企業の業務アプリケーション開発に用いるプログラミングの教授法

田中　宏和*

Teaching method of programming used for the application system development of a company

Hirokazu TANAKA*

Abstract: Most graduates find a job which develops the information system of a company. For programming takes the lead in information education, it needs to make it master the knowledge related to systems development in university. I propose the teaching methods in which a beginner also creates a program simply.

はじめに

プログラミング教育は、情報教育のなかでも中心となる教育であり、優秀なシステムエンジニア(SE)の輩出が期待されている教育機関にとってはとくに重要な教育として位置づけられている。その習得には、プログラミング言語の文法以外にオブジェクト指向などの設計論、プログラミングの対象となるアルゴリズムなどの関連知識が必要である。

静岡大学情報学部では文工融合を教育方針とし、文系学部がもつ社会科学の領域と理系学部がもつ情報科学の領域をカリキュラムのなかに複合的に採り入れている。文系入試で入学した情報社会学科の学生 100 名と理系入試で入学した情報科学科の学生 100 名は、2 年次に進級するときに、情報社会の在り方を学ぶ情報デザイン(ID)プログラム、コンピュータ科学を中心に学ぶ CS プログラム、そしてコンピュータの企業や社会への応用を学ぶ情報システム(IS)プログラムのどれかを選択し、それぞれのプログラムに沿った授業科目を履修する態勢になっている。

IS プログラムを希望する学生の多くは、卒業後はシステムエンジニア(SE)の職業を希望し、IT 企業に就職している。

IS プログラムの学生を対象にしたプログラム教育は 1 年次から C 言語、JAVA 言語の教育が行われているものの、多くの学生において基本的な知識・スキルがあまり身についていないという現状がある。その原因として考えられるのは、学習ステップの初期段階で授業内容がわからなくなると授業についていけなくなり、その結果、学習意欲が低下するという悪循環に陥る点にある。第 2 に、講義をもとに理解を深める演習課題を与えても、課題を解くためのアルゴリズムを自己流で考えてプログラムを作成してしまうため、指導する教員側では学生が考えたアルゴリズムに沿った対応が十分にできない点にある。第 3 に、授業自体が、言語の基礎となる文法の理解を中心に構成されていることである。現在、システム開発の主流になっているオブジェクト指向の考え方やデータ中心アプローチの意義や必要性についてプログラミングの立場からその有用性を理解できる構成にはなっていない。第 4 に、プログラムが適用される業務システムを念頭に置いていないため、プログラミング教育以外に配置されている他の授業科目群との関連性が乏しいことである。本来、情報システム系の学科で扱われているシステム開発技法や手順、さらには、経営学やビジネスモデル等との関連性を意識させる教育が必要である。以上の問題点は、日本の高等教育機関で行われているプログラミング教育でもほぼ共通した課題であると思われる。

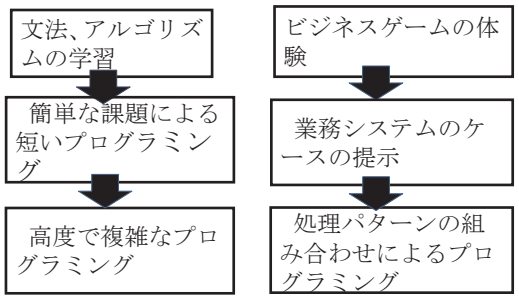

図 1 プログラム教授法の概念比較

われわれは伝統的に行われてきた教授法(図 1 左)に代わる新しい教授法として、最初に企業活動の本質と業務システムの仕組みを理解させた後で、業務システムを対象にしたプログラミング教育を実施する教授法(図 1 右)を数年前から試行錯誤しながら開発してきた。

本稿では、初学者でも意欲をもって、販売管理システムや在庫管理システム、生産管理システムなどのアプリケーションシステムを比較的容易に開発できる方法を提案する。

システム開発のための情報基盤の進歩

一昔前であれば高度な知識・スキルをもったプログラマでなければ業務アプリケーションを開発するのは難しかった。しかし、情報技術が進歩し広く普及してきた現代においては、初学者であってもシステム開発を可能にする情報基盤ができつつある。

①PC とインターネット、クラウドの普及

PC が安価になりインターネットにつながる PC が一

* 東京工芸大学工学部非常勤講師

2014 年 9 月 29 日受理

般家庭にも普及してきた。また、データベースも自前で用意しなくてもITプロバイダが提供するサーバーを借りれば手軽に利用できるようになった。つまり、PCがインターネットに接続している環境であればいつでもどこでもシステム開発ができるようになっている。

