コンピュータとの対話を通したプログラミング教育の授業デザインの提唱とその実践

北島 茂樹[*1]・山中 脩也[*2]・喜田　綾芽[*3]
Email: shigeki.kitajima@meisei-u.ac.jp

*1: 明星大学教育学部教育学科
*2: 明星大学情報学部情報学科
*3: 明星大学大学院情報学研究科

◎Key Words プログラミング教育，コンピュータとの対話，コーディング

1. はじめに

文部科学省（2018）の『小学校プログラミング教育の手引（第一版)』によれば，「コンピュータを理解し上手に活用していく力を身に付けることは、あらゆる活動においてコンピュータ等を活用することが求められるこれからの社会を生きていく子供たちにとって、将来どのような職業に就くとしても、極めて重要なこと」になるという．そのため，今回の学習指導要領の改訂において，小・中・高等学校を通じてプログラミング教育の充実を図り，2020 年度から小学校でプログラミング教育を導入することになっている．

また，諸外国においても，初等教育段階からプログラミング教育を導入する動きが見られるが，イギリスでは 2014 年より「Computing」がナショナルカリキュラムに位置づけられている（2014 年以前は「ICT」）．我が国に先んじてプログラミングに関する教育を行っているイギリスの実践は，たとえそこで育まれる「Computational Thinking」が，文部科学省（2016）の「プログラミング的思考」と厳密には異なるものの，有識者会議において提言された「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」が「コンピュテーショナル・シンキング」の考え方を踏まえつつ，プログラミングと論理的思考との関係を整理したことを考えると，これからの我が国のプログラミング教育の実践に示唆を与えてくれることだろう．

2. 問題の所在

The Royal Society（2012）の報告書『Shut Down or Restart?』によれば，教科「ICT」では，その「ICT」という用語（term）自体が，教科名・学習支援のための技術・MIDI など特定の技術・学校の管理情報システムをサポートする技術・学校の物理的インフラなど幅広い意味で用いられている問題点を指摘している．他にも，教員の専門性の不足や，米マイクロソフト社の Office の使い方などスキル学習に偏ることで子どもにとって退屈な科目になっている現状，それにより子どもがコンピュータサイエンスの魅力に触れることや興味を持つ機会を失なするなどの問題点が指摘されている．これらの問題点は，我が国で 2003 年度に新設された教科「情報」についての田辺（2015）の指摘を鑑みるとき，必ずしも対岸の火事ではないことが分かる．

そこで，『Shut Down or Restart?』では，そうした問題点を踏まえ，ICT という用語を，例えば，英語の読み書き（基本的なリテラシー），言語としての英語（どのように言語が動作するか），英文学（それがどのように使われるか）のように再検討し，

- Digital literacy
- Information Technology
- Computer Science

などの領域に分けて再定義し，それらを「Computing」とすることを提案している．

そうした提案を受け，教科「ICT」を廃止し，「Computing」を 2014 年 9 月より再必修化したイギリスでは，プログラミングに関する学習の割合が大幅に増えるなど子どもの学習の様々な面でプラスの効果が出る一方で，The Royal Society（2017）によれば，必修化したとはいえ，GCSE（全国統一試験制度）のテストで「Computing」が選択される割合は少なく，その理由として，「School didn't offer subject」（学校に GCSE の Computing 科目がなかったから）と答えた生徒が少なくないことが挙げられており，生徒がより高度なコンピュータサイエンスを学ぶ機会が損なわれているなどの問題が浮き彫りになっている．

こうした問題の要因として，The Royal Society（2017）は，知識とスキルを持った「Computing」教師の不足を挙げており，これらイギリスの教科「ICT」と「Computing」における初等・中等教育段階における実践とそれによって明らかになった問題点を踏まえると，我が国の「プログラミング教育」必修化に向けて，子どもにプログラミングの何をどのように学ばせるのかを整理し，さらにプログラミング教育の担い手をどのように育成していくかが解決すべき課題であることが分かる．

