BlockEditor Hinoki:ビジュアル-Java 相互変換システムを利用したオブジェクト指向入門教育の実施と分析

BlockEditor Hinoki: Practice and Analysis of Object-Oriented Programming Education Using Mutual Language Translation Between Visual And Java

大畑 貴史[*1]， 松澤 芳昭[*1]，桐山 伸也[*1]，酒井 三四郎[*1]
Takashi Ohata[*1], Yoshiaki Matsuzawa[*1]，Shinya Kiriyama[*1]，Sanshiro Sakai[*1]
[*1] 静岡大学大学院情報学研究科
[*1]Graduate School of Informatics, University of Educational Systems
Email: Ohata@sakailab.info

あらまし：オブジェクト指向プログラミング(OOP)の授業では，構文エラーによってオブジェクト指向(OO)の概念学習が阻害されてしまう学習者が観察される．本研究ではビジュアル-Java 相互変換システム「BlockEditor」の OO 指向拡張版（コードネーム：Hinoki）を開発し，プログラミング入門教育を修了した大学２年生向けの OOP コースで本システムを利用した授業を実施した．当該年度の言語選択状況やプログラムの質などの年度比較を通じ，本システムの有用性を検証する．
キーワード：プログラミング教育，オブジェクト指向，ビジュアルプログラミング，Java

1. はじめに

オブジェクト指向プログラミング（OOP）の授業では，オブジェクト指向（OO）の概念理解が重要であるが，概念理解が困難な学習者もいる．OO の概念理解が困難である理由の一つに構文エラーが考えられる．学習者は構文の学習やエラー修正に注力してしまい，OO の概念理解が阻害されてしまうためである．

本研究では，松澤らによって提案されているビジュアルプログラミング言語（VPL）とテキスト記述型言語の相互変換によるプログラミング入門教育支援システム（BlockEditor）[(1)]のオブジェクト指向構文への対応と，当システムを利用した OOP の授業を実施した．当該年度の言語選択状況や，プログラムの質などの年度比較を通じて当システムの利用形態と有用性を分析する．

2. OOP の授業での問題点

2.1 本学 OOP の授業

本学の OOP 入門講義は IS（information System）を専攻する学生に対して行われており，文科系の学科の学生と科学系の学科の学生が１：１の割合で受講している．

授業は松澤らによる OOP の教材[(2)]を，この講義用に再編したものを使用している．

2.2 授業での問題点

講義担当の教員からヒアリングを行った結果，次のような学習者の存在が問題として挙げられた．

問題点1 躓きによる課題の蓄積
問題点2 他クラスのインスタンス変数の直接参照

1 の学習者に関して，本学の OOP の講義では，OOP の基本の学習の題材にシューティングゲームを扱っている．ゲームを毎週拡張して学習を行う．課題が回ごとに独立していないため，躓いて課題ができなかった学習者は次の課題に取り組むことが出来ないという問題がある．

2 の学習者に関して，このような学習者は，カプセル化されたクラス設計や，概念学習，利点などの学習機会を損なってしまうという問題がある．

3. 先行研究

松澤らによる BlockEditor は OpenBocks[(3)]を基盤に開発されている．VPL とテキスト記述型言語の相互変換により，言語間のシームレスな移行を可能にしている．しかしオブジェクト指向構文への相互変換には対応しておらず，オブジェクト指向の学習に利用することができない．

4. BlockEditor Hinoki

BlockEditor Hinoki の外観を図 1 に示す．左側の「ファクトリ」からブロックを取り出し，キャンバスに設置することでクラスを設計することが可能である．Eclipse のプラグインとして開発している．

