情報処理学会論文誌　教育とコンピュータ　Vol.5　No.3　35–43 (Oct. 2019)

論　文

子供のプログラミング能力の獲得段階に関する定量的分析：小学校4〜6年生のScratchプログラミングを対象として

太田 剛[1,a)]　加藤 浩[2]　森本 容介[2]

受付日 2018年3月31日，再受付日 2019年1月26日，
採録日 2019年6月22日

概要：新学習指導要領で 2020 年度より小学校からプログラミング教育を実施することが明記されたが，プログラミング能力の獲得段階に関する情報は少ない．そこで本研究では，その獲得段階を明らかにするため，Scratch コミュニティにおいて小学校 4〜6 年生が作成した 900 本近い Scratch プログラムを対象に定量的な分析を行った．分析には，コンピュテーショナル・シンキング概念をもとにした評価基準を用いた．分析の結果，小学校 4 年生では，分岐・ループ・変数などの基本的なプログラミングができること，小学校 6 年生では，分岐・ループの入れ子や，カスタムブロックを使用できるようになることを示した．そして小学校 4・5 年生と 6 年生の間にプログラミング能力が向上することを示した．

キーワード：プログラミング教育，コンピュテーショナル・シンキング，学習分析

Quantitative Analysis for Acquisition of Children's Programming Skills: Scratch Programming of Grade 4–6

Go Ota[1,a)]　Hiroshi Kato[2]　Yosuke Morimoto[2]

Received: March 31, 2018, Revised: January 26, 2019,
Accepted: June 22, 2019

Abstract: Although the new Curriculum Guidelines of Japan announced that programming education is going to start in elementary schools in FY2020, there is few information on the stage of acquisition of children's programming skills. Therefore, this study analyzed nearly 900 Scratch programs created by children of grade 4–6 and made developmental stages of programming clear. Before analyzing, we developed new assessment criteria based on computational thinking concepts. The result shows that children of grade 4 can develop basic program with conditionals, loops and data and children of grade 6 can use nesting of conditionals/loops, and custom blocks. It also shows that children gradually develop their programming skills during grade 4, 5 and 6.

Keywords: programming education, computational thinking, learning analytics

1. はじめに

新学習指導要領 [1] で 2020 年度より小学校からプログラミング教育を実施することが明記された．文部科学省はプログラミング教育の在り方について，“発達の段階に即して，「プログラミング的思考」を育成すること” を示した [2]．また，総務省の有識者ヒアリングと文献調査をもとにした報告では，“評価データは十分には得られていないため，本格的な論理構成を必要とする教育は，9 才頃以降とし，今後，教育の実践と評価を通じて開始時期を調整する必要があると考えられる” と述べられている [3]．

他の教科では，PISA や TIMSS のように国際的に学力が比較できる基準が作成されていることに対して，小学校の

[1] 千葉県立生浜高等学校
Chiba Prefectural Oihama High School, Chiba 260–0823, Japan
[2] 放送大学教養学部
Faculty of Liberal Arts, The Open University of Japan, Chiba 261–8586, Japan
[a)] gohome@v006.vaio.ne.jp

図 1　Scratch プログラミング
Fig. 1　Scratch programming.

子供の思考力や認知能力などの発達段階に即したプログラミング能力の獲得段階（何年生で，どのようなプログラミングが可能かなど）に関する情報は少ない．そこで，自由にプログラミングする場面において作成された大量のプログラムを定量的に分析することにより，子供のプログラミング能力の獲得段階を明らかにすることを試みた．具体的にはブロック型ビジュアル言語である Scratch[*1] [4]（**図 1**）を使って，日本の小学校 4〜6 年生（以後小 4–6 と略す）が作ったプログラムを対象にして，プログラミング能力の獲得段階，および能力獲得に関する要因の 2 つの観点で分析を行った．

本稿では，2 章でビジュアル・プログラミングの定量的分析に関する先行研究を述べ，3 章で評価基準と分析方法を示す．4 章で分析結果を示し，5 章で考察を述べる．

2.　プログラムの定量的分析に関する先行研究

プログラムの分析に関しては，これまで主に高等教育でテキスト言語を対象に，コンパイルや実行時のログを収集・判断する [5]，プログラム内の構造やモジュール間の関係の正しさを判断する [6]，正答プログラムとの類似性を比較して判断する [7] など，特定の課題に対して正しくプログラムを作成できているかを分析する試みが行われてきた．

これに対して，Scratch などのブロック型ビジュアル言語を対象にして，多様なプログラムの内容を定量的に分析・評価する試みが始まっている．森らは小学生が作成した Scratch プログラムのブロック（テキスト形式の一般プログラミング言語の命令にあたる）種別と使用数を分析し，課題によるプログラムの違いを明らかにした [8]．薮田らは，Scratch を使用した課題で，小 5 に比べて小 6 が，分岐構造の使用数が大きく向上していることを示した [9]．

海外では，より体系的な評価基準をもとに大規模に分析を行う研究が進みつつある．Christopher らは，Scratch コミュニティ（Scratch の Web 開発環境に統合された，SNS と作品公開の場）に公開されているプログラムの中からアニメーションを分析対象とし，約 120 個のブロックを 17 のカテゴリーに分類した．一般的な予想に反して，使用ブロックの総数と使用ブロックの種類がプログラミングの経験（Scratch コミュニティで初めてプログラムを公開してからの期間）が進むほど減少することを示した [10]．

