# 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

太田 剛[†1]　森本 容介[†2]　加藤 浩[†2]

**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発達段階は明確ではなく，小中高校を通じた系統的な学習を実施することが難しいと予想される．本研究では，プログラミング能力の発達段階を明らかにするため，既存の教材や実践事例と，Scratch コミュニティの小学校 4-6 生年の作成した 900 本近いプログラムを対象に定量的な分析を行った．そのため，発達的な観点を持つコンピュテーショナル・シンキング概念とプログラムの機能による 2 書類の評価基準を新たに作成した．分析の結果，小学校 4・5・6 年生の間にプログラミング能力が向上するとともに，他人のプログラムを改造するリミックスやプログラムのタイプが能力獲得に関連があることを示した．そして，発達的な観点を持つ評価基準が有効であることを示した．

**キーワード**：プログラミング教育，学習分析，自動採点，教育学習支援環境，コンピュテーショナル・シンキング

# Quantitative Analysis for Developmental Stages and Factors of Programming Capability

GO OTA[†1]　YOSUKE MORIMOTO[†2]　HIROSHI KATO[†2]

## 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]．このプログラミングの発達段階に関する情報は少なく，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]．

このように学習対象の発達段階が明確になっていない現状では，小学校の学年毎に適した教材の準備や指導のための基準が不明確である．そして，小中高校でプログラミング教育の系統性が曖昧であるため，現在の情報教育のワープロや表計算のように各学校段階で同じ学習内容を繰り返すということも予想される．

本研究は，既存のプログラミング教育用の教材や実践事例，インターネットで一般公開されている子供の作成したプログラムを定量的に分析することで，プログラミング能力の発達段階を明らかにするとともに，その能力獲得の要因や分析手法に関する知見を得ることを目的とする．

## 2. プログラムの定量的分析の先行研究

これまで主に高等教育を対象に，コンパイルや実行時の結果を判断する，模範プログラムとの類似性の比較をするなど，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定量的に分析・評価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．

国内では，森らが子供のプログラミング教育で広く使用されている Scratch のプログラムを対象に，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)種別と使用数を測定し，課題によるプログラムの違いを明らかにした[4]．薮田・山本は，同様に Scratch を使用した課題で，小学校 5 年生に比べて 6 年生が，分岐構造の使用数が大きく向上していることを示した[5]．

海外では，より詳細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．Scaffidi・Chambers 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．さらに使用ブロックの総数を“深さ”，使用ブロックの種類数を“広がり”とする尺度で評価を行い，一般的な予想に反して，この深さと広がりがプログラミングの経験が進むほど減少することを示した[6]．

Wilson らはプログラム概念・コーディング書法・デザインの 3 つのカテゴリーと，その詳細として 22 の評価項目

表 1 Scratch 用プログラミング評価のフレームワーク

| コンピュテーショナル・シンキング概念 | コンピュテーショナル・シンキング実践 | コンピュテーショナル・シンキング見方 (Perspective) |
|---|---|---|
| ・順次<br>・ループ<br>・イベント<br>・並列処理<br>・分岐<br>・演算子<br>・データ | ・反復型開発<br>・テストとデバック<br>・再利用とリミックス | ・表現<br>・つながり<br>・質問すること |

注) 文献[8]を表として要約

†1 放送大学 大学院
Master's Program, The Open University of Japan
†2 放送大学
The Open University of Japan

で採点し，Scratch を使用したゲーム作成課題において Primary 4, 5/6, 6/7 年の間でプログラムの得点に違いが無いが，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(この理由として基礎的なプログラミングの学習時間が不足していたと指定している．)[7].

Dasgupta らは， Brennan・Resnick の提案する Scratch 用プログラミング評価フレームワーク[8](表 1参照)をもとに，コンピュテーショナル・シンキング概念(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い，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[9]. また，Xie・Abelson 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 ブロックと非 CTC ブロックに分け，使用総ブロック種類数を“深さ”，新たに使用したブロックの種類数を“広がり”とする尺度を用いて，プログラムを作るごとに，CTC ブロックと非 CTC ブロックがともに, 深さと広がりとして増加していくことを示した[10].

以上のように，この分野の研究は始まったばかりで，どのような評価基準が適切であるか，抽出されたデータをどのように分析・解釈していくかの手法を試行錯誤で検討している段階であると考えられる．筆者らが，子供の Scratch プログラミング場面を観察した範囲で，例えば，小 3-5 ぐらいの初心者でも 1〜2 時間で If 文および If-else 文を使っている．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, Wilson らの先行研究のように単純に基本的なブロックの使用有無でプログラミング能力を判断するだけでは，プログラミングを始めたばかりの子供を対象にしても，その能力の獲得状況を弁別できないのではないかと考える．

また，子供のプログラム作成場面を見ると，CTC を意識することなく, むしろプログラミングの市販書などにある，ブロックを組み合わせて作る機能（例えば得点を計算する，キャラクターをジャンプさせるなど）を覚えていくことによってプログラミングを学習しているようであり，実際の子供同士のプログラミングの教えあいでも，この機能単位の質問・説明がよく行われている．そのため，CTC 以外に子供達が意識する機能の観点からプログラムを分析する方法も有効と考える．

