機能の実行を動的にカスタマイズ可能な Web プログラミング学習環境の提案

M2018SE001 安達有希
指導教員：蜂巣吉成

1 はじめに

プログラミング学習を行う際に，学習の支援を行うことができれば，学習者にとって学習がしやすくなり，より理解を深めることにつながる．プログラミング学習には，プログラミングを学ばせるという主目的の他に，副次的な目的が存在する場合がある．例えば，テストケース設計を学ばせたい，きれいなプログラムを書かせたいなどである [1][2]．プログラミング環境にこれらの学習を行うための機能があれば，プログラミング以外のことも学ばせることができる．学習環境に機能を追加して利用する場合も，支援内容や学習内容などで学習目的が異なり，教員によっては機能を利用させるタイミングを指定して，決められた学習手順で学習を行う．

従来のプログラミング環境は，機能を追加することはできるが，機能ごとに実行方法が決まっており，機能の実行方法を変えて学習者が利用できるようにするのは難しい．複数の機能を制御することも難しいので，学習機能の利用を強制させたりするような学習は行えない．

本研究では，学習機能を追加し，それらを制御することで，学習目的や教員の意図に合わせたカスタマイズが可能な学習環境の提案を行う．本研究で提案する学習環境は，Web を用いた学習環境とし，Web サーバで実行される複数の学習機能を Web ブラウザで統合して学習を支援する．機能を追加するための学習環境のフレームワークを Passive View MVP[3][4] に基づいて設計し，機能間の制御のために各機能の入出力の標準化を行うことで，機能のカスタマイズ方法の統一を行う．機能をカスタマイズする際に，機能の開発者とカスタマイズを行う教員が異なる場合や，教員が学習環境のカスタマイズを行うためのプログラムを記述する知識をもっていない場合がある．教員の負担を減らすために，教員が設定ファイルに課題ごとにデータを記述し，それを読み込むことで課題に合わせたカスタマイズを動的に行えるようにする．統合 CASE ツールの考え [5][6] に基づいて，提案する学習環境を整理し，プロセスプログラミング [7][8][9] などと比較する．

2 プログラミング学習環境

2.1 学習環境の概要

本研究が想定するプログラミング学習環境は基本機能と拡張機能から構成される．基本機能とは，プログラミング学習を行うために必要な機能で，あらかじめプログラミングの学習環境に備わっているプログラム編集やコンパイル，実行，提出などの機能である．

拡張機能とは，基本機能以外の追加される機能である．拡張機能には，プログラムの可視化やヒント表示などのプログラミングの学習を支援する機能とテストケース評価やプルーフリーダなどのプログラミング以外のことを学ばせる機能が想定される．

2.2 カスタマイズの必要性

学習環境をカスタマイズする際に想定されるカスタマイズ内容には，以下のものが挙げられる．

- 拡張機能として追加する機能
- 機能の実行方法
- 機能の実行結果の出力方法
- 機能の制御

例として，プログラムの理解支援のために可視化を行う図式生成機能のカスタマイズを行う場合について述べる．学習者はこの機能を利用し，自身の書いたプログラムの図式を表示させることができる．以下のカスタマイズが想定される．

(1) 学習者が任意に利用できるようにする
(2) コンパイルエラー時に自動的に実行する
(3) 一定時間操作がなかった時に自動的に実行する

文献 [1] で提案されているような，副次的な目的としてテストケース設計について学ばせるために，テストケース評価機能のカスタマイズを行う場合について述べる．学習者がテストケースの入力を行うと，境界値分析や同値クラスに基づいてテストケースを評価した結果がカバレッジとコメントで出力される．以下のカスタマイズが想定される．

(1) 学習者が任意に利用できるようにする
(2) テストをパスしたら提出できるようにする (図 1)
(3) テストケース設計ができたらプログラム編集できるようにする (図 2)