Wilson らは，プログラム概念・不適切なコーディング・デザインの 3 つのカテゴリーを定義し，特定ブロックの使用の有無でプログラムを採点した．そして，Scratch を使用したゲーム作成課題において Primary 4，5/6，6/7 の間でプログラムの得点に違いはないが，学年が上がるにつれて，独自のゲームを作成することを示した [11]．なお，基礎的なプログラミングの学習時間が不足していたことが，得点に違いがなかった原因であると述べている．

Dasgupta らは，Brennan らの提案する Scratch 用プログラミング評価フレームワーク [12]（**表 1**）をもとに，コンピュテーショナル・シンキング概念（Computational Thinking Concepts）（以後 CTC と略す）の分岐やループなどに対応するように Scratch のブロックを分類して分析を行った．その結果，Scratch の Web 開発環境においてリミックス（他人のプログラムをもとにプログラミングする）が，使用するブロックのレパートリーを増やすことに有効であることを示した [13]．また，Xie らは Scratch より高度なビジュアル言語である MIT の Inventor で開発された一連のプログラムが，個人の開発経験が進むと，どのように変化するかを分析した．彼らは Scratch 用プログラミング評価フレームワークをもとに，ブロックを CTC ブロック（分岐・ループ・データ関連）と非 CTC ブロックに分けた．そして，プログラムごとに使用したブロック種類数と，そのプログラムの中で新たに使用したブロック種類数を調べ，プログラムを作るごとに，この 2 つのブロック種類数が CTC ブロックと非 CTC ブロックともに増加していくことを示した [14]．

この分野の研究は始まったばかりであるが，以上のように，特定の学年/年齢のプログラミング能力の獲得状況とその要因に注目する研究が多く，複数の学年/年齢の獲得段階

表 1　Scratch 用プログラミング評価フレームワーク
Table 1　Scratch assessment framework.

Computational Thinking Concepts	Computational Thinking Practice	Computational Thinking Perspective
・sequence ・loops ・parallelism ・events ・conditionals ・operators ・data	・experimenting and iterating ・testing and debugging ・reusing and remixing ・abstracting and modularizing	・expressing ・connecting ・questioning

注）文献 [12] を表として要約

[*1] Scratch では，図 1 のように，一般のプログラミング言語のオブジェクトにあたるスプライト（ネコやネズミのキャラクター）を作り，その動作としてブロックを組み合わせたスクリプトを作り，そしてオブジェクトの集合としてプログラムを作成する．

に注目するものは少ない．著者らが，子供の Scratch プログラミング場面を観察した範囲では，たとえば，小 3–5 の初心者でも 1〜2 時間で if 文および if-else 文を使えるようになる．仮に，多くの子供が短時間の学習で，分岐・ループなどの基本的なブロックを使用できるようになるならば，これらのブロックの使用有無でプログラミング能力を弁別するのは難しいと考える．たとえば Wilson らの先行研究で学年間のプログラミング得点に差異が見られなかったのも，単純に個々のブロックごとの使用有無だけの判断に起因するかもしれない．

3. 評価基準と分析方法

3.1 分析対象

本研究では，Web 上の Scratch コミュニティで公開されている日本の小 4–6 の子供が作成した 871 本のプログラムを対象に分析を行った（**表 2**）．対象の子供の選択は以下のように行った．Scratch コミュニティでは，使用者プロフィールに学年や年齢の情報が記載されていないことが多い．そこで，自己紹介に小 4–6 と記載されている子供のうち，コミュニティ使用が 1 年未満で，かつ 10 個以上のプログラムを公開している子供を対象とした．抽出は 2017 年 4 月 1〜2 日に実施した．

なお，Scratch コミュニティ上のプログラムは，クリエイティブ・コモンズのライセンスに従う．本研究のような分析は，その規定の Transform として許可されると判断する．

3.2 CTC 評価項目の基準

分岐やループなどに対して，簡易な獲得レベルを設定することは，Wilson らによって試みられているが，筆者らは CTC の各能力を段階的に区別する新たな評価基準を使用した．この評価基準は，英国の教科コンピューティングの学習目標体系である Computing Progression Pathways [15] からプログラムに関連した内容を抽出したものと，CTC の考え方をベースにして作成した [16]．それは Scratch のプログラミング内容に対応して，8 分類 4 レベルの，計 32 個の評価項目（以後 CTC 評価項目と呼ぶ）から構成されている（**表 3**）．

CTC 評価項目は，プログラミング能力の獲得段階を明確に区別するための基準であり，各分類のレベル 1〜4 は論理的な前提関係もしくは難易度を示したものである．なお，このレベルの順番は Computing Progression Pathways と，Web 上の Scratch 入門プログラムや Scratch の入門書籍内のプログラムの学習内容の流れを参考に決定した．現時点では年齢とプログラミング能力の獲得との明確な関係が先行研究などで明らかにされていないため，レベルは分類内

表 2　分析対象のプログラム
Table 2　Data source.

