双方向性コミュニケーションを行う ブロック型プログラミングアプリケーションの開発

中原　久志・村長　康太郎

Development of Block Type Programming Application for Bidirectional Communication

NAKAHARA, Hisashi and MURANAGA, Koutaro

大分大学教育学部研究紀要　第 40 巻第 1 号
2018 年 9 月　別刷
Reprinted From
RESEARCH BULLETIN OF THE
FACULTY OF EDUCATION
OITA UNIVERSITY
Vol. 40, No. 1, September 2018
OITA, JAPAN

双方向性コミュニケーションを行う ブロック型プログラミングアプリケーションの開発

中 原 久 志*・村 長 康 太 郎**

【要 旨】 本研究の目的は，小・中学校におけるプログラミング教育において，児童・生徒が活用可能なビジュアルプログラミングを用いた教育用アプリケーションの開発を行うことである。児童・生徒のレディネスおよびプログラミング的思考の要件を満たす仕様を策定し，iOS デバイスで使用するアプリケーションをオープンソースソフトウェアである Blockly を用いて開発した。本アプリケーションは，論理ブロックを並べ替えることで，文字列送信プログラムや自動返信プログラムを自由に作成可能である。児童・生徒にとって身近な題材であり，直感的に操作しやすい UI を工夫することで，アルゴリズムの理解やプログラミング的思考，論理的思考の育成を促すことが期待できる。

【キーワード】 プログラミング教育 アプリケーション開発

１．はじめに

本研究の目的は，2020 年から施行される小学校および中学校学習指導要領において示されたプログラミング教育に関して，小・中学校において児童・生徒が活用可能なビジュアルプログラミングを用いた教育用アプリケーションの開発を行うことである。具体的には，双方向性のあるコミュニケーションのプログラミングをブロック型の UI を用いたアプリケーションを開発し，その利用可能性を検討することである。

文部科学省は 2020 年から，小学校段階でのプログラミング教育を位置付けた [1)]。これは，2016 年の有識者会議において，小学校段階でのプログラミング教育で育成する資質・能力として，①身近な生活でコンピュータが活用されていることや，問題の解決には必要な手順があることに気付くこと，②プログラミング的思考を育成すること，③コンピュータの働きを，よりよい人生や社会づくりに生かそうとする態度を涵養すること，の 3 点が示されたことを背景としている [2)]。ここでいうプログラミング的思考とは，「自分が意図する一連の活動を実現するた

平成 30 年 5 月 31 日受理

＊なかはら・ひさし 大分大学教育学部技術講座

＊＊むらなが・こうたろう 富士通株式会社

めに，どのような動きの組合せが必要であり，一つ一つの動きに対応した記号を，どのように組み合わせたらいいのか，記号の組合せをどのように改善していけば，より意図した活動に近づくのか，といったことを論理的に考えていく力」のことである。また，このようなプログラミングに関する教育は，中学校段階にも位置付けられた。中学校技術・家庭科技術分野の内容「D 情報の技術」において，「生活や社会における問題を，ネットワークを利用した双方向性のあるコンテンツのプログラミングによって解決する活動」が設定された [3]。中学校学習指導要領解説技術・家庭編では，「ネットワークを利用した双方向性とは，使用者の働きかけ（入力）によって，応答（出力）する機能であり，その一部の処理の過程にコンピュータ間の情報通信が含まれることを意味している」とある [4]。この双方向性を持ったコンテンツを題材にすることは，様々な情報機器が身近にあり，実際に使用している小学校の児童に対しても十分に有効であるとともに，興味・関心を持たせることが可能であると考えられる。

そこで本研究では，児童・生徒にとって具体的事例として認識しやすい双方向性のあるコミュニケーションアプリケーションをブロック型の UI を用いたプログラミングで行うことができるツールを開発し，その利用可能性を検討することとした。

2．開発するアプリケーションの仕様

2.1 仕様の策定

アプリケーションの開発に先立ち，仕様を策定した。「プログラミング的思考」および児童・生徒のレディネスに基づき，以下の 4 点を設定した。

①実際のプログラミングコードを記述することなく，感覚的なビジュアルプログラミングを行うことでアルゴリズムの基本構造が理解できること。

②学習者の経験との関連や興味関心を高めうる題材を用いる事で能動的な学習とすること。

③プログラミングへの興味を持った学習者が，テキスト記述型言語についても閲覧・学習が可能であること。

④教育現場の実態に即し，タブレット端末での利用が可能であること。その際，教育現場に導入されているタブレット端末は iOS が多く見られることから，iOS 端末で使用できること。

