Vol.2019-HCI-184 No.14
2019/7/23

プログラミング教育円滑化のための抽象的思考記述システム

又吉康綱[†1]　中村聡史[†1]

概要：小学校から高校においてプログラミング教育が必修化されるなどプログラミングを学ぶ機会が増えつつある．プログラミングの学習は，課題を解決できるようなプログラムを作成させることが一般的であるが，作成できずに挫折する初学者も多くいる．これは，初学者がプログラムの構造に関連する抽象的思考を意識できず，その内容を初学者と教育者の間で共有できていないことが原因であると考えられる．そこで本研究では，プログラムのソースコードとそれに対するコメントを同時に記述し，コメントの配置を入れ替えるだけで直感的にプログラムの構造を試行錯誤可能にする手法を提案する．これにより，初学者は整理しながら手軽に試行錯誤を繰り返せるようになり，教育者は初学者の理解度や意図の把握が容易になることが期待される．実験により，提案手法を用いることで初学者の構造理解を深めることと，教育者の指導への活かすことが可能だと示唆された．

キーワード：プログラミング教育，初学者，抽象的思考

1. はじめに

2020 年度より小学校でのプログラミング教育が必修化されるなど，プログラミングを学ぶ機会が増えつつある．また，旺文社・教育情報センターによる「2018 年度日本の大学データ」[1]によると，大学において主にプログラミングを学ぶ「情報工学」の分野の学科数は 291 学科存在しており，理系の学科の中では最も多い学科となっている．プログラミングを学ぶ人が増える一方で，大学におけるプログラミング教育には多くの課題が存在する．

著者らが所属する明治大学総合数理学部先端メディアサイエンス学科では，1 年次の必修講義としてプログラミングがあるため，100 人以上の学生が一度にプログラミングの講義を履修している．こうした状況において，教師がひとりひとりの理解度に合わせて授業を行うことは不可能である．そのため一般には，TA（ティーチング・アシスタント）が講義のサポートをすることになるが，TA も人数が限られているため 1 人で複数の学生を受け持つ必要がある．また，質問する学生自身が，何に悩んでいるかを簡潔に説明できないことも多く，TA は質問を受ける度にソースコードを読み，どこでつまずいているのかを理解する必要がある（図 1）．しかし，学生のソースコードは，変数宣言や変数の初期値，記述順番や関数のスコープ，インデントや関数の使い方が不適切であるなど状態が複雑に絡み合ううえ，1 つの課題についても多様な解法があるため，ソースコードを読解することは容易ではない．これらのことは，TA にとって大きな負担となっている．また，学生の中には，学んだことを理解しないままにすることで，つまずきを引きずる人も多い．その結果，分からないことが積み重なり，TA に具体的な質問をすることもできず，プログラミングに対して強い苦手意識を感じて挫折してしまう．以上のように，プログラミング教育においては，TA が学生のソースコードを読む負担と，学生がプログラミングに対して苦手意識を感じるという両者の問題が存在する．

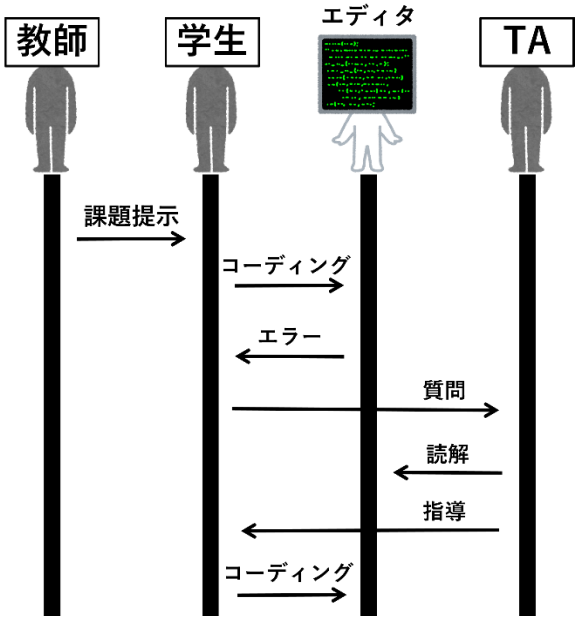

図 1 大学におけるプログラミング教育のフロー

学生のプログラミング能力上達において重要なことは，煩雑な処理を分解し，抽象化や一般化を行う思考を鍛えることであり，抽象的思考[2]と呼ばれている．必修化される小学校のプログラミング教育においても，プログラミングをする前段階の思考力として，抽象的思考をプログラミング的思考[3]と呼び，教育の柱として身につけることを目的としている．

本研究では，プログラミング教育における問題の解決としてプログラミング作成時における抽象的思考に着目する．一般に，TA はプログラミング能力に優れ，抽象的思考が鍛えられている人物が多い．そのため，学生の抽象的思考の内容を直接共有することができれば，ソースコードを読む負担を軽減することが可能である．また学生は，抽象的思

†1 明治大学
Meiji University.