	人数	総プログラム数	総自作数	総リミックス数	平均自作数	平均リミックス数	平均 Scratch コミュニティ参加期間
小 4	8	232	137	95	17.1	11.9	28.6 週
小 5	9	281	208	73	23.1	8.1	24.9 週
小 6	11	358	283	75	25.7	6.8	20.6 週
合計	28	871	628	243	22.4	8.7	24.3 週

表 3　CTC 評価項目
Table 3　CTC assessment criteria.

分類　／　レベル	1	2	3	4
分岐	if	if-else	論理演算子	if(-else)の入れ子
ループ	無限ループ	ループ回数指定	終了条件付きループ	ループの入れ子
モジュール	複数のスプライト*	カスタムブロック**	引数のあるカスタムブロック	クローン***
モデル/モジュール共有	1 スクリプト*で複数のカスタムブロックの利用	異なるスクリプトでカスタムブロックの利用	異なるスプライトで同一カスタムブロック利用	カスタムブロックの再帰的呼び出し
データ利用	変数利用	スプライトで変数共有	リスト型変数利用	クラウド変数利用
発動・トリガー	特定キーによる起動	マウス操作による起動	背景変化による起動	タイマー等による起動
連携・同期	背景変化を介した複数スクリプトでの連携	同一スプライトでのメッセージの利用	他のスプライトへのメッセージでの連携	メッセージ後の Wait での連携
ユーザインタフェース	文字入力の使用	キーの検出	マウスクリックの検出	マウスの座標位置の使用

*　Scratch ではスプライトが一つのオブジェクトであり，その中にプロシジャーとして複数のスクリプトを持つ．
**　一般のプログラミング言語のサブルーチンに対応．
***　自身のスプライトの複製を作る．
注)　文献[16]より引用．

における順序関係のみを想定したものである．

CTC 評価項目は，先に示したすべての先行研究のような単一ブロックの有無のチェックに加えて，入れ子などの複数のブロックの組合せも判断する特徴を持つ．そのため Wilson らの研究においては検出できなかった小学生の学年間の違いが検出できる可能性がある．

3.3　分析ツールと分析方法

著者らは，CTC 評価項目を含むプログラムの分析・評価機能を持ち，その結果を学習者に提示することによりリフレクションを促進することを目的とした Scratch 用学習支援システムを開発した [16]．本システムは，対話型で 1 つ 1 つプログラムを評価する方法と，Excel シートに記述されたプログラム情報から一括して評価する方法の，2 種類のインタフェースを持つ．本研究では，分析対象プログラムの Scratch コミュニティ内でのプログラム ID を手作業で抽出し，分析用 Excel ファイルに記載することにより，後者の機能を使って分析・評価処理を実施した．

CTC 評価項目以外に，プログラム内の有効スクリプト数/有効ブロック数（先頭にイベントブロックがなく，実行されないスクリプトを除外したもの）と，プログラムの属性（プログラムタイプと，自作かリミックスか）を分析に用いた．CTC 評価項目については，個人のプログラミング能力の獲得状況を示す指標として，個々のプログラムの CTC 評価項目の結果を合算（和集合）し，各項目を使用している場合に 1 点を加算して “個人 CTC 評価項目得点” と得点化した．なお，個人 CTC 評価項目得点は自作したプログラムを対象として算出した．このように本分析は自作プログラムを対象としたものである．ただし，リミックスの要因の分析においては，リミックスしたプログラムを対象として算出した得点も用いている．なお，コミュニティ上ではリミックスしたことがプログラム情報として明記される．

4.　分析結果

4.1　獲得段階に関する分析

(1)　CTC 評価項目に関する比較

各学年の個人 CTC 評価項目得点の平均を**図 2** に示す．小 4–6 で分散分析を行った結果，学年の効果は有意であった（$F(2,25) = 3.641$，$p < 0.05$）．そして，テューキー HSD を用いた多重比較によれば，小 6–小 4 と小 6–小 5 の間に有意傾向が見られた（小 5–小 4: $p = 0.95$，小 6–小 4: $p < 0.1$，小 6–小 5: $p < 0.1$）．**図 3** に学年別の個人 CTC 評価項目得点の分布を示す．小 6 は比較的高い得点をとる子供が多いが，小 4・5 でも同様に高い得点の子供がいる．

各 CTC 評価項目について，各学年の使用率を**図 4** に示す（全員その項目を使用していた場合は 100%）．小 4 でも全員が “if”，“ループ回数指定”，“変数利用” を使用していた．小 6 と小 4・5 を比較した場合，“論理演算子”，“ループの入れ子”，カスタムブロックに関する多くの各項目，“ユーザインタフェース” の各項目において，小 4・5 で使用している子供の割合が低いようである．

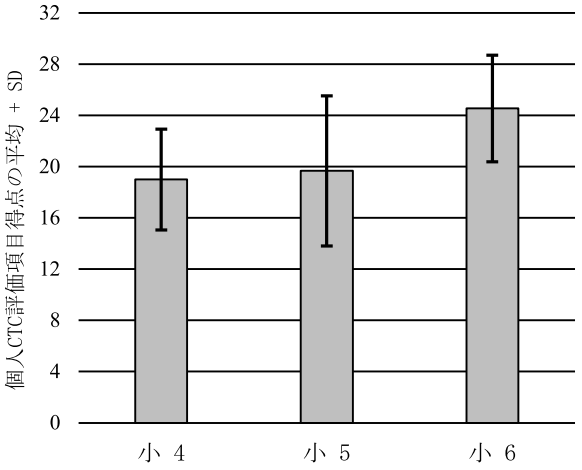


図 2　個人 CTC 評価項目得点の平均の比較
Fig. 2　Average of CTC assessment score with SD.

