プログラミングにおけるログとつまずきの相関 － Slack のサポート履歴を利用した「15分ルール」の妥当性分析

長 慎也[1,a)] 浦上 理[1] 澤本 直輝[1] 市石 舜也[1] 長島 和平[2] 兼宗 進[3] 並木 美太郎[2]

概要：
筆者らは現在，Web ブラウザを用いてプログラミング学習が可能な環境「BitArrow」を，初学者向けのプログラミング学習活動に用いている．BitArrow は，ログの収集機能があり，ソースコードおよび実行結果（エラーを含む）を保存や実行のたびにログの収集を行うことができる．これまでの研究でも，ログの分析を行い学習者の「つまずき」を発見する手法を開発してきた．本研究では「つまずき」を「教員や TA からのサポートが必要な状況」であると定義している．この定義によれば，実際に学習者がつまずいているかどうか，つまり，サポートが必要な状態かどうかはログには直接記録はされていない．そこで，本発表では，オンラインで行われているプログラミングの授業において，Slack による質問を受け付けたタイミングにおいて，受講者のログの状態を解析することにより，つまずいている状態とログの関係を調査した結果を報告する．この調査結果を活用し，今後，ログからのつまずきの自動的な検出の手法を確立させていく．

Relationship between log and ”stucking”

CHO Shinya[1,a)] URAKAMI Satoshi[1] SAWAMOTO Naoki[1] ICHIISHI Shunya[1]
NAGASHIMA Kazuhei[2] KANEMUNE Susumu[3] NAMIKI Mitarou[2]

1. はじめに

プログラミング学習の過程において学習者が，エラーや期待した結果が得られない状況（総称して「トラブル」とする）に陥った場合，受講者の中には自分自身で試行錯誤を試みたあと，解決に至らない場合は教員や TA(以下「教授者」）にサポートを求めて解決を行う者もいる．一方で，トラブルが解決に至らない場合であっても，サポートを求めないまま，解答をあきらめてしまう者もいる．このような受講者を放置しておくと，ドロップアウトの原因となるため，教授者サイドからのサポート（働きかけ）が必要となってくるが，一般的な授業では数十人の受講者がいるため，教授者がそのような状態に陥っている受講者を発見することは困難である．本稿では，このようなまったく質問をしてこない学習者へのサポートを行う手法を確立することを最終目的とし，その予備調査として，学習者への働きかけを行う適切なタイミングについて調査を行った結果を報告する．

筆者らはこれまで，プログラミング学習環境として，Web ブラウザのみで動作可能な BitArrow[1] を開発し，初学者向けの授業におけるプログラミング環境に用いる実践を行っている．

BitArrow は，ログの収集機能があり，ソースコードおよび実行結果（エラーを含む）を次のようなタイミングで収集している．

- プログラムを保存したとき (Save)
- プログラムの実行を指示したとき (Build)
- 実行しようとしたプログラムにコンパイルエラーがあったとき (Compile Error)
- プログラムが正しく実行され，実行結果が返却されたとき (Run)

[1] 明星大学
Meisei University, Japan
[2] 東京農工大学
Hakuyo High School, Japan
[3] 大阪電気通信大学
Osaka Electro-Communication University, Japan
[a)] cho@eplang.jp

- プログラムが実行されたが，実行時エラーがあったとき (Runtime Error)
- プログラムの書き換えを行い，一定時間保存操作がなかったとき (Unsaved)
- その他，名前の変更 (Rename)，削除 (Remove) など

これらのタイミングで，次のような属性値をもつ「ログ項目」が記録される．

- **class** 操作をしたユーザ (以下単に「ユーザ」) のクラス ID
- **user** ユーザのユーザ ID
- **time** 操作をした時刻
- **lang** 操作をした対象となるプロジェクトのプログラミング言語
- **filename** 操作をした対象となるファイルのパス
- **result** 操作の種類（前述のタイミング一覧のカッコ内の識別子．Save, Build など）
- **detail** 実行結果（標準出力またはエラーメッセージ）
- **code** 実行時のプログラムのソースコード．

これまでの研究でも，ログの分析を行い学習者の「つまずき」を発見する手法を開発してきた．本研究では「つまずき」を「教員や TA からのサポートが必要な状況」であると定義している．この定義によれば，実際に受講者がつまずいているかどうか，つまり，サポートが必要な状態かどうかはログには直接記録はされていないため，ログの情報からそのまま「つまずき」を検出するには至っていない．