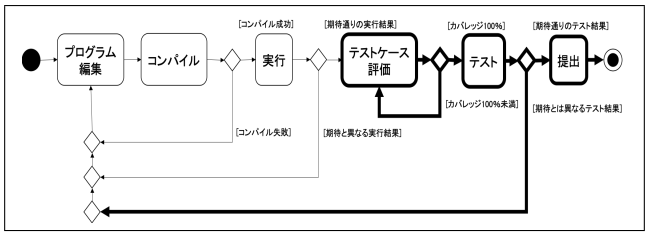

図 1 テストケース評価 (2)

テストケース評価機能を追加した際の学習プロセスは，図 1，図 2 のように表すことができる．機能が増えると組み合わせが複雑になる．また，学習者の行動を規定することも難しい．全体の学習プロセスを定義することは，

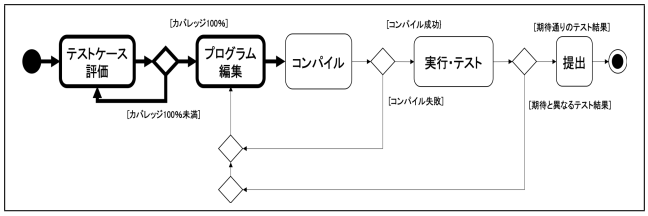

図 2 テストケース評価 (3)

困難であるので，本研究では，機能のカスタマイズを行い，機能の実行方法を変更し，他の機能を制御する．カスタマイズを行うことで，図 1，図 2 の太線部分のような，一部の機能の利用を強制させるような学習を行うことができる．

3 カスタマイズ可能な学習環境の提案

3.1 Web を用いた学習環境

本研究で提案する学習環境は，Web を用いた学習環境とする．学習環境として Web を用いることで，機能の更新などをサーバだけで行うことができる．学習者もコマンドライン環境よりも GUI 環境の方が日常の利用環境と近いので，ギャップを感じることなくプログラミング学習ができる．ブラウザのみで利用できるので，利用者の環境構築にかかる手間を軽減できる．

3.2 学習支援機能の統合

本研究で提案するプログラミング学習環境は，Web サーバで実行される複数の学習支援機能を Web ブラウザで統合して学習を支援する．その考え方は，統合 CASE ツールと類似している．Wasserman は CASE ツールの統合を 5 段階に分類しており [5][6]，その統合に従えば，本研究で提案するプログラミング学習環境は，次のようになる．

1. プラットフォーム統合
   Ajax により Web サーバで実行される学習支援機能を利用する．
2. プレゼンテーション統合
   HTML のフォーム要素でユーザインタフェースを統一し，学習支援機能の実行に必要なボタンなどは動的に生成する．
3. データ統合
   各機能の入出力の標準化を行う．入力はソースプログラムとソースプログラム以外の入力の 2 つ，出力は機能の実行結果と終了ステータスの 2 つに分類した．機能によってはその他のデータも拡張可能である．
4. 制御統合
   複数の機能を実行したり，ある機能の成否によって他の機能の実行を制御する方法を提案する．
5. プロセス統合
   学習者の作業をプロセスとして完全に定義することは難しい．ただし，複数の機能間で依存関係があるものは，制御統合で一部の機能の強制を行う．

3.3 機能の実行方法

実行方法は，機能を実行させるための方法で，本研究で提案する学習環境では，次の 3 通りの実行方法を想定している．

(a) ボタンを押して実行

(b) 他の機能の実行に合わせて実行

(c) 一定時間が経過したときに実行

(a) は，その機能を実行するためのボタンを押すことで，ボタンと対応した機能が実行される．既存のボタンを用いる場合と新しく追加したボタンを用いる場合がある．1 つのボタンから複数の機能を実行させることができる．

(b) は，ボタンを押して実行する機能の実行結果などによって機能を実行させる．単に複数の機能を実行させるだけでなく，ボタンを制御して実行を行えるようにする場合もある．

