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

太田 剛[†1]　森本 容介[†2]　加藤 浩[†2]

概要：2020 年度からの小学校でのプログラミング教育について，教育目標・内容や教師の指導力が問題となっている．筆者らは，これらの問題を解決し，特に，プログラミング経験の浅い教師でも，創造性の育成に有効なメイキングの学習方略で授業が実施できるように，諸外国の情報教育の目標になっているコンピュテーショナル・シンキングをもとにして Scratch 用のプログラミング学習支援システムの開発を行った．子どもがプログラム内でよく使用する機能をサンプルプログラム集として作成・整備した．そして，プログラムが小さな機能から構成されていることを気づかせるため，プログラムに含まれる機能の自動分析機能を作成した．さらに，プログラミングの習得度の発達的観点を持つ評価項目を作成し，子どもが作成したプログラムからその習得度を自動評価する機能も作成した．このシステムは，子どもが複数の機能から構成される複雑なプログラムを作成し，客観的に自分のプログラミング技術を認識することを支援する．さらに教師や指導者は人手でプログラムの解析をしなくても，機能の自動分析・習得度の自動評価の結果から子どもの学習の状態や進歩について把握し，適切な指導を行うことが可能になる．

キーワード：プログラミング教育，学習支援システム，自動評価，コンピュテーショナル・シンキング

Automatic Analysis System of Program Functions and Computational Thinking Concepts for Scratch

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

1. はじめに

世界的に，小学校からのプログラミング教育が本格化し始め，日本でも 2016 年 6 月に政府の産業競争力会議は 2020 年度からの初等中等教育でのプログラミング教育の必修化を示した[1]．文部科学省も 2015 年度にプログラミング教育の問題点を，①担当する教員の指導力，②学習に適した教材，③社会の変化に伴う学習の目標・内容と指摘し[2]，“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”（以後，有識者会議と略す）はプログラミング教育の在り方について，“「プログラミング的思考」などを育成”することを示し，“国や教育委員会による研修と各学校での校内研修が効果的に組み合わされ，教員の資質向上につなげていくことが求められる”や“担当教員の追加配置や専門人材の参画を含めた指導体制の充実”など指導体制の強化を提言した[3]．

このような状況に対して，筆者らは初等中等教育のプログラミング学習支援システムの要求事項の検討後[4]，プログラムの自動診断機能を持つことにより，プログラミング経験の浅い教師でも，創造性の育成に向いたメイキングの学習方略を使用した授業が実施できるプログラミング学習支援システム「コード忍者の里」の開発を行っていて，現在は個人用のシステムを Web 上に公開している[5]．

図 1 Scratch のプログラミング実行画面例

*図中の文字は，個々のスプライトのプログラムが持つ機能を示したもので，筆者が図に追記した．

本システムで対象とするプログラミング言語は Scratch[6][7]を選んだ．これは MIT のメディアラボが開発した子ども向けのビジュアル言語である．例えば図 1 は簡単なシューティングゲームであるが，スプライトと呼ばれるキャラクターを画面上に置き，それぞれのスプライトがどのように動作するかプログラムするものである．そして，Scratch のプログラムはスクリプトと呼ばれ，図 2（上側）のようにブロックを組み合わせて作成する．本システムの中核は以下に示す機能サンプルプログラム集と二種類の診断機能である．本システムは，スクリプトを読み込み，図 2（下側）の例では，プログラム概念（4.2 で詳細を説明）として，分岐はレベル 1，ループはレベル 1，2，4 のコー

†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

図 2 Scratch のスクリプトとシステムの診断結果例
*スクリプトはシューティングゲームのターゲットの例

ディングをしていることと，プログラムの機能としては，「壁反射」，「ポイント」，「サイコロ」のサンプルプログラムと同等の機能を使用していることを診断結果として表示する．本稿では，2 章で小学校のプログラミング教育における自動診断の必要性，3 章と 4 章で 2 種類の診断機能の背景と対象とする教育目標，5 章でシステムの概要と実現方法，6 章で想定利用方法を示す．