折しも，昨今の社会情勢により授業をオンラインで行う例が増えており，そこでのサポートには，Slack などのオンラインのコミュニケーションツールが活用されている．そこで，本発表では，オンラインで行われているプログラミングの授業で，Slack により質問を受け付けたタイミングにおいて，受講者のログの状態を解析することにより，つまずいている状態とログの関係を調査した結果を報告する．この調査結果を活用し，今後，ログからのつまずきの自動的な検出の手法を確立させていく．

2. 「つまずき」の定義と判定基準

まず，本研究は受講者が，トラブルに遭うのを完全に防ぐことを目的としているのではないことに留意したい．受講者自らがそのような状況から適切な解決策を講じたり，試行錯誤をしたりしてこれらのトラブルを解決する経験を積ませることこそが，プログラミング学習の重要な目的の一つであると言えるからである．

つまり，トラブルが発生したからといって，それがそのまま「つまずき」，すなわち，教授者によるサポート・指導が必要な状況であるとは限らないという点に注意しなければならない．

では，どのようなトラブルであればつまずきとみなすべきか？

この質問について，1 つの指標となりそうな事例を挙げると，fast.ai の創立者である Rachel Thomas は，Twitter で「15 分ルール」というツイートをしている[*1]．これは「もし，何らかのトラブルが起きたら，15 分間自分自身で解決を試み，15 分経ったら他の人に助けを求めるべきである．もし 15 分経たずに他の人に聞けば，他の人の時間を無駄にし，15 分経っても聞かなければ，自分自身の時間を無駄にする」という趣旨のものである．この事例から，あるトラブルの発生時点から，進展がないまま一定の時間（例えば，15 分）が経過したあとは，他人のサポートが必要，すなわち「つまずき」の状態に至る，と見なせる可能性が示唆される．

3. 関連研究

プログラミング学習の進捗を管理し，「つまずき」を検出する手法は数多く提案されている [2][3][4][5].

挙手をしない学生に向けたサポートとして，ログから「エラーを頻発させている」ことを検出して，つまずきを早期発見する研究もある [5] が，エラーではないが題意を満たさない場合におきる「つまずき」には対応できない．テストケースを用いて「つまずき」を判定してる例 [2] もあるが，作品制作のような明確な答えがないものや，グラフィックを使用するような明確なテストケースを作成できない場合には検出が難しい．

また，「他の受講者に比べて進捗が遅い」などの手法を用いて「つまずき」を検出しているものもある [2][3][4]．相対的な比較を用いてサポートの必要の有無を検出することは，一斉授業においては有効と考えられるが，今後，オンライン（オンデマンド型）の授業が増え，演習が非同期になる（受講者ごとに異なる時間帯に行われる）と，比較が難しくなる可能性がある（実際，本稿で実践する内容は授業時間外の演習も見られている）．

また，提案システムを用いた「サポートが本当に有用であったか」を検証している例もある [3] が，受講者からの聞き取り調査をベースにしている．これに対して，本稿では，Slack によるオンラインサポートを行った履歴および，コードの編集履歴を用いて，サポートが必要であると考えられるタイミングを，質問を行わない学習者についても自動的に予測しようとする試みである点が異なる．また，オンデマンド型の授業においても，ログを常時残すようにし，サポートも常時行えるようになっていれば，授業時間外であっても，他の受講者の進捗に関係なくつまずきを検出可能であると考えらえる[*2]．

[*1] https://twitter.com/math_rachel/status/764931533383749632

[*2] 実際には，常時サポートを行うことは人的資源の問題から困難ではあるので，「今後の予定」で後述するような機械的なサポートを組み合わせる必要がある．

4. 調査の概要

4.1 調査の目的

図 1 に，本発表で行う調査の概要を示す．1 で述べた「サポートを求める受講者」も，「サポートを求めない受講者」も，トラブルが発生してから一定期間が経過したあとのログの状態には共通する性質があると考えられる．その性質を抜き出すことで，教授者がサポートを求めない受講者に対して働きかけを行うべきタイミングを明らかにすることを目指す．

今回は特に，2 で紹介した「15 分ルール」つまりトラブルが発生してから，進展がないままある一定期間が経過したときに「つまずき」と見なす，という判定基準について，次のことを明らかにする．

- 15 分ルールは妥当か，つまり，学習者がトラブルに遭遇してから，「つまずき」の状態に至るまでの期間はおおむね 15 分か
- 15 分ルールは妥当ではない場合でも，「つまずき」に至るまで期間に一定の傾向が見られるか（「n 分ルール」が成立するか）
- 15 分ルール，あるいは n 分ルールが妥当な場合，判定を自動化することは可能か