(c) は，一定時間の間に操作がなかったときに，自動的に機能を実行させる．具体的には，マウスのクリックや移動，キーボードからの入力がなかった場合に実行される．

2.2 節で述べた図式生成機能のカスタマイズ例の (1)，(2)，(3) がそれぞれ (a)，(b)，(c)，テストケース評価機能のカスタマイズ例の (1)，(2)，(3) がそれぞれ (a)，(b)，(b) に該当する．

3.4 利用者の整理

本学習環境の利用者として想定しているのは，開発者，教員，学習者である．開発者は，学習環境に追加する学習支援機能の開発を行う．学習環境に拡張機能として新たに機能の追加を行うのも開発者である．開発者は追加した機能を確認するために，開発者自身が想定している利用方法で学習者が利用できるように機能の追加を行う．

教員は，学習目的や自身の意図にあわせて，学習環境の機能のカスタマイズを行う．開発者と教員が異なる場合も考えられる．そのような場合でも，教員は，拡張機能として追加したい機能について，利用者としての知識は持っていることを想定している．

学習者は，教員によってカスタマイズされた学習環境を利用してプログラミングの学習を行う．

3.5 フレームワークの設計

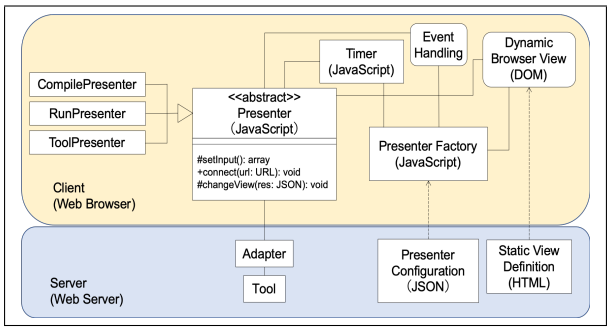

図 3 学習環境の構造

図 3 に，本研究で提案する学習環境のフレームワークの構造を示す．機能の出力によって結果の表示や制御方

法を変えたいので，Model と View を直接対応づけるのようなものではなく，Passive View MVP[3][4] に基づいて設計した．

図3の黄色い部分がクライアント側でDynamic Browser View，Presenter，Presenter Factory，Event Handling，Timer から構成されている．青い部分がサーバ側で Static View Definition，Presenter Configuration，Adapter，Tool から構成されている．四角がファイル，角の丸い四角がブラウザ内での処理を表し，実線がデータの流れ，点線がファイルの読み込みを示している．

Static View Definition が画面構成の定義で，Dynamic Browser View がブラウザが構築する DOM 木である．Static View Definition は，HTML で画面構成の定義が行われており，基本機能のプログラム入力欄や実行ボタンなども定義している．Dynamic Browser View では，Static View Definition や Presenter Factory から追加された内容を読み込んだり，Presenter からのデータを反映させてたりして DOM 木を書き換えて画面の更新を行う．

Presenter は，機能を実行するための処理をクライアント (ブラウザ) で行う．Dynamic Browser View からデータを取得して，Adapter と通信を行い，Adapter から返信されたデータを処理してから，Dynamic Browser View に反映させる．Presenter は JavaScript で抽象クラスとして定義しており，機能ごとにサブクラスを作成する．setInput() で，機能の実行に必要なデータを Dynamic Browser View から取得して連想配列に変換し，connect(url) で，連想配列を実行したい機能の Adapter の URL と通信して送信する．Adapter から返信されたデータを引数にして changeView(res) を呼び出し，Dynamic Browser View に反映させて画面を書き換える．Ajax 通信を用いることで，画面遷移を行わずに画面を更新することができる．

Adapter は，機能の実行のための処理をサーバで行う．Presenter から送信されたデータを利用して機能の実行を行い，その結果を Presenter に返信する．Tool は，各機能を実現したもので，コマンドとして実行される．

Presenter Factory は，Presenter の生成やユーザインタフェースの変更を行う．Presenter Configuration を読み込むことで，動的に学習環境に機能の追加を行い，イベントと Presenter の対応付けを行ったり，必要があればユーザインタフェースの追加を行ったりすることができる．JavaScript で記述している．

