コードの洗練活動を促すコード共有型プログラミング学習プラットフォームにおける静的な評価指標の検討

Investigation of Static Quality Indicators for A Code-Sharing Programming Learning Platform Promoting Code Refinement Activities

前田 新太郎[*1]，古池 謙人[*2]，東本 崇仁[*3]
Shintaro MAEDA[*1], Kento KOIKE[*2], Takahito TOMOTO[*2]
[*1] 千葉工業大学大学院情報科学研究科
[*1]Graduate School of Information and Computer Science, Chiba Institute of Technology
[*2] 京都大学学術情報メディアセンター
[*2]Academic Center for Computing and Media Studies, Kyoto University
[*3] 千葉工業大学情報科学部
[*3]Faculty of Information and Computer Science, Chiba Institute of Technology
Email: front4.shintaro@gmail.com

あらまし: コードの洗練活動を促すコード共有型プログラミング学習プラットフォームはこれまでの評価から一定の学習効果が示唆された．しかしながら，評価指標の妥当性について検証したところ，コードの構造を評価する観点では課題が指摘された．そこで本研究では，コード品質に関するカテゴリを用いた静的な評価指標の実装を検討する．
キーワード：コードの洗練活動，仮想ロボットプログラミング，静的な評価指標

1. はじめに

プログラミング学習では，コードの洗練活動が重要だと考える．コードの洗練活動を促す方法の一つとして，教授者が提示する良いコードの例から学ぶような，他者のコードから学ぶ方法があげられる．一方，このような既存の学習方法は，学習者自身のレベルと学習対象の他者のコードのレベルに大きな差が生じる場合，学習に繋がらない恐れがある．

本研究ではこれまで，学習者の記述したコードベースに，少し良い他者のコードを共有する制御機能を搭載したコード共有型プログラミング学習プラットフォームを開発してきた[(1)]．開発したシステムは，評価から一定の学習効果が示唆された一方，プログラミング熟達者による評価指標の妥当性を検証したところ，システムによるコードの構造の評価について，課題が存在することが示唆された[(2)]．

そこで本研究では，新たな評価指標として静的な評価指標の検討を行う．

2. コード共有型プログラミング学習プラットフォーム

他者からの学びを促すためには，学習者のコードと他者のコードが同じ学習題材であることが望ましい．本研究では，仮想ロボットプログラミングを用いて共通化を試みる．本学習題材は，ロボットに対して「畑に移動し，種を植え，作物を収穫する」プログラミングを行う農作物収穫ゲームを備える．

他者のコードから学ぶ際には，共有されたコードがどのような観点から優れているかを認識することが重要だと考えた．本研究では，前述した学習題材の課題設計に沿って，「収穫数」「コスト」「合計スコア」の 3 つを指標とした評価指標を提案した．収穫数は作物をどのぐらい収穫できたかを表す．コストはロボットの行動コストを表す．合計スコアは，収穫数からコストを引いたスコアである．これにより，作物を多く収穫する（生産性の高い）コードなど，コードの評価の認識を容易にすることが可能となる．

他者からの学びを促すためには，学習者のレベルに近いコードが共有されることが望ましいと考える．本研究では，ランキングを用いた学習者のレベルに近いコードのみ共有を制御する機能を提案した．具体的には，前述した評価指標のスコアを用いてすべての学習者をランク付けし，そのランクに近いコードのみ閲覧できるコード共有制御機能である．レベル差の少ない良いコードから学べるため，学習を阻害せずに段階的なコードの洗練活動につながる．

3. 現状の評価指標における問題点

提案した評価指標について，実際に実験参加者が作成したコードと評価指標によるスコアを一緒にプログラミング熟達者に提示し，妥当性を検証してきた[(2)]．一部，コードと評価について一定の妥当性が示唆されたが，コードの構造に関する評価の必要性が示唆された．そこで本研究では，学習者の記述したコードの構造を直接評価する静的な評価指標の検討を行う．

4. 静的な評価指標の検討

4.1 概要

静的な評価指標を実現する概念の一つとして，リファクタリングがあげられる．ここでのリファクタリングとは，コードが持つある機能の内容を変更せ

ずに，コードの構造をより良いものへ洗練する活動であると考える．Keuning らは，教育的観点からリファクタリングの重要性を述べており，Stegeman らが設計したコード品質に関する評価基準 (3) を基に，リファクタリングに特化した，コード品質のモデルを設計した (4)．本評価は，コードのコメントやインデントに関する評価は含まれず，コードのアルゴリズムと構造に着目している．本研究では，本モデルを元に，静的な評価指標の導入と設計を試みる．