2. 自動診断機能の必要性

2.1 スモールステップとメイキング

筆者らは子どもを対象としたプログラミング教育を調査し，スモールステップ型の学習とメイキング型の学習があることを指摘した[4]．文部科学省のプログラミング教育実践ガイド[8]が“特に，教員は目標を細分化し，小さな目標を達成する体験を積み重ねながら最終目標に近づけるようスモールステップで課題を設定することで，児童生徒の『プログラミングは難しい』という思い込みを払拭させ，自分にもできるという自己効力感を高めさせているようです”と指摘したように，プログラミング教育には，スモールステップで確実に指導する方略がある．例えば Hour of Code 等のイベントで世界的にプログラミング教育を推進している Code.org は，段階的なパズルを解くことによりプログラミングを身につける教材を提供している[9]．ただし，阿部の“(Hour of Code の）パズル自体を自分たちで作りだせるようなメタな仕組みは提供されていない．そのため，解き終わった子供たちは短い時間で飽きてしまった．”という指摘もある[10]．

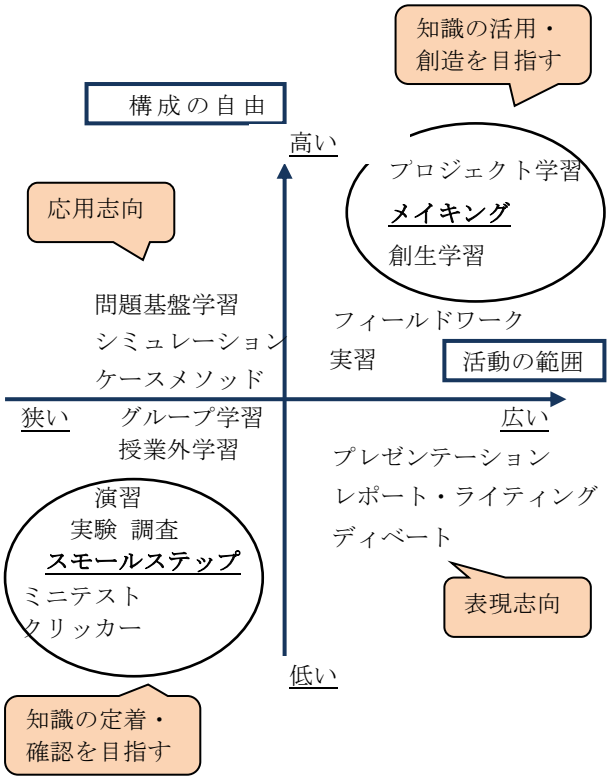

図 3 アクティブラーニングの多様な形態
*山地[14]に追加変更

これに対して，“単なる操作できる教具を与えるだけでなく，学習者が何か意味あるものを作り出す時に教育は最も効果がある”という Papert の構築主義[11]をもとにしたメイキングの考え方に沿ったプログラミング教育も，例えば，CoderDojo[12]に代表される実践が広がっている．メイキングについて Sylvia and Gary は，簡潔でゆるい課題設定において，コンピュータを一つの素材として，“いじくりまわし”，自分の好みで，考えて作っていく活動と考えている[13]．

プログラミング自体は，学習者が実際にプログラムを作成するという行動を伴うためアクティブラーニングの一つであると考えることができる．そこで，筆者らは，山地の「構成の自由」と「活動の範囲」を軸としたアクティブラーニングの分類[14]に，スモールステップとメイキングをマッピングした（図 3 参照）．スモールステップは知識の定着・確認をめざし，メイキングは知識の活用・創造をめざすのに適した学習方法であると考えられる．今後，2020 年度からのプログラミング教育のため，教材の整備が急速に進んでいくが，多くは先に示した文部科学省のスモールステップに対応したものになると筆者らは予想し，創造性の育成を重視しメイキング場面に対応したシステムを構築することを目指した．

2.2 メイキング時の指導上の考慮点

ただし，メイキングでは，兼宗らが“ブロックを適当に組み合わせて不規則に動く作品を子どもの創造性と誤認し

ているものも見受けられる”と過去の失敗を指摘しているように[15]，単に子どもが自由にプログラムを作成するだけではなく，正しくプログラミングを学んでいける仕組みが必要である．国内では，学童保育みどりっ子クラブ内の活動の一つとして，ビジュアル言語であるビスケットを使ったメイキングの学習事例が報告されている[16]．そこでは子どもに自由にプログラミングをさせているので,“習得度にばらつきがでる”や“基本的なテクニックを理解していない”というメイキング場面の問題が指摘されている．そのため，クラブ内で各習得レベルに対応したプログラミング方法のサンプル提供，プログラミング方法を「技」として技一覧表を作り提示，技の習得度合を判定する検定の実施等により子どものプログラミング能力の向上を実現している．メイキングは放任主義とも捉えられがちだが，みどりっ子クラブの対応は知識の定着・確認を保証する方法を示唆していて，本システムでもその考えを取り入れた．

