コンピュータと教育 64－1
（２００２．５．１７）

オブジェクト指向技術者養成のためのカリキュラム

松澤　芳昭[†1]　　中鉢　欣秀[†1]　　岡田　健[†2]　　大岩　元[†3]

[†1] 慶應義塾大学 SFC 研究所　[†2] 慶應義塾大学政策･メディア研究科
[†3] 慶應義塾大学環境情報学部

email: macchan@crew.sfc.keio.ac.jp

概要

企業の新入社員研修等において効果的にオブジェクト指向技術者を育てるための入門教育カリキュラム「オブジェクト指向哲学」を試作した。従来の教育では、知識の伝達を目的として「オブジェクト指向とは何か」を説明することが多い。これに対して本カリキュラムは「何故オブジェクト指向を用いるのか」という議論から、問題解決の手段としてそれらをどう適用するのかに重点をおいて教育する。また、オブジェクト指向の利点のうちプログラムの「可読性の向上」に絞って教育することで、オブジェクト指向の最も基本的な考え方を確実に習得させることを主眼としている。本カリキュラムを用いて実験授業を実施し、アンケート調査を行ったところ、本カリキュラムは有用であるという感触が得られた。

A Curriculum for Object-Oriented Software engineers

Yoshiaki Matsuzawa[†1]　Yoshihide Chubachi[†1]　Ken Okada[†2]
Hajime Ohiwa[†3]

[†1] Keio Research Institute at SFC
[†2] Graduate School of Media and Governance, Keio University
[†3] Department of Environmental Information, Keio University

Abstract

Object-oriented software engineers are required to understand essential views about the technology, to apply the technology to developing quality software. However, Japanese IT industries rear their engineers by training transferring the knowledge. In order to solve this, we developed a curriculum named "Object-Oriented Philosophy" for giving the learners underlying philosophy of object-oriented technology. A trial instruction was made for the engineers of an IT industry, and it was highly appreciated by them.

1. はじめに

オブジェクト指向技術の普及に伴って、オブジェクト指向技術者の需要が高まっている。しかしながら、日本では技術者の供給が追いついていないのが実態である。

文献[1]によると技術者教育は主としてOJT(On the Job Training)によって行われている。OJT による教育は技術者が一人前になるまでに最低でも 4〜5 年かかると言われており、急増するオブジェクト指向技術者の需要を満たすためには非効率である。

加えて、オブジェクト指向技術そのものの難しさが OJT に頼った教育をより困難にしていると考えられる。オブジェクト指向技術を実務で使えるようになるためには、OJT により得られる知識と経験だけではなく、オブジェクト指向の背景にある「考え方」を身に付ける必要がある。

しかしながら、「考え方」を身に付けるために、OJT は適さない。ここに 2 つの理由を挙げる。

1 つ目の理由は、OJT による教育において、学習者は実務をこなすための知識を習得する時間に追われてしまうことである。「考え方」を捉えるためには、知識と経験を体系化する時間が必要である。

2 つ目の理由は OJT の教育者である現場のプロが必ずしも教育のプロであるとは限らないことである。「考え方」を身に付けさせるためには、知識や経験を伝えるだけでなく、体系化された「考え方」を教育できる教育者が必要である。

こうした理由から、OJT を補完する効果的な入門教育カリキュラムが求められている。

また、筆者は入門教育においてオブジェクト指向の概念を「知識」として教育することの効果に疑問をもつ。オブジェクト指向技術者教育においては、単に「クラス」や「継承」といった概念を教えるのみならず、「問題解決の手段」としてそれらをどう適用するのかまで身につけさせる必要がある。しかしながら、既存の入門教育では安易にそれらの概念を知識として説明しているものが多い。

例として、「継承」を教育する方法を取り上げる。既存の教育では(図 1-1)のような例を用いて説明がなされる場合がある。この例は、哺乳類と人間という具体例を用いることによって、継承という概念を一見分かりやすく説明しているように見える。確かに、このような教育を受けた学習者は、継承というものがどういうものかという説明はできるようになる。