②開発ツールの進歩

例えば、Microsoft社が提供するVisualStudioの開発ツールは自動コーディング機能をもつためプログラマが記述するコードのステップ数は少なく済み、またデバッグ(バグ取り)支援機能も充実している。開発ツールの利用は大幅な生産性の向上をもたらしている。

③情報リテラシ能力の高まり

表計算ソフトのExcelを使ってデータを加工編集を行うことが日常的に広く行われるようになってきた。日頃からあるデータと別のデータとのつながりを考える習慣はプログラムを学習するうえでも役立っている。

④ロジカルシンキングの普及

戦略思考、データに基づくロジカルシンキングの知識が広まり、業務システムやシステムの構造を論理的に分析することに対して違和感なく受け入れることができるようになっている。

このような情報基盤の進歩を、大学においても採り入れることが可能であり、むしろ積極的に採り入れることによって、従来とは異なるプログラミング学習法を導入することができる。

ARCSモデルの4要素

プログラミングの学習には、習得したいという強い意欲とその維持が不可欠である。

一般的に、プログラミングを短期間でマスターするためには経験的に以下の要件が望ましいとされている。第1に、プログラミング学習を集中して行える環境を作ることである。そのためには、集中講義の形式でカリキュラムを組むなどの工夫が考えられる。第2に、相談スタッフの配置である。独学で学習を進めてしまうとなかなかプログラミング能力は身につかないので、行き詰まったときは、周りに相談できる人がいると効果があるとされている。第3に、学習をこまめに区切って行うよりも、一気にまとめて全てのことを網羅する学習法が効率的であるとされている。

以上から、プログラミング学習の場合、「習うより慣れろ」という学習方法が適していると考えられている。しかし、それらが実現できたとしても学習する側の意欲が伴わなければ学習効果は低いものになる。

高い学習意欲を引き出し、継続的に学ばせる教育法の枠組みとして、ケラーが提唱したARCSモデルがある(1)。ＡＲＣＳモデルでは、学習意欲の継続性には以下の4つの側面が重要とされている。

①注意（Attention）―学習者に興味をもたせる。

②関連性（Relevance）―学習者に「やりがい」を感じさせ、積極的に取り組めるようにする。

③自信（Confidence）―学習者に成功の機会を与え、自力で成功できるように思わせる。

④満足感（Satisfaction）―目標を達成した学習者を正当に評価し、満足感を与える。

(1) 注意－学習者に興味をもたせる

われわれが提案する教授法では、学習者はまず、仮想企業の経営者となり、実企業で行われている経営意思決定を模擬体験することから始まる。学生は、実企業の業務システムに携わった経験をもたないので、購買－生産－販売の各業務がどのように関連し、モノやカネがどのように動き、流れていくのかの全体像を理解できていない。そこで、プログラミングの対象となる仮想企業の業務プロセスを、ビジネスゲームを用いて体感させることを行う。この模擬体験によって、業務の流れを理解するだけではなく、ユーザの立場からどのような情報が必要なのかについて、業務系システムと情報系システムの両面から関心をもつようになることが期待できる(4)。

(2)関連性

われわれの教授法では、ビジネスゲームのあと、学習者はケース教材をもとに学習を行うことになる。学習者は、業務システムがもつべき機能について主体的に検討を行い、要件定義とシステムの基本設計を行う。これらの学習には在庫管理や顧客管理などの業務知識が必要なため、他の授業科目との関連性がわかるようになる。また、SEがどのように要件定義を行い、それをもとにどのようにシステム設計を行っているのかについても自ら体験することにより理解できるようになる。ケース教材で扱う業務システムは、ビジネスゲームの仮想企業の業務システムを深化させたものであり、また比較的規模の小さな実企業における業務システム上の典型的な問題点を内包させたモデルになっている。

(3) 自信

われわれが提案するプログラミング学習においては、処理の基本パターンの理解と習熟に重点を置いている。プログラムのソースコード例は学習者にすべて紙面で提示されるのでコードの書き方がわからず挫折することがないようにしている。学習者はプログラムコードを自ら書き、完成したシステムがディスプレイに表示され動作することを確認できれば達成感が得られる。また、プログラム学習の進捗がある程度進んだ段階で、マスター系の類似したシステムの作成課題を学習者に与えている。これは、学習者が自らコードを考える余地を残し、作成できたときの喜びと自信が得られるように配慮したものである。