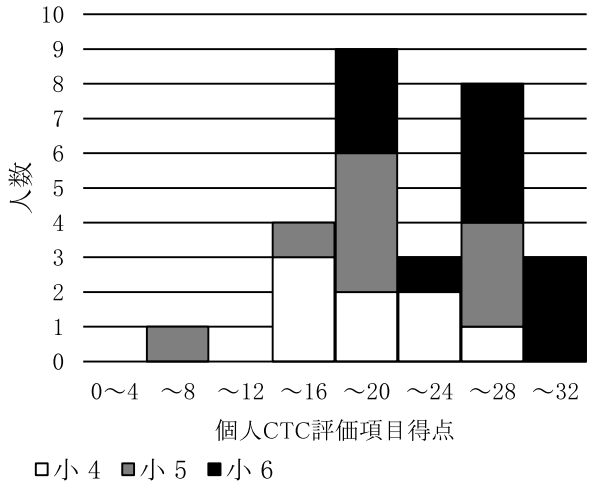


図 3　個人 CTC 評価項目得点の分布
Fig. 3　Histogram of CTC assessment score.

(2)　プログラムの大きさに関する比較

作文では，学年が上がるほど長い文章を書けるように，子供のプログラミングにおいても，学年が上がるほどより大きなプログラムが作成できるようになると考え，学年別のプログラムのブロック数を比較した．Dasgupta らの先行研究 [13] を参考に，ブロック数を対数変換した後，分散分析を行った．その結果，学年の効果は有意であった（$F(2,604) = 6.840$，$p < 0.01$）．そして，テューキー HSD を用いた多重比較によれば，小 6–小 4 と小 6–小 5 の間に有意傾向が見られた（小 5–小 4: $p = 0.95$，小 6–小 4: $p < 0.1$，小 6–小 5: $p < 0.01$）．

