Slackを用いたプログラミング学習者のつまずきの検出支援

浦上 理[1]　長島 和平[2]　並木 美太郎[2]　兼宗 進[3]　長 慎也[1]

概要：これまで我々は，学習者のつまずきを正解行数という指標を用いて検出する研究をしてきた．しかし正解行数は，学習者が課題を完成させていることが前提となった指標であったため，つまずいていた学習者に対してリアルタイムに指導を行うことには適していなかった．本発表では，教員による学習者のつまずき把握を支援するため，プログラミング学習環境 Bit Arrow が検出したエラーを，教員や TA に向けて Slack で配信するような bot を実装した．配信された情報からつまずきを検出し，必要に応じて教員が指導することを支援することで，学習者のつまずきからの復帰をリアルタイムに支援できるようになると考えられる．

1. はじめに

プログラミング学習において，学習者は何らかの問題が発生したときに，解決のために教員や TA に質問を行うことがある．一方で，そのような質問を行わなければ問題を解決できない状況（これを「つまずき」と呼ぶ）に陥っているにもかかわらず，質問などのアクションをとらない学習者もいる．このような学習者に対しては，教員や TA がその学習者のつまずきを検出して，サポートを行う必要がある．

これまで我々は，学習者のつまずきを正解行数 [1] というものを用いて検出する研究をしてきた．BitArrow[5] のログから最後に実行したプログラムを正解と仮定し，その行数からある時点での行数の差分より進捗度を求めるというものであった．

しかし，この方法では学習者が課題が終わっていないとつまずいているかどうかを判断することができないため，授業を行っている最中に即時につまずきを把握することができなかった．このため，つまずいていた場合に指導が遅れることとなった．

本稿では，学習者のつまずきをリアルタイムに，かつ自動的に教員に通知する仕組みを利用して，授業を実践した結果を報告する．

BitArrow のログには，エラーを起こしたかどうかの情報も含まれており，これに関してはリアルタイムに検出することができるため，即時性を上げるためにエラーの情報も併用することを考えた．また，教員への通知をスムーズに行うため，学習者や教員が使い慣れている Slack を用いて通知を受け取れることが望ましいと考えた．

そこで，本発表では，BitArrow が検出したエラーを Slack から教員に自動的に通知する仕組みを提案する．検出したエラーを Slack 経由で教員に通知することで教員が指導し，学習者がつまずいている状態から復帰できるようになると考えた．

現段階では，Slack で通知されたエラーの中から，さらに目視でつまずいている学生を抽出し指導を行っているが，今後は，その部分を自動化したいと考えている．今回の実践から得られた結果（Slack や BitArrow ログ）を用いて，どのような傾向でつまずいているかを分析し，自動指導に向けてのアルゴリズムを構築していく．

2. 関連研究

前述のとおり，これまでのシステムは課題完成後につまずきを検出していた．学習者のつまずきを検出し，教員に把握させるシステムは，いくつか提案されている．

プログラム演習のための進捗モニタリングシステム [2] では，全クラスの進捗表示ページとクラス別進捗ページを出力し，各学生の座席番号，学生番号，課題全体および各課題の評価値とファイル変更日時を表示して進捗状況を確認していた．ここでいう評価値は，事前テストや必須テスト，コンパイル，実行テスト，補足テストがどのくらいできたかを評価し値として出す．

プログラミング入門教育を対象としたリアルタイム授業支援システム [4] では，評価項目としてコンパイル，インデ

[1] 明星大学
Meisei University
[2] 東京農工大学
Tokyo University of Agriculture and Technology
[3] 大阪電気通信大学
Osaka Electro-Communication University

©2021 Information Processing Society of Japan

ント，クラスの形式的記述，ユニットテストで評価し，進行度の算出として学習者のソースコードが，コンパイル・インデント・クラスの形式的記述，ユニットテストの評価項目に対して正解している問題数を完全正解数と置き，その完全正解数が大きい学習者を上位として算出していた．

授業支援システムにおけるプログラミング演習のための学習状況支援機能の設計と評価 [3] では，学習状況分析として，回答開始時刻，コンパイル時刻，実行時刻，正解時刻，提出時刻を現在の時刻と統計分析してつまずいている学生を分析していた．また，正解判定としてシステムが実行時に学生の回答と模範解答を比較して正解，不正解を画面に表示する．

これらの方式では，模範解答やテストケースが明確に定義できる問題であれば有効に働くが，模範解答と違うやり方で題意を満たしていた場合でも不正解扱いにされる問題がある．また，テストケースが設定できないような，作品制作の課題などでは使うのが難しい．これに対して本手法では，まずエラーの発生をつまずきのきっかけと判断しているため，テストケースが設定できない場合でも有効である．もちろん，エラーではないが題意を満たさなくてつまずいている，という事例にも対応する必要がある．これについては「正解行数」を用いることで，事後ではあるが題意を満たさない事例が検出できるため，これらの方式を組み合わせて精度を上げていくことを考えている．

