Vol.2022-CE-164 No.15
2022/3/12

昔話や童話を題材としたオブジェクト指向ソフトウェア設計の学習法

三浦 元喜[1,a)]　片桐 僚太[1]　酒井 遼太[1]

概要：オブジェクト指向ソフトウェア設計を学習するうえでは，オブジェクト指向の考え方や概念を理解したうえで，それを用いて実際の開発に適用したり，説明したりできることが求められる．我々は，学習者が具体的なシナリオや状況を想定しやすくすることと，それらを他の学習者に説明したり，意味を共有したりしやすくするため，昔話や童話といった「物語」を題材とする学習法を提案する．この学習法では，学習者が具体的な昔話や童話を選択し，それを表現するのに必要なクラスを定義することで，物語の設定や関係，状態や心情の変化といったストーリーを表現する．また，設計や実装の詳細を説明することによって理解を深める．実際に大学院の講義で学習者が作成・発表した作品をもとに，提案手法の有効性や可能性について議論する．

A Learning Method for Object-Oriented Software Design based on Old and Fairy Tales

Motoki Miura[1,a)]　Ryota Katagiri[1]　Ryota Sakai[1]

1. はじめに

オブジェクト指向ソフトウェア設計を学習するうえでは，オブジェクト指向の考え方や概念を理解したうえで，それを用いて実際の開発に適用したり，説明したりできることが求められる．本位田ら [1] は「オブジェクト指向アプローチの発想の原点は，現実のモノおよびモノどうしの関係をそのままソフトウェアで表現することによって，現実世界の仕組みをコンピュータ上で再現しようとするものである.」と述べている．この考え方に基づき，(1) オブジェクトモデル・アプローチ，(2) 動的モデル・アプローチ，(3) 機能モデル・アプローチといったアプローチによる分析設計手法が開発されてきた．これらの 3 つのアプローチはオブジェクトモデル・動的モデル・機能モデルのうちのどのモデルの構築に重点を置く（あるいは最初に構築する）かによって分類されており，実際には分析・設計・実装の 3 つのフェーズを行ったり来たりしながら開発する形態がとられている．

しかし，初学者がオブジェクト指向の概念を理解し，実際にソフトウェア設計・開発できるようになるまでの道のりは長く，苦難を伴うこともある．これを緩和するため，筆者の一部は，オブジェクト指向プログラミング言語におけるクラスとオブジェクトの関係や，変数との関係，オブジェクトの振る舞いといった基礎概念の理解を促進させるための視覚化ソフトウェア Anchor Garden を開発し，実践を行ってきた [2]．学習者はマウス操作により視覚的な表現を持つ “変数” や “オブジェクト” を生成したり，変数からオブジェクトにリンクを張ったりする操作体験を通じて，オブジェクト指向プログラミング言語における独特の考え方に接近することができる．クラス継承についても変数からオブジェクトへの参照や，メソッド呼び出しによって一部対応している．しかし，これらの概念を理解したとしても，それらを現実の事象に適用し，分析・設計・実装の 3 つのフェーズを行ったり来たりしながら開発するには，学習者が自分で様々な状況を設定し，概念を適用したり，な

[1] 千葉工業大学 工学部
Faculty of Engineering, Chiba Institute of Technology
2-17-1 Tsudanuma, Narashino, Chiba, 275-0016, Japan
[a)] motoki.miura@p.chibakoudai.jp

ぜそのようにしたのかを説明できるスキルが必要である．

我々は，学習者が具体的なシナリオや状況を想定しやすくすることと，それらを他の学習者に説明したり，意味を共有したりしやすくするため，昔話や童話を題材としてオブジェクト指向分析・設計・実装について学習する方法を提案する．この学習法では，学習者が具体的な昔話や童話を選択し，その物語を表現するのに適切かつ必要なクラスやインタフェース等を定義していく．また，コードとして記述・実装することによって物語に登場する人物やキャラクタ間の関係と動作を表現し，実行によってストーリーの進行と展開が行われるため，動作確認しながら静的モデル/動的モデルの両側面から考察するスキルが習得できる．