特に，②に該当する事項として，プログラミングの学習が，子ども達の生活や日常の延長線上にある内容を取り扱うことで，継続的な学びが可能になると考え，題材を双方向型のコミュニケーションアプリケーションを取り扱うこととした。

2.2 言語の選定

仕様④に関連して言語の選定を行った。iOS アプリ開発に用いられる代表的な言語は「Objective-C」および「Swift」であることが多い。コードの簡易さの観点から，Objective-C でクラスを定義する場合，ヘッダファイルとソースファイルの 2 つを用意する必要がある。しかし同様のコードを Swift で記述すると簡易化を図ることが可能になる。処理速度に着目すると，Swift は高速化を目的として開発された言語であり，Objective-C の最大 2.6 倍，Python の最大 8.4 倍の速度とされている。このような利点から本研究においては，「Swift」が有効であると考え，選定した。

2.3 オープンソースの活用

次に，仕様①に関する事項として，アプリケーションの設計を行った。コーディングのスキルを身につけていない子どもたちが操作および学習が可能なインターフェースとして，ブロックタイプのアプリケーションが存在する。例えばMITメディアラボの「Scratch[5]」や，文部科学省の「プログラミン[6]」などが挙げられる。その中でも「Scratch」や「Blockly[7]」などはオープンソースとして公開されており，本研究においても，オープンソースである「Scratch」をベースに，限られたブロックを組み立てるとメッセージの送受信に必要なサーバやクライアントの動きをする機能の実装を試みた。しかし，一部の不要なブロックや機能を削除し，ブラックボックス化したところ，他の機能に影響を及ぼすことが確認された。そこで，Google のBlocklyを用いて再度試行を行った。

2.4 Blocklyに実装する機能の選定

Blocklyは開発の自由度を高めるため，ブロックの種類は簡単な論理構造のみとなっている。また，Blockly は開発者向けにライブラリを公開している。ライブラリをインポートすることでBlocklyに搭載されているワークスペースとデフォルトのブロックが使用可能になる。ブロックの自作も可能であるが，サーバ構築に必要なブロックの作成は困難であった。そこで，ブロックの組み合わせにより自分の意図した挙動を行い自動でメッセージを返信するプログラムの作成機能を開発することとした。具体的には，既存のチャットアプリケーションのサーバクライアントである「LineMessagingAPI[8]」と「GoogleAppEngine[9]」を使用することとした。

LineMessagingAPIは，スマートフォン・タブレット端末向けのコミュニケーションツール「LINE」の提供する，「Bot」を作成するサービスである。Botはコミュニケーションツールにおける自動返信プログラムの総称である。メッセージをBotに送信することでWebhookトリガーが引かれ，サーバにあらかじめ設定しているプログラムを読み込み，自動で返信を行う。Webhookはイベント（リポジトリにプッシュなど）発生時に，指定したURLにPOSTリクエストを行う仕組みである。

GoogleAppEngineはGoogleの提供しているPaaSである。Java，Pythonなど様々な言語へのサポートを可能としており，本研究では「Go言語[10]」を採用した。

3．アプリケーションの開発

3.1 アプリケーションの概要

3.1.1 ワークスペース画面

アプリケーションを起動するとBlocklyのワークスペース（図1）が表示される。ワークスペース内のツールボックス(図 1，左上の「Logic」と書かれたボタン）にブロックを格納しており，ボタンを押すことで中のブロックが表示される。表示されたブロックを灰色のスペース（図1左の①）へドロップすることでブロックの使用が可能になる。組み合わせたブロックを削除する場合はワークスペース右下にある「ゴミ箱アイコン」へドロップすることで削除ができる。操作を戻す場合はワークスペース左下にある矢印ボタンの操作で可能となる。ツールボックスの作成はXMLファイルを使用する方法と直接プログラムに書き込む方法の2種類があるが，プログラムの簡略化・変更時の簡易さなどの観点から前者を選択した。試作した各ブロ

ックは，①送信してきたメッセージをおうむ返しするブロック，②送るメッセージを選択するブロック③LIneMessagingAPI で作成した Bot の指定と挙動を設定するブロック④メッセージの送信内容（ここでは”Hello world”）を指定するブロック，⑤LineBot のアドレスを格納しているブロックである（図 1 右各番号に対応）。

