プログラムロジック構成躓き推定のためのコンパイル履歴とソースコード間類似度のハイブリッド分析

Hybrid Analysis of Compilation Histories and Similarity Degrees between the Source Codes for Estimation of Stumbling in Construction of Program Logic

立花 隼一[*1], 中山 祐貴[*2], 大沼 亮[*1], 神長 裕明[*1], 宮寺 庸造[*3], 中村 勝一[*1]
Junichi TACHIBANA[*1], Hiroki NAKAYAMA[*2], Ryo ONUMA[*1], Hiroaki KAMINAGA[*1], Youzou MIYADERA[*3], Shoichi NAKAMURA
[*1] 福島大学 共生システム理工学研究科
[*1]Computer Science and Mathematics, Fukushima University
[*2] 早稲田大学 グローバルエデュケーションセンター
[*2]Global Education Center, Waseda University
[*3] 東京学芸大学 教育学部
[*3]Faculty of Education, Tokyo Gakugei University
Email: {s1970028, onuma, kami, nakamura}@sss.fukushima-u.ac.jp, nakayama@aoni.waseda.jp, miyadera@u-gakugei.ac.jp

あらまし: ソフトウェア開発人材に対するニーズの高まりを受け，大学等の教育機関でプログラミング演習が盛んに実施されている．プログラミング演習では，一般に，多数の学習者に対して少人数の教授者で対応する必要がある．プログラミング演習支援に関する研究が多数報告されているが，殆どが，いわゆる文法エラーを扱ったものである．一方，意図する処理などをプログラムとして形作る（ロジック構成）ことへの躓きは，コンパイルエラーとして表出しないため，把握の重要性は認識されつつも，有効な支援が実現されていない．本研究では，コンパイル履歴とソースコード間類似度の分析に基づいて，プログラムロジック構成面の躓きを推定する手法の開発を目指す．
キーワード: ロジック構成，躓き推定，学習履歴分析，ソースコード間類似度，プログラミング学習支援

1. はじめに

プログラミング演習では，学習者は頻繁に問題に遭遇し，その解決に努める．未熟者にとって重要な経験であると同時に, 難しい作業でもある. ここで，学習者が直面する問題は，文法的なエラーと，コンパイルエラーとして表出し難い論理的なエラーに大別できる．文法エラーなど，コンパイルエラーとして表出するため，学習者自身による解決の努力もある程度期待することができるが，コンパイルエラーとして表出し難いエラーは，学習者自身による解決のための模索が難しい．特に，プログラミングの基礎は学習しているものの文法など個別知識にとどまり，自らプログラムを形作る能力が十分でない段階（第 2 の初学段階）では，思い描く処理等をプログラムとして形作る(ロジック構成)面の躓きが多い．この種の躓きは，後のより高度な学習段階にも大きな影響を及ぼし，その把握と指導の重要性が指摘されている[(1)]. 指導・助言のためには，個々の学習状況の把握がとりわけ重要だが，実際の演習授業の制約下では限界がある．

これに対して，プログラミング演習における進捗状況把握支援[(2)]や, プログラム誤り検出システム[(3)]など多くの研究が報告されているが，その殆どが，いわゆる文法エラーを扱ったもので，ロジック構成面の躓きの把握は, 十分な支援が実現されていない．

本研究では，コンパイル履歴とソースコード間類似度の分析に基づいて，プログラムロジック構成面の躓きを推定する手法の開発を目指す．

2. 問題点と方針

2.1. 問題点

本研究では，プログラミング演習における問題点のうち，主に以下の 2 つに焦点をあてる．

（問題点 1）文法エラーに比べ，コンパイルエラーとして表出しないロジック構成面の躓きを学習者自身で解決するのは困難である．

（問題点 2）少人数の教授者で，個々の学習者の躓きを把握するのは容易ではない．

2.2. 方針

上記の問題点に対して，本研究では，まず，実際のプログラミング演習における履歴データや躓きの観察を通して，主な推定方針を定める．これに基づいて，コンパイル履歴とソースコード間類似度の分析による推定手法を開発する（問題点 1 への対応）．次に，分析の過程で対象ソースとその分析戦略自体を動的に切り替える仕組みを開発する．その上で，これらの手法に基づいて，ロジック構成面の躓き状況を視覚化するシステムを開発する(問題点 2 への対応)．これにより，プログラミング演習におけるロジック構成面の躓き把握支援の実現を目指す．