## 3. 分析方法

### 3.1 分析対象

多くの先行研究と同様に，小学校のプログラミング教育で広く使用されると考えられる Scratch のプログラムに関して，次の 2 種類を対象に分析を行った．

**(1) プログラミング教育の実践者の経験知の分析**

Scratch が開発されてから 10 年以上経ち，国内でも実践が進められている．プログラミング教育の実践者らは長年の指導経験から子供のプログラミング学習の発達段階について，各年齢の子供に適した教育内容などの経験知を持っていると想定できる．本研究では，これらの経験知が子供に提供される教材や課題の内容に反映されていると考え，現在国内で広く使用されている Scratch 用の教材や，実際の学校の実践事例内のプログラムを分析の対象とした(表 2 参照).

**(2) Scratch コミュニティ内の子供のプログラムの分析**

Scratch コミュニティで公開されている小学校 4〜6 年生のプログラムを対象に分析を行った．Scratch コミュニティでは，個人情報保護の観点から使用者プロフィールに学年や年齢の情報は記載されていなことが多い．ただし，比較的小学校 4〜6 年生と自己紹介している子供が多くいたため，その中でコミュニティ使用が 1 年未満で 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). 対象は表 3 に示すように 28 名であり，合計 871 本のプログラムの分析を行った．このように収集したプログラムのため，1)Scratch コミュニティに参加している時点でプログラムに興味があると考えられ，一般の子供とは乖離していることが推測される，2)公開しているプログラムなので，実際に作成したプログラム全体ではない，3)公開の順番が必ずしも作成された順番である保証は無い，4)学年は，子供自身が記入したプロフィールから特定した，の制約がある．さらに，3 学年分の被験者数が 28 名と十分でない．しかし，一般の子供が各自 20 本近いプログラムを作るという学習場面を設定すること自体が難しい．本研究は前述した制約はあるが，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ると考える．

### 3.2 評価基準と分析ツール

筆者らは，プログラムの 2 種類の自動評価機能を持つ Scratch 用学習支援システムを開発した[16]. このシステムは会話型で一つ一つプログラムを評価する以外に, Excel シートに記述されたプログラム情報から大量に一括して評価する機能も持つ．本研究では，このシステムを分析ツールとして使用した．以下にその 2 種類の評価基準を示す．

① CTC 評価項目の基準

本システムでは，子供のプログラミング能力の幅広い年齢の発達段階を意識した基準を作成するため，英国の教科コンピューティングの発達段階ごとの学習目標を示した Computing Progression Pathways(以後 Pathways と略す)[17]からプログラムに関連した内容を抽出し(表 5 参照)，それを参考に評価基準を作成した．具体的には，CTC の分岐・ループなどの 8 つの分類のそれぞれについて, 例えば“分岐”

なら，if 文， if-else 文，if 文の入れ子を区別するなど，4 つのレベルの項目(以後 CTC 評価項目とする)を設定した．また，先行研究で用いられている単一ブロックの判定に加えて，入れ子や変数共有などの CTC 評価項目は，複数のブロックの関係を判断する処理となった(表 4 参照).

② プログラム機能の基準

機能に関して，まず 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を分析して，初心者の子供が使用する 63 種類の機能を選び出した．そして，それらに対応したプログラムの特徴を，プログラム機能の検出基準とした(図 1 参照).

本システムを使用した評価結果を分析するにあたり，以上 2 つの基準を次のように数値化した．CTC 評価項目に関しては，プログラムで表 4 の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“プログラム CTC 得点”とした．プログラム機能に関しては，含まれている機能数を

表 2 分析対象の教材および実践事例

| 本稿略称 | タイトル | 種別 | 概要 | プログラム数 | 対象学年 |
|---|---|---|---|---|---|
| Why | Why! プログラミング[11] | テレビ番組 Web | 11 回のテレビ番組で，楽しみながら Scratch とプログラムの考え方を学習．視聴者がプログラムを投稿する Web サイトあり． | 10 | 小中学生 [技術 小 5〜6・中] |
| Koka | Koka プログラミング入門(子供の科学)[12] | 雑誌 Web | 最終的に Scratch でゲームを作ることを目標に，主にゲームを題材にしたプログラム． | 27 | 雑誌の中心読者層は小 5〜6 |
| Waku | 小学生からはじめるわくわくプログラミング 2[13] | 書籍 | Scratch の全くの初心者を対象にした，各教科に対応した内容のプログラム． | 7 | 小学生 初心者 |
| Dojo | Scratch(スクラッチ)でつくる! たのしむ! プログラミング道場 [14] | 書籍 | 世界的なプログラミング教室の CoderDojo の国内各地の主催者が実際に使用している初心者向けプログラム． | 9 | Scratch 初心者 |
| 京陽 | プログラミング学習実践事例集(品川区立京陽小学校)[15] | 実践報告書 | 京陽小学校での小 1〜6 の各学年での実践報告． | 6 | 小学生 |