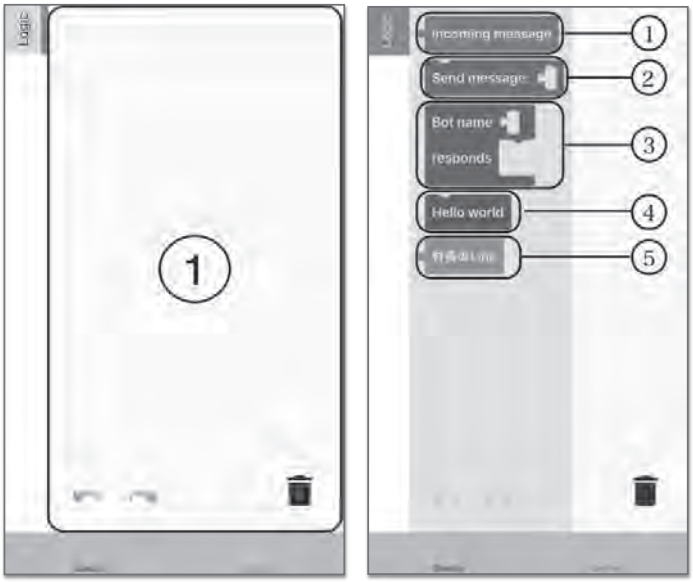

図 1　Blockly のワークスペースとブロックの表示

XML ファイルを使用する方法では，図 2 に示すプログラムを作成した。<category>でツールボックスの内容を指定，name はツールボックスの名前を指定，color はツールボックスを表示する色を指定，<block>でツールボックス内に入れるブロックを指定する。図 3 のコードでは，3 行目で CocoaPods でインストールしたライブラリをインポートしている。6-8 行目ではライブラリ内のワークスペースやその他 Blockly の機能を使用することを定義し，11-12 行目でデフォルトのブロックをロードする。15-28 行目では XML ファイルで作成したツールボックスをロードする。

```xml
<xml>
    <category name="Logic" colour="120">
        <block type="variable_incoming_message"></block>
        <block type="message_text_send"></block>
        <block type="define_bot"></block>
        <block type="send_helloworld"></block>
        <block type="message_sender_name"></block>
    </category>
</xml>
```

図 2　XML を使用したツールボックスのコード（１）

```
import UIKit
import Blockly

class ViewController: UIViewController {
    open var workbenchViewController: WorkbenchViewController = {
        let workbenchViewController = WorkbenchViewController(style: .defaultStyle)
        workbenchViewController.toolboxDrawerStaysOpen = true

        // Load default blocks into the block factory
        let blockFactory = workbenchViewController.blockFactory
        blockFactory.load(fromDefaultFiles: .allDefault)

        // Load toolbox
        do {
            let toolboxPath = "toolbox.xml"
            if let bundlePath = Bundle.main.path(forResource: toolboxPath, ofType: nil) {
                let xmlString = try String(
                    contentsOfFile: bundlePath, encoding: String.Encoding.utf8)
                let toolbox = try Toolbox.makeToolbox(
                    xmlString: xmlString, factory: blockFactory)
                try workbenchViewController.loadToolbox(toolbox)
            } else {
                print("Could not load toolbox XML from \(toolboxPath)")
            }
        } catch let error {
            print("An error occurred loading the toolbox: \(error)")
        }

        return workbenchViewController
    }()
}
```

図 3　XML を使用したツールボックスのコード（２）

3.1.2　カスタムブロックの作成と定義

「Blockly」はデフォルトのブロックの他に自分の作成したブロック（以下，カスタムブロックとする）を使用可能であり，本研究では全てカスタムブロックを作成・使用した。ブロックの作成には，「Blockly Developer Tool[11]」を用いた（図 4）。

「Blockly」のブロックを並べて（図 4，左側）カスタムブロックの定義を作成すると，リアルタイムで JSON ファイルが生成される。生成された JSON ファイルを自分のプロジェクト

に貼り付けて使用する。図 5 における変数等は以下の通りである。

type：ブロックの名前を指定
message0：ブロックに表示するメッセージを指定
Args：インプットセクションを持つブロック（図 6）の設定
name：繋げるブロックの出力名を指定
Check：繋げるブロックの型を指定
Output：アウトプット型のコネクションブロック（図 6）の出力名を指定
previousStatement：ブロックの型を指定
inputInline：他のブロックへの接続の可否を指定（Bool 型）
tooltip：ブロックの説明を追加

