ビジュアルプログラミングにおけるリファクタリングを目的とした支援ツールの構築

米田 浩崇
Hirotaka YONEDA

1　はじめに

ソフトウェアはあらゆる製品に組み込まれており，その重要性は増す一方である．世界的に第四次産業革命が進み，高品質なソフトウェアが求められる中で，優秀なソフトウェア開発技術者の養成は重要な要素の一つである．ソフトウェア開発技術の中でも，その実装技術であるプログラミングに関する教育は，教育機関・民間団体を問わず広く実施されている．

プログラミング初学者に対する教育では，学習者への導入の敷居の低さから，ビジュアルプログラミング（VP）言語が用いられることが多い．VP 言語とは，いくつかの意味のある視覚的なオブジェクトを組み合わせることでプログラミングを行う言語である．森ら（2011）は，VP 言語「Scratch」を用いた授業を構成・展開することで，小学校段階で効果的なプログラミング教育が可能であると報告している [1]．しかし，VP 言語を用いる多くの教育現場では，学習者の興味を惹きつけるまでに留まり，コードの可読性や保守性を重視したプログラミングテクニックを学ぶ機会が少ないのが現状である．一般的なソフトウェア開発における現場では，開発ツールがリファクタリング機能を提供し，コードの可読性や保守性を向上できる．VP ではコードがテキストではなくグラフィックスで表されるため，リファクタリングなどの機能の実装が容易ではなく，同機能を提供するツールは存在しない．

優秀なソフトウェア開発技術者を養成するためには，可読性が高く修正が容易である高品質なコードを記述する能力を養う必要がある．高品質なコードを書く利点として，アルゴリズムやデータ構造の理解が深まることや，作業効率が向上することがあげられ，学習者にとって利益となりうる．そこで本研究では，VP におけるリファクタリング手法を提案し，リファクタリング支援ツールの構築およびその有効性の検証を行う．

2　VP におけるリファクタリング

2.1　Scratch

本研究では，VP 言語の一例として Scratch を対象とする．Scratch は，MIT メディアラボが開発したプログラミング言語学習環境である．2019 年 4 月時点でのユーザー数は 3900 万人であり，日本でも 39 万人のユーザーが存在する．Scratch によるプログラミングの一例を Fig. 1 に示す．初学者が詳細な構文の記法を覚えることなく視覚的にプログラミングが行え，動作結果がインタラクティブなアニメーションにより得られるという特徴がある．無料で利用でき，かつ環境導入も容易であるため教育機関やワークショップで広く用いられており，文部科学省も Scratch を用いた教育カリキュラムを作成している [2]．

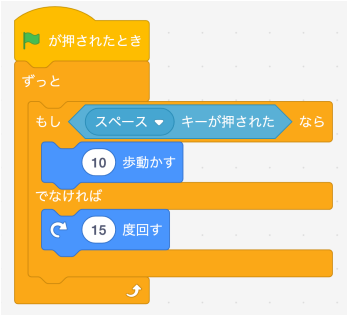

Fig.1　Scratch を用いたプログラミングの例

2.2　リファクタリング

リファクタリングとは，ソフトウェアが抱える設計上の問題を，ソフトウェアの外部的な振る舞いを変更することなく取り除くことをいう．主にコードの可読性や保守性の向上を目的として行われる．リファクタリングの結果，機能追加やバグの修正が容易になり，生産性が向上するといわれている．

あるコードにおいてリファクタリングが必要か否かの判断基準の一つに「コードの臭い」と呼ばれる概念がある．コードの臭いとは，プログラミングにおいてソースコードに問題が存在することを示す兆候のことをいう．コードの臭いの例として下記のものがあげられる．

- Long Method
- Duplication
- Large Class
- Data Clumps

Long Method は，一つのメソッドが長い状態であり，可読性が低くなる．また Duplication は，同様の動作が複数箇所に存在する状態であり，保守性が低くなる．このようなコードの臭いのうち，いずれかあるいは複数が該当するコードは，改善が必要であると判断できる．

2.3　Scratch におけるリファクタリングの有効性

VP ではコードがテキストではなくグラフィックスで表示されるため，解析が容易ではないという課題がある．VP の中には Blockly や MOONBlock など，コードを JavaScript や Python などのコードに変換できるものも存在するが，Scratch ほど教育現場で広く使われているとはいえない．

コードの臭いは主にオブジェクト指向型言語を対象とした概念である．Scratch は，スプライトごとにコードを記述する．スプライトとは，ベクター形式の画像であり，スプ