支援の概要を図 1 に示す．学習者によるコンパイル作業の推移とソースコード間の距離推移から，躓いている学習者を推定する．当該学習者を教授者に

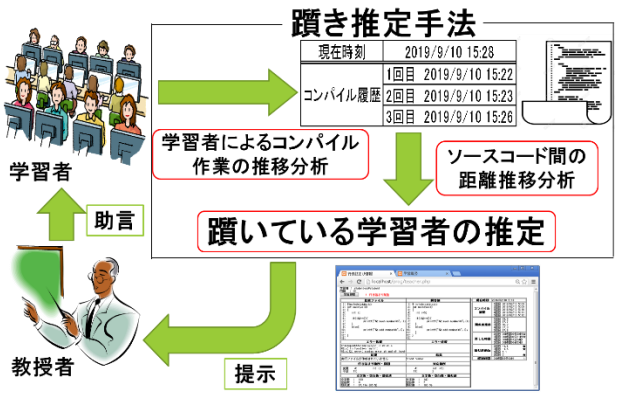

図１ 支援の概要

提示することで，プログラミング演習において，ロジック構成面の躓きを起こしている学習者を把握する作業を支援する．

3. ロジック構成面の躓き推定手法

3.1 躓き推定の手順

学習者の躓きを以下の手順で推定する．

1）コンパイル履歴（コンパイルしているが，コンパイルエラーの出ていない状態が一定以上続いている）を基に，該当する学習者，コンパイル日時，ソースコードを一次候補として抽出する．
2）1）で抽出したものについて，該当ソースコードと正解（解答例）を比較し，類似度を算出する．
3）一定以上の間，正解コードに対する類似度が上昇していない場合，ロジック構成躓きと判定する．

3.2 コンパイル履歴の分析による一次候補抽出

躓き推定手順（1）の例を図 2 に示す．1 回目のコンパイル時点ではエラーがあり，2 回目ではエラーがなくなっている．しかし，3 回目，4 回目とエラーがないにも関わらず，コンパイルを繰り返している．このような状態が一定以上続いている場合に，該当する学習者，コンパイル日時，ソースコードを一次候補として抽出する．

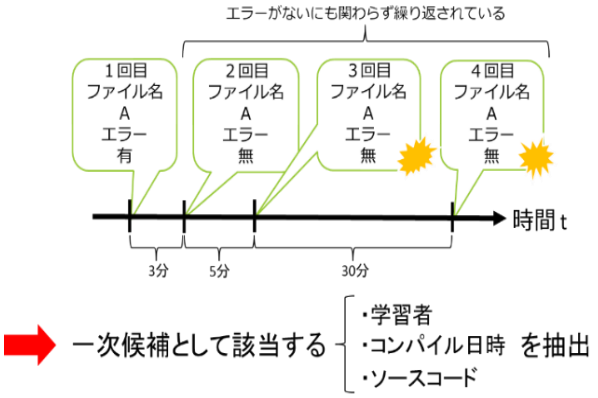

図２ コンパイル履歴の分析による一次候補抽出

3.3 ソースコード間の類似度算出

ソースコード間の類似度の算出には，性質の異なる複数の手法を用いる．具体的には，ソースコードをブロック単位に分割したもの（命令順列と呼ぶ）の類似性を算出する方法，および，ブロック構造を木に見立てた構文木の類似性を算出する方法をとる．ここでは，Order Leveshtein Distance (OLD), Tree Edit Distance (TED)，Jaro-Winkler Distance (Jaro)，Overlap Coefficient (OC)を用いる．

＜**OLD**＞ 2 つの命令順列間のレーベンシュタイン距離を求めることで類似度を算出する．つまり，学習者の命令順列を正解の命令順列に一致させるために，学習者の命令順を何回挿入・削除・置換を要するかを表す値を求める．

＜**Jaro**＞ 2 つの命令順列間のジャロウィンクラー距離を求めることで類似度を算出する．つまり，学習者のソースと正解ソースの間の文字列一致状況，置換の要・不要から算出する類似度である．

＜**OC**＞ 2 つの命令順列間のシンプソン係数（OC）を求めることで類似度を算出する．つまり，ある 2 つの集合(命令順列) のうち要素数が少ない方の要素数と共通要素数の割合から値を求める．

＜**TED**＞ 2 つのツリーの間のレーベンシュタイン距離を求めることで類似度を算出する．つまり，正解の構文木に一致させるために，学習者を何回・削除・置換を要するかを表す値を求める．

3.4 類似度算出法の動的切替

本研究では，いくつかの手法で類似度を算出する体制を準備し，状況に応じて相応しい算出方法を動的に採用する方針をとる．そのために，実際のソースコードを用いた検証を行い，どの算出方法を用いれば最も効率よく正確なデータを得られるか比較・検証する．この作業を通して，各類似度算出法の特徴の把握に努め，手法切替の「指標」を整備する．指標に基づいて動的切替を担うモジュールを開発し，それを用いた反復的な検証と改選を試みる．

4. おわりに

本稿では，コンパイルエラーは発生していないものの，思い描く処理像をプログラムとして形作ることができず躓く状態，即ちロジック構成面の躓きの推定手法の提案をした．今後は，実際のデータを用いて提案手法の検証を重ね，動的切替手法の開発につなげたい．

参考文献

(1) P.Kinnunena，B.Simonb，“My program is ok – am I? Computing freshmen’s experiences of doing programming assignments”，*Computer Science Education*, Taylor & Francis, Vol.22, No.1, pp. 1-28 (2012).
(2) 井垣宏，斎藤俊，井上亮文，中村亮太，楠本真二，“プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案”，情報処理学会論文誌，Vol.54, No.1, pp.330-339 (2013).
(3) 櫻井桂一，“プログラミング実習を支援するための C プログラム誤り検出システムの開発－コンパイラによって検出されない誤りを中心として－”，日本教育工学会誌，Vol.25, pp67-70 (2001).