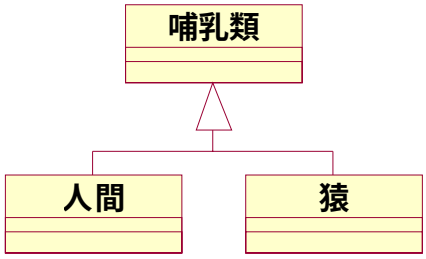

図 1-1 継承を教育する際に利用される例

しかし、この例が継承を適用することによって問題に対する何らかの解法を示しているとは一概に言い切れない。哺乳類という抽象化が問題解決に寄与するかどうかはコンテクストによって変わってしまうからである。

例のような入門教育を受けた技術者は、問題解決の手段としてオブジェクト指向の概念を適用するために、OJT により得られる経験だけを頼りとする。従って、単に継承の概念の知識を得るだけではオブジェクト指向の長所を生かしたプログラムが書けるようになるとは限らない。

以上のような問題意識から、本論文ではオブジェクト指向技術を効果的に教育できる入門教育カリキュラムの提案を行う。

このカリキュラムは、ガニェによる学習目標の分類理論に基づいてオブジェクト指向技術の基礎的学習目標を構成し、それらを効率よく教育できるようにした。本カリキュラムを用いて実験授業を実施し、アンケート調査を行ったところ、本カリキュラムは有用であるという感触が得られた。

2. 既存教育の効果が上がらない理由

本章では、既存教育の効果が上がらない理由を「学習目標の明確化と分類」という側面から分析する。

学習目標の明確化と分類の作業を行うことで、作業によって明らかとなる学習目標を、既存の学習目標と比較することができる。これによって、既存の教育における問題点をより明らかにできる。

2.1. 学習目標の分類

本研究において、学習目標の分類には、ガニェによる分類[2-6]を用いた。ガニェによると、学習目標は言語情報、知的技能、認知的方略、態度、運動技能という 5 つの種類に分類される。(表 2-1)

言語情報 (Verbal Information)	述べることができる知識 knowing what
知的技能 (Intellectual Skills)	やってみせられる知識 knowing how
認知的方略 (Cognitive Strategies)	学習の方法に関する知識・技能、メタ認知
態度 (Attitudes)	個人の選択行動を支える内的な状態
運動技能 (Motor Skills)	筋肉の運動を伴う技能

表 2-1 ガニェによる学習目標の分類

「言語情報」とは、記憶によって答えることができる、いわゆる「知識」のことを指す。例えば、りんごをみて apple と答えることができることは、言語情報の学習目標である。

「知的技能」とは、実際にやってみせられる知識のことを指す。法則、概念の理解やそれらを適用した問題解決能力などがそれにあたる。

「認知的方略」とは学び方そのものの学習のことを指す。例として、注意して話を聞く、仮説を立てる、評価する、などが挙げられる。

「態度」は人が行動を選択するときの内的な状態のことを指す。例えば、「環境を大切にする」といった行動選択の基準を学習するものである。

「運動技能」は筋肉の運動を伴う技能、例えばタイピング学習がそれにあたる。

これらの分類をすることによって、次のような利点がある。

第 1 に、分類された学習目標によって効果的な学習方法が異なるために、分類によって効果的な教育方法の知見が得られることである。例えば、言語情報は、情報を意味付けして学習することが効果的であるのに対し、運動技能は反復練習が効果的な学習方法である。

第 2 に、学習目標にふさわしい教育効果の測定方法を得ることができることである。例えば、言語情報は言葉で述べられることで達成できているかが確認できる。知的技能は未知の例に適用できることで初めて達成できたとみなされる。

2.2. オブジェクト指向技術における学習目標の分類

本研究では、オブジェクト指向技術教育における主な学習目標を言語情報、知的技能、認知的方略の 3 つの分類に注目して分類する。(表 2-2)