図 5 に示すように，各学年で千ブロック以上の大きなプログラムを作成できる子供がいる．また，プログラミング能力が高くなるとモジュール化を行い個々のモジュールが

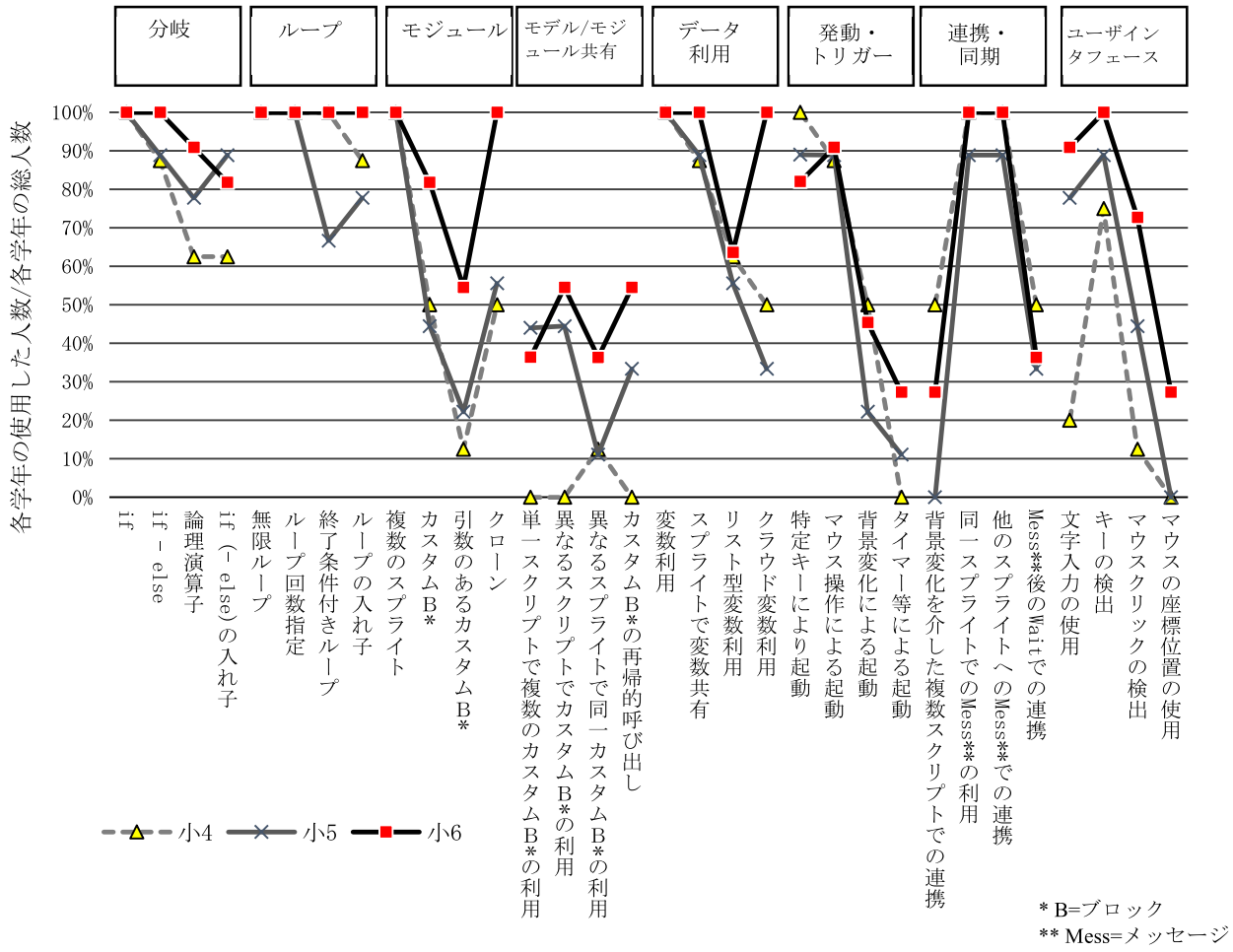


図 4　各 CTC 評価項目の使用率

Fig. 4　Percentage of number of children who use each of the CTC assessment criteria.

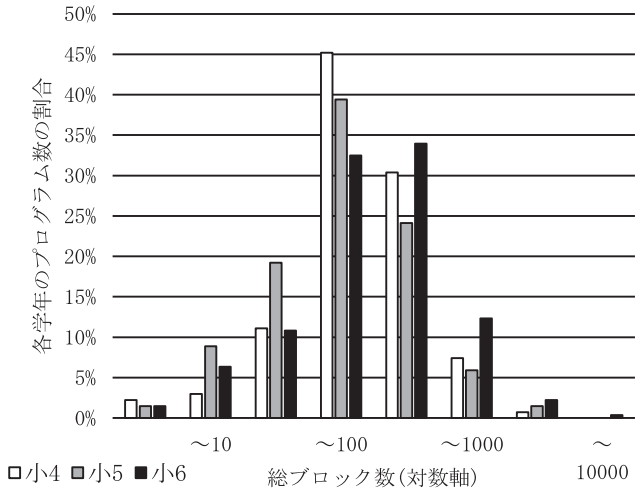


図 5　プログラムの総ブロック数

Fig. 5　Number of blocks in each program.

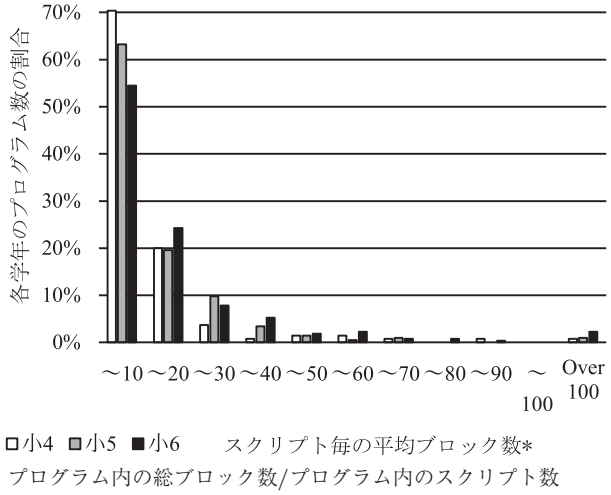


図 6　プログラムのスクリプトごとの平均ブロック数

Fig. 6　Average number of blocks per scripts in each program.

適切なサイズになることが予想されるため，プログラムの大きさだけでなく，プログラムごとのスクリプトあたりの平均ブロック数（図 **6**）を調べた．その結果，小 4–6 では 20 以下のプログラムが多いが，100 を超えるものもあった．

4.2　プログラミング能力獲得の要因に関する分析

Scratch コミュニティの小 4–6 のプログラミング能力獲得の要因についての分析結果を以下に示す．

(1)　**プログラミングの学習量**

Christopher らの研究と同様に，コミュニティの参加期間

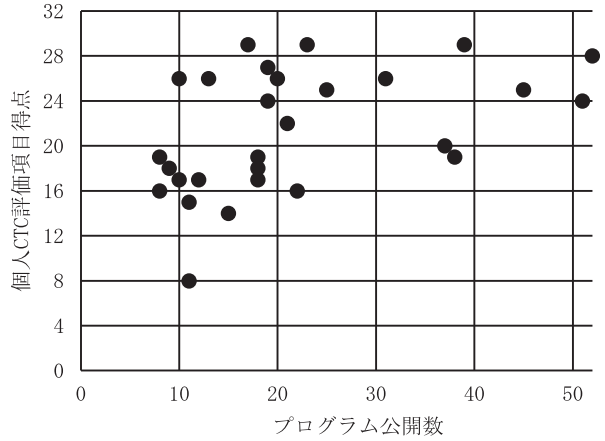


図 7　プログラム公開数と個人 CTC 評価項目得点の関係
Fig. 7　Scatter chart of CTC assessment score in number of shared programs.