2.3 自動診断によるメイキングの支援

スモールステップでは，学習内容がストーリーとして予め決められていることが多く，子どものプログラミングの確認も，ストーリーと合っているかの判断になり，授業の中でプログラミング経験の浅い教師でも実施可能である．また，Code.org のパズルも子どもの入力したプログラムが予め用意されている解答と一致するか自動採点する機能を含んでいるが，単純なマッチングで実現可能である．

これに対してメイキングでは，子どもが別々のプログラムを開発することになり，教師や指導者は，そのプログラムを解析して，個々の子どもの学習状況を把握し，個別に適切な指導をする必要がある．これには，時間と手間がかかり，またプログラムを読むためには十分なプログラミング知識が必要とされる．そこで，本システムでは，人手によるこれらの対応を自動化することを試みている．

プログラム自動評価については，今まで，主に高等教育を対象として，誤り分析や自動採点等の課題等に対して正しいプログラムを作成したかを判断するシステムが開発・利用されてきた．初等中等教育で使用される Scratch でもプログラムを自動評価する試みが始まり，正誤ではなく，どのようなプログラムを作っているか判断するシステムが開発されている．Wolz らは Scratch プログラムでどのようなブロックを使っているかを分析するツールを開発し，Scrape として Web で公開した[17]. その後, Boe らは Scratch のプラグインモジュールとして Hairball を開発し，ブロックの分析に加えて，変数が初期化されているか等のプログラム書法/作法について自動評価した[18].

これらのツールをもとにして Moreno-León らは Dr. Scratch[19]を開発し，Web に公開している．Dr. Scratch の特徴は，プログラム概念の個々のカテゴリー（例えば if 文や if-else 文の有無，複数のスプライトの使用やサブルーチンの使用等）の習得状態を得点化し，総合得点として学習者にフィードバックしていることである．そして，実際に数週間 Scratch を学習した生徒に対して Dr. Scratch を使用する 1 時間のワークショップを実施し，その評価を行った結果，Dr. Scratch の使用に生徒は肯定的な意見を持つことと，ワークショップの前後の比較でプログラム概念の総合得点が上がったことを示し，ツールがプログラミング習得に有効であることを示唆した[20].

3. 情報教育の目標とプログラム機能の自動分析

有識者会議で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力”と定義され,“いわゆる「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ながら提言された定義である”と示されているが[3]，その意味するところは今後検討が必要である．筆者らは，プログラミング教育の目標・内容をより具体化し，システム開発の方向性を明確化するため，このコンピュテーショナル・シンキングの意味を明確化し，その能力の獲得を支援するためにプログラム機能の自動分析機能をシステムに持たせた．

3.1 諸外国の情報/プログラミング教育の目標

諸外国では初等教育段階からのプログラミング教育が始まり[21][22]，特に英国とオーストラリアでは，プログラミング教育を含む新しい情報教科が小学校で必修となった．例えば，英国とオーストラリアのナショナルカリキュラムは，Wing のコンピュテーショナル・シンキングに関するエッセイ[23]に影響を受けていて[24][25][26]，英国の教科コンピューティングは,ナショナルカリキュラムの冒頭に「生徒たちがコンピュテーショナル・シンキングと創造性を用いて，世界を理解し変革していく」と明記している．

表 1 コンピュテーショナル・シンキングをもとにした各国カリキュラムのキー概念

Computing(UK)	Digital Technologies (Australia)	CSTA(USA)*
Abstraction	Abstraction	Abstraction
Decomposition		
Generalization		Modeling and Simulation
Algorithmic thinking	Specification, algorithms	Algorithms
		Problem solving
	Data collection, representation and interpretation	Data representation
Evaluation		

*米国にはナショナルカリキュラムが無いため，実質的な情報教育の標準カリキュラムである CSTA を比較で使用した[27].