2. 関連研究

兼宗らは，学校教育用オブジェクト指向言語ドリトルを開発している [3]．ドリトルでは，初等教育での導入を考慮してプロトタイプ方式を採用し，クラスや継承などの高度な抽象概念の理解を不要にしている．また [4] では，中等教育への展開と，実践評価を行っている．ドリトルは初学者向けのオブジェクト指向言語であるが，言語体験を重視しプロトタイプ方式を採用しているため，本研究が対象とするクラスや継承といった高度な抽象概念の導入と適用には向いていないと考えられる．

渡辺らは，物語導入型教材コンテンツによるオブジェクト指向プログラミング教育の実践を行っている [5]．[5] では，教材コンテンツにキャラクターを登場させ「物語」を持たせることで，学習者が，(1) 学習対象の概念が用いられる状況や課題の糸を理解しやすくすること，(2) 教材に親しみを持ち，学習内容に入り易くすること，を目指している．具体的には，架空の会社に入社した新人が，先輩社員とのやりとりを通して，オブジェクト指向プログラミングを習得していくという設定としている．部長が設計した宣言部を提示し，それに合わせて新人が書いた実装部の誤り部分を指摘する問題や，新人になったつもりでメインプログラムを書く問題などである．高井らによる実践 [6] では，物語導入型教材コンテンツ導入により，オブジェクト指向の考え方に適合するプログラムを提出する学生の割合が増加したと報告している．カプセル化，メッセージとその反応としてのメソッド，クラスとインスタンス，例外処理，継承などを理解することを学習目的としている点は本研究と類似している．[5] では，課題への導入や課題を解く上での制約を説明するために，教員が教育的観点に配慮しながら物語を作成していた．しかし，本研究で提案する「昔話や童話を題材とした手法」は，学習者自身が物語を自由に選定したうえで，その世界観やストーリーを表現するための設計と実装を行う点が異なる．

早川らは，クラス図に対応したオブジェクト図を自動生成し学習者に提示するシステムを開発した [7]．学習者が自ら作成したオブジェクト図と比較照合するユースケースと，クラスモデルの多重度誤りの発見を支援するユースケースについて検証および有用性の評価を行っている．

Tanielu らは，VR 空間内で「家」と「設計図」のアナロジーを取り入れたオブジェクト指向概念学習システムを構築している [8]．VR を用いることで，没入感と深い関わりを促進する．メソッド呼び出しと戻りは部屋への出入りで表現され，引数は窓の下枠に箱を置くことで表現している．

赤山らは，オブジェクト指向モデリング教育におけるモデル駆動開発 (MDD) ツールを活用する利点を示している [9]．MDD ツールを用いることで，クラス図やステートマシン図を描いた後，すぐにコード生成と動作確認を行える．MDD ツールを用いた場合と用いなかった場合を比較したところ，MDD ツールを用いたグループは機能ごとの動作確認が行えるためボトムアップ的な開発を行う傾向があり，MDD ツールを用いなかったグループはトップダウン的に開発する傾向があったと報告している．

Knuth の「文芸的プログラミング (Literate Programming)」[10] は，プログラムと解説文書をひとつの文芸的ファイルに混在させて記述することで，その両者の整合性を保ちながら開発を行えるプログラミングパラダイムである．プログラムを文芸的に記述するという観点では一部類似しているが，本研究が提案する手法は，オブジェクト指向プログラミングのパラダイムを用いて物語を表現することによって，学習者の知識理解および定着をはかるものである．

Web 記事「ストーリーテリングなコードを書こう」[11] では，オブジェクト指向の具体的なメリットを説明するため，方向性と思想が異なる 2 種類のコードを示しながら，「ストーリー感」とその重要性について説明している．ストーリー感を重視したプログラムは「自然と，クラスは主語となり，メソッドは述語となり，引数は目的語となる」と述べており，本研究提案における基本的な思想と一致している．本研究では，オブジェクト指向ソフトウェア設計の学習法として，昔話や童話といった具体的な物語を学習者自身が想起し，そのストーリーをどのようにプログラムで表現するかを考えてもらうことによって，オブジェクト指向特有の概念や考え方と，その具体的な実装の両面について学ぶ方法を提案している．