これらのことを明らかにするにあたり，学習者の質問行動の記録を用いることにした．質問をしてくる学習者は，何らかの合理的判断によって，自分自身が「つまずいた」つまりサポートが必要と思った，とみなすことができる．そこで，学習者がトラブルに遭遇してから，Slack などを通じてサポートを求めるまでの時間に一定の傾向があれば，質問をしていない学習者に対しても，トラブルが発生してから同程度の時間が経過したときに，働きかけを行うことが望ましいと考えらえる．

実際には，本来すべき試行錯誤を行わず早めに質問を行ったり，必要以上に試行錯誤を行いすぎた後に質問を行っていたりしている学習者もいると考えられるが，今回は，そのような学習者も含めた多くの事例を集めることで，全体の傾向から「15 分」という時間が妥当性のあるものかを検証する．

4.2 調査の手法

トラブルの発生については，BitArrow のログの内容を分析することによって，次のようなトラブルの状態にあることを確認することが可能である．

- コンパイルエラー・実行時エラーが発生した
- エラーではないが，課題が要求する仕様（題意）を満たしていない（ロジックエラー）

「コンパイルエラー・実行時エラーが発生した」については，BitArrow のログの **result** 属性にエラーであるかどうかが記載されているので，容易に判別可能である．

また，「ロジックエラー」についても，これまでの研究で「正解行数」という尺度を提案している [6]．これは，ログに残されているある時点でのソースコードについて，同一ファイル名の最後のソースコード（学習者が最終的にはその課題に正解していることを前提としている）からの差分を用いた値を用いて算出されるものであり，正解行数が増えているほど，正解に近づいていると判断できる．

したがって，これらのトラブルが「進展がないまま」かどうかを，コンパイルエラーについてはエラーの状態が続いていることで，ロジックエラーについては正解行数が増えていない状態が続いていることで，判別可能と考えた．

さらに，「つまずいた」，つまり教授者のサポートが必要である状況にあるかどうかは，ログからは直接わからないため，Slack でのやりとりの履歴を組み合わせた．

今回の授業は，ほとんどがオンラインで行われたため，質問がある場合，Slack から教授者宛に DM(ダイレクトメッセージ) を送るように指示をしておいた．このため，質問があったタイミングを「つまずき，教授者によるサポートが必要な状態」であったとみなし，これを質問者の当該ログと突き合わせることで，トラブルが発生してから，つまずきの状態に至るまでの所要時間を割り出すことが可能となる．

この所要時間を割り出すことは，「サポートを求めない」受講者に対して，どのタイミングで「つまずき」と見なし，教授者サイドからの働きかけを行うべきか，ということを知る指標となりうる．

BitArrow のログ閲覧機能の一部を**図 2** に示す．ログ項目ごとに，エラーや正常実行かの区別を提示するほか，先述した「正解行数」の値を表示している．正解行数が当該課題の最大値を更新した瞬間を「進展した」とみなし，♪マークで示している．また♪マークが表示されてから一定の時間が経過すると，色を変えて進展していない旨を警告するようにしている．