4.2 コード品質のモデル

Keuning らが提案したコード品質のモデルを表 1 に示す．まずカテゴリとして，「Flow」「Idiom」「Expressions」「Decomposition」「Modularization」の 5 つを提案している． Flow はコードの構造や流れに関する問題点を評価する．例えば，if などの条件分岐が複数回呼び出されることでネストが深くなるコードや処理が複数回繰り返されているコードがあげられる． Idiom はコードの構造に関連する問題点を評価する．例えば，配列を用いて記述できる処理を，複数の if を用いて記述しているコードがあげられる．Expressions は式の表現に関連する問題点を評価する．例えば，意図的に分割できそうな式を分割せずに宣言しているコードがあげられる．Decomposition は，コードのアルゴリズムにおける分割に関連する問題点を評価する．例えば，一つの関数内で長大な処理を記述したコードがあげられる．最後に Modularization はモジュール化に関連する問題点を評価する．例えば，目的が明確でないクラスが記述されたコードがあげられる．

本研究では，これらのカテゴリを静的な評価指標と定め，それぞれの評価指標を開発したシステムにおいてどのように適応するかを検討する．

4.3 コード品質のカテゴリを基にしたシステム実装の検討

Flow はネストが深いほど，良くないコードとして評価する．本システムでは，独自のロボットの関数として，作物の成長度を取得する GetGrowhLevel 関数や，前面の状態（畑や空き地など）を取得する GetFrontPanel 関数など用意しているため，初学者がネストの深いコードを生成する可能性は高い．よって，学習者が記述したコードからネストの深さを探索し，スコアとして算出する仕組みを検討中である．また，条件分岐の数に基づいたコードの複雑性を表す循環的複雑度による評価の実装を検討している．

Decomposition について本研究では，メイン関数あるいは学習者が作成した関数において，長大な処理を検出することで評価の実現を試みる．具体的には，学習者の記述したコードから関数を検出し，その関数の中で記述されている関数の数をカウントする．その数が多くなるに連れて，Decomposition の評価を加算する．対して，適切な関数の分割に対する評価も検討しているが，学習者があらゆるケースで関数化の実行を行ってしまうと，一切の影響を受けずに良い評価であるとシステムは評価してしまうため，それを防ぐ手法の検討が要求される．

表 1 Keuning らによるコード品質のカテゴリ

カテゴリ	内容
Flow	ネストやパス，コードの重複，到達不可能なコードなど
Idiom	不適切な制御構造，不適切なライブラリ関数の利用
Expressions	複雑な式や不適切なデータ型の使用
Decomposition	メソッド内での長大な処理の記述，
Modularization	目的が不明確なクラスやメソッドの記述

上記の問題点を解消する評価方法として Modularization があげられる．これは，ある関数が明確な目的を持っているかどうかを評価する．つまり，前述した“あらゆるケースで関数化”を実行した関数は，その関数は目的があきらかではないため，Modularization を用いて適切に評価を行うことが可能となる．しかしながら，現状のシステムでは学習者が記述したコードに対して，システムがどのような機能を持つかを解釈できる仕組みは実現できていない．よって，機能を解釈できるような仕組みを提案することが今後の課題としてあげられる．

現状では，配列の操作や式の記述を学習者に促す課題設計にはできていない．よって，Idiom と Expressions について，システム実装は未検討である．本研究では，既存の課題設計の拡張を検討している．

5. おわりに

本稿では，コード共有型プログラミング学習プラットフォームにおける静的な評価指標の導入を検討した．今後の課題として，これまでの実験参加者学習者が記述したコードを参照し，具体例をベースに検討することが挙げられる．

謝辞

本研究の一部は JSPS 科研費 JP22K12322，JP21H03565，JP20H01730 の助成による．

参考文献

(1) 前田新太郎， 茂木誠拓， 古池謙人， 東本崇仁: “仮想ロボットプログラミングを用いたコード共有プラットフォームの開発と評価”，教育システム情報学会誌, Vol.40, No.3, in press（2023）

(2) 前田新太郎， 茂木誠拓， 古池謙人， 東本崇仁: “プログラミングを対象とした競争型知識共有プラットフォームにおける評価指標の妥当性検証”，人工知能学会研究会資料 先進的学習科学と工学研究会, pp.12-17（2021）

(3) Stegeman, M., Barendsen, E. and Smetsers, S.:“Towards an empirically validated model for assessment of code quality”, Proceedings of the 14th Koli Calling international conference on computing education research, pp.99-108 （2014）

(4) Keuning, H., Heeren, B. and Jeuring, J.:“Code Quality Issues in Student Programs”, ITiCSE Conference on Innovation and Technology in Computer Science Education, pp. 110–115 （2017）