Presenter Configuration は，Presenter Factory の設定ファイルである．ボタンの追加や Presenter の生成を行うための情報が JSON 形式で記述されている．

Timer は，一定時間ごとに機能を実行させるための仕組みである．

3.6 Presenter Configuration

機能とイベントの対応づけは Presenter Factory で行われるが，Presenter Configuration を読み込んで，その内容を Presenter Factory として反映させることができる仕組みを実装した．Presenter Configuration は，機能のカスタマイズに必要な情報で，学習環境に追加するボタンと機能に関する情報を教員が記述する．2.2 節で述べた図式生成機能を，学習者が任意に実行できるようにする場合の例を Listing 1 に示す．

ボタンの追加を行うために必要な情報は，次の 3 つである．

- ボタン ID，ラベル，初期状態

ボタン ID は，追加するボタンに対する ID で，ラベルは，ブラウザで表示されるボタンにつけられるものである．初期状態は，機能の制御をカスタマイズするための情報で，あらかじめボタンを無効にしておく場合に disabled と記述する．

機能の追加を行うために必要な情報は，次の 5 つである．

- 機能の名前，Presenter の名前，Adapter の URL
- ボタン ID，時間

機能の名前は，複数の機能を追加する場合に，区別するためのものである．Presenter の名前は，追加する機能に対する Presenter クラス名で，Adapter の URL は，追加する機能と対応した Adapter の URL である．追加する機能のカスタマイズについては，これらの情報を記述するだけで，学習環境に機能の追加を行うことができる．ボタン ID と時間は，追加した機能の実行方法をカスタマイズするための情報である．ボタン ID は，同じボタン ID を持ったボタンが押されたら機能が実行されるようになる．複数の機能で同じボタン ID を記述すれば，1 つのボタンから複数の機能を実行させることができる．時間は，ミリ秒単位で記述し，その間操作がなかった場合に自動で機能が実行されるようになる．両方記述すれば，記述したボタンと一定時間の両方で実行ができる．

Listing 1 2.2 節の (1) の Presenter Configuration

```
{"question1":[
   {"button":[
      {"id":"syndia","label":"SYNDIA","initial":""}
    ],
    "tool":[
      {"name":"syndia",
       "presenter":"SyndiaPresenter",
       "adapter":"php/syndiaAdapter.php",
       "buttonid":"syndia",
       "time":""
      }
    ]
   }
 ]
}
```

3.7 カスタマイズ方法

機能のカスタマイズを行う方法は，カスタマイズする内容によって異なる．3.3 節で述べた (b) のように，他の機能の出力や実行結果による機能間の制御をカスタマイズする場合は，Presenter の変更を行い，拡張機能として追加する機能や 3.3 節の (a)(c) のような機能の実行方法，初期状態のボタンの制御をカスタマイズする場合は，Presenter Configuration の変更を行う．

Presenter を変更するカスタマイズを行う場合，教員が機能の追加を行う際に，追加する機能のサブクラスを定義して，changeView(res) をオーバーライドする．例えば，2.2

節で述べた図式生成機能を，コンパイルが失敗した時に実行されるようにする場合は，Listing 2 のように Compile Presenter のサブクラスを定義し，changeView(res) にその処理を定義することで実現できる．

Listing 2 2.2 節の (2) の CompilePresenter のサブクラス

```
class CompPresenterS extends CompilePresenter {
  constructor(url) {
    super(url);
  }
  changeView(returnValue){
    var prog=document.getElementById("answer");
    prog.innerHTML=returnValue.answer;
    if(returnValue.status!=0){syndia.connect();}
  }
}
```

Presenter Configuration を変更してカスタマイズを行う場合は，追加するボタンや機能の情報を JSON 形式で記述する．

4 評価・考察

Presenter Configuration で，教員がカスタマイズを行うための記述を行うことで，学習環境に機能のカスタマイズを行うことができる．Presenter Configuration は，JSON 形式で記述を行うので，プログラムを書く必要がなくなり，より容易に機能のカスタマイズを行うことができる．