図 1 BlockEditor Hinoki の外観

4.1 設計目標

OOP の授業で利用できるブロック型言語とテキスト記述型言語の相互変換能力があり，学習者自身

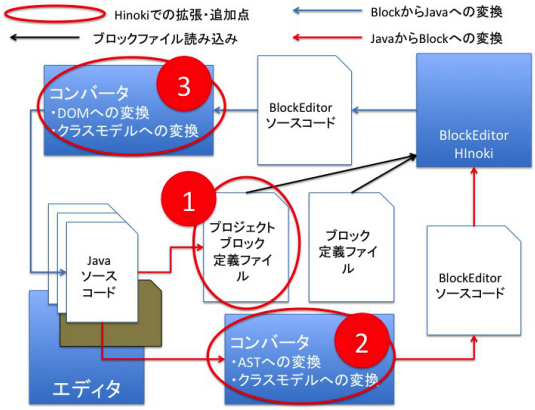

図 2 BlockEditor Hinoki のアーキテクチャ

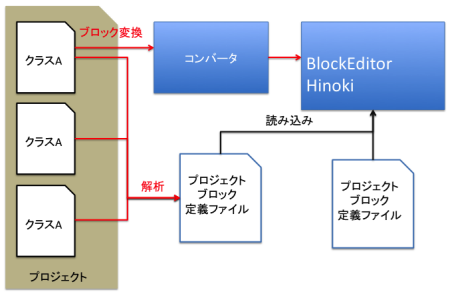

図 3 動的ブロック作成のアーキテクチャ

が設計したクラスをブロック型言語で利用できることが設計目標である.

4.2 アーキテクチャ

BlockEditor Hinoki のアーキテクチャを図 2 に示す. アーキテクチャの変更点は，1). プロジェクトを動的に解析し，「プロジェクトブロック定義ファイル」を作成する，2). 式や文の抽象構文木(AST)への変換の拡張，である.

4.3 動的なブロックモデルの作成

図 3 に動的なブロックモデル作成のアーキテクチャを示す.

BlockEditor Hinoki では同一プロジェクト内の Java ファイルを解析し，学習者の作成したクラスのブロックモデルの記述された「プロジェクトブロック定義ファイル」を作成している．基本的なブロックモデル（変数など）は，「ブロック定義ファイル」に記述されている.

この 2 つのファイルを読み込むことで，基本的なブロックと学習者自身が設計したクラスオブジェクトのブロックを利用可能にしている.

4.4 OO 構文への相互変換能力の拡張

OO の Block から Java へ変換する際の AST への変換，Block から Java へ変換する際の DOM への変換に対応した．これにより表現できるプログラム構成要素が増え，OOP に対応した表現が可能である.

5. ブロックの仕様

ブロックはプログラムの処理を抽象的な日本語で

図 4 銀行口座クラスをブロックに変換した例

表現するように設計している．ブロックで表現された銀行口座クラスを図 4 に示す．このクラスには，残高を表すインスタンス変数とコンストラクタ，預金するメソッドが定義されている.

6. 評価と今後の展望

BlockEditor は本学の OOP コースで利用されている 48 個の構文要素のうち 47 個に対応している. 未対応な構文要素はコメント文である．これは他の方法で代用可能であるため，OOP の授業に利用可能な相互変換能力があると評価できる.

現在本学の学部 2 年生を対象とした OOP の授業で BlockEditor を利用している．現在授業で提出されたソースコードの年度間の比較，分析を行う．プログラム作成履歴の分析には Programming Process Visualizer(PPV)[4]を利用する.

2.2 節での問題 1 に関しては，課題の提出率を昨年のものと比較し，期限内に課題を提出できる学習者が増えることを検証している．問題 2 に関しては，PPV を用いて学習者のソースコードを分析し，private の インスタンス変数が増えることを検証している.

その他，学習者の BlockEditor の利用推移や，記述された処理を分析し，学習者の利用形態を分析する. 本発表ではこれらの検証結果に関して報告を行う.

参考文献

(1) 松澤芳昭，保井元，杉浦学，酒井三四郎：“ビジュアル-Java 相互変換によるシームレスな言語移行を指向したプログラミング学習環境の提案と評価 “，情報処理学会論文誌，No.55，No.1，pp.57-71，2014.

(2) 松澤芳昭,大岩元:情報系学生を対象としたオブジェクト指向までのプログラミング入門教育の実践と課題，情報教育シンポジウム(SSS2009)，pp199-206.

(3) Ricarose Vallarta Roque. Openblocks: An extendable framework for graphical blockprogramming systems. Master thesis at MIT, 2007.

(4) 松澤芳昭,岡田健,酒井三四郎: Programming Process Visualizer:プログラミングプロセス学習を可能にするプロセス観察ツールの提案，情報教育シンポジウム 2012 論文集(2012).