3. 研究の目的と方法

3.1 研究の目的

本研究の目的は，「プログラミング教育」必修化に向け，「子どもにプログラミングの何をどのように学ばせるのか」，そして「プログラミング教育の担い手をどのように育成するのか」を明らかにすることである．

3.2 研究の方法

本研究の目的を達成するために，次の方法でその解

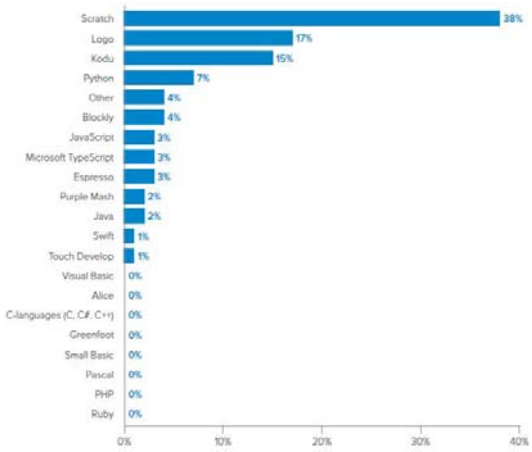
決を行う.

- コンピュータサイエンスへの接続を踏まえ，子どもが学ぶべきプログラミングにおける概念を精査し，その学び方を開発する.
- プログラミング教育の担い手の育成とプログラミング教育を担う現職教員の研修のための組織を立ち上げ，持続可能で効果的な運営方法を検討する.

4. コンピュータとの対話を通したプログラミング教育の授業デザイン

4.1 ビジュアル型言語をプログラミング教育に用いることの有効性とその限界

The Royal Society (2017) によれば，小学校では，図 1 のように Scratch が 38%と最も割合が高く，Logo (17%) やゲームデザインツールの Kodo (15%)，Python (7%) がそれに続く．このことから，全体としては Scratch のようなブロックベース (blocks-based) の言語（ビジュアル型言語）の割合が高いものの，小学校段階から Python のようなテキストベース (text-based) の言語（テキスト記述型言語）も用いられていることが分かる.



出所：The Royal Society「After the reboot」(2017.11)

図 1　小学校の Computing で使用されている言語

Bau ら (2017) によると，ビジュアル型言語におけるブロックベースのインターフェースでプログラミング言語を学習することで，後に従来のテキスト記述型言語の学習を改善できるのだという．例えば，9 年生で Scratch のコースを受講した生徒の方が，10 年生で C# や Java を早く学習できており，その理由として，ブロックが大きなカタマリ (chunk) として「ループ」などの構文の読み方を示すことによって，初心者の認知負荷を軽減していることなどが挙げられている.

出所：文部科学省「小学校プログラミング教育の手引（第一版）」(2018.3)

図 2　正三角形をかくプログラムの例

文部科学省 (2018) は，『小学校プログラミング教育の手引（第一版）』においてビジュアル型言語を用いた例を紹介しており（図 2），ビジュアル型言語である Scratch を用いたプログラミング授業も，森秀樹ら (2011) の実践のように既に小学校で行われている.

また，松澤と酒井 (2013) は，大学におけるプログラミング教育において，MIT で開発された OpenBlocks フレームワークを利用してビジュアル型言語とテキスト記述型言語を相互変換できるシステムを開発し，ブロックベースのプログラミングインターフェースで組み立てられた言語を Java に変換したり，逆に Java をブロックに変換したりする中で，文科系の学生がどのようにプログラミングを学ぶのかについて研究を行った．その結果，学生はビジュアル型言語の制御構文のわかりやすさを支持しており，ビジュアル型言語がテキスト記述型言語 (Java) 習得の足場かけ (Scaffolding) として機能していることが明らかになった．その一方で，自由記述欄には，ビジュアル型言語が「Java に慣れるまでは使いやすく慣れるとどんどん使いにくく」なることや「メソッドあたりから直接 Java で書いた方がわかりやすく」なるなどのコメントも見られた．これは，大学生についての例ではあるが，ビジュアル型言語がテキスト記述型言語の習得に有効である半面，その限界を示していると考えることもできる.