ライトごとにその動作をプログラミングする手法が取られている．したがって，スプライトを一つのオブフェクトとみなすことができ，コードの臭いの概念を適応できると考えられる．Hermans ら（2016）は，Scratch におけるコードの臭いの影響を検証し，Long Method や Duplication を含むコードは，コード編集時のパフォーマンスを低下させると報告している [3]．Scratch において，規模の大きな作品ではステップ数が 500 を超えるものも少なくなく，リファクタリングによる効率化が有効であると考えられる．

2.4 Scratch におけるリファクタリングの例

Scratch を用いたリファクタリングの例として，単純なゲームを作成する場合を考える．ここでは，Pong と呼ばれるゲームを実装する．Pong は，マウスでバーを動かし，画面内を跳ね返るボールが画面下部の赤いバーに当たらないようにするゲームである．この実装において，ボールの二通りの実装を Fig. 2 に示す．Fig. 2 の右は Long Method を抱えた実装であり，左側はリファクタリングを行なった場合の一例である．メソッドを複数に分解することで条件分岐のネストが減り，可読性が向上している．ボールの動作に新たな機能を加える際も，コードの編集が容易となる．

3 支援ツールの構築に向けた課題

3.1 コード解析の処理方法

コード内でリファクタリング可能な部分を検出するためには，コードの分析が必要である．そのため，VP 言語で記述されたコードを解析可能な形式に変換する必要がある．Scratch プログラムの保存には，プログラム内で使用するグラフィック情報や音源，スクリプトをパックした独自形式のファイルが用いられる．これらのうち制御情報については，JSON ファイルとして独自形式ファイル内に含まれている．本研究ではこの JSON ファイルからスクリプトを抽出し，予め定めたパターンにマッチングするかをテキスト分析処理で行う．制御パターンの例を Table. 1 に示す．

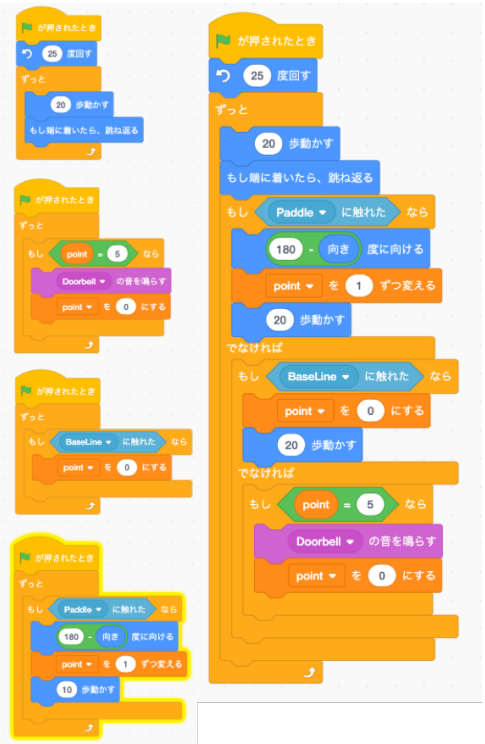

Fig.2 ボール動作における二通りの実装例

Table.1 制御パターンと検出パターンの例

制御パターン	検出パターン
条件分岐	”control_if_else”
繰り返し	”control_repeat”
X 歩動かす	”motion_movesteps”
変数への代入	”data_setvariableto”

Scratch 上の各ブロックは，付与されたユニークな ID で判別され，あるブロックは，次のブロックや入れ子になるブロックの ID を保持する．この ID を追うことで論理構造を解析することができる．

3.2 リファクタリング箇所の検出

リファクタリング支援ツールの構築にあたり，種々のコードの臭いの中から，どのパターンをリファクタリング対象とするかの選定が必要となる．またそれらコードの臭いが，どのような Scratch 上の制御パターンと一致するかを定義する必要がある．Scratch では多くのプログラムがネット上に公開されており，ダウンロードし編集することができる．したがって，これらのコードを収集し登場するパターンを分析することで，Scratch においてリファクタリングが有効なパターンの特定ができると考えられる．

4 今後の研究方針

本稿では，VP 言語の一つである Scratch においても通常のプログラミングと同様に，リファクタリングが可能かつ有効であることを示した．今後は，コード内のリファクタリングが必要な箇所を具体的に特定するための解析システム構築を行う予定である．また，その結果を踏まえ，VP におけるリファクタリング支援ツールを作成する．その後，被験者実験を行い，作成した支援ツールの有効性の検証を行う予定である．

参考文献

1) 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, Vol. 40, No. 3, pp. 131–142, 2011.
2) 小学校プログラミング教育に関する研修教材, http://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1416408.htm, 文部科学省, 参照 25.July.2019.
3) F. Hermans, E. Aivaloglou, Do code smells hamper novice programming? A controlled experiment on Scratch programs, 2016 IEEE 24th International Conference on Program Comprehension, pp. 1–10, 2016.