今回は，受講者から Slack 経由での質問があった時刻を「つまずき」の瞬間と見なし，次の時間を測定した．

- 「当該課題着手開始」から「つまずき」までの操作時間
- 「トラブル発生」から「つまずき」までの操作時間
- 「最後の進展」から「つまずき」までの操作時間

「当該課題着手開始」については，当該課題のファイル[*3]を初めて操作したログ項目の時点を指す．

「トラブル発生」については，質問内容で示されたトラブルが起きた直接の原因となるコードが書き込まれた時点を，著者がログ閲覧機能を用いて目視で確認したものである．

また，「最後の進展」については，質問があった時刻よ

[*3] BitArrow のファイル配布機能により，各受講者の操作するファイル名は課題ごとに統一されている

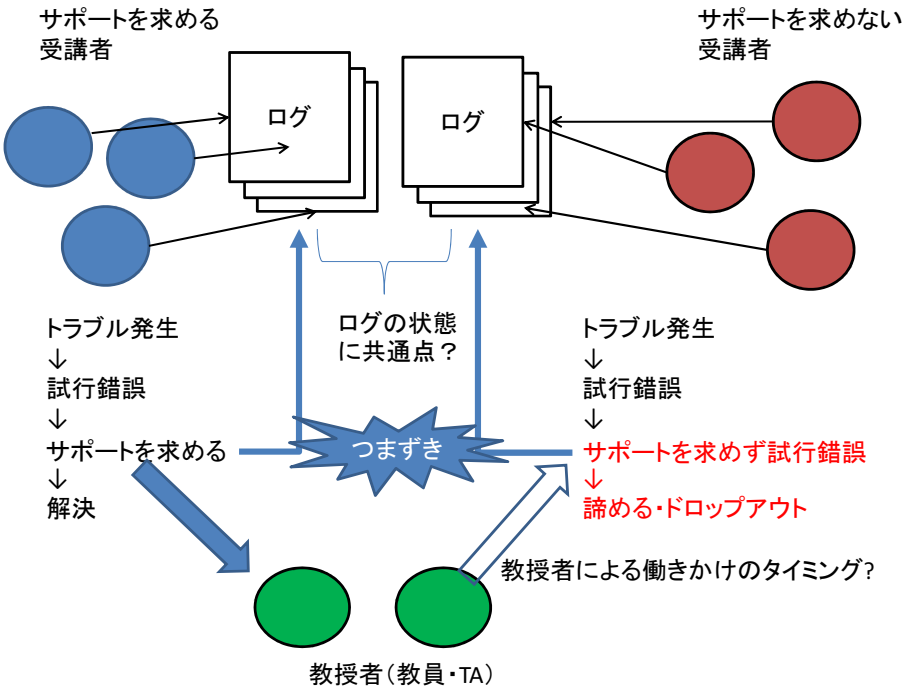

図 1　つまずきのログからの推定

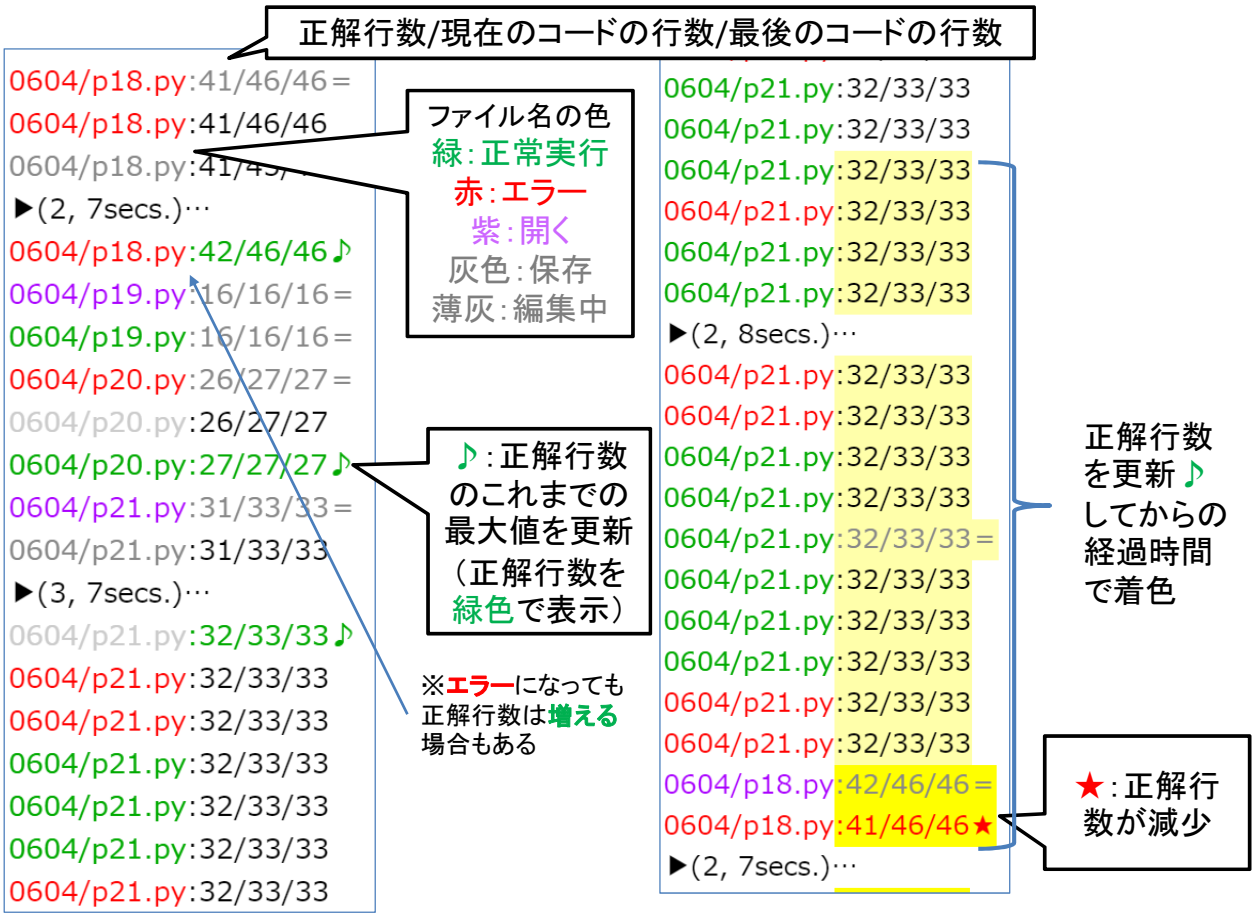

図 2　ログビューアのつまずき推定機能

り前で，もっとも新しい♪マークがついている当該課題のファイルのログ項目の時点を指す．そのような♪マークがない場合は「当該課題着手開始」と同じ時点とする．

ここで述べた「時点」には，2 つの尺度を用いた．1 つは通常の「時刻 (ログ項目の time フィールドの値)」，もう 1 つは「実働時間 (actual time)」である．これは，次のように計算される．