表 3 分析対象の Scratch コミュニティの子供とプログラム

| | 人数 | 総プログラム数 | 総自作数 | 総リミックス数 | 平均プログラム数 | 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 | 29.0 | 17.1 | 11.9 | 28.6 週 |
| 小 5 | 9 | 281 | 208 | 73 | 31.2 | 23.1 | 8.1 | 24.9 週 |
| 小 6 | 11 | 358 | 283 | 75 | 32.5 | 25.7 | 6.8 | 20.6 週 |
| 全体 | 28 | 871 | 628 | 243 | 31.1 | 22.5 | 8.7 | 24.3 週 |

表 4 CTC 評価項目と教材および実践報告の分析結果

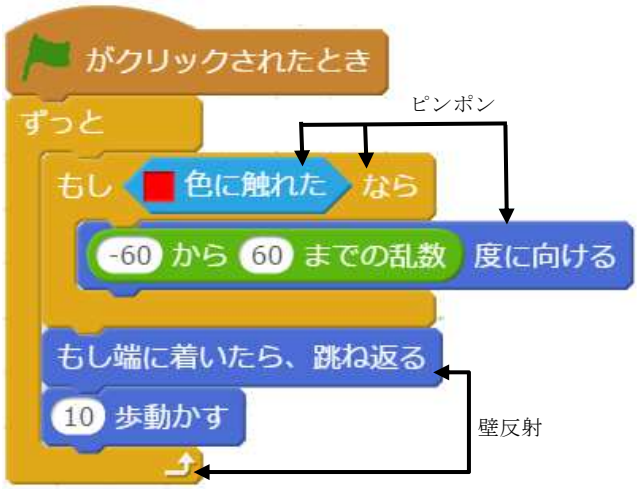
| 分類 | | CTC 評価項目 | Waku | Dojo | 京陽 | Koka | Why | 分類 | | CTC 評価項目 | Waku | Dojo | 京陽 | Koka | Why |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 分岐 | 1 | If | ○ | ○ | ○ | ○ | ○ | データ利用 | 1 | 変数利用 | ○ | ○ | ○ | ○ | ○ |
| | 2 | If - else | ○ | ○ | ○ | ○ | ○ | | 2 | 異なるスプライトで変数共有 | ○ | ○ | | ○ | ○ |
| | 3 | 論理演算子 | | | | ○ | ○ | | 3 | リスト型変数利用 | | ○ | ○ | ○ | |
| | 4 | If (- else)の入れ子 | | | | | ○ | | 4 | クラウド変数利用 | | | | | |
| ループ | 1 | 無限ループ | ○ | ○ | ○ | ○ | ○ | 発動・トリガー | 1 | 特定キーによる起動 | ○ | ○ | ○ | ○ | ○ |
| | 2 | ループ回数指定 | ○ | ○ | ○ | ○ | ○ | | 2 | マウス操作による起動 | | ○ | | ○ | ○ |
| | 3 | 終了条件付きループ | ○ | ○ | | ○ | ○ | | 3 | 背景変化による起動 | | | | ○ | ○ |
| | 4 | ループの入れ子 | ○ | | ○ | ○ | ○ | | 4 | タイマー等による起動 | | | | | |
| モジュール | 1 | 複数のスプライト* | ○ | ○ | ○ | ○ | ○ | 連携・同期 | 1 | 背景変化を介した連携 | | | | ○ | ○ |
| | 2 | カスタムブロック** | | | | ○ | ○ | | 2 | 同一スプライトでのメッセージの使用 | | ○ | ○ | ○ | ○ |
| | 3 | 引数のあるカスタムブロック | | | | ○ | | | 3 | 他のスプライトへのメッセージでの連携 | | ○ | ○ | ○ | ○ |
| | 4 | クローン*** | | ○ | | ○ | ○ | | 4 | メッセージ後の Wait での連携 | | ○ | | ○ | |
| モデル／モジュール共有 | 1 | 単一スクリプトで複数のカスタムブロック | | | | | | ユーザインタフェース | 1 | 文字入力の使用 | | ○ | ○ | ○ | |
| | 2 | 異なるスクリプトでカスタムブロックの利用 | | | | | ○ | | 2 | キーの検出 | | ○ | | | ○ |
| | 3 | 異なるスプライトで同一カスタムブロックの利用 | | | | | ○ | | 3 | マウスクリックの検出 | | | | ○ | |
| | 4 | カスタムブロックの再帰的呼び出し | | | | | | | 4 | マウスの座標位置の利用 | | | | | |

* 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.