言語情報	・プログラミング言語の文法 ・モデリング言語の文法
知的技能	・オブジェクト指向の概念 ・プログラミング ・モデリング
認知的方略	・オブジェクト指向の学び方

表 2-2 オブジェクト指向技術教育における主な学習目標の分類

言語情報	・プログラミング言語の文法 ・モデリング言語の文法 ・オブジェクト指向の概念 ・プログラミング ・モデリング

表 2-3 既存の教育から推測される分類

1.言語情報としての学習目標

プログラミング言語/モデリング言語の文法、知識（例えば、for 文の書式を言い当てること）は、記憶によって答えることができるものである。したがってそれらは言語情報に分類される。

2.知的技能としての学習目標

クラス、継承、カプセル化などに代表されるオブジェクト指向の概念は未知の例に適用され、それにより問題解決がなされねばならない。従って、それらの概念を理解し、適用する能力は知的技能に分類される。

プログラミング/モデリングの学習は、実際にプログラミング/モデリングができるようになることを目的とするため、知的技能である。文法と実際にプログラミング/モデリングができることを区別することが大切である。

3.認知的方略としての学習目標

認知的方略の教育は効果的なオブジェクト指向技術の教育方法を考える上で非常に重要である。認知的方略とは、学び方を学習するものであり、学習者はこれらの学習を通して自力で学習するための能力をつけることができる。

オブジェクト指向技術者が生涯にわたって学習者であり続けるために、この能力を身に付けることは非常に望ましい。また、自力で学習するための能力をつけることには OJT の負担を軽くできるという利点もある。

2.3. 既存の教育の効果が上がらない理由

前節までの議論を踏まえ、既存の教育では何故効果が上がらないのか考察する。

１つめの理由はオブジェクト指向の概念を言語情報として教育していることである。1 章で取り上げたような継承の概念を教育する例のように、「オブジェクト指向とは何か」という説明に終始してしまっている場合、学習者は概念の定義が述べられるようになるものの、実際に適用できるようにならない。これは言語情報として教育してしまっているために、知的技能としての、概念をコンテクストを吟味して適用する能力がつかないことを示していると分析される。

2 つ目の理由は、認知的方略を教育していないことである。入門教育においては、応用的な概念を学ぶための基礎となる認知的方略こそ教育するべきである。学び方を教育しなければ、教育する効果が上がらないのは当然である。

また、学び方を教育しなければ学習者が独学できるようにならない。このことは、OJT の効率を下げ、一人前になるまでに長い年月を要する原因ともなっていると考えられる。

既存の教育方法から推測される学習目標の分類を(表 2-3)に示し、既存教育の効果が上がらない理由を次のようにまとめる。

1) 知的技能として教育するべきものを言語情報として教育してしまっていること
2) オブジェクト指向技術を教育する上で重要だと考えられる認知的方略を教育していないこと

3. 試作カリキュラム

本研究では、既存のオブジェクト指向技術者教育の問題を解決するために、試作カリキュラムを開発した。本章ではその概要について述べる。

3.1. カリキュラム概要

本カリキュラムのタイトルを「オブジェクト指向哲学」とした。このタイトルには、オブジェクト指向技術の背景にある「考え方」を教育するカリキュラムという意味が込められている。

対象：

- 将来技術者になる企業の新入社員
- 基礎的なプログラミング能力を有する者

講義形式：

- 議論中心の対面教育

教師の能力にもよるが、1 クラス 10 名～20 名程度で構成される。

3.2. 方針

3.2.1. 基本方針

本カリキュラムは、オブジェクト指向技術の背景にある「考え方」を教育することを主眼としている。ここでいう「考え方」とは、2.2 節で述べられたオブジェクト指向技術の学習目標のうち、オブジェクト指向の概念（知的技能）とその学び方（認知的方略）を指す。

「考え方」を主眼とするため、オブジェクト指向技術を表現するために必要なプログラミング言語/モデリング言語は、必要なときに必要最低限だけ教育するという方針としている。

