，子供達がモデル的な理解を作り上げたり，子供達が日常経験から得たデータを入力してモデルを検証したりといった学習は，プログラミングを用いることで行いやすくなる．その過程には，ものごとを抽象化したり，抽象化したモデル的な対象を直接操作したりといった，子供にとって難易度が高い課題も含まれる可能性が高い．一人では難しい課題だからこそ，子供達同士が互いに意見を出し合いながら協調的に議論する必然性も生まれる．プログラムから実行結果が返されるため，子供が自分で自分のモデルを検証しより良いものへと作り変えていく「前向き」な学びも促しやすい．

このような学習環境を実現するためには，堀田氏が述べるように，企業や NPO といった学校外の組織との連携が有効だと考えられる．もしそれが実現すれば，新しい「学び方」を子供達に提供できる可能性が大いにある．そこで以下の節では，こうした学習環境を設計するためにプログラミングの特長をどのような場面で具体的に活かすことができるかについて，筆者の考えを示す．

5.2. 子供達の多様な思考を外化する

子供達が作成したプログラムを観察し，より良い授業づくりのための手掛かりとして活用できる可能性がある．プログラムは，子供がどのような考えを持っているのか，どのような手続きでその考えを実現しようとしているのか，といったことが表された外化物とみなせる．教師にとっては，子供がどのようなプログラムを作ったかを見ることで，その授業で子供達がどこまで到達したかを形成的に評価しやすくなる．評価結果は次の授業のやり方を調整するための手掛かりとなる．前向きアプローチの授業で，問いを設定するために，子供たちの現状を把握するための手段として有効だと考えられる．

子供にとっては，子供同士でプログラムを参照し合うことで，互いのプログラムの差異について「なぜそのようになるのか」について話し合う活動のきっかけとなる．プログラムは限られた種類の部品を用いて作るため，プログラムの差異を見つけるのは自然言語よりも容易な可能性がある．

また，次節とも関連するが，子供自身がプログラムの動きを追って自分の考えの誤りを見つ

ける手助けをすることもある．戸塚氏の実践で育った子供は，プログラムの最後から順に一行ずつ遡って確認を行うようになっていた(20)．この手法は，その子供が筆算のたしかめを行う際にも自然と用いられていたという．このように，プログラミングが，学び方を学ぶ手段になる可能性もある．

5.3. 子供達がモデルを自分自身で吟味する

子供が自身のモデル的な知識を吟味するための手段としてプログラミングを位置付けられる可能性がある．プログラミングでは，プログラムと実行結果の両方が明示される．実行結果が期待通りでないならば，「自分が間違っている」（太田氏）ことを認めて，自身のモデル的な知識の現れであるプログラムを修正することが求められる．プログラムの修正結果も，即座に実行結果として確認できる（阿部氏）．

ただし，「場当たり的な試行錯誤が動機づけられる課題では興奮は起こらないのではないか」（松澤氏）という見解も踏まえると，子供が手探りでプログラムを実行したり修正したりしているだけの場合には注意すべきだと考えられる．作成したプログラムを実行しても思い通りに動かなかった場合，プログラミングに不慣れな子供は，全てのプログラムを消して最初から書き直す傾向も示されている(21)．子供が自分のモデルの間違いを認め，その原因を見つけて修正していく過程そのものが促されるように，プログラミングが用いられることが期待される．小学校の場合，この追究の対象が教科の本質的な問いと重なるような設計をすることが求められるだろう．

最後に，プログラミングという行為がすべからく高次の認知的な能力（例えば「論理的思考力」等と呼ばれるもの）を育成するわけではないことを改めて強調しておきたい．1980 年代に数多く行われたプログラミング教育で示されたように，プログラミング自体が子供のプランニングや，論理的に考えることのパフォーマンスを向上させるわけではない(40)．プログラミングがその目的と照らして適切に位置付けられる必要がある．これはプログラミングに限らず，学習を支援するツールやシステム等に共通して言えることである．

謝辞

本シンポジウムのご講演をお引き受けくださった堀田龍也氏，太田剛氏，冨永浩司氏，阿部和広氏，松澤芳昭氏，田村学氏（登壇順）に記して感謝する．講演者の皆様および戸塚滝登氏には，本要約について確認・修正を賜った．本シンポジウムの実施には，静岡大学工学振興基金および静岡大学情報学部の支援を得た．磐田市立城山中学校には，シンポジウムの広報にご尽力を賜った．また，本シンポジウムは，静岡大学情報学部地域連携推進室（岡田安功室長）をはじめとする多くの皆様の協力によって実現した．本稿のとりまとめには静岡大学情報学部竹内勇剛氏より助言をいただいた．

文献

(1) 文部科学省 (2016).『議論の取りまとめ（案）：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』．文部科学省 . http://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1371901.htm (2016.11.20 参照)

(2)European Schoolnet (2015). *Computing our future: Computer programming and coding*. Priorities, school curricula and initiatives across Europe. http://www.eun.org/c/document_library/get_file?uuid=3596b121-941c-4296-a760-0f4e4795d6fa&groupId=43887 (2016.11.20 参照)

(3) 総務省 (2015).『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』. http://www.soumu.go.jp/main_content/000361430.pdf (2016.11.20 参照)

(4)Wing, J. M. (2006). Computational Thinking. *Communications of the ACM,* 49(3), 33-35.

