ビジュアルプログラミングにおける リファクタリング支援ツールの構築に向けて

米田 浩崇[†1,a)]　槇原 絵里奈[†1,b)]

概要：プログラミング教育において，Scratch に代表されるビジュアルプログラミング環境が広く用いられている．ビジュアルプログラミングに慣れた学習者は，大規模で複雑なプログラムを作成することも少なくない．一般的に，大規模なプログラムは複雑なロジックを持ち，可読性や保守性が低下する．しかし，ビジュアルプログラミング環境における教育では，コードの可読性や保守性を向上するためのツールが存在せず，高品質なコードの記述を学ぶ環境が整っていない．そこで本研究では，ビジュアルプログラミング環境におけるリファクタリングに注目し，可読性や保守性の高い高品質なコードの記述を支援するツールを提案する．

1. はじめに

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が重要視されている．プログラミングに関する教育として，教育機関や民間団体，企業によって多くのプログラミング教室が実施されている．

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（以降 VP）言語が広く用いられている．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．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，可読性や保守性を重視したコードの記述方法を学ぶ機会が少ない．一般的なソフトウェア開発における現場では，開発環境がリファクタリングやデバッグなどの機能を提供し，コードの可読性や保守性の向上を支援する．一方 VP 環境では，主な対象が初学者である観点から，開発環境を単純に扱えることが重視されており，リファクタリングやデバッグを始めとする開発支援ツールが整っていない．

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る．また，適切なコードの記述方法を知ることで，学習者がテキストベースのプログラミングに移行する際の障壁を小さくすることに繋がると考えられる．そこで本研究では，VP 環境におけるリファクタリングに注目し，可読性および保守性の高い高品質なコードの記述を支援するツールを提案する．

[†1] 現在，同志社大学
Presently with Doshisha University
[a)] hyoneda@mikilab.doshisha.ac.jp
[b)] emakihar@mail.doshisha.ac.jp

2. 準備

2.1 本研究で対象とする学習者のレベル

VP 環境はプログラミング初学者を対象とした学習環境であるが，意欲を持って取り組んでいる学習者は，規模の大きなゲームのような複雑なロジックを持つコードを記述することがあり，実際の作例も多く公開されている．

一般的に，プログラムが大規模であるほどロジックは複雑になり，可読性や保守性が低下するため，リファクタリング支援による効用が大きいと考えられる．したがって，本研究で提案するツールが対象とするのは，VP 環境に関する基本的な操作を習得済みで，プログラミング行為に大きな興味を持つ学習者である．

2.2 Scratch

本研究では，VP 言語の一種である Scratch[*1] 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り，様々な機能を持つブロックを組み合わせることでプログラミングを行う．2019 年 4 月時点でのユーザー数は 3900 万人であり，多くのプログラミング教育の現場で使用されている．無料で利用でき，かつ環

[*1] https://scratch.mit.edu/

境導入も容易であるため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 [1].

2.3 Scratch におけるコード解析の処理方法

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ベクター画像や音源，ブロック情報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，ブロックの組み合わせが示すロジックについては，JSON ファイルとして独自形式ファイル内に含まれている．Scratch 上の各ブロックの種類，パラメータ，順序等が JSON で表されており，例えばブロックの種類は**表 1** に示すような値で区別されている．JSON ファイルに対してテキスト分析処理を行うことで，組まれたブロックが示す動作を機械的に取得できる．

3. リファクタリング支援ツールの構築に向けた課題と考察

3.1 リファクタリングすべき箇所の定義

リファクタリング支援ツールを構築するにあたり，どのようなコードをリファクタリング対象とすべきかを定義する必要がある．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「コードの不吉な臭い」がある [2]．コードの不吉な臭いとは，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．例えば，一つのメソッドが長い状態を Long Method と呼び，これは可読性の低下につながる．また，同じコードが複数箇所に存在する状態を Duplication といい，これは保守性の低下につながる．このような特徴が該当するコードは修正が必要であると判断でき，関数化や変数の抽出を行うことで可読性や保守性を改善できる．コードの不吉な臭いは，主にオブジェクト指向言語を対象としたものであるが，いくつかの特徴は VP においても応用可能である．実際に Long Method や Duplication を含むコードは，学習者のコード編集時のパフォーマンスを低下させると報告されている [3].

リファクタリングすべき箇所の提示にあたり，種々のコードの不吉な臭いの中から，どのパターンが VP においても適応できるかを調べる必要がある．また，それらの特徴が，Scratch 上のどのようなブロックの配置に対応するかを定義する必要がある．Scratch では多くの作例がインターネット上に公開されており，ダウンロードすることができる．したがって，これらのコードを収集し，コードの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であると考えられるパターンを絞り込むことができると考えられる．

表 1 制御パターンと対応する値

制御パターン	値
条件分岐	”control_if_else”
繰り返し	”control_repeat”
スプライトの移動	”motion_movesteps”
変数への代入	”data_setvariableto”

3.2 リファクタリングすべき箇所の提案方法

リファクタリング支援ツールは学習者に対し，プログラム内でリファクタリングが可能な箇所および修正案を提案する．具体的には，開発ツール上にボタンを設置し，学習者が任意の時点でボタンをクリックした際に，問題箇所と修正案を表示するものを考えている．

3.3 Scratch 固有の課題

Scratch が提供する機能のうち，外部インターフェースに相当するのはブロック定義とメッセージパッシングである．ブロック定義では，特定の動作を行うブロック群を新たな一つのブロックとして定義できる．一般的なプログラミング言語における関数の定義に相当し，引数を与えることができるが戻り値を持たせることはできない．メッセージパッシングでは，プログラム内で特定のメッセージを発信することで，そのメッセージに対応するブロック群を実行できる．これも関数の定義に相当するが，引数および返り値を持たせることはできない．また，Scratch ではプログラムの動作出力が音やアニメーションで表示される．そのため，コードの単体テストが容易でないという課題がある．リファクタリングの前後で振る舞いが変わっていないことを機械的に判定する仕組みが必要である．

4. おわりに

本稿では，VP 言語の一つである Scratch を対象としたリファクタリング支援ツールを提案した．本ワークショップでは，課題として挙げた項目に中心に，リファクタリングの実施を学習者に促す具体的な方法や，既存のリファクタリングに関する研究の VP 環境への応用に関して議論したい．

参考文献

[1] 文部科学省:小学校プログラミング教育に関する研修教材，入手先 ⟨http://www.mext.go.jp/a_menu/shotou/zyouhou/detail/1416408.htm⟩ (2019.08.08).

[2] Fowler, M.: Refactoring: Improving the Designof Existing Code, Addison Wesley(1999).

[3] Hermans, F. and Aivaloglou, E.: Do code smells hamper novice programming? A controlled experiment on Scratch programs, *2016 IEEE 24th International Conference on Program Comprehension*, pp. 1–10 (2016).