表 5 英国の教科コンピューティングのプログラミング教育に関する学習目標と分析対象の教材の対応

| 学年 | Year 1-6 (入学年齢は 5 歳より) | | | | Year 7-9 | | Year 10-11 |
|---|---|---|---|---|---|---|---|
| 発達段階 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 分岐 | ループや if 文等の分岐を使った簡単なアルゴリズムを設計. | if-else 文を含む分岐の流れ, をプログラムの中で使用 | if 文と if-else 文の違いを理解し, それらを適切に使用 | | 入れ子(ネスト)になった分岐文を使用 | | |
| 演算子 | ステートメントの中で算術演算子を使用 | | 変数と比較演算子を, ループ終了判定を制御するために使用 | 論理型等の演算子と数式をプログラムの制御で利用 | (ビット)反転の演算子を理解し使用 | | |
| ループ | プログラムの中でループを使用 | until 等の後判定ループを使用 | | 繰り返しがループであることを理解 | | 前判定と後判定の違いを理解し利用 | While ループと For ループの違いを理解 |
| モジュール | | | 問題分割し, 個々の部分に対しての個別の解決方法を作成 | | 引数を持つ関数の必要性を認識し, 独自の関数を作成 | 引数の受け渡しについて理解して利用 | 再帰を利用した問題の解決 |
| データ利用 | | 変数の宣言と割り当て | | 適切なデータの型を選択 | 一次元配列構造を理解し利用 | 変数のスコープを確認 | 二次元配列構造を理解し利用 |

注) Waku の内容に対応　　Waku と Why の内容に対応




| 使用例 | 機能 | Scratch 内部プログラムの検出パターン |
|---|---|---|
| 壁反射 | 画面枠での反射 | `"['doUntil' \| 'doForever'\| 'doRepeat' …['bounceOffEdge'"` |
| ピンポン | 特定の色に触れた場合の反射 | `"['doIf'\|'doIfElse' ['touchingColor:' … ['heading: … ]"` |

図 1 プログラム機能例

“プログラム使用機能数”とした．また，子供の評価について, その子供が今までどの CTC 評価項目と機能をプログラム内で使ってきたか一括して把握できるように，個々のプログラムの結果を合算(和集合)して，“個人 CTC 得点”と“個人使用機能数”とした．なお，自作プログラムとリミックスを分けて集計した．

## 4. 教育実践者の経験知の分析結果と考察

表 4 に，各教材および実践事例のプログラムが CTC のどの項目を含んでいるかを示した．初心者向けの Waku と Dojo で，“分岐”は if-else, “ループ”は終了条件付きループ,“データ”は単純変数のスプライト(オブジェクト)での共有と，基本的な項目を使用している．ただし，Scratch では Pathways の Year 7 にある一次元配列はリスト型変数であり，初心者向けの“Dojo”では，これを使用している．これはクイズ形式のプログラムが入っているためで，Scratch では質問内容をあらかじめリスト型変数に手作業で格納する実装が一般的であり，リスト型変数に対する複雑な操作は含んでいない．これらに対して，初心者から小学校高学年を対象とした Koka と Why では，“分岐”での論理演算子の使用や入れ子なども含む．さらにモジュールも使用しており, スプライトを起動する“発動・トリガー”もマウス操作や背景変化など多様になっている．なお，ループの入れ子はプログラムで幾何学的な模様を描く場合に使用されることが多い．

これらの一般的な教材に対して，学校現場で実際に作成する京陽のプログラムに関しては，CTC として絞り込まれた内容を扱っているようである．これは阿部がこのカリキュラムのポイントとして“いずれも子供たち自身が自分なりにプログラムを書いて表現していることである”と述べているように[18]，あえて CTC として簡単なプログラムで子供達の創造性に重点を置くようにしているとも推測できる．但し，小 6 ではリスト型変数をループの入れ子で操作