- 当該課題着手開始時点での実働時間は 0 とする．
- 同じファイルを対象とする (filename フィールドが等しい) ログ項目 (A と B) が連続している場合で，かつ A から B への経過時間 (d) が 5 分以内の場合，d を B の実働時間に足しこむ (B.actualTime=A.actualTime+d)．
- 前述した d が 5 分以上の場合，あるいは A と B の間に他のファイルのログ項目が挟み込まれている場合，B の実働時間は A のそれと等しいものとする．

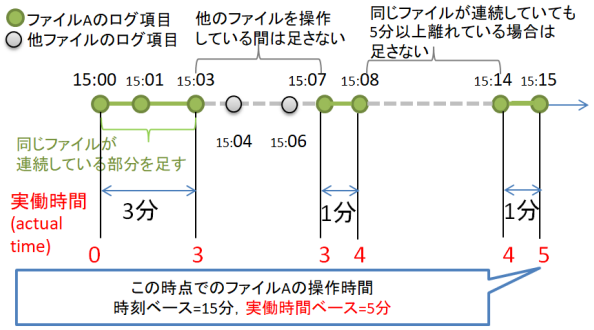

図 3　実働時間の計算方法
Fig. 3　Actual Time

計算方法を図 **3** に示す．実働時間は，他のファイルを操作していたり，操作が止まっていたりする間を数えない方式とみなすことができる．

4.3　対象とする授業

調査の対象となった授業は，明星大学　情報学部の 2 年生向けの「プログラミング演習 3」という授業である．言語には Python を用い，オブジェクト指向プログラミング (OOP) の基本的な概念を習得するのを目的としている．受講者はすでに Python の基本は 1 年生で学んでいる．受講者は約 40 人である．

授業内容を**表 1** に示す．演習教材は，「コンピュータとの対話」を重視した教材 [7] に準拠しており，1 回あたりおおむね 20 から 30 程度のプログラムが提示されるが，そのうち約半分は，完成しているプログラムを実行することによって，その回で学習する内容を習得していくものである (これを「演繹パート」と呼ぶ)．また，残り半分は空のプログラム，または一部が空欄になっているプログラムであり，プログラムを埋めることで指定された出力結果を得る演習を行う (これを「仮説形成パート」と呼ぶ)．本授業では，演繹パート，仮説形成パートを交互に繰り返すことによって，学習を進めていく．

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	OOP：クラス・フィールド・コンストラクタ
8	OOP：メソッド
9	OOP：オブジェクト同士の関係
10	OOP：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

本学は 2021 年度からは対面授業を再開し，本授業も対面で始めたものの，緊急事態宣言の影響などにより，ほどなくオンラインに戻る運びとなった．オンライン授業におけるサポートを充実させるため，Zoom での参加を義務化した．ただし，Zoom では授業に関する簡単なアナウンスを最初に行ったり，必要に応じてヒントなどを知らせたりする以外は使用せず，質問を受け付けるのはもっぱら Slack を使用した．質問がある受講者は，Slack のダイレクトメッセージ (DM) を使用し，教授者に質問を行い，サポートを受けた．

5.　調査結果と考察

Slack にて行われた質問それぞれについて，トラブル発生から質問までの経過時間および，「正解行数」から判別した最後の進展からの経過時間を調査した．**表 2** に，調査結果を示す．

- 「事例#」は，Slack への質問があった場合に付番している．付番の順序は時系列ではない．
- 「受講者#」は，質問をしてきた受講者の通し番号である．質問した回数が多い順に付番しており，学籍番号などとは無関係である[*4]．
- 「課題」は，Slack で質問された内容に関係するプログラムのファイル名である．授業を行った日付と，教材上に登場する順序でつけられている．
- 「最初-質問」は，「当該課題着手開始」から「つまずき」までの時間である．後述するグラフ中では「first」で表記する．
- 「発生-質問」は 「トラブル発生」から「つまずき」ま