第二言語としての英語を体験的に学習する手法として，「英語劇」という手法があり，小学校 [12] や大学 [13] で活用されている例が報告されている．本研究は，自然言語ではなく，プログラミング言語であるが，慣れ親しむためにストーリーを用いるという点では類似性がある．

3. 手法

本研究で提案する「物語を題材としたオブジェクト指向ソフトウェア設計学習」は，学習者（個人またはグループ）

自身が，特定の「物語」を想定したうえで，その物語が進行する様子をオブジェクト指向の考え方を適用しながら図やソースコード，実行結果で表現するものである．また，表現をしたうえで，それらを学習者が説明する活動も含むものである．

3.1 内容

物語には，通常登場人物やキャラクタが登場し，相互に関わりあいながら時間軸に沿って進展していく．登場人物やキャラクタは味方や敵，人間や動物といった種別や，年齢性別，所有しているアイテム，抱いている感情といった属性によって構成することができる．また，関係性を静的に表現したり，態度や対応を動的に変化・生成したりすることもある．アイテムについても状態や関係をもち，ある条件を満たしたときだけ効力を発揮するといった設定が行われることがある．こうした物語中の「種別」や「属性」，「状態」や「関係」の変化を，オブジェクト指向ソフトウェアの分析・設計・実装によって表現することは，日常や身の回りの生活状況下で考えるのと同等か，それ以上の自由度があると考えられる．また具体的な「物語」の前提条件や状況設定が適切な制約となり，共通認識や理解が促進されることで，コミュニケーションが行いやすい．

3.2 想定されるメリット

提案手法で「物語を題材とする」ことによって想定されるメリットとして，以下の点が挙げられる．

- 【ストーリー自体の魅力】 既存の昔話や童話といった「物語」は，長い年月を経て伝承され親しまれてきたものであり，ストーリー自体の魅力が高い．
- 【多様性と自由度】 題材となる物語は無数に存在するため，物語の選定自由度は高い．
- 【具体性】 物語を選定したあとは，登場人物や事象などが具体的に定まるので分析・設計・実装しやすい．
- 【工夫やアレンジが可能】 物語をそのまま表現するほかに，「もし〜だったら」といったアレンジを加えることも容易である．
- 【物語とプログラム表現のギャップ】 古典的な物語を現代的なプログラムで表現するギャップが新奇性や味わいを生み出すこともある．
- 【他の学習者と意図を共有しやすい】 物語自体の知名度が高い場合は，他の学習者に説明するときに意味・意図を共有しやすい．

これらに加えて，物語に含まれる静的な関係だけでなく，時間軸に沿った動的な側面の分析・設計・実装を自然かつ横断的に扱える点も有効と考えられる．統合開発環境 (IDE) を用いれば，設計や実装の適切さを簡単に確認することができる．

3.3 懸念される問題点

「物語を題材としたオブジェクト指向ソフトウェア設計学習」で懸念される問題点について以下に挙げる．

- 【物語をうまく選べない】 物語選定の自由度が高すぎることで，学習者が物語の選定時点でつまずいてしまう可能性もある．また，物語の選定に時間をかけすぎてしまう恐れがある．
- 【オブジェクト指向やプログラミングの概念を適用できない】多くの物語において，そのストーリーやプロットの一部にオブジェクト指向やプログラミングの概念を適用させることは可能であると考えている．しかし，選定する物語の展開によっては，うまく適用できない場合も考えられる．
- 【あまり知られていない物語を題材とすると，効果が低減する】物語の知名度が低い場合，物語が知識として共有されている，という条件を満たさなくなる．そのため，上記メリットで挙げた「他の学習者と意図を共有しやすい」という効果が低減する恐れがある．

4. 実験講義

提案する「物語を題材としたオブジェクト指向ソフトウェア設計学習」が学習者に与える影響を調査するため，我々は講義のなかで演習を行った．

4.1 講義および課題の内容

大学院の講義「システムソフトウェア特論」（後期全 13 週，1 週 120 分，2021 年度は完全オンライン実施）の前半では，Java 言語を用いたプログラミングを通じて，オブジェクト指向の基礎とデザインパターンについて説明と演習を行っている．この前半部分の学習のまとめとして，第 5 回 (10 月 19 日) の最初（デザインパターン導入の前）に受講者に以下の課題を与え，取り組みかたやレポート作成方法について 40 分程度の説明をした．