さらに，Bau ら (2017) も「ブロックだけではプログラミング言語を学習可能にするには不十分」であり，「さらに学習上の課題に直面」することを述べている．そこで，現状，テキスト記述型言語でしか行なえないことが多い点を踏まえると，いずれ直面するテキスト記述型言語の習得に向け，これからの社会を生きていく子どもたちにとって，ビジュアル型言語だけで学習を進めることには，やはり限界があるといえる.

4.2 コンピュータとの対話

Papert (1980) によれば，従来の教育は「“正しいか・間違っているか”に焦点を当てていたため，多くの子供たちは尻込みしてしまって」おり，さらに「問題は…正しいか間違っているかではなく，“そのバグを修正できるかどうか”に」あるという．つまり，プログラミング教育の効果は，子どもたちが間違いに怯えることなく知識とその習得の仕方を学ぶなど，「学習方法を大きく変革する可能性」にあるのではないだろうか.

その上で，文部科学省 (2018) の述べる「コンピュータに意図した処理を行わせるために必要な論理的思考力」を捉えるなら，そうした「論理的思考力」の育成は，何らかの間違いが含まれることを前提に，コーディングを介することでバグの存在をコンピュータに教えてもらい，またそのフィードバックをもとにバグを特定し，修正していくという試行錯誤を通して，子どもが獲得していくものであると考えることができる.

そうしたコンピュータとのインタラクティブなやり取りを通してこそ，子どもは「コンピュータは何ができるのか」を体験していき，結果として「コンピュータに意図した処理を行わせる」ための思考法が身についてくるのではないだろうか．さらに，その過程でコーディングの補助機能であるコンパイルを使用できる

ようになり，その結果として，コンピュータとのインタラクションが促進されるのではないだろうか.

こうした子どもとコンピュータとの一連のインタラクティブなプロセスを，本研究では，「コンピュータとの対話」であると考える.

4.3 「Play Programming ‼」による授業デザイン

文部科学省（2018）によれば，「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」のことである．この前提を支えているものは，本質的にコンピュータが計算機であるという事実であり，またそれが「思考」である以上，その獲得には，コンピュータとともに思考してみることが欠かせないプロセスになるのではないだろうか.

そこで，子どもたちが「コンピュータとの対話」を通してプログラミングを学ぶことができる教材として，「Play Programming ‼」を開発した．採用した言語は，Python（前半）と Processing（後半）である.

まず，「Play Programming ‼」では，「順次・反復・分岐」の概念を子どもが学ぶべき内容であると考えた．なぜなら，コンピュータとの対話において，最も重要なのはそれら３つの概念だからである．そうした「順次・反復・分岐」を学ぶプロセスにおいて，コンピュータにどのようにインプットするとどのようなアウトプットが返ってくるのかを確かめたり，あるアウトプットをするためにはどのようなインプットをしたらよいかを試したりすることを，子どもがコンピュータとインタラクションしつつ，これらの概念を構成できるよう指導する順序やワークシート等をデザインした.

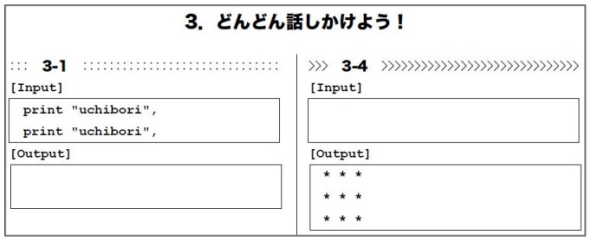


図3「Play Programming ‼」のワークシート

また，指導方法としては，コンピュータや教師･アシスタントの役割を表 1 のように設定し，教師は授業開始時に最低限の指示を与えるとともに，「間違えてもいいから徐々に正しさに近づいていくことが大切」という価値を伝えた上で授業を行うようデザインした.