考を意識しながらプログラムを書くことで，自身の理解について確認しながら鍛えることが可能になる．

そこで本研究では，プログラムの抽象化とその抽象化をベースとした TA による学生の指導を可能とするため，プログラムの各処理をツリーのノードとして明示し，そのノード操作とプログラムを連動することによって抽象化しながらプログラミング可能な仕組みを実現する．ここでは，ユーザ（学生）がツリーに追加したノードがプログラム内に構造を考慮したコメントとして生成され，そのコメントに関連した部分に処理を書き込んでいくことで，１つのプログラムを完成させることができる. TA はそのノードを見ることで学生の意図を即座に把握することができ，学生も抽象化によって自身のやりたいことを整理することが可能となる．また，抽象化されたノードの移動などを指導することで，質問に対して効率良く回答ができるようになると期待される．さらに，プログラム言語に依存するエラーと切り離して抽象的思考の内容を直接教授できるようになり，より高度な指導が可能になると考えられる．本研究ではプロトタイプシステムを実装し，評価実験を実施することでシステムの有用性について検討する．

2. 関連研究

2.1 プログラミング授業支援

大学における教師や TA が学生に対してプログラミング授業を行う際の支援に関する研究も多く存在する．Durrheim ら[4]は，教師の正解のプログラムと学生のプログラムとの違う点を行単位で指摘することで，正解へと誘導するシステムを提案している．また，安田ら[5]は，TA が学生に教えている状況を録画し，学生と TA 間で共有することができるプログラミング演習支援システムを開発している．同様に TA の効率化が可能な研究として，Elice[6]は，プログラミングの手順を 6 ステップに分けて，提案システム上でステップの進捗度を管理することによって, TA の補助が必要な学生を発見できることから TA の効率化が可能になる手法を提案している．一方，西田ら[7]は，プログラミングの基礎を短時間で習得することを目指し，日本語表現が可能な xDNCL を記述するプログラミング学習環境 PEN を開発している．

これらのシステムは，本研究と類似した目的をもっているが，授業中に構造を考え理解させつつ，学生からの質問の受け答えを容易にするように設計されていない．

2.2 プログラミングの構造の理解に関する研究

初学者の学習支援となるプログラミング環境の提案も多く存在する. Patrice[8]は, ソースコードを書くことなく，検索や並び替えに関するアルゴリズムに特化して，設計し実行する AlgoTouch を開発している．ブロックとして実行できる研究も存在し，Kayama ら[9]は，プログラムの構造を変数宣言ブロック，計算ブロック，条件分岐ブロック，繰り返しブロックなどの構造ブロックとして扱い，入れ子をもたせながらアルゴリズムを記述出来るシステムを提案している．さらに，今泉ら[10]は，学生にプログラム構造の動作過程を理解させることを目的とし，ソースコードの構造と動作過程を可視化し，対応させるシステムを開発している．提案システムと同様に自然言語でアルゴリズムを記述できる AlgoWeb[11]は，ポルトガル語をベースとしたプログラム記述言語でプログラムの構造を記述することができ，アルゴリズムの実行，演習をすることができる．

これらの提案は，プログラムの構造の可視化や自然言語による構造の記述により，初学者がプログラムの構造を考え，理解することを支援している．しかし実際のプログラミングの前に実施することを前提にしており，プログラムを記述しながら構造を考えることが考慮されていない．本研究ではプログラミングの環境に併存する形でこれらの構造を記述できるシステムを考える．

これらの研究のようにプログラミングの授業や学習を支援する研究は多く存在するが，初学者の抽象的思考とプログラミングを同時に支援することには成功していない．そこで，本研究では，この抽象的思考とプログラミングを同時に支援することによって初学者と教育者の双方が恩恵を受ける手法を目指す．

3. 提案手法

3.1 抽象的思考とその記述

プログラミング教育において，初学者が初めに習うことは，プログラムの基本的な変数や関数の利用方法や単純な四則演算などであり難易度は高くないが, 学習を進めると，条件分岐や配列，繰り返し，関数やクラスの作成といったように難易度が上がっていく．そして学習が進むと，プログラム全体の完成図を意識しながら考える必要が生じる．つまり，変数の流れやクラスの構造，分解，記述の効率化のための再利用について考えながら記述しなければならない．これらの思考方法が，抽象的思考[4]と呼ばれる．プログラミング学習においては，応用問題や新たな単元の問題を解く際に抽象的思考が要求されることが多いため，これを鍛えることが重要である．

抽象的思考のための記述として，フローチャートが考えられる．ここでフローチャートは全体像のモデリングや設計を行うことで構造や仕様を確定させることで，バグがない品質の高いプログラムが書けるようなるが，これらの方法を初学者に対して実践させるには独特な記法を覚える必要があるため難しく，現実的ではない．一方，プログラムのソースコードに直接記述できるコメントは処理を簡潔に記述することが可能な手法であり，日常的に使う自然言語によって記述できるため初学者も簡単に利用することがで

