M-028

ビジュアルプログラミングによる視覚的な非同期処理の学習支援システムの提案と試作
A Proposal of Learning System for Asynchronous Processing with Visual Programming Environment

玉木 英鴻† 平山 勇太‡ 横井 翔太‡ 早川 智一‡
Hidehiro Tamaki Yuta Hirayama Shota Yokoi Tomokazu Hayakawa

1. はじめに

近年，Web サイトや Web アプリケーションの開発で広く用いられている言語 [1] として JavaScript が存在する．この理由として，JavaScript が Web ブラウザ上で動作するほぼ唯一のプログラミング言語であることが挙げられる．

JavaScript では，非同期処理――処理 A を開始したら完了を待たずに他の処理 B を開始して，処理 A が完了したら通知を受け取る方式――が用いられている．これは，JavaScript の仕様（単一のスレッドしか使用できない環境）下で並行処理を実現するためである．たとえば，Web アプリケーションが利用者の操作に反応したり Web サーバと通信したりする際に非同期処理が用いられている．

しかし，JavaScript の非同期処理は同期処理と比べて学習が困難という課題が存在する．主な理由として，(1) 非同期処理は同期処理と比べて処理の流れが複雑で理解しにくいこと，(2) JavaScript の非同期処理には複数の記法があり使い分ける必要があること――が挙げられる．

この課題を解決するための先行研究として，非同期処理の難しさに着目した可視化の研究は存在するが，学習支援を主目的とした研究は見当たらなかった（2 章）．

本論文では，前述の課題を解決するための学習支援システムを提案する．具体的には，(1) ブロックタイプのビジュアルプログラミング（以下，ブロックプログラミング）による JavaScript の非同期処理の問題を適切な順番で出題し，(2) それらの問題を学習者に解かせることで JavaScript の非同期処理の段階的な理解の促進を図る．

2. 関連研究・関連技術

2.1 関連研究

諏訪ら [2] は，非同期処理の難しさを解決するために，プログラミング言語 Liquid を提案している．諏訪らは，非同期処理が難しい原因として，(1) 従来の非同期処理モデルは非同期操作を明示的に記述する必要があること，(2) 非同期処理と同期処理とが待ち合わせるタイミングに注意が必要なこと――を挙げている．諏訪らの研究と本研究とは非同期処理の難しさを緩和する点で類似しているが，Liquid は JavaScript の非同期処理の学習が主目的ではない点で異なる．

†明治大学大学院理工学研究科 Graduate School of Science and Technology, Meiji University
‡明治大学理工学部 School of Science and Technology, Meiji University

Tominaga ら [3] は，コードの実行順序および Promise や async/await の動作を可視化する AwaitViz を提案している．Tominaga らは，非同期処理が難しい原因として，async/await の動作の複雑さを挙げている．Tominaga らの研究と本研究とは非同期処理の難しさを緩和する点で類似しているが，Tominaga らの研究は初学者の学習支援が主目的ではない点で異なる．

2.2 関連技術

ブロックを用いた視覚的なプログラミングを実現している技術としては Blockly [4] 等が挙げられる．これらは，変数・論理式・ループなどをブロックで提供しており，利用者はブロックを繋げるだけでプログラミングできる．しかし，これらは JavaScript の非同期処理に関するブロックを提供していない点で本研究とは異なる．

3. 提案手法

本論文では，前述（1 章）の課題を解決するために，JavaScript の非同期処理の難しさをブロックプログラミングで緩和する学習支援システムを提案する．ここで，ブロックプログラミングを採用した理由として，(1) ブロックプログラミングによって利用者が非同期処理の学習のみに集中できると考えたこと，(2) ブロックプログラミングは利用者の年齢とプログラミングの経験レベルに応じて適切に拡張され，利用者の独創性をサポートするのに十分な表現力がある [5] こと――が挙げられる．

図 1 に，提案システムの利用の流れを示す．学習者は，(1) システムによって出題される非同期処理に関連する問題に対してプログラミングをする，(2) プログラムが完成したら，「実行」ボタンを押して作成したプログラムの動作確認を行う，(3)「答え合わせ」ボタンを押してプログラムの正誤判定を行う，(4) 正解したら，Web ブラウザ上に表示されている JavaScript のコードを確認する，(5) 次の問題に移動して，(1) ～ (4) を繰り返す．

図 2 は提案システムの画面である．画面左上には，非同期処理などの問題を解くために必要なブロックがある．画面右上では，ブロックを用いてプログラミングをする．画面左下には，ブロックプログラミングしたものが JavaScript のコードとして表示される．画面最下部には 5 つのボタンがある．「実行」ボタンで作成したブロックプログラムを実行する．「答え合わせ」ボタンで作成したブロックプログラムが正解しているかどうかを判定する．