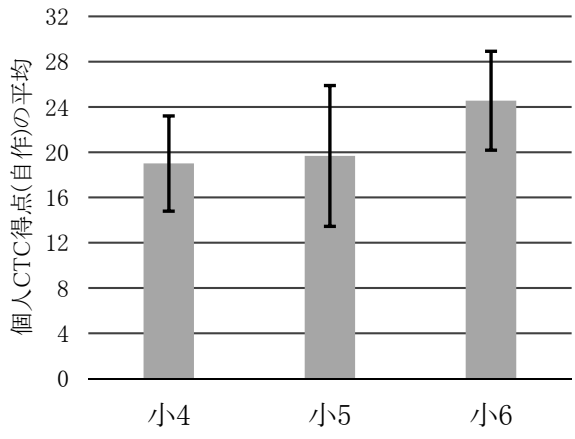



図 2 学年別の個人 CTC 得点(自作)の平均±SD

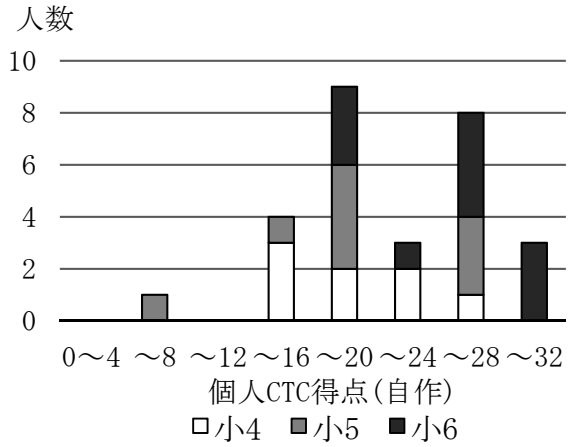



図 3 個人 CTC 得点(自作)の分布

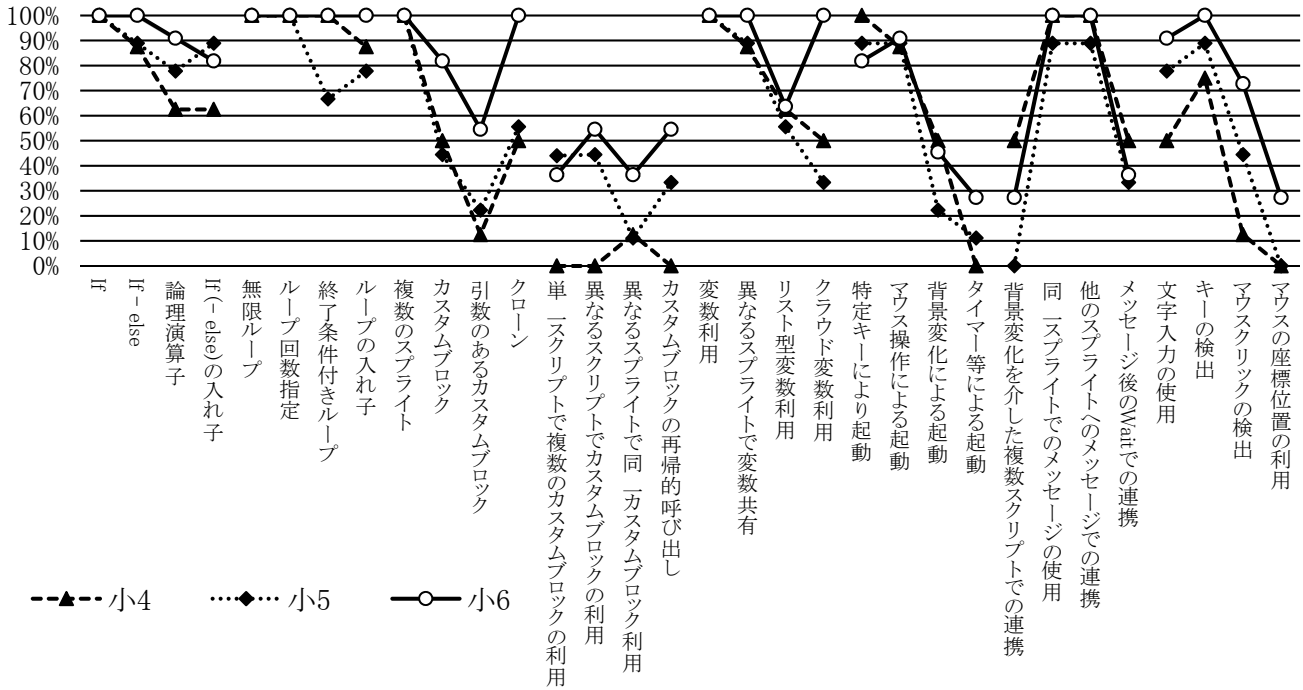



図 4 学年ごとのCTC評価項目の使用率(学年で使用した子供/学年の子供の人数)

し素数を求めるという，やや高度な課題であった．

英国の Pathways に，Waku と Why が含む CTC 評価項目をマッピングした(表 5 参照)．Pathways 自体が，その発達段階の根拠について明確にしていないため情報教育に関わる有識者の経験から作成したと考えられるが, Waku と Why とある程度の類似性がある．また引数のあるカスタムブロックは, Koka には含まれ, Why には含まれないが, Pathways では Year 7 に割り当てられることから，Pathways の表中の発達段階 5 及び 6 の学習目標が，国内の小学校高学年から中学校の発達段階に位置づけられることが推測される．ただし，再帰的呼び出しについては，Pathways では Year 10-11 の学習内容になっている．分析の結果，小 4 では全く使用せず，小 6 で半数が使用することがわかり，Pathways で定義したよりも低い学年でも使用可能であると推測される．

## 5. 子供のプログラムの分析結果と考察

### 5.1 発達段階に関する分析

#### (1) 学年間の CTC 評価項目に関する比較

学年別の個人 CTC 得点(自作)の平均を図 2 に示す. 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)．ただし，テューキーHSD を用いた多重比較によれば，各学年の間に有意差はなかった(小 5-小 4: P=0.959, 小 6-小 4: P=0.062, 小 6-小 5: P=0.095)．図 3 に学年別の個人 CTC 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．

さらに各 CTC 評価項目について, 各学年で使用している子供の割合を図 4 に示す(全員その項目を使用していた場合は 100%)．小 6 と小 4・5 を比較した場合，論理演算子，if の入れ子，終了条件付きのループ，ループの入れ子，カスタムブロックに関する各項目，“ユーザインタフェース”の各項目において，小 4・5 で使用している人の割合が低いようである．この点は，前述した，プログラミング教育の実践者の経験知において, 高度な分岐やループの使い方と，カスタムブロックに関する項目を高学年で学習するようにしていることと一致しているようにも考えられる．