表1 コンピュータと教師･アシスタントの役割

項目	役割
コンピュータ	その入力が「正しいかどうか」を子どもに教える.
教師・アシスタント	子どもが「コンピュータとの対話」できるようファシリテートする.

5. コンピュータとの対話を通したプログラミング教育の実践

5.1 COPERU PROJECT とその活動

コンピュータとの対話を通したプログラミング教育を実現していくために，本学の情報学部・教育学部・理工学部が連携し，プログラミング教育に関連する学部の教職員と学生，現役の学校教員によって構成されるプロジェクトチーム「プログラミング教育連携研究ユニット」（COPERU= COllaborative Programming Education Research Unit）を昨年発足させ，COPERU PROJECT の活動を行ってきた.

本ユニットの中核である「プログラミング教育ゼミ」には，情報学部・教育学部・理工学部の有志の学生が集まり，小・中学校におけるプログラミング教育の学習指導案の作成をし，小・中学校で行うプログラミング教育の授業にアシスタントとして参加している.

表2 プログラミング教育ゼミの構成人数

	2年生	3年生	4年生	院生	計
情報	9		1	1	11
教育	6	5	1		12
理工	1				1

学生の多くは，教職課程を選択しており，また，学生たちにプログラミング教育の指導技術の学習環境を提供し，小･中学校の校内研修にも参加させることで，自らがプログラミング教育の担い手になるとともに，プログラミング教育のコンテンツ作成や指導のノウハウを，校内研修を通して同僚に伝えていく指導者（メンター）にもなれるよう成長することが期待される.

5.2 地域の小・中学校との連携

COPERU PROJECT では，明星大学のある日野市をはじめ，八王子市や相模原市など近隣の地域と連携し，専門知識をもった大学教員と学生がゲストスピーカーとして学校に赴き，校内で「Play Programming ‼」を実施するとともに，その実践をたたき台に，大学や小・中学校の教員，学生がともにプログラミング教育について議論できる場を提供する中で，指導計画や効果的な教育，適切な教材，アイディアやベストプラクティスを共有することを試みている.

5.3 「Play Programming ‼」の実践

2017 年 9 月～2018 年 2 月に「Play Programming ‼」の実践を行った日野市内の公立小学校の教員にインタビューを行ない，子どもたちに次のような変化があったとの回答を得た．なお，この「Play Programming ‼」はクラブ活動の時間帯に行われ，小学校 4～6 年生 33 名が参加した.

- Scratch では味わえないような強い楽しみを覚えているように感じた.
- 「数字を変えてみたらどうなるのかな」と自主的に試すことが，教室のあちこちで行われていた.
- 一つの間違いが，逆に新しい発見や気づきにつながり，「ここを入れてみたらこんな風になったよ」

と伝え合っていた.
- 困っていたら隣の子が助けており，また誰かが発見したら「あ，そのやり方いいね」というようにお互いに情報交換しながら教え合っていた.
- テキストベースでも，子どもたちはそれほど驚きを感じておらず，むしろパズルに近い感覚で，「どうやったらテキストで打って表現できるだろう」と試行錯誤しながらコンピュータに接していた.
- 本当に自然にコンピュータと向き合い，コンピュータや近隣の子どもたちと会話ができていた.
- 教員が長いスパンで学んだことを，子どもたちは短いスパンで学んでおり，それが抵抗感なくできている.

図4「Play Programming ‼」における子どもの様子

また，キーボードの操作（大文字と小文字，半角と全角など）にも子どもたちはすぐに慣れ，英単語にもゲームのコマンドのように接している様が見られた.