(4) 満足感

学習者はプログラミングスキルを習得したいという動機から受講している。作成したアプリケーションシステムがすべて正しく動作することを確認し、さらにそのシステ

ムについて他者から良い評価を受けることができれば、大きな満足感を得られることが期待できる。

われわれが提案する教授法では ARCS モデルの「注意」と「関連性」に重点を置いて学習意欲を引き出しさらにプログラムづくりでは、プログラミングの基本パタンを繰り返し学習させることに注力し、「自信」と「達成感」が得られるように考慮されている。

プログラミング学習のデザイン

(1) 模擬体験としてのビジネスゲーム

本稿で扱うビジネスゲームは、学習者が 2,3 名でチームを組み、「街のケーキ店」の「経営者」となって、他のチームと業績を競い合うものである(2)。

ビジネスゲームでは、学習者は食材を発注し、届いた食材の在庫管理を行う。ケーキは、食材数と販売予測数をもとに前日に生産計画を立てる。製造したケーキは店舗で販売し、売れ残ると廃棄するという単純なものである。しかし、購買管理、生産管理、在庫管理、販売管理、資金繰りがゲームに内包されているので、ひととおりの業務の流れを理解することができる。

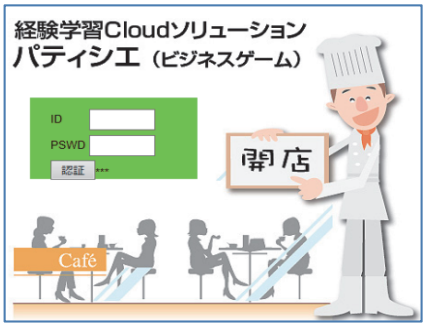

図 2 ログイン画面 1

なお、ビジネスゲーム自体は、Web アプリケーションとなっておりインターネットに接続できる PC があれば簡単に利用できる。図 2～図 5 にはビジネスゲームの利用画面の一部を例示的に示している。

図 3 入力画面例

ゲーム終了後に、学習者はゲームの振り返りを行う。その過程で、企業活動において経営管理機能の良し悪しが業績に大きな影響を与えること、また、業務システムの運営が経営管理の質を左右することを体感的に気づくようになる。図 6 にビジネスゲームの仕様を提示している。なお、ゲームの所要時間は 4 時間程である。

図 4 業績閲覧画面例

期	発注	計画	期首	入庫	出庫	期末	販売	廃棄	受注	受注残
0	100	100	0	0	0	100	0	0	0	0
1	140	100	100	0	100	0	100	0	159	59
2	150	140	0	100	100	0	100	0	188	88
3	180	150	0	140	140	0	140	0	144	4
4	120	150	0	150	150	0	150	0	197	47
5	160	150	0	180	150	30	150	0	174	24
6	170	150	30	120	150	0	150	0	155	5
7	160	170	0	160	150	10	150	0	197	47
8	180	170	10	170	170	10	170	0	176	6
9	180	180	10	160	170	0	170	0	176	6

図 5 在庫管理の画面例

ビジネスの仕組み

(1)材料を発注すると翌々日(2 日後)の早朝に納品される。納品時に代金を支払う。ケーキ 1 個を生産するのに、材料 1 個を使用する。

(2)資金が足りないときは不足分を金融機関から借り入れる。借入金は翌日に全額を返済する。そのときに利息も支払う。利息は 5%とする。

(3)資金が余ったときは、財務戦略の一環として全額を資金運用する。運用レートは 5%とする。

(4)商品が売れ残ったときはブランドを維持するためにすべて廃棄する。商品の品切れは受注残となるが翌日に持ち越さない。

(5)スタッフ数が多ければ顧客サービスが向上する。

(6)広告は、西口と東口の駅前で通行者に配布することができる。六本木の商圏は鉄道路線で東西に分断されているが人の往来は可能である。

(7)材料は大型冷蔵庫で保管するので長期保存が可能であるが、在庫品の維持には、1 個あたり 50 円の費用が毎日かかる。

(8)翌日の生産数は前日に計画を立てる。生産可能な材料数(期首在庫+入庫数)以上の計画を立てることはできない。期末在庫がマイナスになった場合は、その日の製造後に営業停止処分を受けるので、その日は休業しなければならない。