そして，英国ではコンピュテーショナル・シンキングを“複雑かつ雑然で，一部しか見えない現実問題を，知能のないコンピュータだけで問題に取り組めるような指示に変換する，いくつかの知的能力の集合”と定義し[28]，いつくかの能力から構成されることを示している．また Wing のエッセイでは抽象化が最も重要とあると述べているが，多様な見方を示しているため，コンピュテーショナル・シンキング自体の明確な共通認識は無く，その定義について研究・論議が続けられている[29]. そして，表 1 のように，各国カリキュラムでもコンピュテーショナル・シンキングのキー概念の考え方や表現は異なっている．

3.2 本システムでのコンピュテーショナル・シンキングの考え方とプログラム機能の自動分析

筆者らはプログラミング的思考の具体的な能力は，コンピュテーショナル・シンキングの中にあると考え，本システムの目標として以下の具体的なキー概念を選んだ．

- 抽象化（Abstraction）：問題解決では，問題を解きやすくするため，問題や事象の適切な側面・性質だけを取り出し，他の部分を捨てること．
- モデル化（Modeling）：抽象化の結果であり，問題解決では問題や事象を簡潔に理論的に表現すること．
- デコンポジション（Decomposition）：大きなまたは複雑な問題や事象を理解または解決可能な細部に分割すること．
- アルゴリズム（Algorithms）：問題を解決するための明確な手順で，同様の問題に共通して利用できるものである．

有識者会議の示したプログラミング的思考は，“動きの組合せ”からアルゴリズムが中心とも考えられるが，“自分が意図する一連の活動を実現する”ためには抽象化・モデル化・デコンポジションも必要である．また，コンピュテーショナル・シンキングはプログラマー的思考法と訳されることもあるように，実際のコンピュータシステムの開発では，これらのキー概念はプログラマーにとって必要な能力であり，子どもがプログラムを作るときも，これらの能力がないと，複雑なプログラムを正しく作成することは困難である．そこで，本システムでは子どもが本格的なプログラムを作りたくなるような，または作れるような環境を構築することにより，これらの能力を獲得する機会を提供することを目指した．

例えば，図 1 の簡単なシューティングゲームでも，子どもが Scratch でプログラムを作る時，初めにゲームがどのような機能の組み合わせで構成されるかデコンポジションする必要がある．そして個々の機能を作るため，その本質を抽象化して，それをモデル化してプログラムの形で表現する．そこで，本システムは子どもがプログラムを作る場合に必要な個々の機能（例えば得点を計算する，当たりの判定をする等）を含んだ小さなプログラムをサンプル集として提供することで，プログラム作成作業を手助けすることにした．これにより，子どもにプログラムがいろいろな機能から構成されることを気づかせ，子どもは作りたいプログラムに必要な機能をサンプル集の中から探して組み込むことで，大きく複雑なプログラムも容易に作成できると考えられる．それに加えて，本システムに，プログラムがどのサンプルプログラムと同等の機能を使用しているか調べる自動分析機能を持たせた．これで，子どもは自分のプログラムに入っている機能を客観的に把握でき，プログラムに多くの機能を組み込む動機になることが期待される．さらに，Scratch コミュニティには多くの人がプログラムを公開しているので，子どもがお手本となるプログラムを見つけた時，それがどんな機能から作られているか知ることができる．また，教師や指導者にとっても，子ども達が作成したプログラムが機能面（質および量）からどのようなものであるか簡単に把握することができる．

また，前述したように，Scratch 用の自動評価システムはいくつか開発されているが，多くはプログラム概念について評価するもので，本システムのように，機能分析を行うことは新しい試みである．

4. プログラミング教育の学習目標と自動評価

4.1 発達的観点からのプログラミングの学習目標

前述したように各国の新しい情報教育はコンピュテーショナル・シンキングという大枠の目標を持っているが，狭義のプログラミング教育に注目すると分岐・ループ等のアルゴリズムやプログラム作成手順等のプログラミング自体の評価基準を明確化する研究・検討も始まっている．

Brennan らは Scratch のプログラミング学習場面をもとにした評価基準（表 参照）を提案している[30]（表 2 の順次，ループなどの“概念”に対応するものを，本稿ではプログラム概念と呼んでいる．なお Brennan の原著では Computational Thinking Concepts である）．また，Wilson らは，プログラム概念，プログラミング書法/作法（例えば，スプライト名を既定値から変更していない．トリガーが無く実行されないブロックがある等），ユーザビリティの評価基準を設定し，Scratch のプログラムの評価を行った．この評価において例えば，条件文について if の使用，if-else の使用等の発達的観点からレベル分けした[31].