3. つまずきの検出方法

本実践では，学習者が文法エラーや実行時エラーを発生させた際に，それを学習者のつまずきの前兆とし，Slack を通じて教員に通知するシステムを実装し使用した．

3.1 学習環境で検出されるエラーの情報

ソースコードの編集履歴を取得するためプログラム開発環境『Bit Arrow』のログ収集機能を用いた．Bit Arrow のログは実行のたびに収集しており，実行内容の各項目には，次の情報が含まれている．

- ユーザ ID
- 実行時刻
- 実行時のソースコード
- 実行結果
- エラーメッセージ
- ファイル名

課題ごとにファイル名が指定されている場合[*1]は，そのユーザ名と課題ファイル名で学習者のある課題に対する行動を見ることができる．

[*1] Bit Arrow には教員によるソース配布機能があり，全学習者に課題用ファイルを同じ名前で配布できる．

3.2 教員や TA に配信される内容

それぞれの学習者がどの箇所でつまずいているかを把握するため，Slack に BitArrow が検出したエラーを教員や TA に向けて配信するような bot を実装した．配信内容には，次の情報が含まれている．

- エラーを出したログの URL
- 学籍番号
- 直近の 5 分間で起きたエラー件数
- ファイル名
- エラー内容

ログの各実行から特定の学習者のユーザ ID および調査対象となっている課題で指定されているファイル名を持つものを抽出し，その学習者がエラーを出したのちにそのエラーを解決できなかった場合に配信することとする．

また，配信間隔は 5 分間隔で，一番エラー件数の多い学習者を一番下に表示するようにした．これは，配信の際に上の方だとエラーを出した人が多い際に見つけづらいからである．

実際に Slack に配信される情報を**図 1** に示す．

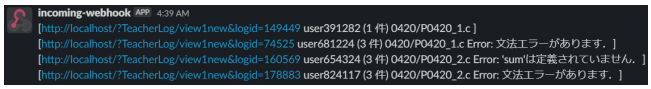

図 1 Slack に配信されるエラーの発生状況

順調に課題を進めている場合は，エラーが出ないと考えられる．また，理解度の高い学習者はエラーが発生しても自力で解決することができると考えられる．そのため，図 1 を見たときに，直近 5 分間で発生させたエラーの件数が多い学習者は，つまずいている可能性が高いという仮説を立てた．

4. 評価

Slack から教員に学習者のエラーを自動的に通知したことで，つまずいている状態からの復帰を支援できたかを評価する．

4.1 対象授業

2021 年度に明星大学情報学部の 2 年次配当で開講された『プログラミング演習 3』の，筆者担当講義を対象とした．この講義は，Python を用いたオブジェクト指向を学習する講義である．授業内容は**表 1** に示した通りである．この講義には約 40 名の受講者がいる．受講者は 1 年次に Python を用いた入門教育を受けている．授業体系としては，登校者を減らす取り組みでその日に出された課題が翌週の講義までに全部できている人は，Zoom を使用したリモートでの参加にし，課題ができていない人に対しては対面で行う予定であったが，東京都に緊急事態宣言が出されている関係で全員リモートでの参加に変更となっている．この講義で

©2021 Information Processing Society of Japan

表 1　授業内容

回	学習内容
1	ガイダンス
2	復習 1：変数・入力
3	復習 2：条件分岐
4	復習 3：繰り返し
5	復習 4：配列
6	復習 5：関数
7	オブジェクト指向：クラス・フィールド・コンストラクタ
8	オブジェクト指向：メソッド
9	オブジェクト指向：オブジェクト同士の関係
10	オブジェクト指向：継承
11	オブジェクトの配列
12	ファイル IO
13	グラフィックスの基本
14	オブジェクトとグラフィックス・アニメーション
15	振り返り

は Slack も使用しており，本発表で提案した bot の通知以外にも，学習者側からの講義に関する連絡・質問や教員側からの難しい問題のヒントを出す際にも使用されている．

4.2　評価方法

今回の実践では，エラーのみを抽出してリアルタイムで学習者をどの程度までサポートができるかどうかを評価する．

本授業でのサポート体制を図 2 に示す．現段階では，エラーの検出を本発表で提案した Slack の bot 機能で行っているが，それ以降の「つまずき」かどうか，つまり，サポートが必要かどうかの判断は現時点では目視で行っている．具体的には，bot が配信されたエラーの中から，特にサポートが必要と思われる事例を抽出し，bot が配信したチャンネルとは別のチャンネル（つまずき管理チャンネル）に教員や TA が転送を行っている．