きる．しかし，これらの記述を手間だと感じることや，記述の仕方には厳密な仕様が決められているわけではないため，構造を示すためには不十分なコメントも多いという問題がある．

また，簡潔にプログラムの構造を表す記法として擬似コードが知られている．擬似コードとは，自然言語とプログラムのコードを交えて記述するものであり，アルゴリズムを表現するために使われる．例えば，図 2 は FizzBuzz の処理を擬似コードによって記述したものである．この疑似コードは基本情報技術者試験においても使用されている．しかし，疑似コードは実際のプログラミング環境で実行出来ないため，ソースコードとは別に用意する必要があり，挙動の確認ができずに設計のミスが起こる問題が予想される．そのため，初学者が擬似コードと実際のソースコードの両方を記述しつつプログラミングを行うことは困難であると考えられる．

以上より，プログラム構造を考えるための抽象的思考を記述することにおいて，ソースコードの内部でコメントをすることと，擬似コードのように自然言語とソースコードの構造とを対応させて記述できることが重要であると考える．そこで本研究では，これらを実現するための手法として，自然言語によって記述するコメントをノードとし，実際のソースコードと同じ構造をツリーで表示を行うコードツリー手法を提案する．

3.2 コードツリー手法

本研究で提案するコードツリー手法は，ツリー上のノードと，ソースコード中のコメントを結びつけ，プログラムの構造をツリー状で表示することで，考えやすく，ノードの並び替えや追加の操作を可能とする手法である．この手法により，自身および他者の抽象的思考を俯瞰することができソースコードの理解や試行錯誤を深めるとともに，TA とのコミュニケーションを円滑化することが可能になる．

コードツリーによる表現のイメージを図 3 に示す．図 3 では，プログラムの各処理について，その内容を示すコメントがノードとして存在しており，ソースコード側の 3 行目から 7 行目の条件分岐は，コードツリーのノードにおける「もし x が 3 と 5 の倍数なら」と記述されたノードが対応している．また，プログラムの処理には入れ子，つまり親子関係が存在しており，「Fizz を表示する」のノードは，子ノードであり「そうでなく，もし x が 3 の倍数なら」が親ノードであると言える．この入れ子はプログラムの構造において特に重要であり，これらの関係を視覚的に捉えることが抽象的思考の理解に繋がると考えられる．

以上よりこのコードツリー手法をソースコードのエディタと同時に扱えるようにすることで，プログラミングの妨げにならずに，入れ子を意識しつつ，ノードの入れ替えや親子関係の編集をすることで，構造についての理解が深まることが期待される．また，教育者にとっても，コードツリーのノード同士の関係を見ることによって，初学者がどこまでプログラムの構造を認識しているのかを確認することが可能になり，ノードベースの入れ替え指示も行えることで従来よりも円滑に学生とのコミュニケーションが可能になると考えられる．

```
x を 1 から 100 まで 1 ずつ増やしながら，
    もし x%3=0 かつ x%5=0 ならば
        "FizzBuzz"を表示する
    を実行し，そうでなくもし x%3=0 ならば
        "Fizz"を表示する
    を実行し，そうでなくもし x%5=0 ならば
        "Buzz"を表示する
    を実行し，そうでなければ
        x を表示する
    を実行する
を繰り返す
```

図 2　擬似コードによる FizzBuzz の例

図 3　コードツリーとソースコードの対応関係

4. プロトタイプシステム

3 章で説明したコードツリー手法を利用したプロトタイプシステムを実装する．著者の所属する学科や多数の大学の初年次プログラミング講義で採用されている Processing[12]を採用し記述・実行するシステムを構築する．

4.1 実装

プロトタイプシステムを Web アプリケーションとして実装した．実装には JavaScript, MySQL を用いた．

クライアントサイドは JavaScript で実装しており，抽象的思考を記述可能な入れ子を持ったコメントやコードを入

表 1　プロトタイプシステムに関する学生アンケート結果

	質問事項	評価値の分布					平均値
		-2	-1	0	1	2	
Q1	提案システムが使いやすい	0	4	1	2	2	0.2
Q2	提案システムの方が課題を解きやすい	2	2	2	3	0	-0.3
Q3	提案システムで課題に集中してスムーズに取り組めた	1	3	1	3	1	0.0
Q4	提案システムの方が課題の理解が深められた	0	1	4	3	1	0.4
Q5	提案システムの方が課題を早く解けた	2	2	4	0	1	-0.4
Q6	提案システムを今後も使いたい	1	2	4	1	1	-0.1
Q7	提案システムを継続利用することでプログラミング力を向上できると思う	0	1	1	5	2	0.9