表 2 プログラミング教育の評価フレームワーク

概念	実践	見方・考え方 (Perspective)
・順次 ・ループ ・イベント ・並列処理 ・分岐 ・演算子 ・データ	・反復型開発 ・テストとデバック ・再利用とリミックス	・表現 ・つながり ・質問すること

表 3 英国の教科コンピューティングをもとにした，発達段階を考慮したプログラム概念の学習目標

学年	Year 1-6				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-then-else 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，終了判断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		繰り返しがループのようなプロセスであることを理解		前判定と後判定のループの違いを理解し，利用	While ループと For ループの違いを理解
モジュール			問題を分割し，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
モデル			プロシージャーは，下位の解決方法の詳細を隠すために使用できることを理解	状況における類似性と差異を識別できて，それらを問題解決に利用	いくつかの問題が同様の特徴を共有し，同じアルゴリズムを使用することを認識	問題解決の一般化において，情報をどこで取り除くことができるか認識	サブルーチンとして，どこでも再利用可能なモジュールブログラムをデザイン，作成
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

本システムでも，先に示したプログラム機能の使用状況だけでは，子どものアルゴリズム等の習得度が明確に評価できないと考え，このプログラム概念による評価も取り入れた．まず，表 1 で示した国のプログラミング教育の学習目標を調査した結果，英国の教科コンピューティングで実質的に詳細の学習内容を定義している Computing Progression Pathways[32]にプログラミングの記述が多いので，これを抽出し，Brennan のプログラム概念で再整理し，発達的観点を持つ評価基準を作成した（表 3 参照）．

4.2 本システムのプログラム概念の考え方と自動評価

本システムの自動評価システムの基準の作成は，表 3 で示した評価基準を，Scratch のプログラム言語体系で評価できる内容に変換し，子どもに理解しやすいように 4 つの段階にした（表 4 参照）．なお，この基準の段階は，例えば 2 の分岐の“if-else”とデータ利用の“スプライトで変数共有”が，同じ難易度であることを示したものではなく，個々の要素の中の獲得される想定上の順位を示すものである．今後，本システムの機能を使い，国内において，子どもの発達段階またはプログラミングの学習期間に対応したプログラミング概念の習得状況を調査することにより，この基準が精査されていくと考える．

表 4 本システムのプログラム概念評価基準

要素	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	ブロック[サブルーチン]	引数のあるブロック	クローン
モデル/モジュール共有	1 つのスクリプトで複数のブロック	異なるスクリプトでブロックの利用	異なるスプライトで同一ブロック利用	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	特定キーにより起動	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使用	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利用

図 4 コード忍者の里 トップ画面

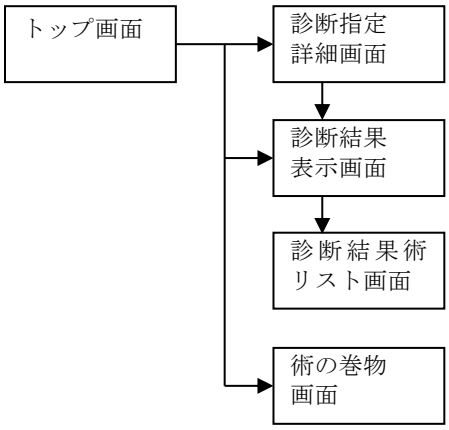

図 5 画面遷移図

5. システム概要

5.1 ゲーミィフィケーション要素の利用

これまで説明してきた内容をシステムの開発方針および機能とするシステムを開発した．図 4 に示すように，子どもが親しんで学習できるよう，システムの外装はゲーミィフィケーションの要素を取り入れ，“コード忍者の里”として Web サイトを構築し，教材である機能サンプルプログラムは“術”とした．そして，子どもは，術を組み合わせて好きなプログラムを作るという世界観を持つことになる．図 5 の画面遷移図が示すように，システムは教材である機能サンプルプログラム集の画面である“術の巻物”画面と自動診断関係の画面から構成される．

5.2 術の巻物: 機能サンプルプログラム集

サンプルプロクラムの準備は，Scratch サイトのチュートリアルプログラム，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，子どもが使用する機能を選び出し，単独で動作するような小さなプログラムを新規に約 60 種類作成した．これらのプログラムは Scratch コミュニティに共有可能なプログラムとして登録してあるため，子どもは術の巻物画面のリンクから内容を見ることができる．