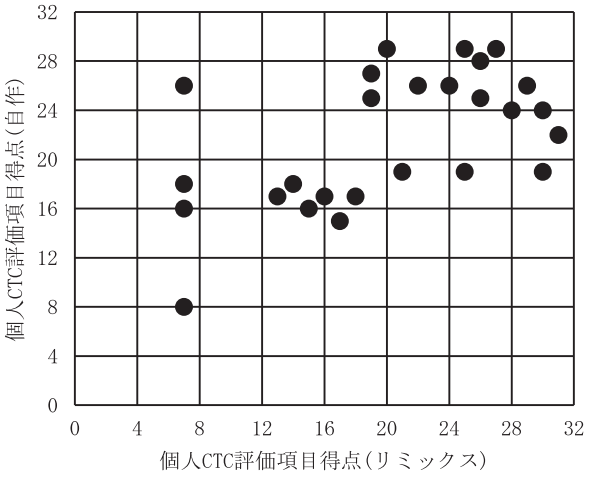


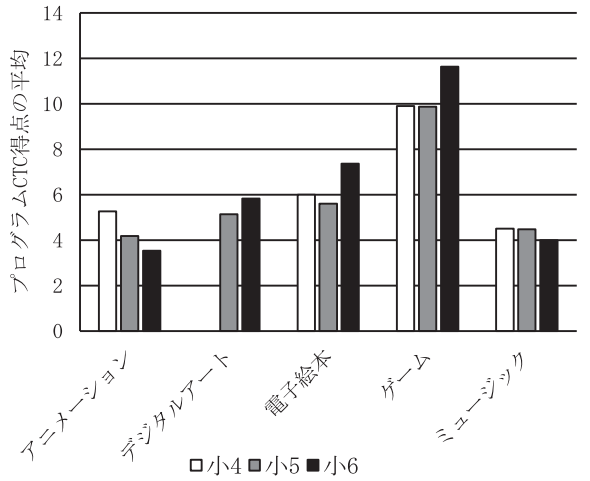
図 8　個人 CTC 評価項目得点の自作とリミックスの関係
Fig. 8　Scatter chart of CTC assessment score of orginal programs in remix programs.

とプログラミング能力の関係を分析した．その結果，参加期間と，公開プログラム数（$r = 0.02$）および個人 CTC 評価項目得点（$r = 0.07$）には，ほとんど相関が見られなかった．また，多くのプログラムを作るほどプログラミング能力は向上すると推測されるが，公開プログラム数と，個人 CTC 評価項目得点には，やや相関が見られた（$r = 0.46$）（**図 7**）．対象者の中の毎日プログラムを公開する子供では，参加期間が短くても多くのプログラムを作成していた．このように，学習量としては単なる期間ではなく，作成したプログラム数が適切な指標になることが示唆される．ただし，作成したプログラムがすべて公開されるのではないため，公開数が少なくても作成数は多い場合が考えられる．図 7 に示すように，多くのプログラムを公開していて個人 CTC 評価項目得点の低い子供はいなかったが，公開数が少なくても得点が高い子供はいるということは，この仮説と矛盾しない．

この結果から，前述した各学年の個人 CTC 評価項目得点の違いは，小 6 が多くのプログラムを作っていたなどの理由が考えられる．表 2 より，小 4 に比べて小 5・小 6 の方が多くのプログラムを公開しているように見えるが，個人のばらつきも大きい．分散分析を行った結果，学年の効果は有意でなかった（$F(2, 25) = 1.999$，$p = 0.503$）．また対象者のコミュニティ参加平均期間は，小 4 が 28.6 週，小 5 が 24.9 週，小 6 が 20.6 週で，大きな差はない．

(2)　リミックス

リミックスは他人のプログラムを手本にして学習する機会と考えられている．そこで，リミックスしたプログラムに多くの CTC 評価項目が含まれていれば，子供が自作したプログラムにも多くの CTC 評価項目が含まれるようになると考えられる．分析の結果，自作とリミックスの個人 CTC 評価項目得点には，やや相関が見られた（$r = 0.55$）（**図 8**）．なお，2 名の子供が自作しか公開していなかったため，本分析から除外した．



* 個人 CTC 評価項目得点と同様に，プログラムごとに各項目を使用している場合に，単純に 1 点として加算して得点化した．

図 9　タイプ別のプログラム CTC 評価項目得点の平均
Fig. 9　Average of program CTC assessment score by types.

(3)　プログラムのタイプ

Scratch の開発元である MIT メディアラボは Scratch のプログラムを，アニメーション，ゲーム，ミュージック，デジタルアート，電子絵本などのタイプに分類している．そこで，対象の全プログラムの内容を筆者が判断し，これら 5 つのタイプに分類し，タイプとプログラムで使用されている CTC 評価項目の関連を分析した．**図 9** に示すように，他のタイプに比べて，ゲームのプログラムは，より多くの CTC 評価項目を含むような複数の要素を必要とすると考えられる．なお，小 4 ではデジタルアートのプログラムはなかった．

5.　考察

本研究では，Scratch コミュニティ上の子供のプログラムを，主に CTC の観点から分析した．その結果，小 4 で

は基本的な分岐・ループ・変数などが使用できること，小6になると，入れ子やカスタムブロックなど，より高度なプログラミングが可能になることを示した．

ただし，本研究の対象とした子供はScratchコミュニティに参加していることからプログラムに興味があると考えられ，分析対象のプログラムも普通の子供のプログラムとは乖離していることも推測される．そこで，普通の子供を対象とする学校実践として使用された京陽小のプログラム[17]の内容をCTC評価項目で分析した[18]．その結果，小5まではif-else，ループ回数指定，他のスプライトへのメッセージの連携などを使用しており，小6ではループの入れ子とリスト型変数など，やや高度な項目を使用していた．また，カスタムブロックや“モデル/モジュール共有”は使用されていなかった．このような，京陽小の小5までと小6のプログラムの難易度設定は，本研究で得られた知見と類似したものであると考えられる．本研究は子供の数多くのプログラムを定量的に分析する，国内では初めての試みであり，普通の子供のプログラミング能力の獲得段階を知る手がかりとして有用であろう．