力するエディタ部分で，ソースコードの編集や実行などを行う．サーバーサイドは Node.js と MySQL で構築し，ユーザが入力したソースコードの保存や呼び出し，入力されたソースコードの構文チェックなどを行う．

4.2 システムの利用方法

システムの左側にプログラムの構造を示すコードツリー，中央にソースコードを書くエディタ，右側に Processing 実行画面，下部にコンソールが表示される．上部には，実行，停止，コード補完，コードツリー有効無効，保存，録画ボタンがある（図 4）．コードツリーとエディタは常に同期しており，一方を書き換えるともう一方も自動的に更新される仕様になっている．

コードツリーでは，ノードのテキストを書き換えることができると同時に，ノードを入れ替えることや，入れ子を深くしたり，改行することで新たなノードを追加したりすることが可能である．エディタからは，ソースコードを直接編集が可能である．ただし，「開きカッコの数と閉じカッコの数が一致しない場合」や「コメントが不足している場合」などのコードツリーが成立しない状態では，コードツリーの機能が停止しノードの入れ替えなどが出来なくなる．これを解消するためには，表示されるエラーメッセージをもとにエディタから原因の解決を行う必要がある．

5. 評価実験

初学者であるプログラミングを学び始めた学生がシステムを利用してプログラミングすることによって，学習の支援になるかどうかを明らかにする．また，教育者側の TA が学生からの質問に対して，システムを利用することによって思考や意図を理解しやすくなるかを検証する．

5.1 実験手順

評価実験は，大学 1 年の学生 9 名に対して行った．これらの学生は大学からプログラミングを学び始めた者が多く，プログラミング学習経験が 1 年未満である．そのため，本研究が対象とするプログラミングの構造について十分に理解しきれていない者が多いと考えられる．また，TA として

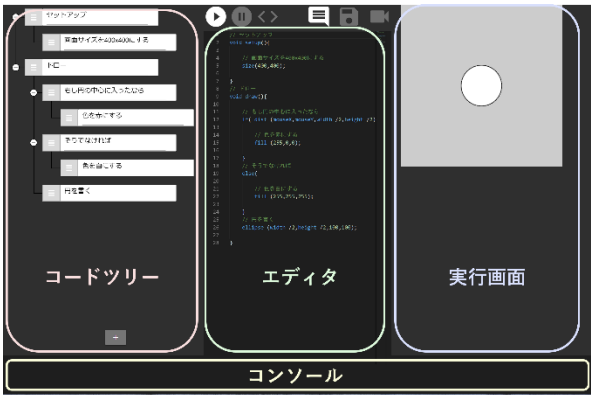

図 4 システムスクリーンショット

大学院生 3 名が評価実験に参加した．

評価実験では，2 種類の問題を解いてもらった．プロトタイプシステムのコードツリーを無効と有効にした状態それぞれでプログラミングをしてもらった．また，どちらの問題に対してもプロトタイプシステムを利用した状況にするため，学生をグループ 1（4 名），グループ 2（5 名）の 2 グループに分け，別々の問題でプロトタイプシステムを利用してもらった．問題を解くのに 1 問あたり 30 分の制限時間を設けた．さらに，システムへの慣れが必要と判断し，事前にプロトタイプシステムを利用したチュートリアル問題を解いてもらった．

なお，先にプログラムの構造を考えた後に対応するコードを書くことが理解の促進に繋がると考えているため，システムの利用においては，なるべくコードツリーのコメントを書いてからコードを書くように指示した．2 問終了後に，5 段階評価形式のアンケートと自由記述形式のアンケートに回答してもらった．また，実験中はプログラミングを行っている画面の録画を行い，著者が内容やプログラミングについて確認を行った．

5.2 結果

5.2.1 学生による評価

学生に行ったプロトタイプシステムに関するアンケート調査（Q1 から Q7）の結果を表 1 に示す．また，5 段階（-2：まったくそう思わない～2：とてもそう思う）で評価

を，その分布と平均値も示す．

既存手法と提案手法とを比較した表 1 の Q1 から Q7 の結果より，継続利用に関する Q7 では，平均値 0.9 と高い結果が得られたが，課題の解きやすさに関する Q2 と Q5 では，平均値が負の値になってしまい，提案手法よりも既存手法が良いとの結果になった．

ここで既存手法と提案手法の比較について，コードツリーの使用頻度が高いグループ(4 名)と低いグループ(5 名)に筆者の主観で学生を分け，質問項目ごとの平均値を出した結果を表 2 に示す．その結果，コードツリーの使用頻度が高いグループの方が使用頻度の低いグループよりも全体的に平均値が高い値になった．

表 2　コードツリーの使用頻度ごとのグループと各アンケート結果の平均値

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
使用頻度高	1.50	0.50	0.75	0.75	0.50	0.50	1.25
使用頻度低	-0.80	-1.00	-0.60	0.20	-1.20	-0.60	0.60