図 4 Blockly Developer Tool

```
[
 {
 "type": "define_bot",
 "message0": "Bot name %1",
 "args0": [
          {
          "type": "input_value",
          "name": "BOT_NAME",
          "check": "BotName"
          }
          ],
 "message1": "responds %1",
 "args1": [
          {
          "type": "input_statement",
          "name": "RESPONDS",
          "check": "String"
          }
          ],
 "colour": 223,
 "inputsInline": true
 },
 {
 "type": "variable_incoming_message",
 "message0": "incoming message",
 "output": "String",
 "colour": 30,
 "tooltip": "Returns the value of incoming message."
 },
 {
  "type": "message_text_send",
  "message0": "Send message: %1",
  "args0": [
           {
           "type": "input_value",
           "name": "VALUE",
           "check": "String"
           }
           ],
  "previousStatement": "String",
  "colour": 10,
  "inputsInline": true
  },
 {
    "type": "send_helloworld",
    "message0": "Hello world ",
    "previousStatement": "String",
    "colour": 230,
    "tooltip": "",
    "helpUrl": ""
},
{
    "type": "message_sender_name",
    "message0": "村長のLine",
    "output": "BotName",
    "colour": 80,
    "tooltip": "Returns the name of the sender of this communication",
    "helpUrl": ""
}
]
```

図 5　カスタムブロックの定義（JSON ファイル）

図 6　使用するブロック
（左：インプット，中：アウトプット，右：コネクション）

3.1.3　画面間でのデータの受け渡し

ワークスペース内で並べたブロックの状態を XML で変換し，サーバに POST リクエストで送信している。この際，ワークスペースとは別の画面に POST リクエストを行うボタンを配置しており画面間の XML データの受け渡しは以下の通りに設定している。

画面間のデータの受け渡しはクラス間の変数共有を行う AppDelegate メソッドと，Storyboard での画面遷移を行うメソッド Segue を使用した。AppDelegate.swift ファイルに変数を指定し，message 変数を String 型で作成した。

Segue は StoryBoard を用いて，Tab Bar Controller からワークスペースへ繋げる。Segue を繋げることで画面遷移時のメソッド ViewDidDisappear が使用可能になる。

3.1.4　通信方法

HTTP POST リクエストを使用して，生成された XML データを Google App Engine に送信する。HTTP POST リクエストは HTTP のプロトコルを用いてクライアントから Web サーバへ情報を登録する時に使用する。図 7 において，27 行目で xml の文字列を数値に変換する。今回は UTF-8 を使用した。29 行目でサーバの URL を設定し，34 行目でヘッダファイルを設定する。37 行目で POST メソッドを指定し，40 行目で送信するデータを格納する。43-50 行目で URLSessionTask メソッドを指定する。iOS での HTTP 通信はライブラリを使用するこ

とが多いが，今回は単純な POST リクエストなので URLSessionTask メソッドを使用した。GoogleAppEngine では受信した POST リクエストを解析し，LineMessagingAPI を読み取れる形式へ変換する。

```
@IBAction func tappedButton(_ sender: Any) {

    //Encoding UTF-8
    let xmlData = workspaceXml.data(using: String.Encoding.utf8)

    //Set URL
    let url = URL(string: "https://kotobuu-gae.appspot.com")!
    var request = URLRequest(url:url)

    //Set XML header
    request.setValue("application/xml", forHTTPHeaderField: "Content-Type")

    //Set POST
    request.httpMethod = "POST"

    //Set data
    request.httpBody = xmlData

    //Use URLSessionDataTask
    let task = URLSession.shared.dataTask(with: URLRequest(url: url),completionHandler: { (data, response,
        error) in
        if error != nil{
            print(error!.localizedDescription)
        }else{
            print(NSString(data: data!, encoding: String.Encoding.utf8.rawValue) as Any)
        }
    })
    task.resume()
    }
```

図 7　HTTP POST リクエスト

3.2　実装時の動き

①Hello world を送信する場合

”Bot name”の横に使用する LineMessagingAPI のアカウントを格納したブロックを繋げることで，レスポンスをする Line のアカウントの設定を行った。“responds”の横に挙動の設定を行うブロックを繋げる。図 8 では，”Helloworld”を送信するので”Helloworld”ブロックを繋げる。