つまずき管理チャンネルに転送された項目については，つまずきが解消できるまでの推移を教員や TA がスレッド機能を使い，管理している．そこでつまずきが解消できない場合は，教員が実際につまずいている学生にダイレクトメッセージを送り，修正して正解につなげていくという流れである．

図 2　授業でのサポート体制

その学習者の BitArrow 上のログと，Slack の指導の履歴を追跡し，改善が見込まれた場合つまずきが解消されたものとする．リアルタイムなサポートを行わなかった学習者の中にもつまずいている者がいたかの判断は，正解行数と合わせて授業終了後に確認を行い，つまずいていたのに指導が行き届かなかった場合がなかったかどうかを調べる．本稿執筆時には授業が進行しているので，調査はまだ行われていない．

4.3　エラー検出とサポートの事例

本授業では，習熟度別によって 4 つのクラスに分けられている．

本原稿執筆時（6 月 18 日まで），bot 機能で配信されたものが 1483 件（全クラス），つまずき管理チャンネルに転送されたものが 74 件（筆者担当クラスのみ），実際にダイレクトメッセージを送ったものは 12 件である．

次に，実際にサポートを行った事例を紹介する．

事例 1

```
def total2(a):
    t=0
    for x in a:
        if isinstance(x,list):
            t+=x+a #★
        else:
            t+=x
    return t
a=[1,[10,5],3]
print (total2(a))
a=[1,[10,5,[7,9]],3]
print (total2(a))
```

図 3　0514/p35.py(指摘前)

図 3 は，第 6 回の授業 (5 月 21 日) において，bot を通じて通知されたプログラムである．これは，配列 a の合計値を再帰的に合計する関数であり，★で示した箇所を適切に記入できていない．この通知があったのが 15:02 であり，つまずき管理チャンネルに転送したのが 15:09 である．検討した結果，15:16 に教員より，当該学生に次のようなダイレクトメッセージを送った．

> ちょっと前回の課題でお困りのようですので．
> そこは「再帰呼び出し」です．再帰呼び出しが何かは，LMS にアップロードした資料を一度ご覧ください．

この結果，当該学生は 15:21 に正解のプログラムを完成させ，15:30 には解決した旨の返信があった．

この学生は学期開始からこの時点までは，Slack による質問を自ら行ってはこなかったが，次週の 28 日は自ら 1 件の質問を行っている．

事例 2

図 4 のプログラムは，第 9 回の授業 (6 月 11 日) において，bot を通じて通知されたプログラムである．これは，`Circle` クラスのメソッド `move` の中身（★）を定義する課題である．なお，メソッド呼出（☆）の際には，`Circle` クラスの 内部のフィールドである `o` は使用してはいけない，という制限を課していた．本プログラムでは，★で示し

```
class Point:
    def __init__(self,x,y):
        self.x=x
        self.y=y
    def __str__(self):
        return "Point({},{})".format(self.x,self.y)
    def move(self, dx,dy):
        self.x+=dx
        self.y+=dy
class Rectangle:
    def __init__(self, o, width, height):
        self.o=o
        self.width=width
        self.height=height
    def move(self,dx,dy):
        self.o.move(dx,dy)
    def __str__(self):
        return "Rectangle(o={}, width={}, height={})".
            format(self.o, self.width, self.height)
class Circle:
    def __init__(self, o, radius):
        self.o=o
        self.radius=radius
    def move(self,dx,dy):
        move(dx,dy)   # ★
    def __str__(self):
        return "Circle(o={}, radius={})".
            format(self.o, self.radius)
ra=Rectangle(Point(3,5), 6,4)
ca=Circle(Point(6,3), 2)
print(ra)
print(ca)
ra.move(1,3)
ca.move(-3,4) # ☆
print(ra)
print(ca)
```

図 4 0611/p06.py(指摘前)

た箇所を適切に記入できていなかった．bot からの通知があったのが 15:00，つまずき管理チャンネルに転送したのが 15:03 である．15:10 に次のようなダイレクトメッセージを送った．

> p06 で苦戦しているようですが，これは Rectangle のものと同じでいいんですよ．

これに対して，当該学生から「o を使うことになるが問題はないか」という確認が Slack を通じて送られてきた．この学生は，先述した制限事項（呼出時に o を使えない）の意味を取り違えていたため，このままアドバイスを行わなかった場合，正解であるはずのプログラムに対して悩み続けた可能性もある．学生は 15:11 にはすでに正解のプログラムを入力し終え，15:19 に解決した旨の知らせがあった．なお，この学生は授業全体を通じて，Slack による質問を自ら行ったことは現時点では 1 回もない．