**(2) 学年間のプログラム機能に関する比較**

学年別の個人使用機能数(自作)を図 5 に示す．学年が上がると増えるように見えるが，個人のばらつきも大きく，分散分析を行った結果，学年の効果は有意でなかった（$F(2,25)=1.999$ ,$p = 0.156$）．

また，個人 CTC 得点(自作)と個人使用機能数(自作)には，やや相関がみられた($r = 0.60$)．図 6 に示すように，CTC 得点が低く使用機能数が多い子供はいないが，CTC 得点が高く使用機能数が少ない子供はいる．前者については，今回使用した各プログラム機能自体に CTC 評価項目の要素を含むため, 多くの機能を使用した子供は同時に個人 CTC 得点も高くなると考えられる．後者については，例えばデジタルアートに多くみられるように，あまり機能を使わずに高度なプログラムを作っていることも推測される. さらに，同一機能でも様々な実現方法があり，現在の検出パターンでは検出できないことも十分考えられる．

**(3) 学年間のブロック数に関する比較**

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と，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．なお，ブロック数については，Dasgupta らの先行研究を参考に対数変換を行った. 図 7 に, ブロック数ごとのプログラム数の分布を示す. 各学年の差異について明確な違いは断定できないが，少なくとも小 5・6 では千数百ステップ以上の大きなプログラムの作成ができる子供がいる．

### 5.2 プログラミング能力獲得の要因に関する分析

**(1) コミュニティ参加期間と作成プログラム数**

コミュニティの参加期間が長いとプログラミング学習時間が長く，プログラミング能力が向上すると推定されるが，参加期間と，公開数(r=0.02)および個人 CTC 得点(自作)(r=0.07)にはほとんど相関がみられなかった. 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．分析の結果，公開数と個人 CTC 得点(自作)には，やや相関がみられた(r=0.46)．対象者の中に毎日プログラムを公開する子供がいたように，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と考えられる．ただし，図 8 に示すように，多くのプログラムを公開していて個人 CTC 得点(自作)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．3.1 節で述べたように，公開数よりも多くのプログラムを実際は作成している子供がいると考えられる．

**(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仮定した．分析の結果，自作とリミックスの個人 CTC 得点には，やや相関がみられた(r=0.55)(図 9 参照)．ただし，相関が仮にあったとしても，子供がリミックス時に他人のプログラムから，新しいプログラミング方法を習得したという因果関係を説明したわけではない．また，キャラクター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からどのように変更したか，リミックスの CTC 評価項目やプログラム機能が, 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．

**(3) プログラムのタイプ**

MIT 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，電子絵本などにタイプに分類している[19]. そこで, 筆者らは対象の全プログラムの内容を確認し，タイプの情報を付加して，タイプとプログラミング能力の関連を分析した．図 10 に示すように，他のタイプに比べて, ゲームはより多くの CTC 評価項目を含むプログラムが作成されていると考えられる．Scaffidi・Chambers の先行研究はアニメーションを対象としたが，これらのプログラムが高度なプログラミングを必要としないと仮定すると，学習が進んでも，あまりプログラムに変化がなかったと推測される．

## 6. まとめと今後の予定

本研究では，プログラミング能力の発達段階を明らかにするため，主として Scratch コミュニティ上の子供のプログラムを CTC の観点から分析し，小 4・5 では基本的な分岐・ループ・変数などが使用できることと, 小 6 になると，さらに入れ子やカスタムブロックなど，より高度なことが可能になることを示した．そして，これらの結果は教材や Pathways の教育実践者の経験知とも一致し，これらの経験知自体を検証したとも考えられる．ただし，ブロック数に