課題の説明

具体的な物語（昔話やおとぎ話） を 1 つ選び，そのストーリーを表現するのに必要なクラスを複数定義したうえで，インスタンス（オブジェクト）の具体的な挙動によって話の流れを表現して動作を確認できる Java プログラムを作成せよ．

この課題を提示した目的は，前半で学習した「オブジェクト指向の基礎」の知識や概念を身につけており，自分独自の対象に適用してクラスを設計することができるかと，設計したクラスを実際に使うことで，設計の有効性および問題点に気づく能力を問うためである．

この課題には，受講者は個人またはグループで取り組んでもらった．受講者は，物語を選択し，それをクラスでど

う表現するかを考えつつ，プログラムを作成した．また，レポートとして，UML で記述した図やソースコード，実行結果などを掲載した Web ページを作成してもらった．その Web ページを受講者間で共有しながら，発表会を第 8 回 (11 月 9 日) の講義時に行った．つまり，課題提示から発表会までは 3 週の期間があった．発表会では 1 テーマあたり 5 分程度で説明するように指示した．

この課題を提示する際，受講者に「桃太郎」を題材とした以下のソースコード例と動作例を示して説明した．**図 1** は，桃太郎のメインメソッドを含むクラスの定義であり，桃太郎の冒頭部分をプログラムで表現している．**図 2** と **図 3** は，それぞれ桃太郎に登場する「モノ」と「生き物」に関するクラス定義である．このプログラムを実行した結果を**図 4** に示す．このような例を示しつつ，課題に取り組むにあたって以下のことを説明した．

- 「物語」は，皆がよく知っているものがのぞましい．
- 考え方としては，最初に登場人物や，登場するモノをオブジェクトで表現するため，クラスを作成する．その際，階層関係があれば継承で表現し，共通のメンバやメソッドは親クラスで定義する．
- 登場人物のあいだでのやりとりや，モノの受け渡しなどを，メソッド呼び出しで表現する．そのために必要なメソッドを追加していく．
- 主語. 動作 (述語) の形式で呼び出すほうが，プログラムの可読性は高くなる．

設計ツールや開発環境はとくに制限・指定しなかったが，多くの受講者は講義で導入した Visual Studio Code と，その拡張である Extension Pack for Java を利用したものと思われる．

```java
public class MyApp {
    Run | Debug
    public static void main(String[] args) throws Exception {
        // むかしむかし、あるところに、おじいさんとお婆さんが住んでいました
        Ojiisan ojiisan = new Ojiisan();
        Obaasan obaasan = new Obaasan();
        // お婆さんは川で桃を拾ってきました
        Momo momo = new Momo();
        // 桃から、桃太郎が生まれました
        Momotaro taro = momo.bornMomotaro();
        // お婆さんは、桃太郎に、きび団子の入った袋を渡しました
        obaasan.tsukuruKibi();
        obaasan.watasuFukuro(taro);

        // 犬がいました。桃太郎は犬にきび団子を渡しました
        Inu inu = new Inu();
        taro.watasuKibi(inu);
        // 猿がいました。桃太郎は猿にきび団子を渡しました
        Saru saru = new Saru();
        taro.watasuKibi(saru);
        // キジがいました。桃太郎はキジにきび団子を渡しました
        Kiji tori = new Kiji();
        taro.watasuKibi(tori);
    }
}
```

図 1 桃太郎のメインメソッド

```java
import java.util.ArrayDeque;
public class Mono {
    String name = "";
    public Mono(String name){
        this.name = name;
    }
    @Override
    public String toString(){
        return name;
    }
}
class Momo extends Mono {
    Momo(){
        super("桃");
    }
    Momotaro bornMomotaro(){
        return new Momotaro(); //桃太郎が生まれる
    }
}
class Kibidango extends Mono {
    Kibidango(){
        super("きび団子");
    }
}
class Fukuro extends Mono {
    ArrayDeque<Mono> fukuro;
    Fukuro(){
        super("きび団子が入った袋");
        fukuro = new ArrayDeque<Mono>();
    }
    public void add(Mono mono){
        fukuro.add(mono);
    }
    public Mono pop(){
        return fukuro.pop();
    }
    public int size(){
        return fukuro.size();
    }
}
```