学生からの自由記述として，「コメントを書きやすくどうしたら良いか解りやすかった」，「入れ替えなど直感的に操作ができて楽だった」など肯定的な意見を多くもらった．一方で，「プログラムが長くなりがちになる」，「長くなるとソースコードとコメントとの対応が取りづらい」，「もう少し自由に書きたい」など改善点やシステムの UI に対する提案も多くえられた．

5.2.2 TA による評価

学生からの質問に対応した TA に対して，システム状況下での使用感に関するアンケート調査を表 3 に結果を示す．なお，5 段階（-2：まったくそう思わない～2：とてもそう思う）で評価を行った．その結果，ミスの発見に関する Q8 のみ平均値が負の値を示したが，意図の汲み取りに関する Q9 と講義での利用に関する Q10 は正の値になった．

表 3 システムに関する TA へのアンケート回答

質問項目		平均値
Q8	提案システムの方が学生の間違っている箇所を発見しやすい	-0.3
Q9	提案システムの方が学生の質問の意図を汲み取りやすい	0.7
Q10	提案システムを実際の講義で利用したい	1.0

TA からの自由記述として，「構文に悩んでいるのか，問題の理解に悩んでいるのか判断しやすかった」，「コードのミスチェックがしやすかった」，「バグが見つけやすかった」などの好意的な意見があった．一方で，「ソースコードの方を結局は見た」，「システムの仕様を理解しないと TA がコードツリーを直せない」，「質問が少なかったので判断できない」といった否定的な意見や反省点が得られた．

5.3 考察

5.3.1 学生アンケート結果からの考察

既存手法と提案手法との比較において，平均値が正の値となった課題の理解に関する Q4 と継続利用に関する Q7 については，提案手法のプログラミング構造の直感的な操作が可能になった影響により，既存手法よりも良い結果となったと思われる．一方で，平均値がゼロ未満の値になった課題の解きやすさに関する Q2 と Q3，Q6 は評価値の分布が広がっており，個人差があると考えられる．課題の解く早さに関する Q5 に関しては，提案手法の都合上，コメントを書くのに時間をかけるため，負の値になったと考えられる．

さらに，学生をコードツリー使用頻度によってグループに分けてアンケートを比較した結果，使用頻度が高い学生ほど，全体的に高い評価をしており，システムを好意的に感じていることが分かる．このことから，提案手法による初学者のプログラミング学習の支援が可能だと言える．

5.3.2 TA アンケート結果からの考察

TA のアンケート評価 Q8 が低かった原因として，自由記述にもあるように，プログラミングが得意な学生が多く，TA へのプログラミングに関する質問が多く出なかったことが原因だと考えられる．また，学生からの質問の中に，エラーの理由をシステムが出力できないものがあり，TA が問題解決できずに困ったからだと考えられる．また，意図の汲み取りに関する Q9 と講義での利用に関する Q10 が高かった理由としては，コードツリーを用いたことにより，プログラムの構造を自然言語で把握することが可能だったからだと思われる．このことより，プロトタイプシステムのコードツリーが優位にはたらいたことが示唆された．

5.3.3 実験時の動画による観察からの考察

実験中の学生がどのようにシステムを使ってプログラミングをしているかを画面の録画をした動画を元に観察した結果，システムの使い方に対して，コメント先行記述タイプとソースコード先行記述タイプの 2 つのタイプに分けられた．

コメント先行記述タイプでは，プログラムの流れを日本語でコードツリーやコメントから入力した後にコードを書いていた．しかし，ソースコード先行記述タイプでは，コメントを書かないことで起こるコードツリーの停止に悩まされたり，実験終了直前に全てのコメントを書いていたりした．本実験では，なるべくコメントを先に書くように指示を行ったが，コメント先行記述タイプ 4 名，ソースコード先行記述タイプ 5 名に分けることができた．なお，これはコードツリーの使用頻度のグループ構成と一致し，使用頻度が高い学生はコメント先行記述タイプだった．

観察より，コメント先行記述タイプは，コードツリーを用いてコメントから書くことによって，プログラムの構造を考えることができていた．学生の中には，詳細なコメントを記述しており，自分のソースコードのミスに気づいて修正する動作が見られた．また，ノードの入れ替えをよく行っていた．

一方で，ソースコード先行記述タイプは， Q1 の評価値をゼロ以下としており，表 1 の既存手法と提案手法を比較した質問項目も低く評価していた．これは，エディタでソースコードを書いてからコメントを書いたことにより，コメント不足が原因でコードツリーが停止し，解決することが出来なくなったからだと考えられる．また，コードツリーを一切使わずに正解に近づく学生もいた．

以上のことから，ソースコード先行タイプはシステムを煩わしく感じるが，システムに頼る必要がなく自分でプログラムの構造を考え，記述できるだけのスキルを既に身に付けていると言える．そのため，提案手法が対象とする初学者ではないと言える．