事例 3

次の事例は，検出されたもののサポートを見送ったものである．

通知が来た時点でのソースコードを**図 5** に示す．この通知を受け取ったのは 17:16 である．★の箇所において，メソッドの使用方法を理解していないような書き方を行っており，17:19 頃，TA と教員でサポートを行うかどうか協議をした上，様子を見ていたところ，17:20 に**図 6** のようなプログラムに変化し，突如自己解決した．

```
#前略
class Circle:
    #中略
    def contains(self, p):
        if Circle(def __init__)<=self.radius+: #★
            return True
        else:
            return False
#後略
```

図 5 caption=0604/p21.py（通知した時点でのソースコード, 抜粋）

```
#前略
class Circle:
    #中略
    def contains(self, p):
        if p.distance(self.o)<=self.radius:#★
            return True
        else:
            return False
#後略
```

図 6 0604/p21.py（自力解決した後のソースコード，抜粋）

5. 考察

本原稿執筆時点では，すべてのサポート事例についての検証は行っている途中であり，前節で挙げた 3 つの事例だけからの考察となるが，事例 1，2 については，サポートを必要としている（つまずいている）学生に対して，bot がつまずきを自動的に検出し，教員側からサポートを行うことで，この先に起こるであろうより深刻なつまずきを未然に防ぐことができた可能性が示唆される．特に，これらの事例で挙げた学生は，自分から質問をほとんどしない学生であるため，このような学生に対してもつまずきからの復帰をさせる手段を新たに増やすことができたと考えられる．

一方で，事例 3 のように，bot からの通知があったとしても，最終的にはサポートが必要でなかった事例もある．ただ，このような事例を検出しないようにすることによって，本当にサポートが必要であるにもかかわらず，検出されなくなる事例が発生してしまうことを考慮すると，ある程度広く検出を行い，教員や TA で選別を行う，現状の方法のほうが望ましいと考える．

逆に，本手法を使用してもリアルタイムに検出できないつまずきがないかどうかの検証を行うことが重要であると考える．課題が完成した状態においては，正解行数を使っ

©2021 Information Processing Society of Japan

たログの分析が可能である．正解行数は，順調に完成に近づいている場合は単調に増加し，何らかのトラブルがある場合は横ばい，または減少することが確認されている [1]．これを利用して，トラブルが発生していて，かつ，相当の時間が経過している状態においても，bot による通知がなかった事例の有無を今後調査していく．

これらの調査を行い，つまずきに共通する特徴を明らかにすることで，教員や TA による目視でつまずきを確認する負担を軽減し，より多くの学習者のつまずきからの復帰を支援できると考えられる．

「つまずきに共通する特徴」にどのようなものがあるかは複数考えており，3.2 で述べた直近 5 分間で発生させたエラーの件数のほか，プログラムに取り組んでいる時間，ログ総数や，他人のプログラムとの違いがつまずきと関係しているか考えている．これらの要素も含めてつまずきを今後調査する．

6. まとめ

本実践は，BitArrow のエラーを Slack から教員や TA に自動的に通知する仕組みを実装し，授業で使用した．通知されたエラーからつまずきを目視で確認し，学習者に直接指導を行った．実際に行った指導を調査したところ，自分から質問をほとんどしない学生に対して必要な指導を行えていた事例が見つかった．本システムを用いることで，これまで把握が難しかったつまずいている学生に対するサポートができるようになったと考えられる．今後この実験により学生のつまずき傾向を見つけ出し，直接学生につまずいた部分の自動配信が実現できる可能性を見つけ出し，教員が目視でつまずきを確認する負担を軽減することを目指す．

この実験によりこれまでつまずいていた人がつまずきから救うことができると考える．

参考文献

[1] 浦上理, 長島和平, 並木美太郎, 兼宗進, 長慎也：プログラミング学習者のつまずきの自動検出, 情報処理学会研究報告コンピュータと教育（CE）,2020-CE-154,Vol.4,pp.1-8(2020)
[2] 内藤広志, 齊藤隆：プログラム演習のための進捗モニタリングシステム, 情報処理学会研究報告,2008-CE-93,pp. 33-40(2008)
[3] 加藤利康, 石川孝：授業支援システムにおけるプログラミング演習のための学習状況支援機能の設計と評価, 情報処理学会研究報告,Vol.2012-CE-113,No.6,pp. 1-8 (2012)
[4] 長谷川伸, 松田承一, 高野辰之, 宮川治：プログラミング入門教育を対象としたリアルタイム授業支援システム, 情報処理学会論文誌,Vol. 52,No.12,pp. 3135-3149(2011)
[5] 長島和平, 長慎也, 間辺広樹, 兼宗進, 並木美太郎：Web ブラウザを用いたプログラミング学習支援環境 Bit Arrow の設計と評価, 情報処理学会論文誌教育とコンピュータ,Vol. 4,No. 1,pp. 57-69 (2018).

©2021 Information Processing Society of Japan