今回の実践にアシスタントとして参加した学生へのインタビューによると，子どもたちが課題を解決できるようコミュニケーションを取る中で「子どもたちがどのように問題を解決したいのか把握して解決していくために，どのようにすれば良いのかを，一緒に考えるようになった」のだという．最初は，子どもたちに「プログラミングの仕方」を教え，自身が「知っている」正しい答えに導くことが自身の役割だと考えていたようである．しかしながら，「どのようにプログラミングを行っていくかは，子どもによって異なる」ことを知り，さらに，最初は課題を解決するための「考え方」を質問していた子どもたちが，「自身の考えをコンピュータに行わせるためにどのようにコンピュータと対話したらよいかを」を質問するようになってきたことで，学生自身も認識を変えるに至ったのである.

6. 研究のまとめと今後の課題

6.1 研究のまとめ

子どもにコンピュータとの対話を通して「順次・反復・分岐」を学ばせる「Play Programming ‼」を開発し実践を行った結果，子どもたちは，コーディングをあまり抵抗なく受け入れており，試行錯誤しながらコンピュータとともに思考し，それを子ども同士で共有する姿が見られた．また，COPERU PROJECT を発足させ，プログラミング教育の担い手となる現場の教員や学生とともに実践を行い振り返る中で，自身もプログラミング教育に関わるスキルを身につけると共に，「教員が知識とスキルを教える」から「子どもが，自身でできるようになるよう学ばせる」に認識が変化していった.

テキスト記述型言語を用いたコーディングによるプログラミングの導入は，小学校では敬遠されがちであるが，プログラミングの知の集合であるコンパイラを教育的に利用することで，ビジュアル型言語によるプログラミングとは別の効果を得ることができた.

コンパイラとは，人間が理解しやすい言語や数式でプログラムを記述可能な人工言語で書かれたプログラムを，コンピュータが直接的に実行できる機械語に変換するプログラムのことであり，作成したコードが変換可能か確認する機能を持つ．特に，正しく変換が行なえない場合に，どの部分のコードが変換できないかを詳細に表示する機能があり，この機能を教育に効果的に利用することができたのではないだろうか.

また，ビジュアル型言語によるプログラミング教育では，こうした部分を避けるように構成されているように感じるが，あえてこの部分を子どもたちに体験させることにより，コンピュータとの対話の方法を知り，そのインタラクションがコンピュータとの共同作業を強力に推進する武器として身につけていくことが，あらゆる活動においてコンピュータ等を活用することが求められるこれからの社会を生きていく子どもたちにとっては，必要となってくるのではないだろうか.

6.2 今後の課題

今後の課題は，まず，「Play Programming ‼」の内容を校種や学年間でより効果的に行えるよう精査・改善していくことや，それを学んだ子どもの経年変化を追跡していくことにある．また，小学校におけるテキスト記述型言語導入の効果のさらなる検証と，ビジュアル型言語の導入時期の検討も行っていく必要がある.

参考文献

(1) Bau, D. ら：“Learnable Programming: Blocks and Beyond”, Communications of the ACM, pp. 72-80 (2017).

(2) 松澤芳昭, 酒井三四郎：“ビジュアル型言語とテキスト記述型言語の併用によるプログラミング入門教育の試みと成果”, 研究報告コンピュータと教育 (CE), pp. 1-11 (2013).

(3) 文部科学省：“小学校プログラミング教育の手引（第一版）”, http://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1403162.htm (2018).

(4) 森秀樹, 杉澤学, 張海, 前迫孝憲：“Scratch を用いた小学校プログラミング授業の実践”, 日本教育工学会論文誌 34 (4), pp. 387-394 (2011).

(5) Papert, S. A.：“Mindstorms: Children, Computers, And Powerful Ideas”, Basic Books, Inc (1980).

(6) The Royal Society：“Shut Down or Restart?”, https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf (2012).

(7) The Royal Society：“Computing education”, https://royalsociety.org/topics-policy/projects/computing-education/ (2017).

(8) 田辺亮：“高等学校の教科「情報」の問題点”, 教育情報研究 31 巻 2 号 pp. 3-14, 日本教育情報学会 (2015).