5.4 評価実験のまとめ

プロトタイプシステムを利用してプログラミングを学び始めた学生がプログラムの構造について直感的に扱うことが可能となり，理解を深めながらプログラミングが可能かどうかに加え，教育者である TA が学生の理解度や思考の意図を把握しやすいかを検証するために評価実験を行った．その結果，システム上でのプログラミングの書き方に対して，コメント先行記述タイプとソースコード先行記述タイプがあり，前者に対してプログラミングの抽象的思考の支援を与えることが可能であった．一方，後者は，システムを用いなくても独自の抽象的思考のスキルを身につけているため，システムの対象とする初学者ではなかった．

また，TA からのアンケートより，学生の構造に関する理解度や思考の意図をスムーズに読み取ることができる可能性が示唆された．しかし，学生から TA に対する質問が少なく，システムが指導のコミュニケーションにどのような影響を及ぼしたのか具体的な内容が明らかになっていない．そこで 6 章では，TA と学生間のコミュニケーションに着目した追加実験を行うことで TA に与えるシステムの影響を調査する．

6. 追加実験

評価実験より提案手法を用いたプロトタイプシステムが学生の抽象的思考の支援を行うことが可能であることが明らかになった一方で，教育者である TA にとってどうのように影響を与えるかが不明確である．そこで，追加実験では，学生と TA とのコミュニケーションがどのようになるのかを観察することで TA に与える影響を明らかにする．

6.1 実験手順

追加実験は大学 1 年の学生 4 名に対して行った．また，教育者側の TA として大学院生 2 名が参加した．追加実験は，システムの動作や挙動に関する質問があがらないよう慣れるために長期間実施した．具体的には，2 日間に分け 4 問を解いてもらった．4 問全問において，プロトタイプシステムのコードツリーを有効にした状態にし，1 問あたり 40 分の制限時間を設けた．また，学生と TA とのコミュニケーション回数を増加させるために，問題の難易度をあげるとともに，キー入力を 1 分放置した場合や制限時間 15 分を切った場合には，TA から学生に悩んでいる点を聞くようにした．なお，制限時間内に完成させ，コードツリーのコメントを書いてからコードを書くように指示した．4 問終了後に学生と TA に自由記述形式のアンケートに回答してもらった．実験中はシステムを使用している画面の録画と，部屋全体の録画を行い，著者らが学生と TA とのコミュニケーションについて確認を行った．

6.2 質問の分類結果

学生と TA とのコミュニケーションについて，筆者の主観により質問の分類を行った．質問の分類は，プログラムの問題が解けないことによって発生する質問として，「解法に関する質問」，「プログラムエラーに関する質問」，「問題の趣旨を理解していない質問」の 3 種類に分類した．また，その他にも「問題の仕様に関する質問」および「システムの仕様に関する質問」とし，合計 5 種類の分類を行った．実験によって得られたそれぞれの質問件数を表 4 に示す．

表 4 分類ごとの質問件数

質問分類	質問件数
解法に関する質問	11
プログラムエラーに関する質問	7
問題の趣旨を理解していない質問	0
問題の仕様に関する質問	3
システムの仕様に関する質問	4

解法に関する質問とは，問題の仕様を満たすような挙動を作るためにはどのようにプログラムを書いたら良いのか分からないことや，思い通りの結果にならない場合行われた質問を指す．実験では，分類の中では最も多くの質問が上がった．

プログラムエラーに関する質問とは，プログラムの構文エラーやコンパイルエラーなどのプログラムのルールを理解していない場合に行われた質問を指す．実験では，スコープ外での変数宣言によるコンパイルエラーや変数名のタイプミスによるエラーなどがあった．

問題の趣旨を理解していない質問とは，学生が問題やソースコードの内容について全く理解が及ばず，問題解決に

向けて方針を立てることが困難で具体性のない質問を指す．これらは学生が全く手を出せない状態にあり，このような場面にシステムが優位にはたらくと期待されていた．しかし，今回は，既に学生がプログラミングの授業を履修しており，ある程度プログラミングに対する知識や経験があったため，これらの質問はされなかった．

問題の仕様に関する質問とは，問題文では不明確なプログラムの挙動に関する質問を指し，TA が学生のソースコードを読むことはなく，挙動や口頭でのみ説明する質問を指す．実験では，クリックの挙動に関する質問が上がった．

システムの仕様に関する質問とは，システム上のコードツリーが停止した原因を解決出来ない場合やシステムのバグなどが原因で解決を求める質問を指す．実験では，入れ子の数が合わない状態などがあった．

6.3 アンケート結果

まず学生からのアンケートでは，5 章で得られたような「プログラムの順番をまとめて入れ替えることができて良かった」や「ひとつひとつの動作を言葉にして確認できるので理解が深まった」など，システムにより支援された内容が多かった．TA からの指導に関して，システムにより支援されたというような記述は得られなかった．