能力獲得の要因を考える場合，学習量の指標としては作成プログラム数が適切であることが示唆された．ただし，個人CTC評価項目得点と公開プログラム数に相関が見られたが，能力が高いから多くのプログラムを作るのか，またはプログラムを多く作るから能力が高くなるのかなどの因果関係は不明である．そして，プログラムのタイプと使用しているCTC評価項目の関係についての結果は，今後のプログラミング教育で，設定する学習目標に対応して，どのようなタイプの課題を児童に与えるか考慮する必要があることを示唆するものである．

6. まとめと今後の課題

本稿では，子供のプログラミング能力の獲得段階を明らかにするため，CTC評価項目を使用し，Scratchコミュニティ上のプログラムを分析することにより，小4–6の獲得状況の差異を示すとともに，子供のプログラミング能力獲得に関すると考えられる要因を示した．

ただし，本研究のように自由に作成されたプログラムを定量的に分析する試みは始まったばかりであり，いくつかの課題も明らかになった．

まず初めに，今回の結果をもとにCTC評価項目を再検討する必要があると考えられる．前述したように，CTC評価項目は論理的な前提関係もしくは難易度を想定したものであるが，個々の子供を見ると，レベルの低い内容より高い内容をプログラムで先に使用することも見受けられる．たとえば，一般的なオブジェクト指向言語において，オブジェクトの複製は，引数などを持つオブジェクトの生成・利用の後に学習するものと考え，Scratchのクローンの使用をレベル4に設定した．しかし，図4に示すように，多くの子供が早い時期にクローンを使用していることが分かった．レベルの逆転の理由の1つとして子供の使用する教材の内容が考えられる．たとえば，“リスト変数利用”では，リストに質問と答えを設定するというクイズ型プログラムが，初心者用の教材にも多く含まれている．この場合，子供はリスト変数の複雑な処理を理解していなくても，プログラム内で使用することになる．今回の分析においては，プログラムで各CTC評価項目を1回でも使用していれば，その項目を獲得したと判断した．手本を模倣して使用することと，理解して高度に使用することの区別をする必要があると考える．本分析では，小4–6のある程度の獲得段階の違いを示すことができたが，このように，CTC評価項目の各項目の獲得の判断定義や，各分類内のレベルで示した獲得順番などを見直していく必要がある．さらに，各CTC評価項目は分岐・ループ・データ利用などの多くのプログラミング言語で普遍的に使用される要素が中心であるが，“背景変化による起動”や“クラウド変数利用”などのScratchに特化した項目が混在している．Scratchは子供のプログラミング教育に特化した言語であり，同様にゲーム開発やスマートフォンのアプリケーション開発に特化したプログラミング言語も教育場面で利用されはじめている．そのため，プログラミング言語で普遍的・共通的に使用される要素と各プログラミング言語で特化した要素を明確に分離する評価体系を構築すれば，どのようなプログラミング言語を使用した場合でも普遍的な発達段階の把握が可能になると考えられる．

次に，個人CTC評価項目得点において自作とリミックスの間に相関があることを図8で示したが，子供がリミックス時に他人のプログラムから，何を学習したか明らかにする必要がある．たとえば，キャラクタの絵を少し変えただけでもリミックスとなるため，今後は，元になったプログラムをどのように変更したか，リミックスで使用されていたCTC評価項目が，その後の自作プログラムに使用されていたかなどの，リミックス自体の質を含めて多面的に確認する必要がある．

さらに，本稿では小4–6の限られた年齢の子供のプログラムを分析しただけであるが，今後，中学生や高校生も含めた幅広い年齢を対象とした分析を行うことによって，小学校から高校までの各年齢のCTC評価項目の獲得状況を定義することにより，系統的なプログラミング教育はどのようなものにすべきであるかの指針を示せるだろう．京陽小の事例[17]では，プログラミング教育の経験豊富な専門家が教員の指導を行ったため，学校現場の実践においても普通の子供が理解可能な課題を設定できたと考えられる．

筆者は，本研究で得られたような獲得段階をもとに，小学校のプログラミング教育向けの教材を試作している[19]．より発達段階が広範囲・詳細に明らかになれば，小学校から高校において，プログラミング経験の浅い教師でも各学

年に適したプログラミング課題の作成または選択が可能になると考える．ただし，新学習指導要領では小学校のプログラミングは特定の教科を設置することなく，各教科の中で実施することが指示されている．そして，そのプログラミング例としてあげられている三角形の描画においては，中学で学習する外角の概念が必要になり，ほかに座標での負の数や変数の利用など，小学校の学習指導要領外の概念を子供がプログラミングで使用している場面も見受けられる．このように今後のプログラミング教育の授業を考える場合は，単にプログラミングの獲得段階だけでなく，従来の教科の学習内容との整合性なども考慮する必要があるだろう．