図 6 診断結果術リスト画面

5.3 プログラム診断機能

図 2 に示すようにプログラム診断機能は，機能の自動分析結果を“プログラム内の術”で，プログラム概念の習得度の自動評価結果を“プログラマーの技”として表示する．また，他人のプログラムがどのような機能から構成されているか解りやすくするため，診断結果画面からプログラムに含まれる術だけを，図 6 のようにリスト化して表示することができる．これらの診断結果は e-learning で使用されるバッジシステムと同様に学習者の動機づけになると考えられる．ただしバッジシステムは，時には学習よりバッジの収集自体が目的化される恐れがある．実際に，みどりっ子クラブの報告において，習得度の判断試験にあたる検定の通過が目的化し，プログラミングに興味を失った子どもがいたことが示されたので，子どもの射幸心をあおらないため，前述した Dr. Scratch のように評価結果を得点化して表示することは、本システムでは行わなかった．

自動診断プログラムとして，子どもが独自に作った部分と他人のプログラムからコピーした部分をどう判断するかが大きな問題である．特に，Scratch コミュニティにはリミックスと呼ばれる，他の人が作ったプログラムをベースに改良を支援する機能があり，プログラム活動において他の人とつながることを推奨している．これにより，Scratch を使い始めた子どもでも，他のプログラムをもとすれば，結果として高度な機能を持つプログラムがすぐ出来上がってしまう．そこで，リミックスで作成されたプログラムなのか判断が必要であるが，プログラム自体からこの情報が取得できないため，この対応として，診断対象プログラムの指定時に，それが自分で作成したプログラムであるか指定するようになっている．

5.4 プログラム解析の処理方法

Scratch では使用するスプライトのグラフィック情報，スクリプト，使用する音源，開発画面でのスプライトやスクリプトの位置情報などをパックしてプロジェクトとして 1 ファイル化し，これらのうち制御情報については，JSON ファイルとして記録されている．またスプライトのブロッ

表 5 機能サンプルと検出パターン

サンプルタイトル	機能	Scratch プログラムの検出パターン
サイコロ	乱数の使用	"['randomFrom:to:'"
壁反射	画面枠での反射	"['doUntil' | 'doForever'| 'doRepeat' …['bounceOffEdge'"
ポイント	得点の計算	"['doIf'|' doIfElse', [' …['changeVar:by:', … , 1]"
迷路	方向キーによる左右の移動	"['whenKeyPressed', 'left arrow']… ['changeXposBy:', -" & "['whenKeyPressed', 'right arrow'] … ['changeXposBy:', "
ピンポン	特定の色に触れた場合の反射	"['doIf'|' doIfElse' ['touchingColor:' … ['heading: …]"

クは対応するコマンド情報に変換されている．本システムでは，JSON ファイルからスクリプトを取り出し，機能分析や自動評価において，予め決めたパターンにマッチングするかテキスト分析処理で行っている（表 5 参照）．例えば，サンプルプログラムの“ポイント”の場合，“doIf”または“doIfElse”コマンド内に，“'changeVar:by:', （任意の変数），1”があれば，この機能が使用されていると判断する．Scratch のプログラミングにおいて，意味もなくブロックを配置することは可能であるが，本システムでは，スクリプトが例えばマウスクリックをもとに起動される，メッセージの送受信関係が確立している，変数がスクリプト内で使用されている等のチェックを行い，プログラムとして、一切使用されないブロックは除外している．ただし，マウスクリックで起動されるスクリプトでも，そのクリック自体がプログラムとして意味を持つかなどの論理的な判断は行えない．

先行システムではプログラム書法/作法として，使用していないブロック等を情報提示している．本システムでも，これらの情報は処理上保有しているが，小中学生には，初めに形式にとらわれない自由なプログラミング活動を楽しんでもらいたいので，情報提示はしていない．

5.5 実装環境

今後は，学習履歴をもとにした指導情報の自動生成の機能拡張を計画しているため，機械学習系のライブラリーが充実しており，Scratch 関連のツールがあることから Python 3.4 で実装した．Scratch 自体はスタンドアロン版の Ver. 1.4 とスタンドアロン/Web 版の Ver. 2.0 の両方が使用されているが，今回は Web 上のコミュニティを利用することを想定しているので，Ver2.0 を対象とした．なお，Ver. 1.4 から 2.0 のコンバータは Python 2.x 系で利用可能であるため，今後，Ver. 1.4 のプログラムも診断対象とすることを検討する．