3.2.2. 知的技能を教育する方針

オブジェクト指向の概念は知的技能である。従って、本カリキュラムでは「オブジェクト指向とは何か」という説明ではなく、オブジェクト指向の概念を適用する利点（以下、「オブジェクト指向の利点」とする）を議論することにより、コンテクストを吟味して概念を適用することを教育する方針を貫いている。

「可読性の向上」に絞る

本カリキュラムの特徴は、オブジェクト指向の利点を「可読性の向上」に絞って教育することである。

一般的にはオブジェクト指向の利点は可読性の向上のほかに、「再利用性」、「拡張性」、「保守性」の向上だと言われている。しかし、初学者に対して可読性の向上以外の利点を教えるのは以下の理由により効果的でない。

第 1 の理由は、初学者は保守性、再利用性や拡張性そのものに対する理解が弱く、また初学者が書くような小さいプログラムではその利点が明らかになりにくいため、利点について深く議論することが難しいためである。そのため、どうしても「カプセル化すると保守性が高まる」、「モジュールの凝集性を向上させることにより再利用性を高める」というような説明となってしまう。このような説明をした場合、学習者はこれらの利点を言語情報として認識してしまうため、知的技能としての教育効果が期待できない。

第 2 の理由は、時として可読性と再利用性や拡

張性は相反する場合があるためである。そのようなケースでは、それらの利点や欠点を考慮した上での解決策が求められる。しかし、初学者にそれらの解決策を議論させるのは非常に難しい。初学者が再利用性や拡張性を意識しすぎてしまった結果、本来のプログラムの目的を失い、可読性、再利用性や拡張性とすべてが失われてしまうことはよく起こる。

これらの理由により、本カリキュラムでは「可読性の向上」に絞って議論の主題を明確にし、理解を深めてもらう方針としている。

前提技能

ガニェによれば知的技能を教育する方法として、より基礎的な技能（下位目標となる前提技能）から段階的にひとつずつ習得させていくのが有効とされる。[5]本カリキュラムでは、オブジェクト指向の利点を議論するための前提技能として、以下の 3 つの技能を設定している。

1) プログラムにそのプログラムの意味(目的)が意識されたコメントがつけられる
2) データ構造とアルゴリズムを理解している
3) 構造化手法を理解している

1) が設定される理由は、可読性が向上するという利点を捉えるために、可読性の高いプログラムについての基準が必要なためである。可読性が高いプログラムとプログラムの意味･目的を考慮するということには、非常に深い関係がある。(3.2.3 項で詳しく述べる)

2) が設定される理由は、オブジェクト指向の大きな利点が、データ構造とアルゴリズムが融合することであるために、その利点を明確にするためにデータ構造とアルゴリズムの関係を議論する必要があるためである。

3) が設定される理由は、オブジェクト指向の利点を議論する際の比較対照になるためである。オブジェクト指向は構造化手法の問題点を解決するために生まれた手法であるため、比較することで利点を明確にすることができる。

3.2.3. 認知的方略を教育する

本カリキュラムのもう一つの特徴は、カリキュラム全体を通して、オブジェクト指向の学び方（認知的方略）を教育することである。学習者は学び方を身に付けることでその後の学習の基盤を構築することができ、学習効果を高めることができる。

3.2.1 項で述べられたように、本カリキュラムは、オブジェクト指向の利点を議論することを中心に進められる。議題は「可読性」である。その際に、学習者が「可読性の高いプログラム」についての基準を身に付けることが非常に重要である。学習者がこの基準を身に付けることによって、議論が円滑に進むようになるばかりか、その基準を用いて自力で判断することができるようになる。つまり、この基準を身に付けることが、学び方を習得することにほかならないのである。

本カリキュラムの最大の特徴は、この基準が身に付きやすいように、「可読性の高いプログラム」とは「意味が明確なプログラム」だと一意に定めていることである。ここでいうプログラムの「意味」とは人間にとっての意味のことであり、プログラムの「目的―手段」の関係における目的と同義である。