一方，TA からのアンケートでは，「個人差の可能性もあるが，実際の授業時よりも学生の質問がまとまっており答えやすかった」や「学生にコメントを書く習慣が出来ていそう」と肯定的な意見があった一方で，「特定の計算や処理ができないことによる質問では，ソースコードの全体を見る必要がないためコードツリーをあまり見なかった」や「コードツリーのノードの多さに抵抗感を感じた」などのシステムの問題点に関する意見をもらった．

6.4 考察

分類した質問の中で，「解法に関する質問」が最も多かった．これは，問題を理解して分かるところまで進めたが，思うような挙動にならない状態がプログラミング中には頻繁に発生するからである．本研究ではこのような，プログラミング問題の解き方について TA と学生が円滑にコミュニケーションを行えることを目的としている．今回の実験ではこれらの質問に対して，コードツリーを用いながら指導している場面は存在しなかった．学生と TA とのやりとりでは，学生が質問を聞く際にエディタをポインティングしており TA の視線がエディタに向いており，システムを用いてない状態と同様な，普段どおりの指導を行っていたと考えられる．そのため，TA がコードツリーを用いて教えることに対する慣れや指導が必要だったと考えられる．

また，「問題の趣旨を理解していない質問」も本研究で支援するべき質問であったが，実験においては質問がされなかった．今回の実験では既に全員がプログラミングの授業を履修しているため当該質問が出なかったと考えられる．

一方で，「プログラムエラーに関する質問」は，プログラミング言語の構文エラーなどによって発生する質問であるため，本研究で解決したい質問とは異なる．また，「問題の仕様に関する質問」と「システムの仕様に関する質問」についても本研究の目的とする抽象的思考の可視化が影響を及ぼす質問分類ではない．しかし，実験において，これらの質問が存在したことは，問題について学生に伝わりづらい部分があったことや，システムに慣れるためのチュートリアルが不足していたためだと考える．

アンケート結果より，学生は TA への質問のコミュニケーションに対してシステムに支援されている印象を持っていなかったが，TA は，個人差と思いながらも質問の内容がまとまっていたと感じていた．これは，5 章で明らかにした学生の思考が支援された事で，学生自身が的確に疑問点をまとめられるようになったとも考えられる．

TA への質問の最中にコードツリーが使われなかった原因として，実験協力者が既にプログラミングの授業を 1 年間履修した状態であり，全体を見る必要がある質問がされなかったからだと考えられる．さらに，問題を難しく設定し，厳しい制限時間を設けたため，残り時間が少なくなると情報量が乏しい最低限のコメントを書くようになっていた．この状態で TA がコードツリーを見てもプログラム全体を理解することは不可能であることから，コードツリーが質問で使われなかったと考えられる．

実験より，プロトタイプシステムを用いることで TA が学生からの質問に対して理解しやすく，指導する際に教えやすいということが期待されたが，学生がプログラミングの基礎を履修した状態であったため，問題の趣旨を理解していないような質問がされず，明らかにすることは出来なかった．そこで，7 章ではプログラミングを習いたての初学者に使ってもらってもらい，システムがどのような影響を及ぼすのかを調査するためにユーザースタディを行う．

7. 初学者でのユーザースタディ

追加実験より，プログラミングを習いたての初学者に対してシステムがどのような影響を及ぼすのかを検証するためにユーザースタディを行った．

7.1 手順

プログラミングを習い始めて 3 ヶ月の大学 1 年生 12 名に対して行った．TA として大学院生 2 名が参加した．

プログラミングの授業で習い終えた範囲（条件分岐）までの内容に関する問題を 2 問解いてもらった．追加実験と同様に，チュートリアルを行いシステムに慣れてもらった後に実施し，終了後に自由記述式のアンケートに回答してもらった．実験中はシステムを使用している画面の録画と，部屋全体の録画を行い，学生と TA とのコミュニケーションについて確認を行った．なお，TA にはコードツリーを使って学生の質問にできる限り答えるように依頼した．

7.2 アンケート結果

学生からは，「横にノートでメモ書きしている感覚でやりやすい」や「先に日本語で記すので勉強に向いている」，「コメントの順番を変えるだけでプログラムも順番変わるのが簡単で良い」などの肯定的な意見が多く得られた．一方，「操作が難しい」や「コメントがめんどくさい」などと共に，「どこまで同じツリーに入れるのか，やツリーの順番を決めるのが難しい」と言った意見があった．

TA からのアンケートでは，「コードツリーを使っての説明が普段よりも伝わりやすい」や「改善点をコードツリーに絡めて説明でき良かった」と言った肯定的な意見や「問題が簡単なので，全体を見なくても質問の意図が理解できてしまう」と言った意見があった．