謝辞　本研究は JSPS 科研費 17K18665 の助成を受けたものです．

参考文献

[1] 新学習指導要領（平成 29 年 3 月公示），入手先 ⟨http://www.mext.go.jp/a_menu/shotou/new-cs/1383986.htm⟩（参照 2018-03-06）．

[2] 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），入手先 ⟨http://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm⟩（参照 2018-03-06）．

[3] 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，入手先 ⟨http://www.soumu.go.jp/main_content/000361430.pdf⟩（参照 2018-03-06）．

[4] Scratch, 入手先 ⟨https://scratch.mit.edu/⟩（参照 2018-03-06）．

[5] 市村　哲，梶並知記，平野洋行：プログラミング演習授業における学習状況把握支援の試み，情報処理学会論文誌，Vol.54, No.12, pp.2518–2527 (2013).

[6] 式見　遼，若林智徳，松浦佐江子：WebStudy：ソフトウェア開発技能育成を目的としたプログラムの実行・評価機構を持つ Web 教科書の開発，電子情報通信学会論文誌，Vol.J96-D, No.10, pp.2464–2475 (2013).

[7] 渡辺博芳，荒井正之，武井恵雄：事例に基づく初等アセンブラプログラミング評価支援システム，情報処理学会論文誌，Vol.42, No.1, pp.99–109 (2001).

[8] 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～小学生を対象としたプログラミング教育の再考～，日本教育工学会論文誌，Vol.34, No.4, pp.387–394 (2011).

[9] 薮田挙美，山本朋弘：中学校技術科教員による小中連携でのプログラミング学習の展開，第 42 回全日本教育工学研究協議会全国大会論文集，pp.192–195 (2016).

[10] Christopher, S. and Christopher, C.: Skill progression demonstrated by users in the scratch animation environment, *International Journal of Human-Computer Interaction*, Vol.28, No.6, pp.383–398 (2012).

[11] Wilson, A., Hainey, T. and Connolly, T.: Evaluation of computer games developed by primary school children to gauge understanding of programming concepts, *Proc. 6th European Conference on Games-Based Learning* (2012).

[12] Brennan, K. and Resnick, M.: New Frameworks for Studying and Assessing the Development of Computational Thinking, *Annual Meeting of the American Educational Research Association* (2012).

[13] Dasgupta, S., Hale, W., Monroy-Hernandez, A. and Hill, B.M.: Remixing as a pathway to computational thinking, *19th ACM Conference on Computer-Supported Cooperative Work and Social Computing* (2016).

[14] Xie, B. and Abelson, H.: Skill progression in MIT app inventor, *Proc. IEEE Symposium on Visual Languages and Human-Centric Computing 2016*, pp.213–217 (2016).

[15] CAS Computing Progression Pathways KS1 (Y1) to KS3 (Y9) by topic, 入手先 ⟨http://community.computingatschool.org.uk/resources/1692⟩（参照 2018-03-06）．

[16] 太田　剛，加藤　浩，森本容介：コンピュテーショナル・シンキング概念に基づくプログラム自動評価機能を持つ Scratch 用学習支援システムの開発，教育システム情報学会誌，Vol.35, No.2, pp.204–214 (2018).

[17] プログラミング学習実践事例集（品川区立京陽小学校），入手先 ⟨http://school.cts.ne.jp/data/open/cnt/3/956/1/keiyoprograming.pdf⟩（参照 2018-03-06）．

[18] 太田　剛，森本容介，加藤　浩：プログラミング能力獲得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（SSS2017），pp.85–92 (2017).

[19] 太田　剛：系統的学習を考慮した小学校プログラミング教育用教材の試作，日本 STEM 教育学会　第 1 回年次大会 (2018), 入手先 ⟨https://www.j-stem.jp/wp/wp-content/uploads/2018/10/R03.pdf⟩（参照 2018-12-20）．

太田 剛　（正会員）

1981 年筑波大学第二学群人間学類（心理学専攻）卒．2018 年放送大学大学院修士課程（情報学プログラム）修了．千葉県立生浜高校非常勤講師．1981 年より国内外の教育システムとデジタル教材の開発，遠隔教育，情報教育，教育の情報化に従事．現在は主に，高校情報科教育と子供のプログラム教育を研究対象としている．教育システム情報学会，日本教育工学会等の各会員．

加藤 浩　（正会員）

1983 年慶應義塾大学大学院工学研究科修士課程修了．同年 NEC 入社．1999 年東京工業大学大学院社会理工学研究科博士課程修了．博士（工学）．2000 年より，メディア教育開発センター・助教授．2006 年同・教授．2009 年改組により放送大学・教授．2001～2017 年総合研究大学院大学文化科学研究科併任．現在教育工学，認知科学の研究開発に従事．日本教育工学論文賞を複数回受賞．日本教育工学会，日本科学教育学会，ヒューマンインターフェース学会，日本テスト学会，電子情報通信学会，アメリカ教育学会，ACM 各会員．

森本 容介　（正会員）

2005 年東京工業大学大学院社会理工学研究科博士課程修了．博士（工学）．メディア教育開発センター助手等を経て，現在，放送大学准教授．東京医科大学兼任准教授．情報処理学会教育学習支援情報システム研究運営委員会運営委員．電子情報通信学会，日本教育工学会，教育システム情報学会各会員．