本カリキュラムが提供する「意味が明確なプログラム」という基準は非常に強力である。なぜなら、この基準だけでオブジェクト指向における主要な概念の利点を説明できるからである。ここでは、そのことについて 3 つの例を挙げて説明する。なお、ここで例にあげるプログラムは Java 言語である。

1) クラス

例として、配列を使った要素リストに 1000 を追加するというプログラムを考える。(図 3-1)

構造化プログラミングでは、アルゴリズムとデータ構造を分離して考えるため、a)のようなプログラムになる。（add というメソッドは、配列の空きを探して要素を代入するメソッドだと考えて欲しい）配列とそのアルゴリズムを持つ配列リストクラスを作った場合は、b)のようなプログラムになる。

これらのプログラムを比較して、どちらが「意味が明確な」プログラムかを議論する。ここで注目して欲しいのは、プログラムに書かれたコメントである。コメントにはプログラムの意味（目的）を書くように指導する。するとこの 2 つのプログラムの意味は変わらず、手段として書かれたプログラムだけが異なることが分かる。ここで、書かれたプログラムのどちらが意味が明確か、つまり、コメントに近いプログラムはどちらなのかを議論する。この議論によって、クラスという「語彙」を作ることがプログラムの可読性に寄与することを学習者に理解させることができる。再利用性や拡張性ということには触れずに、クラスの本質

```
//要素リストに 1000 を追加する
int[] array = new int[100];
add(array,1000);
```

a) 構造化手法で add メソッドを作った場合

```
//要素リストに 1000 を追加する
ArrayList arrayList = new arrayList();
arrayList.add(1000);
```

b) List というクラスを作った場合

図 3-1 クラスを作る利点の説明例

```
//要素リストに 1000 を追加する
ArrayList arrayList = new ArrayList();
arrayList.add(1000);
```

a) 継承を使わない場合

```
//要素リストに 1000 を追加する
List list = new ArrayList();
list.add(1000);
```

b) List という抽象クラスを作った場合

図 3-2 継承を使う利点の説明例

```
A a = new A();
a.x = 100;                    //x を設定
System.out.println(a.x);      //x を取得
```

a) カプセル化しない場合

```
A a = new A();
a.setX(100);                  //x を設定
System.out.println(a.getX());//x を取得
```

b) カプセル化した場合

図 3-3 カプセル化の利点の説明例

を理解させることができるのである。

2) 継承

例として、1)で挙げた、要素リストに 1000 を追加するというプログラムを考える。(図 3-2)ここで、b)のプログラムでは、ArrayList クラスのスーパークラスとして List クラスを定義する。

ここでも、どちらのプログラムがコメントを反映した意味の明確なプログラムなのかを議論する。継承を利用しない場合は、a)のように「配列リストに 1000 を追加する」というプログラムを書かねばならないのに対し、継承を利用した b)のプログラムは、「要素リストに 1000 を追加する」というプログラムを書くことができる。

注目すべきことは、再利用性や拡張性に触れずに継承の利点を議論していることである。

第 1 回　コメントの書法
第 2 回　配列
第 3 回　構造化手法
第 4 回　クラス/インスタンス
第 5 回　連結リスト
第 6 回　データ構造とアルゴリズムの結合
第 7 回　継承とインターフェイス
第 8 回　サーチ
第 9 回　ソート
第 10 回　スタックとキュー

表 3-1　カリキュラム構成

3)カプセル化

例として、配列を使った要素リストに 1000 を追加するというプログラムを考える。(図 3-1)

a)のプログラムにおけるクラス A の x という変数はカプセル化されていない。b)のプログラムにおけるクラス A の x 変数は private 宣言され、設定と取得のメソッドを介して操作を行うようになっている。ここでも、コメントに書かれているプログラムの意味を考えると、「設定する」「取得する」という意味が明確なプログラムは b)である。