図 6 ビジネスゲームの仕様

(2) ケース教材の導入

学習者はビジネスゲームを体験した後、次の段階としてケース教材に取り組むことになる。ケース教材は、米菓の詰合せをする中小企業をモデル化したものである。ケース教材はビジネスゲームを深化させた内容となっているのと同時にプログラム作成の対象となるものである。

1. 商品は数種類の製品を詰め合せて構成する。商品ごとにレシピが作成されている。
2. 製品は複数の仕入先から調達する。入荷時に検品して製品在庫として保管する。
3. 詰合せ作業は、
①商品レシピをもとに製品在庫から必要数を出庫して準備する。
②商品に完成させる。
4. 検品作業は、
①商品が正しく詰め合わされているかを全数チェックする。
②商品在庫として保管する。
5. 出荷作業は、
①得意先から注文があると商品在庫から出庫し、包装・梱包した後、出荷する。
②出荷時に納品書を添付する。得意先への出荷は同時に売上げとなる。

図 7　米菓会社の業務システムの概要

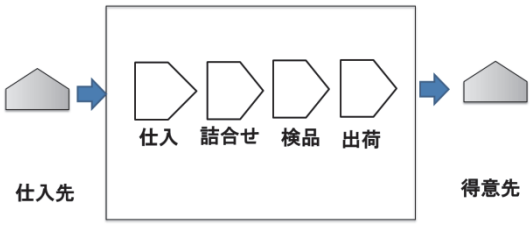

図 8 ケース教材の仕様

ケース教材を使った学習の目的は、業務システムにおける情報の流れに着目し、どのような情報の流れが業務システムの生産性を阻害しているのか、またその流れをどのように再構築すれば生産性が向上するのかを学習者自身が検討することにある。

ケース教材の概要は以下の内容になっている。

① A社は仕入先から米菓を個別に購入し、それらを「詰合せセット」に仕上げて得意先に販売している。

② A社の業務プロセスは、仕入れ、詰合せ、検品、出荷の 4 つの業務から成り立っている(図 8)。

A社の業務の流れを記述したものを図 7 に示している。A社の業務上の問題点は実務経験がない学習者でもビジネスゲームを通じて学習しているので比較的簡単に指摘できる内容である。学習者が個人学習でまとめたあと、図 9 に示す解答例を提示する。この解答例を解決するためにシステムを開発していくことを学習者に説明し、プログラミング学習のステップに進むことになる。

1. 注文があっても商品在庫が足りないため、欠品になることがある。
2. 詰合せ作業時に、必要な製品が一部足りないことが発覚する場合がある。
3. 仕入先からいつ届くのかがわからなかったり、同じものを誤って二度発注してしまうことがある。
4. 得意先から届く支払案内書と当社の請求データが合わないことがある。
5. 仕入先から届く請求書と当社の支払データが合わないことがある。
6. 得意先から商品不良を指摘されたとき原因究明ができず得意先に報告できないことがある。
7. 表計算ソフトを使って集計しているが月ごとの得意先別販売実績を把握するのに時間がかかる。
8. 梱包指示は工場長が差配しているが、経験とカンに頼っているので、間違う場合がある。その結果、廃棄対象となる過剰在庫と顧客の信用を失う欠品が生じている。
9. 鮮度が求められるので在庫は多く抱えたくないが欠品も避けたい。
10. 商品ごとに売上高はわかっても利益がわからない。このため、得意先、仕入先に対する価格交渉ができないし、商品政策も立てられない。
11. 作業者の多くはパートの主婦である。勤務シフト表を早めに作成して欲しいとの要望がある。

図 9　A社の問題点の解答例

プログラム学習の基本的な考え方

プログラムは、コンピュータ言語を問わず、順次処理、分岐処理、繰り返し処理の 3 つしかないので本来、プログラムは容易に習得できる筈である。これを困難にしているのは、学習者の意欲以外に、①プログラムの作成方法、および②指導者の対応力に問題があると思われる。

(1) プログラムの作成方法

従来の教育法では、文法の解説を行う講義のあと、文法の理解を深める演習課題を与え、これらを交互に繰り返すことでプログラミングの知識とスキルを習得させる教育法が採られてきた。しかし、この方法では、学習者は自己流のやり方でプログラムを作成してしまうことになり、このことがプログラムのバグの発見や修正を難しいものにしている(図 1 左)。