7.3 考察

学生からのアンケートでは，5 章 6 章と同様な意見が得られた．また，ツリーの順番を決めるのが困難という意見があり，これはプログラムの構造を組み立てる力，つまり，抽象的思考をまだ鍛えきれてないことが原因と思われる．録画から学生と TA とのコミュニケーションの確認より，この意見を記述した学生は，コードツリーに絡めて説明を行っていた TA に質問をしていた．このことより，システムが抽象的思考を鍛えきれていない学生の抽象的思考を TA と共有でき，指導に活かせることが示唆された．

今回，プログラミングの授業で習い終えた範囲が狭く問題が単純だったため，TA が質問の意図を簡単に理解できてしまった．そのため，想定していた問題の趣旨を理解していないような質問はされなかった．

ユーザースタディより，プログラミングを習いたての初学者に対しても，システムが有効にはたらくことが明らかになった．また，抽象的思考が鍛えきれていない学生と TA とのコミュニケーションにおいても，システムを通して抽象的思考を可視化し共有することで指導に活かせることが示唆された．

8. まとめと今後の展望

本研究では，プログラム内のコメントからツリーを作成することで，ソースコードと同様な構造を持つ日本語で言語化された抽象的思考を可視化する手法の提案を行った．提案手法を元にしたプロトタイプシステムを用いた実験により，初学者に提案手法が優位にはたらき，プログラミングの構造を考えることにおいて支援することができた．

また，教育者側の TA が学生のプログラミング構造の理解度や思考の意図を理解，指導への支援が可能と示唆された．しかし，追加実験やユーザースタディにおいて，学生から問題の趣旨を理解していないような質問がなかったため十分な評価を行うことができなかった．

実験の中で，ある TA が一方的に構造（必要な変数など）を教える状況があった．その際に，答えを教えるのではなく，学生自身にプログラムの流れを読ませ，考えさせるようなアドバイスをすることによって，学生の理解が深まると考えられる．しかし，TA が学生のソースコードの構造から正解に誘導する力が必要と考えられるため，これらがプロトタイプシステムを用いることで容易になるかを今後の実験によって明らかにしていく．また，システムを卒業した際に，適切なコメントを書く習慣が身についているかなど，システム支援後の影響についても調査する必要がある．

謝辞 本研究の一部は，JST ACCEL（グラント番号 JPMJAC1602）の支援を受けたものである．

参考文献

[1] “日本の大学数は 768 大学 私立大が約 8 割！｜旺文社教育情報センター”．http://eic.obunsha.co.jp/resource/viewpoint-pdf/201807.pdf, (参照 2019-6-27).
[2] Kanamori, H., Tomoto, T. and Akakura, T.. Development of a Computer Programming Learning Support System Based on Reading Computer Program. Human Interface and the Management of Information. 2013, p. 63-69.
[3] “参考資料2 小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）”．http://www.mext.go.jp/b_menu/shingi/chukyo/chukyo3/053/siryo/__icsFiles/afieldfile/2016/07/08/1373901_12.pdf, (参照 2019-6-27).
[4] Durrheim, S. M., Ade-Ibijola, A. and Ewert, S.. Code Pathfinder: A Stepwise Programming E-Tutor Using Plan Mirroring. Communications in Computer and Information Science. 2016. vol. 642. p. 69-82.
[5] 安田光，井上亮文，市村哲．学生とティーチングアシスタント間でトラブル解決過程を共有できるプログラミング演習支援システム．情報処理学会論文誌. 2012. vol. 53, no. 1, p. 81-89.
[6] Kim, S., Kim, W. J., Park, J. and Oh, A.. Elice: An online CS Education Platform to Understand How Students Learn Programming. Learning at Scale. 2016. p. 225-228.
[7] 西田知博，原田章，中村亮太，宮本友介，松浦敏雄．初学者用プログラミング学習環境 PEN の実装と評価．情報処理学会論文誌. 2007. vol. 48, no. 8, p. 2736-2747.
[8] Patrice, F.. A Teaching Assistant for Algorithm Construction. Innovation and Technology in Computer Science Education. 2015, p. 9-14.
[9] Kayama, M., Satoh, M., Kunimune, H., Niimura, M., Hashimoto, H. and Otani, M.. Algorithmic Thinking Learning Support System with eAssessment Function. 2014 IEEE 14th International Conference on Advanced Learning Technologies. 2014, p. 315-317.
[10] 今泉俊幸，橋浦弘明，松浦佐江子，古宮誠一．ブロック構造の可視化環境によるプログラミング学習支援．研究会報告ソフトウェア工学. 2010. vol. 2010-SE-167, no. 2, p. 1-7.
[11] Dorneles, V. R., Picinin, D. J. and Adami, G. A.. ALGOWEB: A Web-Based Environment for Learning Introductory Programming. 2010 10th IEEE International Conference on Advanced Learning Technologies. 2010, p. 83-85.
[12] “Processing.org”. https://processing.org/, (参照 2019-6-28).