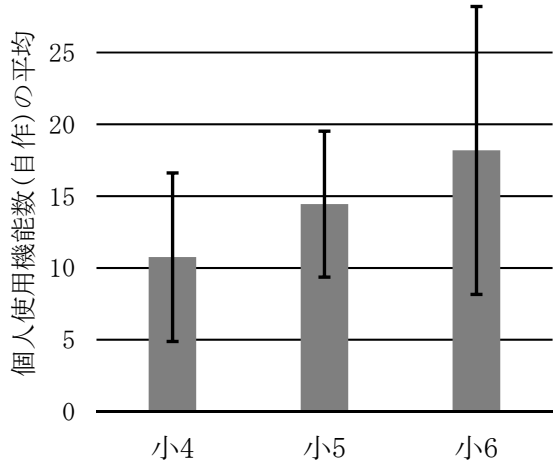



図 5 学年別の個人使用機能数(自作)の平均±SD

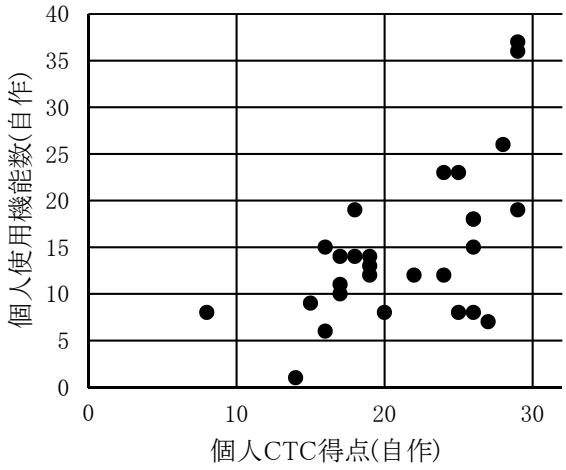



図 6 個人 CTC 得点(自作)と個人使用機能数(自作)の関係

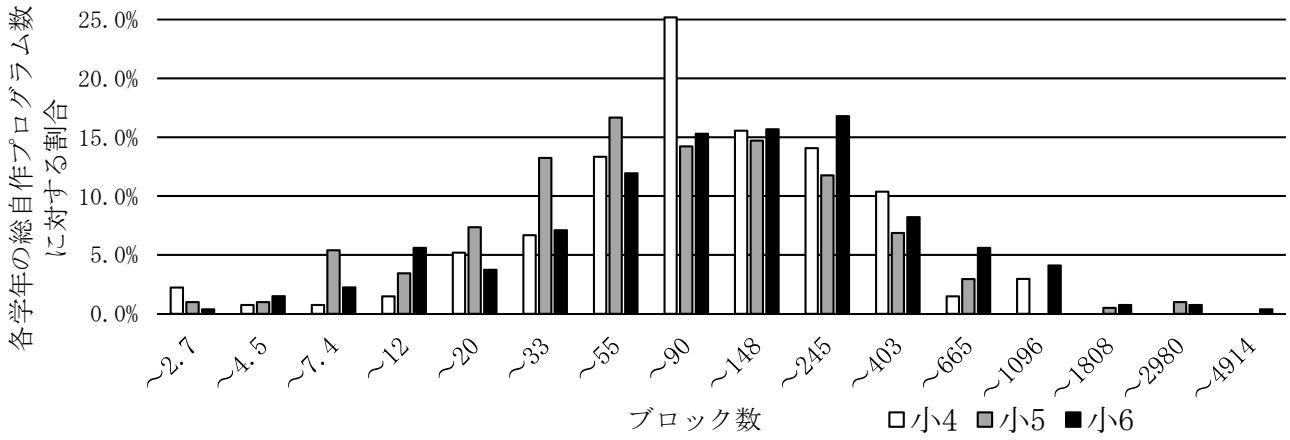



図 7 学年別の自作プログラムのブロック数の分布(各学年の総自作プログラム数に対する割合)