この議論では、カプセル化における「保守性」という側面を教育できないのではないかという意見があるかもしれない。しかし、ここで行った議論は意味が明確なインターフェイスを利用してクラスにアクセスするという議論であり、データの隠蔽と関係が深い議論である。そのため、保守性について理解する前提技能として価値があると考える。

また、初学者に保守性という側面から議論をさせるのは非常に難しい。(3.2.2 項を参照)これに対して、意味が明確なプログラムという基準はカリキュラム全体を通して一環しているため、この基準を利用した議論は比較的容易に進めることができる。

3.3. カリキュラム構成

本カリキュラムは、自動販売機システムの構築をテーマとして、全 10 回で構成されている。

カリキュラム構成表を(表 3-1)に示す。

3.3.1. オブジェクト指向の利点を明確にする

本カリキュラムの構成は、学習者がオブジェクト指向の利点を明確にできるように、いくつかの工夫がなされている。

学習者がオブジェクト指向の利点を効果的に理解するために、実体験するということが非常に重要である。ここでいう実体験とは、学習の対象となる概念を適用した場合、しない場合それぞれのプログラムを実際に書き、プログラムを比べてみるという体験のことである。プログラムを実際に書いてみた上で、それらを比較することによって、利点をより明確にすることができる。

効率的にプログラムを比べることができるように、構造化手法から、徐々にオブジェクト指向の概念を導入していくという、ソフトウエア工学の歴史的な流れに沿ってカリキュラムが構成されている。

また、プログラムを比較する際に、比較する 2 つのプログラムの機能が異なる場合は比較が難しく、利点が明確にならない場合がある。そのため、プログラムの機能は変えずに新しい概念を使って実装しなおすという、いわばリファクタリング[7]というべき作業を行わせる。

3.3.2. コメントの書法を徹底させる

本カリキュラムでは全体を通して「意味が明確なプログラム」という基準で議論を行う。そのため、初期の段階でその土台を作ることができるかどうかが、その後の議論の成否を分ける。

その土台を作るために、本カリキュラムでは、1 回目の単元でコメントの書法についての授業を行う。この授業では他人が書いたプログラムにコメントをつけるという作業を行わせる。この作業により、プログラムの意味(目的)は、書いた本人にしか分からないということを実体験でき、プログラムの意味を明確にすることが（この場合コメントをつけることが）可読性の向上につながるということを理解させることができる。

コメントをつける作業を自分の書いたプログラムに対して行わせてしまうと効果がない。自分の書いたプログラムの意味(目的)は自分で分かっているため、それらを考慮することにより可読性が高まる理由をつかみにくいためである。

4. 実験授業

試作カリキュラムの有用性を評価するために、実際のソフトウエア開発企業において、20 名の社員に対して試作カリキュラムを用いた実験授業を行った。

今回は、本カリキュラムの対象である新入社員を集めるのは困難であったため、1〜2 年間実際に技術者として仕事をされている方を対象とした。そのため、以下の視点によるアンケート調査を行うことで、本カリキュラムの有用性を評価した。

- 自分が過去に受けた新入社員教育と比較して今回提案するカリキュラムは有用であるか
- 自分が今技術者として仕事をしているという観点から、将来技術者になるための入門教育として今回提案するカリキュラムが適当かどうか

非常に役に立つ	15%
役に立つ	75%
普通	5%
役に立たない	5%

表 4-1 Q1.カリキュラムは技術者教育に有用か

非常に良かった	15%
良かった	55%
普通	20%
(無記入)	10%

表 4-2 Q2.カリキュラムのスタイルについて

4.1. アンケート結果

アンケート結果のうち興味深いものを抜粋する。(表 4-1)は、本カリキュラムが技術者教育に有用かどうかという質問である。
(表 4-2)は、「何故オブジェクト指向を用いるのか」という議論を中心としたカリキュラムのスタイルについての質問である。