6. 学校授業場面での想定利用方法

有識者会議において“各教科等で育まれる思考力を基盤としながら基礎的な「プログラミング的思考」を身につけること”と，プログラミング教育を実施する新しい教科を設けず，既存の教科内で学習することが提言された[3]．これに対して筆者らは，例えば Scratch を使用する場合でも各教科に共通する基本的な操作やプログラミング能力を習得した後に，各教科で特有の使用方法をすることが望ましいと考える．そのため，基本的なプログラミング能力の学習のため，本システムをメイキング場面で使用することが有効であると考える．メイキングの考え方から自分の好きなものを作っていくことは，全く新しい授業の形態ではなく，すでに小学校の図画工作科で実践されてきて，図画工作科の学習指導要領の領域「A 表現」には“「表したいことを絵や立体，工作に表す」は，およそのテーマや目的をもとに作品をつくろうとすることから始まる．”とあり，図画工作科は，基本的なプログラミング能力を学習するのに適していると考える．また数時間の総合的な学習の時間でも，基礎的な学習が実際に検証されている[8].

なお，本システムで Scratch 初期操作の学習を支援することは考えていない．例えば，マインクラフトは非常に子ども達に人気があり，学校や塾で習ったわけでもないのに，子ども達自身が教えあいながら操作方法を習得している．このように子ども自身が興味のあるものは自ら習得する能力があり，Scratch はマインクラフト同様に子どもにとって魅力のあるものと考える．また，初期操作については，すでに多くの教材が開発されていて，必要であれば，それらを利用すればよい（例えば，NHK for School の“Why!?プログラミング”[33]等）．

また，Dr. Scratch では評価基準を Web で公開していて子どももその意味を見ることができる．これは習得すべきプログラム概念が何であるか理解しやすいが，安易にプログラム概念を含むプログラムを作成し得点を稼ぐこともできてしまう．このため，本システムでは Web には評価基準の解説は公開せず，生徒自身に個々のプログラム概念は何であるか考えさせることを想定している．また有識者会議において“順次，分岐，反復といったプログラムの構造を支える要素についても，（中略）小学校教育では（中略），知識として身につけることを指導のねらいとしたりする段階ではないと考えられる．”と示されたことから[3]，必要に応じて教師がそれぞれのプログラム概念は何であるか生徒に示すかどうか判断することになる．

7. 今後の予定

基本的な機能は個人用ツールとして公開済みであるが，一般の e-learning システムが持つ学習履歴機能が未サポー

トなので，診断結果を記録するため，紙ベースの 2 種類の生徒用ワークシートを提供している．現在，子どものプログラミング能力の学習状態の推移をより把握するため，履歴機能を付加したクラスルーム用システムを開発中である．

本システムでは，プログラム概念の評価基準を作成したが，まず獲得順番や獲得可能な年齢などの妥当性について検証する必要がある．そのため，子どもの作成した Scratch プログラムを収集して，その分析を行うことを計画中である．そして，クラスルーム用システムのリリース後，実際の教育場面でのシステムの有効性を検証するとともに，システム機能の見直しを行う予定である．

最終的には，プログラミング教育に関する詳細の学習履歴データを収集し，LAK（Learning Analytics and Knowledge）システムとして，適切な学習を進めるために教師・指導者および学習者向けのアドバイスを自動生成することを目指している．この機能を提供することにより，学校現場でプログラミング教育に経験の浅い教師でも，本当に良い授業を運営することが可能になることを考えている．