[*4] BitArrow や Slack によるログ分析を行うことについては受講者から事前に許可をとっている

での時間である．グラフ中では「occur」で表記する．

- 「♪-質問」は 「最後の進展」から「つまずき」までの時間である．グラフ中では「adv」で表記する．
- t がついているものは「時刻」ベース，a がついているものは「実働時間」ベースで計算したものである．グラフ中ではそれぞれ「/t」と「/a」で表記する．

これらの値を時間の種別，尺度ごとに集計した結果を**図 4** に示す．また，横軸を受講者番号，縦軸を事例ごとの時間としてプロットした結果を**図 5**，**図 6** に示す．

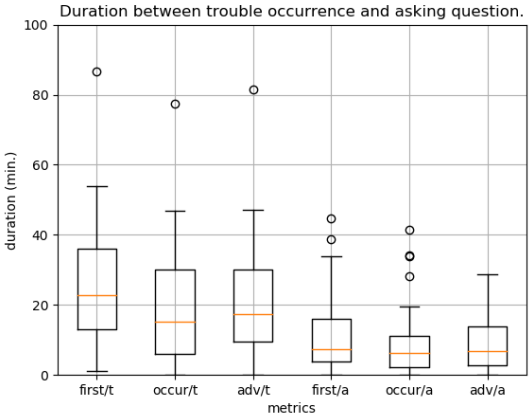

図 4 質問までの時間の分布

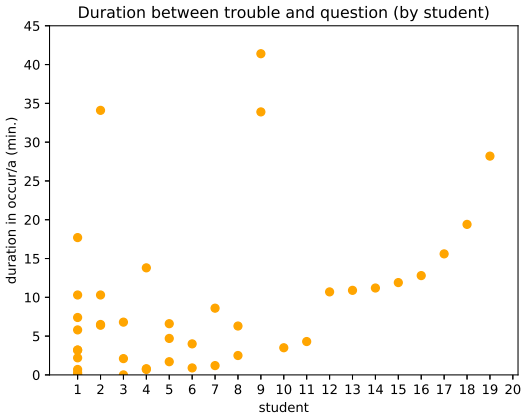

図 5 トラブル発生から質問までの時間（受講者別，実働時間）

集計結果から，「15 分ルール」の妥当性を検証する．

図 4 の「occur/t」の項目に着目すると，中央値が 15.2 となる．これはつまり「(目視で確認した）トラブル発生時点から，Slack に質問するまでの時間（時刻ベース）の中央値がおおむね 15 分である」ということを表しており，先述した「15 分ルール」をある程度支持する結果となった．

さらに，「adv/t」の中央値が 17.3，つまり「(ログから算出可能な）最後に♪が現れた時点から，質問するまでの時間の中央値がおおむね 17 分である」という結果や，図 4 における occur/t と adv/t および，occur/a と adv/a の分布

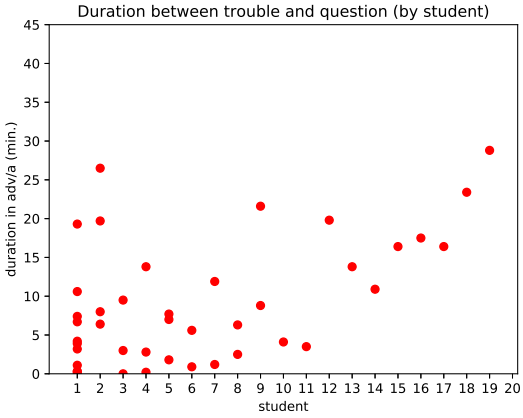

図 6 最後の進展から質問までの時間（受講者別，実働時間）

に注目すると，「15 分ルール」に基づいて「つまずき」を検出する際に，目視で確認した場合と，ログから機械的に算出した場合で大きな差がないことを示す．受講者別に見た図 5 と図 6 を比較しても，分布に大きな差がないことが確認できる．このことから，つまずきの状態をログの状態から自動的に推定することが可能であることが示唆された．

また，尺度については「時刻」と「実働時間」を用いたが，実働時間で計測した場合は全体的に時間がおおむね半分になり，トラブル発生から 7 分程度の実働時間で質問を行ってきている．この間には，他の課題を参照して参考にしている場合が多くみられた．また，一旦トラブルが発生している課題を飛ばして，先の課題に着手している場合もあった．

図 4 だけを見ていると，「時刻」で集計した値を単純に半分程度にすれば「実働時間」に近い分布になっているように見えるが，図 6 と，同じ散布図を時刻ベースでプロットした**図 7** とを比較すると，時刻ベースのほうが外れ値が多いことがわかる．これは，授業時間外などに作業を行っている場合に質問までの時間が長くなる（質問は原則常時受け付けているが，返答がすぐもらえる授業中に質問が集中する傾向にある）ためであると考えられる．このような外れ値を収斂させるためには実働時間での集計が有効であるといえる．