次に、コメントのうち、興味深いものを取り上げる。

- カリキュラムについて

a) オブジェクト指向を使用するに至る理由や利点・欠点をまず洗い出してから、その結果オブジェクト指向だという展開なので無理がない
b) 今回のように、（知識をつけるための教育ではなく）考えさせられるような講座は、業務においても、利用できる
c) プログラミングの本質を考えることができるのでよい。言語教育は文法中心になりがちだが、それを打破した講座でよい
d) プログラムの目的というものを明確にしながらコーディングしていく習慣が身につく
e) プログラムを作成するときただ作成するのではなく、機能別にまとめて考えることを重視している教材なので、受講後のプログラム作成の考え方にいい影響が出る

- 「何故オブジェクト指向を用いるのか」という議論を中心としたカリキュラムのスタイルについて

f) 質問を受講者に多く投げることにとって常に考えていく姿勢が自然と身につく内容でした。皆に問いかけを行う方法はよい

g) プログラミングの文法や慣習の述べる前に、議論の時間をもち、考えさせようとするのは理解しやすい

h) みんなでディスカッションというのはよかった。案外、当たり前の事を知らないので、実際に説明しようとしてもできないということがわかった

- プログラミングの基礎とオブジェクト指向の考え方の関係について

i) ある程度プログラミングの勉強をしてからではないとプログラミング演習はできない

j) 文法を理解しているのなら、オブジェクト指向プログラミング入門として丁度よいと思います

k) プログラムの基礎部分の学習が短期間だと少し難しいと思います

4.2. 考察

カリキュラム全体として、オブジェクト指向技術者教育に有用かという質問に 90%の受講者から「役立つ」または「非常に役立つ」という評価を得た。

また、「文法中心よりも考え方中心のほうが良い」「利点を議論するのは良い」「目的を考えさせるのは業務にいい影響が出る」など、今回提案するカリキュラムの方針は良いというコメントを多数得た。

これらの評価より、今回提案するカリキュラム「オブジェクト指向哲学」は総じて有用であるという評価を受けたと考えられる。

しかし、問題点も指摘された。

1) 教師の役割

議論中心の教育方法に対して、「教師の質問があまりに抽象的過ぎる」、「講師の解答への導き方が強引すぎる」というような指摘があり、このような授業を行う際に、教師の役割を明確に定める必要があることが分かった。

2) プログラミング能力との関係

また、カリキュラムを受講する学習者にとって、「プログラミング能力がオブジェクト指向の概念の理解に影響されるのでは」という意見が多数あった。これらの指摘は本カリキュラムを受講する際の事前条件について、もう少し具体的に記述すべきであることを示していると考えられる。

5. まとめと今後の展開

アンケート調査により、今回提案するカリキュラムは総じて有用であるという評価を得た。本研究で提案した教育方法論の方向性は概ね正しいと考えられるので、更なる改良を行ない、本年 5 月の新入社員研修において使用する予定である。

6. 謝辞

本研究を進めるにあたり、(株)EXA の児玉公信様を始めとする技術部の皆様には大変お世話になりました。また、実験授業として参加してくださった皆様に感謝いたします。

7. 参考文献

[1] "平成１３年版労働経済の分析",厚生労働省,2001
http://www.mhlw.go.jp/wp/hakusyo/roudou/01/
[2]Leslie J.Briggs, Kent L.Gustafson, Murray H.Tillman, "Instructional Design Principles and Applications",1991
[3]Patricia L.Smith, Tilman J.Ragan, "Instructional Design Second Edition", John Wiley & Sons, inc., 1999
[4]沼野 一男,平沢 茂 "教育の方法・技術",学文社,1989
[5]B.レイヒ,M.ジョンソン,"教室で生きる教育心理学",新曜社,1983
[6]藤土 圭三,"心理学から見た教育の世界",北大路書房,1994
[7]マーチン・ファウラー, "リファクタリング: プログラミングの体質改善テクニック",ピアソン・エデュケーション, 2000