謝辞　本研究は JSPS 科研費 26282058 の助成を受けたものです．

参考文献

[1] “名目 GDP600 兆円に向けた成長戦略（次期「日本再興戦略」）【案】”．
http://www.kantei.go.jp/jp/singi/keizaisaisei/skkkaigi/dai26/siryou1.pdf, (参照 2016-5-11).
[2] “情報に関わる資質・能力についての参考資料”．
http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/061/siryo/__icsFiles/afieldfile/2016/02/01/1366444_2_2.pdf, (参照 2016-5-11).
[3] “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) ”．
http://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm, (参照 2016-7-7).
[4] 太田剛，森本容介，加藤浩． 初中等教育のプログラミング教育支援システムのデザインの検討. 2016，日本教育工学会 研究報告集, 16-1, p. 561-567.
[5] “コード忍者の里 for Scratch”．
http://tk2-249-34225.vs.sakura.ne.jp/ncv4s/, (参照 2016-7-10).
[6] “Scratch”. https://scratch.mit.edu/, (参照 2016-5-11).
[7] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, M., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y.. Scratch: programming for all. 2009, Commun. ACM, 52(11), p. 60–67.
[8] “プログラミング教育実践ガイド”．
http://jouhouka.mext.go.jp/school/programming_zirei/, (参照 2016-5-11).
[9] “Code.org”. https://code.org/, (参照 2016-5-11).
[10] 阿部和広．プログラミング入門をどうするか：3. 子供の創造的活動とプログラミング学習．情報処理. 2016, Vol.57(4), p. 349-353.
[11] Papert, S.. Constructionism: A New Opportunity for Elementary Science Education. MIT, Media Laboratory, Epistemology and Learning Group: National Science Foundation. 1986.
[12] “CoderDojo”. https://coderdojo.com/, (参照 2016-5-11).
[13] Sylvia, L. M. and Gary, S.. Making, Tinkering, and Engineering in the Classroom. Constructing Modern Knowledge Press, 2013. (邦訳 作ることで学ぶ．オイリー・ジャパン, 2015)
[14] 山地弘起．アクティブ･ラーニングとはなにか．大学教育と情報 JJUCE Journal. 2014, No.1, p. 2-7.
[15] 兼宗進，阿部和広，原田康徳．プログラミングが好きになる言語環境．情報処理学会誌. 2009, Vol.50, No.10, p. 986-995.
[16] 原田康徳，勝沼奈緒実，久野靖．公立小学校の課外活動における非専門家によるプログラミング教育．情報処理学会論文誌. 2014, Vol.55 (8), p. 1765-1777.
[17] Wolz, U., Hallberg, C. and Taylor, B.. Scrape: A tool for visualizing the code of scratch programs. Poster presented at the 42nd ACM Technical Symposium on Computer Science Education. 2011, p. 4-5.
[18] Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P. and Franklin, D.. Hairball: Lint-inspired static analysis of scratch projects. Proceedings of the 44th ACM Technical Symposium on Computer Science Education. 2013, p. 215-220.
[19] “Dr. Scratch”. http://drscratch.programamos.es/, (参照 2016-5-11).
[20] Moreno-León, J., Robles, G. and Román-González, M.. Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. M RED - Revista de Educación a Distancia. 2015.
[21] “Computing our future”．
http://fcl.eun.org/documents/10180/14689/Computing+our+future_final.pdf, (参照 2016-5-11).
[22] “諸外国におけるプログラミング教育に関する調査研究”．
http://jouhouka.mext.go.jp/school/pdf/programming_syogaikoku_houkokusyo.pdf, (参照 2016-5-11).
[23] Wing, J. M. Computational thinking. Commun. ACM. 2006, 49: p. 33–35.
[24] “National curriculum in England: computing programmes of study”．
https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study, (参照 2016-07-10).
[25] “Computational Thinking A guide for teachers”．
http://community.computingatschool.org.uk/files/6695/original.pdf, (参照 2016-5-11).
[26] “The Australian Curriculum: Digital Technologies (F‐10) ”．
http://www.australiancurriculum.edu.au/technologies/digital-technologies/structure, (参照 2016-5-11).
[27] “CSTA K‐12 Computer Science Standards”．
https://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf, (参照 2016-5-11).
[28] “The Chartered Institute for IT (2014) Call for evidence - UK Digital Skills Taskforce”．
http://policy.bcs.org/sites/policy.bcs.org/files/BCS%20response%20to%20UKDST%20call%20for%20evidence%20final.pdf, (参照 2016-7-7).
[29] HU, C.. Computational thinking: what it might mean and what we might do about it. Proceedings of ITiCSE 2011. 2011, p. 223–227.
[30] 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.
[31] 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.
[32] “CAS Computing Progression Pathways KS1 (Y1) to KS3 (Y9) by topic”． http://community.computingatschool.org.uk/resources/1692, (参照 2016-07-12)
[33] “NHK for School Why!?プログラミング”．
http://www.nhk.or.jp/gijutsu/programming/, (参照 2016-7-10).