②送信したメッセージをおうむ返しする場合

“Send message”ブロックの横にサーバに送られてきたメッセージを表示する”Incomming message”を繋げる（図 9）。

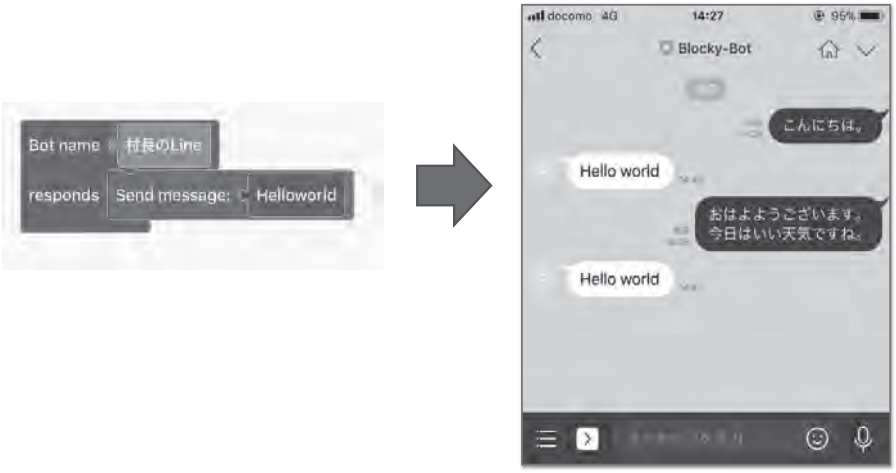

図 8　Hello world を送信するブロックの組み合わせと動作例

図 9　送信したメッセージをおうむ返しするブロックの組み合わせと動作例

4．まとめと今後の課題

本研究では，小・中学校において児童・生徒が活用可能なビジュアルプログラミングを用いた教育用アプリケーションの開発を行った。開発したアプリケーションは，論理ブロックを並べ替えることで，双方向性コミュニケーションツールとしての自動返信プログラムを自由に作成可能である。児童・生徒にとって身近な題材を設定し，iOS のタブレット等で活用可能であるとともに，直感的に操作しやすい UI を工夫することで，アルゴリズムの理解やプログラミング的思考，論理的思考の育成を促すことに期待できると考えられる。

しかし，本研究にはいくつかの課題がある。まず，現段階では動作確認にしか至っておらず，試行的実践を通して，その教育効果を測定し，アプリケーションの改良および活用例を例示する必要がある。また，条件分岐，単語検索機能（特定の文字または文字列が含まれている場合，特別な挙動を行う機能等），インターネット検索機能（単語検索機能を用いて特定の文字列を送信することであらかじめ設定していたサイトで単語の検索を行う機能）といった機能の追加の検討も必要と考えられる。これらについては，今後の課題とする。

参考文献

1) 小学校学習指導要領（平成 29 年告示）：文部科学省 http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/05/07/1384661_4_3_2.pdf（最終アクセス：2018.5.31）
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）：小学校段階における論理的思考力や創造性、問題解決能力等の育成とプログラミング教育に関する有識者会議，http://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm（最終アクセス：2018.5.31）
3) 中学校学習指導要領（平成 29 年告示）：文部科学省 http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/05/07/1384661_5_4.pdf（最終アクセス：2018.5.31）
4) 中学校学習指導要領（平成 29 年告示）解説技術・家庭編：文部科学省 http://www.mext.go.jp/component/a_menu/education/micro_detail/__icsFiles/afieldfile/2018/05/07/1387018_9_1.pdf（最終アクセス：2018.5.31）
5) Scratch：MIT メディアラボ，http://scratch.mit.edu/（最終アクセス：2018.5.31）
6) プログラミン：文部科学省，http://www.mext.go.jp/programin/（最終アクセス：2018.5.31）
7) Blockly：Google，https://developers.google.com/blockly/（最終アクセス：2018.5.31）
8) Messaging API：LINE，https://developers.line.me/ja/services/messaging-api/（最終アクセス：2018.5.31）
9) Google App Engine：Google,https://cloud.google.com/appengine/?hl=ja（最終アクセス：2018.5.31）
10) The Go Programming Language：Google,https://golang.org/（最終アクセス：2018.5.31）
11) Blockly Developer Tools：Google, https://developers.google.com/blockly/guides/create-custom-blocks/blockly-developer-tools（最終アクセス：2018.5.31）

Development of Block Type Programming Application for Bidirectional Communication

NAKAHARA, Hisashi and MURANAGA, Koutaro

Abstract

The purpose of this study is to develop educational applications using visual programming that children and students can utilize in programming education. First, we formulated specifications that fit the requirements of students' readiness and programming thinking. Next, we developed an application using Blockly which is open source software. In this application, it is possible to freely create a “string transmission program” and an “automatic reply program” by rearranging logical blocks. The content of the application is designed to be familiar to children and students and can be intuitively and easily thought out by devising the appropriate UI. The use of this application can be expected to encourage understanding of algorithms, programming thinking, and development of logical thinking.

【Key words】 Programming Education, Application Development