図 2 桃太郎に登場する「モノ」に関するクラス定義

発表会終了直後に，我々は受講生に対して，以下の設問項目からなるアンケートを実施した．

(1) オブジェクト指向を説明する「ものがたり」を決めるのに，どれくらい時間がかかりましたか？
(1) 1 時間程度 (2) 2 時間程度 (3) 3 時間程度 (4) 4 時間程度 (5) それ以上

(2) 決めた「ものがたり」を表現するクラスを設計するのに，どれくらい時間がかかりましたか？ (UML 図などを作成する時間は除く)
(1) 1 時間程度 (2) 2 時間程度 (3) 3 時間程度 (4) 4 時間程度 (5) それ以上

(3) クラスを設計したあと，「ものがたりのストーリー」を表現するプログラムを作成するのに，どれくらい時間がかかりましたか？ (1) 1 時間程度 (2) 2 時間程度 (3) 3 時間程度 (4) 4 時間程度 (5) それ以上

(4) 「ものがたり」を表現するプログラムを作成するときに，難しかった点を回答してください．可能であればクラス定義と，ストーリーを表現するプログラムに分けて回答してください．

```java
public class Ikimono {
    String name = "";
    void printName(){
        System.out.println(name);
    }
    void morau(Mono mono){
        System.out.println(name + "が "+mono.toString()+" をもらった");
    }
}
class Momotaro extends Ikimono {
    Fukuro kibiBukuro;
    Momotaro(){    name = "桃太郎";   }
    void morau(Mono mono){
        super.morau(mono);
        if (mono instanceof Fukuro) kibiBukuro = (Fukuro)mono;
    }
    void watasuKibi(Ikimono dareka){
        Kibidango oneKibi = (Kibidango)kibiBukuro.pop();
        dareka.morau(oneKibi);
        System.out.println("  残り "+kibiBukuro.size()+" 個");
    }
}
class Ojiisan extends Ikimono {
    Ojiisan(){  name = "お爺さん";   }
}
class Obaasan extends Ikimono {
    Fukuro fukuro;
    Obaasan(){
        name = "お婆さん";
        fukuro = new Fukuro();
    }
    void tsukuruKibi(){
        fukuro.add(new Kibidango());
        fukuro.add(new Kibidango());
        fukuro.add(new Kibidango());
    }
    void watasuFukuro(Ikimono dareka){
        dareka.morau(fukuro);
        fukuro = null;
    }
}
class Inu extends Ikimono {
    Inu(){   name = "犬";    }
}
class Saru extends Ikimono {
    Saru(){  name = "猿";    }
}
class Kiji extends Ikimono {
    Kiji(){  name = "キジ";   }
}
```

図 3　桃太郎に登場する「生き物」に関するクラス定義

```
桃太郎が きび団子が入った袋 をもらった
犬が きび団子 をもらった
  残り 2 個
猿が きび団子 をもらった
  残り 1 個
キジが きび団子 をもらった
  残り 0 個
```

図 4　桃太郎プログラムの実行結果

(5) オブジェクト指向の考え方で「ものがたり」を表現するプログラムを作成することで，どのような知識や技能が習得できたとおもいますか？
(6) 「ものがたり」以外に，プログラムで表現してみたいものや，プログラムで表現したら学習になるとおもう「対象」があったら書いてください．
(7) その他，苦労した点や，今後の改善案があれば書いてください．

4.2　受講者の作品

表 1 に，受講者が作成したプログラムのクラス数，行数と，発表会における説明時間を示す．わらしべ長者以外については，すべて動作する Java プログラムをもとに説明を行っていた．わらしべ長者については，物々交換を行う概念コードと，登場キャラクタ，アイテムに対応するクラス定義が提出された．いずれの作品も，継承関係にある 5 個以上（平均 9 個）のクラスを定義していた．また「かぐや姫」と「金の斧銀の斧」は，それぞれ第 5 回後半で説明した Singleton と Template Method を利用し，発表時に言及していた．「大きなカブ」は，再帰的な構造を生かしたプログラミング技術の高い作品であった．「一寸法師」は，小槌をつかうと身長が 2 倍になるという設定を取り入れていた．いずれの作品および発表も，設計の意図について明確に説明できていた．