Presenter Configuration の記述で，3.3 節で述べた (a) のように，1 つのボタンから複数の機能を実行させる場合，現時点で想定している学習機能は，実行に依存関係がなく，どの機能から実行しても結果は変わらない．3.3 節の (b) のように，他の機能の実行結果に合わせて機能を実行させるなど，機能の実行に順序が求められる場合は，Presenter に記述し変更することで実現できる．

カスタマイズを行うための Presenter Configuration の整合性は，記述を行なった教員が行う必要がある．機能の実行で用いるボタン ID が実際に存在するかなどの誤りがないかなどを実行前に静的にチェックする仕組みは今後の課題である．

5 関連研究

プロセスプログラミングは，ソフトウェアを開発するプロセスを手続き的なプログラムとして記述して実行しようという考えで，多くのプロセスを記述する形式言語やモデル，支援環境が提案されている [7][8][9]．この考え方に基づいて，プログラミング学習の学習手順を定義することで，決められた学習手順にしたがって学習機能の利用を促すことができる．しかし，プロセスをプログラムとして記述できるのは全体から見た限られた部分であり，不確定な行動を確定的なものと捉えることも難しいとされている [8]．学習者の行動も不確定なものであり，学習プロセス全体を記述するのは難しいので，本研究では，Web における機能の実行方法を整理し，一部の機能間で制御する方法を実現した．

BPEL[10] は，実行可能なビジネスプロセスを記述するための言語で，複数の Web サービスを連携させることで，複雑なプロセスフローを定義することができる．BPEL でも複数の機能を制御する記述が可能である．しかし，学習者の演習プロセス全体を記述することは難しく，ボタンの追加などのユーザインタフェースの記述はできない．

機能の実行方法を変更することはアスペクト指向プログラミングと似ているが，本研究で提案する方法では，ユーザインタフェースの変更を行うことができ，課題ごとに異なるカスタマイズを行い，動的に変更することができる．これらをプログラムの記述ではなく，設定の記述で変更することができる．

6 おわりに

本研究では，学習目的や，教員の意図にあわせて，カスタマイズが可能な学習環境の提案を行った．機能を追加するための学習環境のフレームワークを設計し，機能間の制御のために各機能の入出力の標準化を行った．教員がより容易にカスタマイズを行うことができる仕組みを提案し実装した．教員がカスタマイズを行う際の変更箇所や記述量を減らしたり，簡単に記述ができるようになった．

今後の課題として，Presenter Configuration に Presenter のカスタマイズを記述できるような拡張や実行前に静的にチェックする仕組みの実現があげられる．

参考文献

[1] 蜂巣吉成，小林悟，吉田敦，阿草清滋：プログラミング演習におけるテストケース評価システムの提案，コンピュータソフトウェア，Vol.34，No.4，pp.54-60 (2017).

[2] 蜂巣吉成，吉田敦，桑原寛明，阿草清滋：プログラミング学習用プルーフリーダの試作，コンピュータソフトウェア，Vol.35，No.4，pp.129-135 (2018).

[3] Mike Potel: MVP:Model-View-Presenter the taligent programming model for C++ and Java, Taligent Inc., Tech. Rep. (1996).

[4] K. Sharan: Model-View-Controller Pattern, in Learn JavaFX 8, pp.419-434 (2015).

[5] Anthony I.Wasserman: Tool integration in software engineering environments, LNCS, vol.467, pp.137-149, Springer (1990).

[6] Ian Sommerville: Software Engineering (5th Edition), Addison-Wesley (1995).

[7] L. Osterweil: Software processes are software too, Proc. of the 9th International Conference on Software Engineering, pp.2-13 (1987).

[8] 玉井哲雄:ソフトウェア工学の基礎, 岩波書店 (2004).

[9] G. E. Kaiser: Experience with Marvel, Proc. of the 5th international Software Process Workshop, pp.82-88 (1990).

[10] OASIS:Web Services Business Process Execution Language Version 2.0,（オンライン）入手先〈http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf〉（2007）