注) 横軸の数値は対数変換に対する元の整数/小数

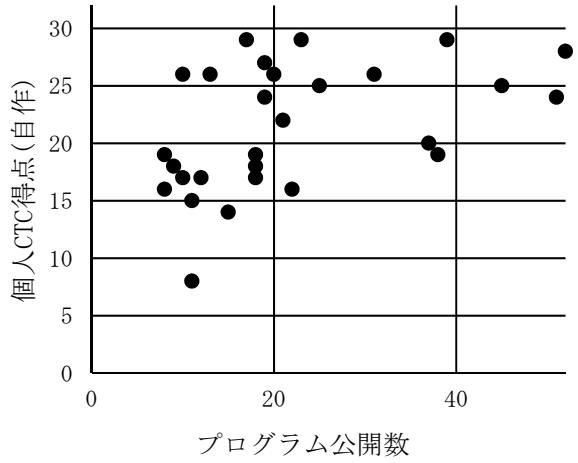



図 8 プログラム公開数と個人 CTC 得点(自作)の分布

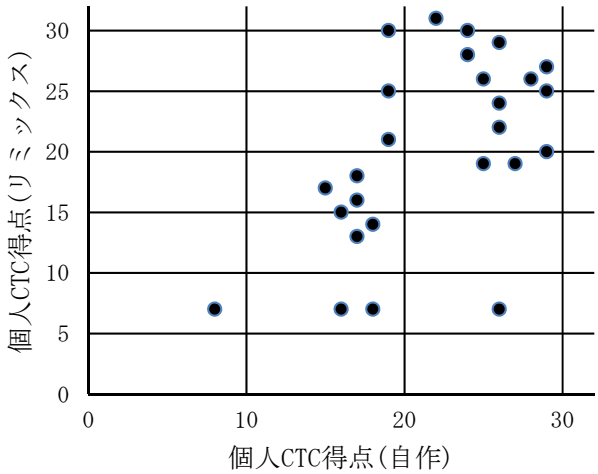



図 9 自作とリミックスの個人 CTC 得点の関係

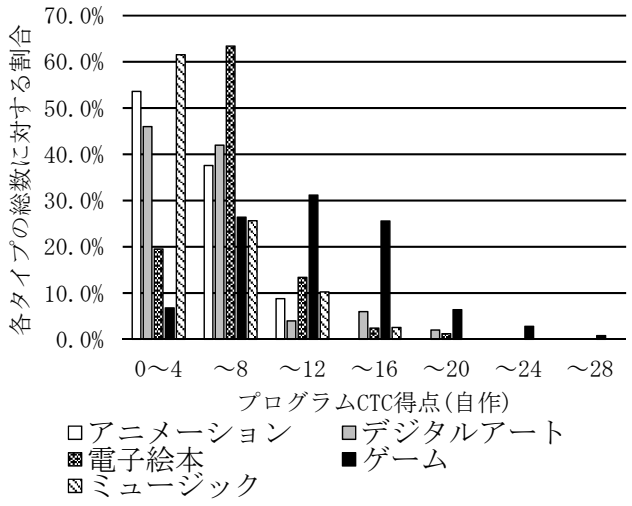



図 10 タイプ別のプログラム CTC 得点(自作)の分布

関しては，小 5・6 で大きなプログラムを作成する子供がいた．筆者らが，これらのプログラムの内容を確認した範囲では，単純にブロックを増やしただけでなく，カスタムブロックでプログラムを分割するなど，小 5・6 からプログラムの質の変化が見受けられた．しかし，標本数が少ないため，この知見は今後の検証課題とする．そのため今後，発達段階については，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必要がある．

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を示した．また，個人 CTC 得点において自作とリミックスの間に相関があることと，プログラムのタイプによって使用する CTC 評価項目数に違いがあることを示した．特にリミックスは，Scratch コミュニティの設計思想で重視されている．子供にとってプログラムを学ぶ手本と考えられるが，前述したように，リミックスで何を学習したかを分析する必要がある．

そして，分析手法として新しく作成した評価基準について，複数ブロックの関係を判断し発達段階を考慮した CTC 評価項目は，小 4-6 の違いを検出できたように有効であると考えられる．ただし，プログラム機能については，ある程度のプログラム能力の違いを示すことができたが，例えば，同一機能について複数の検出パターンを用意するなど，今後改善が必要である．

本研究では，Scratch コミュニティから抽出した一部のプログラムを分析しただけである．今後筆者らは，主催するプログラミング教室に参加する子供から，学習の都度プログラムを収集して分析する予定である．これにより，本研究のより詳細な分析が可能になるとともに，個人のプログラミングの学習過程とその要因，および年齢による明確な発達段階の有無について分析を行いたい．

## 参考文献

[1] “新学習指導要領”. 2016, http://www.mext.go.jp/a_menu/shotou/new-cs/1383986.htm, (参照 2017-5-19).
[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”. 2016, http://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm, (参照 2017-5-19).
[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”. 2016, http://www.soumu.go.jp/main_content/000361430.pdf, （参照日 2017-5-19).
[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. 2011, Vol.34 (4), p. 387-394.
[5] 藪田挙美，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. 2016, p. 192-195.
[6] Christopher, S. and Christopher, C.. Skill progression demonstrated by users in the scratch animation environment. International Journal of Human-Computer Interaction. 2012, Vol.28 (6), p. 383–398.
[7] Wilson, A., Hainey, T. and Connolly, T.. Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. Proceedings of the 6th European Conference on Games-Based Learning. 2012.
[8] 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.
[9] 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.
[10] Xie, B. and Abelson, H.. Skill progression in MIT app inventor. Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing 2016. 2016, p. 213-217.
[11] “Why! プログラミング”. http://www.nhk.or.jp/gijutsu/programming/, （参照日 2017-5-19).
[12] “Koka プログラミング入門”. http://www.kodomonokagaku.com/magazine/programming/,（参照日 2017-5-19).
[13] 倉本大資，阿部和広．小学生からはじめるわくわくプログラミング 2．日経 BP 社. 2016.
[14] 角田一平，とがぞの，高村みづき，若林健一. CoderDojo Japan 公式ブック Scratch(スクラッチ)でつくる! たのしむ! プログラミング道場．ソーテック社. 2017.
[15] “プログラミング学習実践事例集(品川区立京陽小学校)”. http://school.cts.ne.jp/data/open/cnt/3/956/1/keiyoprograming.pdf,（参照日 2017-5-19).
[16] 太田剛，森本容介，加藤浩．プログラム機能の自動分析機能とプログラム概念の自動評価機能を持つ Scratch 用プログラミング学習支援システムの開発．情報教育シンポジウム(SSS2016). 2016, p. 106-113
[17] “CAS Computing Progression Pathways KS1 (Y1) to KS3 (Y9) by topic”. http://community.computingatschool.org.uk/resources/1692,（参照日 2017-5-19)
[18] 阿部和広．子供の創造的活動とプログラミング学習．サイエンティフィック・システム研究会 教育環境分科会 2015 年度第 2 回会合 資料. 2015, p. 3-5.
[19] “Scratch Wiki, Project Types”. https://wiki.scratch.mit.edu/wiki/Project_Types,（参照日 2017-5-19)