6. 今後の予定

考察から，「15 分ルール」はある程度妥当性がある（実際に 15 分その課題に集中しているわけではなく，以前の課題を見直す，先の課題を行っている，休憩している，など他のこともしているので，実質的な作業時間は約半分の 7 分程度）ことがわかったが，実際の分布を見れば明らかなように，15 分より短い事例，長い事例もたくさん発見された．

トラブル発生から質問までが短い事例を見ると，課題が示す文章の意味を取り違えていて，自分が正しいと思い込

表 2 トラブル発生から質問までの時間

事例#	受講者#	課題	t 最初-質問	t 発生-質問	t ♪-質問	a 最初-質問	a 発生-質問	a ♪-質問
46	1	0416/p19	16.8	15.8	16.8	1.1	0.0	1.1
47	1	0423/p26	1.7	0.7	0.0	1.7	0.7	0.0
2	1	0507/p11	28.8	28.4	28.8	10.6	10.3	10.6
6	1	0507/p21	32.0	31.1	32.0	6.7	5.8	6.7
8	1	0514/p32	6.2	5.8	5.8	0.7	0.3	0.3
21	1	0521/p05	13.3	11.4	13.3	4.2	2.2	4.2
17	1	0521/p08	30.6	30.6	30.6	7.4	7.4	7.4
24	1	0521/p11	32.2	30.6	32.2	19.3	17.7	19.3
16	1	0521/p20	25.9	6.8	6.8	4.3	3.2	3.2
26	1	0528/p15	13.1	0.0	9.8	1.0	0.0	0.3
51	1	0618/p09	23.0	21.7	22.4	4.5	3.2	3.9
12	2	0514/p32	39.8	35.3	27.7	38.6	34.1	26.5
33	2	0528/p14	30.9	24.0	30.9	8.0	6.5	8.0
35	2	0604/p18	22.9	7.7	21.1	21.2	6.4	19.7
52	2	0618/p21	12.9	12.0	7.9	10.6	10.3	6.4
1	3	0507/p13	14.6	11.2	12.3	10.0	6.6	7.7
10	3	0514/p19	4.0	2.6	4.0	1.8	1.7	1.8
31	3	0528/p06	10.9	14.6	10.6	7.2	4.7	7.0
28	4	0528/p06	17.8	17.8	17.8	13.8	13.8	13.8
29	4	0528/p11	5.7	3.6	5.7	2.8	0.7	2.8
30	4	0528/p20	22.2	2.3	1.7	1.6	0.8	0.2
23	5	0521/p17	22.5	22.5	22.5	0.0	0.0	0.0
27	5	0528/p18	10.5	4.3	9.4	3.8	2.1	3.0
39	5	0604/p14	9.9	7.2	9.9	9.5	6.8	9.5
3	6	0507/p28	210.5	210.5	17.4	33.9	33.9	8.8
7	6	0514/p03	53.8	46.9	23.3	44.8	41.4	21.6
25	7	0528/p20	50.9	1.2	1.2	2.6	0.9	0.9
38	7	0604/p04	23.4	17.0	23.4	5.6	4.0	5.6
22	8	0521/p19	1.2	1.2	1.2	1.2	1.2	1.2
40	8	0604/p18	20.0	8.7	18.6	12.8	8.6	11.9
15	9	0521/p19	9.5	9.5	9.5	6.3	6.3	6.3
36	9	0604/p18	18.5	5.0	15.7	4.6	2.5	2.5
4	10	0507/p31	44.6	40.3	41.4	16.7	15.6	16.4
42	11	0604/p18	41.5	27.7	32.3	21.1	12.8	17.5
34	12	0604/p20	38.7	38.0	38.7	4.1	3.5	4.1
13	13	0514/p03	138.7	19.1	103.0	30.2	11.9	16.4
49	14	0514/p35	47.1	30.7	47.1	19.8	10.7	19.8
44	15	0416/p09	15.1	12.1	15.1	13.8	10.9	13.8
5	16	0423/p28	37.3	34.7	35.3	29.2	28.2	28.8
50	17	0618/p12	86.5	77.5	81.5	28.4	19.4	23.4
48	18	0528/p11	29.0	17.6	17.3	14.2	11.2	10.9
14	19	0514/p28	6.1	3.7	2.9	6.7	4.3	3.5

んでいる結果と異なる結果が課題に提示されているため，課題の内容に間違いがないか確認したくて質問した，という例もあった．一方，長い事例においては，「授業開始前に他の課題を終わらせて，わからないものだけをじっくり考えている」「一応正解のプログラムが書けたが，よりよい書き方を追求している」という場合もあり，一概に早めのサポートが必要とは考えられないものもあった．