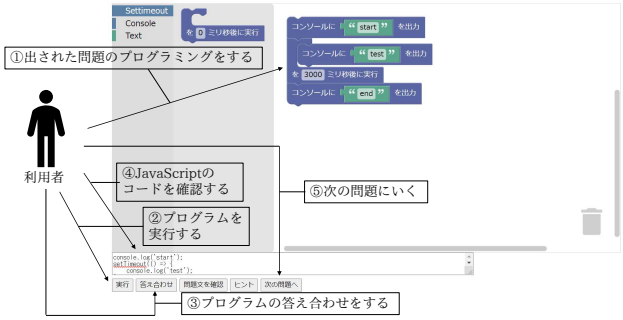

図 1　提案システムを利用する流れのイメージ

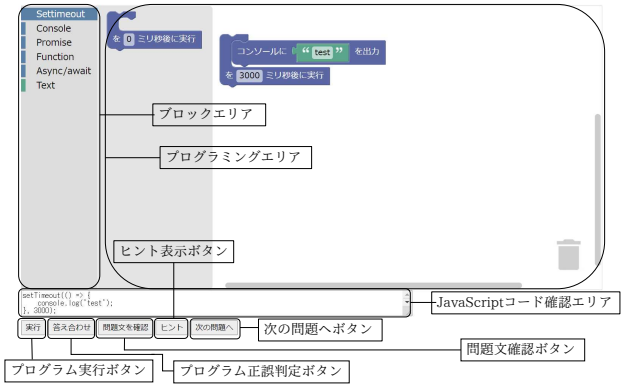

図 2　提案システムの画面

「問題文を確認」ボタンで問題文を確認する．「ヒント」ボタンで問題を解くためのヒントを表示する．「次の問題へ」ボタンで次の問題に移動する．

4.　設計

我々は，提案システムを Web ベースで利用できるように設計した．これは，利用者の環境構築の負担を軽減して，学習障壁を可能な限り低減するためである．

提案システムは，JavaScript の 3 つの非同期処理（コールバック・Promise・async/await）に対応するブロックを提供し，これらのブロックを使う問題を出すことで学習を支援する．具体的には（1）同期処理，（2）コールバック，（3）コールバック地獄，（4）Promise，（5）async/await――の順番で解かせることで学習効果の向上を図る．

また，問題の正誤判定の処理の流れは次のとおりである：（1）プログラミングエリアにあるブロックを読み込む，（2）各ブロックに対応する JavaScript のコードを生成する，（3）ブロックの繋がりに基づいて，生成した JavaScript のコードを結合する，（4）結合したコードと問題の答えとを比較する，（5）正解だった場合は次の問題へのボタンを表示し，不正解だった場合はその旨を伝える．

5.　実装

我々は，提案システムを実装するために，Blockly（2.2 節）を用いた．これは，Blockly はブロックから JavaScript コードを生成する機能を備えることから，非同期処理の学習支援環境の構築に最適と考えたためである．

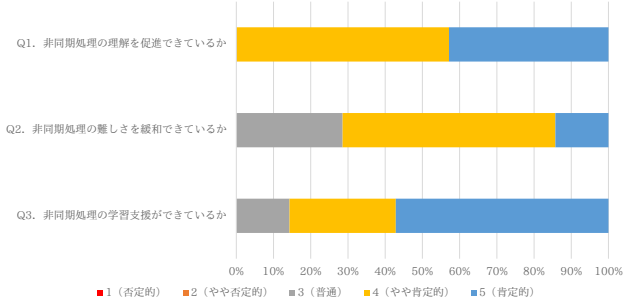

図 3　評価結果

6.　評価

我々は，提案システムの有用性を確認するために評価を行った．具体的には，明治大学ソフトウェア工学研究室の学生 7 名に提案システムを利用させ，次の 3 点についてアンケート評価を行った：（1）非同期処理の理解を促進できているか，（2）非同期処理の難しさを緩和できているか，（3）非同期処理の学習支援ができているか．

評価の結果を図 3 に示す．非同期処理の理解の促進と非同期処理の学習支援に関しては，最低評価が「やや肯定的」であることから概ね達成できたという結論を得た．この理由として，ブロックプログラミングで非同期処理を学習する方法が効果的だったためだと考えた．一方で，非同期処理の難しさの緩和については他の 2 つと比べると評価が低くなっている．この理由として，Promise と async/await は複数のブロックを適切に繋ぐ必要があるために，利用者からするとプログラミングが複雑になってしまったことが原因だと考えた．

7.　おわりに

本論文では，ビジュアルプログラミングによる視覚的な非同期処理の学習支援システムの提案と試作を行った．評価結果として，提案システムが非同期処理の学習を支援する一定の効果があることを確認した．今後は，非同期処理ブロックの単純化による非同期処理の難しさの緩和とさらなる評価を行う予定である．

参考文献

[1] IEEE: The Top Programming Languages, https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2020.
[2] 諏訪重貴，福田浩章，篠埜功：Liquid: 非同期処理の暗黙的同期を行う型付きプログラミング言語，Vol. 2019, No. 1, pp. 203–204 (2019).
[3] Tominaga, E., Arahori, Y. and Gondow, K.: AwaitViz: a visualizer of JavaScript's async/await execution order, Vol. 2019, No. 1, pp. 2515–2524 (2019).
[4] Google Developers: Blockly, https://developers.google.com/blockly.
[5] Feng, A., Tilevich, E. and Feng, W.: Block-based programming abstractions for explicit parallel computing, *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*, pp. 71–75 (2015).