表 1　受講者が作成したプログラムのクラス数，行数と説明時間

物語の題材	クラス数	行数	説明時間
金の斧銀の斧	8	162	5 分 36 秒
三匹の子豚	11	110	3 分 27 秒
大きなカブ	9	138	5 分 58 秒
わらしべ長者	14	117	3 分 55 秒
かぐや姫	7	238	4 分 37 秒
一寸法師	5	76	3 分 58 秒
平均	9.0	140.2	4 分 35 秒
標準偏差	3.2	55.9	60 秒

4.3　アンケート結果

設問項目 (1)〜(3) に対する回答に基づき，受講生が物語の決定，クラス設計，プログラム作成におおよそどれくらいの時間を費やしたかを**図 5** に示す．この結果から，ほとんどの受講生は 1 時間程度で物語を決定できており，クラス設計やプログラム作成にそれよりも多くの時間を費やしていることが読み取れる．なお，受講生番号と表 1 に示した物語の題材の並び順は対応していない．

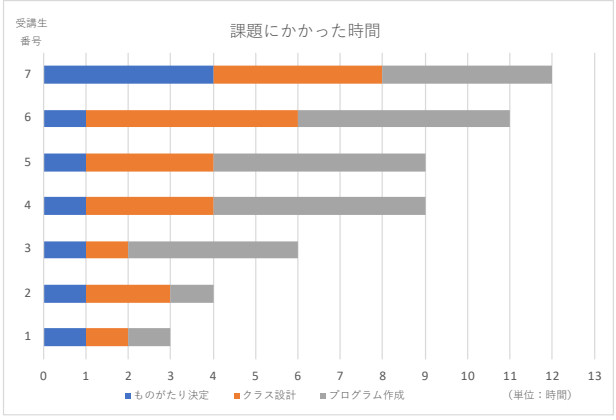

図 5　アンケート項目 (1)〜(3) の結果

設問項目（4）に対する，学生の回答を以下に示す．

- 頭の中である程度構想がまとまっていたが，実際に書き始めてみると色々な書き方で表現できることを学んだので，思うようにプログラムを完成させるのが難しかったです．
- クラス定義の際にどこまで汎化，特化を行えばいいか少し迷いました．
- 親クラスのメソッドになにを定義してあげるか悩んだ．
- ストーリーを表現するプログラムの時，同じような動作をさせるときに，条件分岐をしたとき if 文だと複雑になりすぎて把握しづらくなる

設問項目（5）に対する，学生の回答を以下に示す．

- 実際に考えながら書くことで，「この知識があればここの部分はこう変えることで良くなるな」という風に学んだ知識を使うことができたと思います．また，考える力と分からないことがあったときに的確にその情報を調べることが少しできるようになったと思います．
- デザインパターンを使わず，ものがたりを作ったが調べるうちにデザインパターンはこういう場合には使えるんだと理解することができた．
- これといって習得できたことはありませんが，実際にオブジェクト指向を用いて何かを作ることで理解がより深まりました．
- デザインパターンを用いることで形がある程度決められる．

設問項目（6）に対する，学生の回答を以下に示す．

- ゲームの有名なワンシーンを表現してみたいと思いました．
- 会社などの組織
- ものがたりをプログラムで表現するアイデアはとても学びになると思いました．ほかに思いつきません．

4.4 議論

今回実験講義で提案手法に基づく課題を提示した結果，すべての学習者は課題の意図を理解し，オブジェクト指向設計の考え方を物語に適用することができたといえる．また，課題遂行時間の全体における物語選定時間の割合は多くても 1/3 程度であった．このことから，物語の選定から学習者に考えさせる方法をとったとしても，大学院レベルの学生であれば問題なく課題遂行できることがわかった．