(5)Wing, J. M. (2011). Research notebook: Computational thinking - What and why? The Link Magazine. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why (2016.11.20 参照)

(6) 文部科学省 (2016).『幼稚園、小学校、中学校、高等学校及び特別支援学校の学習指導要領等の改善及び必要な方策等について（答申）』. 中央教育審議会 . http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/1380731.htm (2016.12.30 参照)

(7) 遠山紗矢香・松澤芳昭・横山昌平・高口鉄平・竹内勇剛 (2016). 協調的なプログラミングを促す PBL 型学習環境の構築 .『HCG シンポジウム 2016 予稿集』, 228-235.

(8) 遠山紗矢香 (2015). 協調的な学びとプログラミング .『高校教科「情報」シンポジウム 2015 秋資料集』, 1-7.

(9) 遠山紗矢香 (2017). 第五章 : プログラミング教育の動向 .『資質・能力を育成する教育課程の在り方に関する研究報告書 4：ICT リテラシーと資質・能力 , 国立教育政策研究所 .

(10)Hmelo-Silver, C. E., Chinn, C. A., Chan, C. K. K. & O' donnell, A. (Eds.) (2013). *The International Handbook of Collaborative Learning*. New York: Routledge.

(11)Sawyer, K. R. (Ed.) (2014). *The Cambridge Handbook of the Learning Sciences 2nd edition.* Cambridge: Cambridge University Press.

(12)Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. NY: Basic Books.

(13)Resnick, M. (2007). Sowing the seeds for a more creative society. Learning and *Leading with Technology*, 35(4), 18-22.

(14) 阿部和広 (2013).『小学生からはじめるわくわくプログラミング』. 東京 : 日経 BP 社 .

(15) 阿部和広・橋爪香織・谷内正裕 (2014).『5 才からはじめるすくすくプログラミング』. 東京 : 日経 BP 社 .

(16) 阿部和広・倉本大資 (2015).『小学生からはじめるわくわくプログラミング 2』. 東京 : 日経 BP 社 .

(17) 久野靖・和田勉・中山泰一 (2015). 初等中等段階を通した情報教育の必要性とカリキュラム体系の提案 .『情報処理学会論文誌教育とコンピュータ』, 1(3), 48-61.

(18)Montessori, M. (1982). *The secret of childhood*. New York: Ballantine Books.

(19) 戸塚滝登 (1989).『クンクン市のえりちゃんとロゴくん』. 東京 : ラッセル社 .

(20) 戸塚滝登 (1995).『コンピュータ教育の銀河』. 東京 : 晩成書房 .

(21) 三宅なほみ (1985).『教室にマイコンをもちこむ前に』. 東京 : 新曜社 .

(22) 佐伯胖 (1986).『コンピュータと教育』. 東京 : 岩波新書 .

(23) 寺尾敦 (2012). ICT を活用して深い学習を支援する .『コンピュータ＆エデュケーション』, 33, 28-33.

(24)Scardamalia, M., Bransford, J., Kozma, R. & Quellmalz, E.（2012）. New assessments and environments for knowledge building. In P. Griffin, B. McGaw, & E. Care (Eds.), *Assessment and Teaching of 21st Century Skills*. Dordrecht: Springer, 231-300.

(25) 白水始 (2016). 前向き授業と ICT.『学習情報研究』, 251, 4-7.

(26)Wood, D., Bruner, J., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Child Psychiatry,* 17, 89-100.

(27) 三宅なほみ（2015）. 三宅なほみ 最後の論文 .『認知科学』, 22(4), 542-544.

(28)CoREF (2016).『自治体との連携による協調学習の授業づくりプロジェクト 平成 27 年度

活動報告書 協調が生む学びの多様性 第 6 集 ―私たちの学習科学を育てる―』. 東京大学大学発教育支援コンソーシアム推進機構 . http://coref.u-tokyo.ac.jp/wordpress/wp-content/uploads/2016/05/H27houkoku_all.pdf (2016.11.20 参照)

(29)Gleitman, H. (1991). *Psychology: Third Edition.* London: Norton & Co.

(30) 野島久雄 (1982). 手続き的バグの診断・生成・除去 . 波多野誼余夫（編）『認知心理学講座 4 学習と発達』, 東京 : 東京大学出版会 , 219-229.

(31)Bransford, J. D., Brown, A. L. & Cocking, R. R. (2000). *How people learn: brain, mind, experience and school*. Washington D. C.: National Academy Press.

(32) 国立教育政策研究所 (2016).『資質・能力理論編』. 東京 : 東洋館出版社 .

(33)NGSS Lead States (2013). *Next Generation Science Standards: For States, By States*. Washington D. C.: National Academy Press.

(34)Grover, S. & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher,* 42(1), 38-43.

(35)Miyake, N. (1986). Constructive interaction and the iterative process of understanding. *Cognitive Science,* 10(2), 151-177.

(36) 三宅なほみ . (2000). 建設的相互作用を引き起こすために . 植田一博・岡田猛 (編著)『協同の知を探る』, 東京 : 共立出版 , 40-45.

(37)Clement, J. (2008). The role of explanatory models in teaching for conceptual change. In S. Vosniadou (Ed.), *International handbook of research on conceptual change*. New York: Routledge.

(38) 三宅芳雄・三宅なほみ（2014）. 実践の科学としての教育心理学 . 三宅芳雄・三宅なほみ（編）『教育心理学概論』. 東京 : 放送大学教育振興会 , 11-23.

(39)Wiggins, G. & Mctighe, J. (2005). *Understanding by design*. Alexandria: Association for Supervision & Curriculum.

(40)Pea, R. D. & Kurland, M. D. (1984). On the Cognitive Effects of Learning Computer Programming. *New Ideas Psychology,* 2(2), 137-168.