今回の調査では，「質問者は適切なタイミングで質問を行っている」という仮定をおいて調査を行ったが，質問までの時間が短いものの中には，適切に試行錯誤を行っておらず，単に「わかりません」という質問をしてくるものもあり，適切に試行錯誤した後のソースコードや実行結果を含めた質問をしているものと，そうでないものを分けて集計する必要があると考えらえる．また，質問までの時間が長い場合には，試行錯誤が適切でない（無闇にコードを書き換えているだけで，学習者にとって「時間の無駄」になっている）ものがなかったかどうか，なども精査しなければならない．

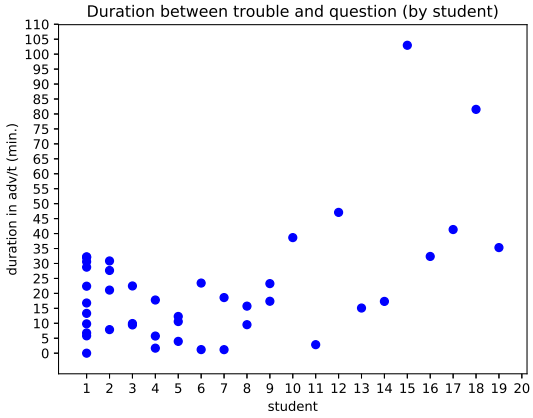

図 7 最後の進展から質問までの時間（受講者別，時刻）

本研究の最終的な目的は，まったく質問をしてこない学習者へのサポートをどう行うか，であるため，まずはそちらの調査を優先して行っていく．実際，約 40 人の受講者中，表 2 によれば質問をしているユニークな受講者は 19 人であるため，約半数は質問を一切していないことになる．これらのユーザについては，ログを分析し，「最後の進展」から「15 分」を目安に質問を行うべきタイミングになっていないか，について調べる予定である．

つまずきの検出を自動化するにはまだいくつか障壁がある．「最後の進展」を判定するための「正解行数」は，当該ファイルの最後のプログラムが「正解」である，という前提に立っているため，授業中にリアルタイムな分析を行うことが難しい．これについては，今回の調査で得られた「つまずき」のタイミングで，ログに現れる共通の性質を抽出し，機械学習などの手法も取り入れて，「正解行数」以外の尺度で「つまずき」を検出することを検討していく．

7. まとめ

プログラミング学習の過程において学習者がトラブルに陥った場合であっても，教授者にサポートを求めないまま，解答をあきらめてしまう場合がある．そのような学習者に対しては，適切なタイミングで教授者側からサポートを行う必要がある．そのような状態に陥っている受講者を発見するための予備調査として，「BitArrow」のログの収集機能を用いて集めたログと，自らサポートを求めてきた学習者の Slack によるサポートの履歴を突き合わせて，トラブル発生から，質問を行ってくるまでの時間を調査した．

その結果，プログラム内で何らかの進展があってから，おおむね 15 分程度何も進展がない状態が続くと，サポートを求めてくる（つまり，「つまずいている」）という傾向を掴むことができた．また，「何らかの進展」があるかどうかは，ログを事後に解析することで自動的に検出できることも示唆された．

今後，つまずいている状態とその時点でのログの状態を分析し，将来的にはサポートが必要なタイミングをリアルタイムに，自動的に通知できるシステムの確立を目指していく．

謝辞 本研究は JSPS 科研費 19K03153 の助成を受けたものです。

参考文献

[1] 長島和平，本多佑希，長 慎也，間辺広樹，兼宗 進，並木美太郎：オンラインで複数言語を扱うことができるプログラミング授業支援環境，情報教育シンポジウム 2016 論文集，Vol. 2016, pp. 1–9 (2016).
[2] 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).
[3] 井垣 宏，齊藤 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).
[4] 加藤利康，石川 孝：プログラミング演習のための授業支援システムにおける学習状況把握機能の実現，情報処理学会論文誌，Vol. 55, No. 8, pp. 1918–1930 (2014).
[5] 市村 哲，梶並知記，平野洋行：プログラミング演習授業における学習状況把握支援の試み，情報処理学会論文誌，Vol. 54, No. 12, pp. 2518–2527 (2013).
[6] 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，技術報告 4，明星大学, 東京農工大学, 東京農工大学, 大阪電気通信大学, 明星大学 (2020).
[7] 長 慎也，山中脩也，北島茂樹，今野貴之：情報系初年次のプログラミング演習における，コンピュータとの対話を重視したコースデザインと支援システム，技術報告 20，明星大学, 明星大学, 明星大学, 明星大学 (2019).