また，設計や実装には唯一の解答があるわけではなく，状況や目的によって適切な設計や実装が異なることへの気付きも得られていた．

講義では第 5 週から第 7 週にかけてデザインパターンについて説明したため，デザインパターンを関連させた実装や発表もみられた．アンケート結果からもデザインパターンに関する言及がみられた．物語を表現する課題にどう取り組むかを考えながら講義を聞くことで，知識を活用する意識が強まった可能性がある．

5. さいごに

物語中の「種別」や「属性」，「状態」や「関係」の変化を，オブジェクト指向ソフトウェアによって表現することによる分析・設計・実装の学習法を提案した．提案する学習法を大学院の講義における課題として課したところ，学習者は課題の意味を理解し，物語を自分たちで選定したうえで，適切に適用できることがわかった．また，課題において，学習者はデザインパターンやアルゴリズム上の工夫などを追加で適用し，作成した作品がプログラム的により多くの意味を持つようなアレンジやアイデアを取り入れていた．これらのことから，提案する手法は大学院レベルの教育では有効な方策であるといえる．

謝辞

本研究の一部は JSPS 科研費（課題番号 19K03056）の支援によるものです．

参考文献

[1] 本位田真一, 山城明宏. オブジェクト指向分析・設計. 情報処理, Vol. 35, No. 5, 1994.
[2] 三浦元喜, 杉原太郎, 國藤進. オブジェクト指向言語における変数とデータの関係を理解するためのワークベンチ. 情報処理学会論文誌, Vol. 50, No. 10, pp. 2396–2408, October 2009.
[3] 兼宗進, 御手洗理英, 中谷多哉子, 福井眞吾, 久野靖. 学校教育用オブジェクト指向言語「ドリトル」の設計と実装. 情報処理学会論文誌プログラミング (PRO), Vol. 42, No. SIG11 (PRO12), pp. 78–90, 2001.
[4] 兼宗進, 中谷多哉子, 御手洗理英, 福井眞吾, 久野靖. 初中等教育におけるオブジェクト指向プログラミングの実践と評価. 情報処理学会論文誌プログラミング (PRO), Vol. 44, No. SIG13 (PRO18), pp. 58–71, 2003.
[5] 渡辺博芳, 佐々木茂, 高井久美子, 武井惠雄.「物語」導入型教材コンテンツを用いたオブジェクト指向プログラミング教育の実践例. 情報教育シンポジウム 2002 論文集, Vol. 2002, No. 12, pp. 133–138, 2002.
[6] 高井久美子, 佐々木茂, 渡辺博芳, 荒井正之, 武井惠雄.「物語」導入型教材コンテンツを活用したセルフラーニング型授業—オブジェクト指向プログラミング教育における実践例—. 教育システム情報学会誌, Vol. 24, No. 2, pp. 106–116, 2007.
[7] 早川勝, 野沢光太郎, 松澤芳昭, 酒井三四郎. オブジェクト指向モデリング教育のためのオブジェクト図自動生成システムの設計と評価. 情報処理学会論文誌, Vol. 54, No. 1, pp. 66–79, 2013.
[8] Tevita Tanielu, Raymond 'Akau'ola, Elliot Varoy, and Nasser Giacaman. Combining analogies and virtual reality for active and visual object-oriented programming. In *Proceedings of the acm conference on global computing education*, pp. 92–98, 2019.
[9] 赤山聖子, 久住憲嗣, 部谷修平, 福田晃. オブジェクト指向モデリング教育におけるモデル駆動開発ツールの活用方法の検討. 情報処理学会論文誌, Vol. 55, No. 1, pp. 72–84, 2014.
[10] Donald Ervin Knuth. Literate Programming. *The Computer Journal*, Vol. 27, No. 2, pp. 97–111, 1984.

[11] Roku. ストーリーテリングなコードを書こう. https://zenn.dev/ad5/articles/6780d514ed8cda6bdf0f, September 2020. (2022 年 2 月 2 日確認).
[12] 西崎有多子. 小学校外国語活動における「桃太郎」を使った授業展開-英語劇化への過程と民話としての側面. 東邦学誌, Vol. 41, No. 3, pp. 1–21, 2012.
[13] 日高真帆. 英語劇の上演と大学教育への応用. 2012.