業務システムを対象にしたプログラム作成の場合、処理ルーチンごとに定型的な、ある決まった基本パターンが存在している。その基本パターンを習得し、そのパターンを組み合わせることにより比較的容易にプログラムを作りあげることが可能である。業務システムのプログラム作成において必要とされる文法の知識は全体の一部しか利用しない場合がほとんどである。われわれが提案するプログラム学習では、第 1 に文法の知識よりも基本パターンの理解と習得に重点を置いている。

第 2 に、優れた開発支援ツールの利用である。プログラムコードの入力作業を効率化し、コードミス(コンパイルエラー)に気づき修正するには開発支援ツールの利用が欠

かせない。開発支援ツールの利用によって学習者はストレスをあまり感じることなくプログラムを作成できる。

なお、われわれはMicrosft社のVisualStudioを開発ツールとして利用している。

第3に、日本語の活用である。変数や関数の名称はなるべく日本語で記述することで、処理全体を日本語の文章のように表現することである。学習者は、どういう処理をしているのか理解しやすくなり、デバッグ(バグ取り)をしやすくなる。図10には、認証画面のコード例を表示している。変数名は名詞で、関数名は動詞を使うことが基本となる。

```
bool 照合チェックする()
{
  foreach (DataRow row in ds.M 社員)
  {
      if ((string)row["社員 CD"] == txtID.Text&&
                          (string)row["PSWD"] ==
                          txtPSWD.Text)
      {
          lblM.Text = "…";
          lblM.ForeColor = Color.Black;
          return true;
      }
   }
   lblM.Text = "認証できません";
   lblM.ForeColor = Color.Red;
   txtID.Text = "";
   t xtPSWD.Text = "";
   return false;
  }
private void btn 認証_Click(object sender,
                                    EventArgs e)
{
     if (照合チェックする())
     {
         F01_メニュー F01 = new F01_メニュー();
         F01.Show();
      }
}
```

図10 認証画面のコード例

(2) 指導者の対応力

プログラミングの演習課題は個別指導が必須となる。個別指導を困難にしている大きな原因は、学習者がプログラムを自己流で作成してしまい、「スパゲティ状態」に陥ってからプログラムの不明な点を指導者に援助を求めることにある。

構造化プログラミングとは、処理全体を小さなモジュールに分解し、モジュール群を階層的に構造化することによって全体のプログラミングを構成することをいう。

ステップ数の少ないプログラムであっても構造化プログラミングによって作成されるべきである(3)。処理パターンを組み合わせることは構造化プログラミングを実現することでもある。これにより、作成する課題が同じであれば、結果的に誰でも同じ構造をもったプログラムになることが期待できる。

さらに、変数や関数の命名規則を標準化し学習者に徹底させることにより、構造だけではなくコード自体が同じプログラムになることが期待できる。われわれの教授法においては、変数名と関数名に日本語を用いることにより、第三者でも理解しやすいプログラムに仕上げることができる。

例えば、図10で示した認証プログラムの場合、ユーザが入力したIDとパスワードが正しければメニュー画面に遷移し、正しくなければエラーメッセージを画面に赤字で表示するプログラムである。エラー処理は別関数(照合チェックする)として分離させることで、プログラム全体が見やすくなり、同時に仕様変更に伴う修正作業を効率的に行うことができる。

われわれは、指導者の対応力を補う方法として、演習課題を与える前に、類似プログラムのソースコードを学習者に開示することを行っている。例えば、社員マスターの画面を作成し、「追加」や「修正」、「削除」などの処理を行うプログラムを指導する際には、ソースコードを印刷して学習者に配布している。そのあと、演習課題として社員マスターと類似している部門マスターの作成を与えるという方法を採っている。この方法の利点は、第1に、学習者は、文法規則を再度確認・理解するとともにパターンの使い方を繰り返し習得できる点にある。第2に課題を解くためのアルゴリズムを考えることを学習者に求めていないため、プログラムを作成する意欲さえあれば挫折することなく安心して課題に取り組めるようになっている点にある。

アプリケーションシステムの開発

A社を対象にした業務アプリケーションは、マスター系とトランザクション系のサブシステムに分かれている。システムはデータベースを用いたクライアントサーバー方式を採り入れており、データベースは、Microsoft社のSQLServerを用いている。学習者は自分のPCにデータベースをインストールし、そのデータベースを使ったアプリケーションを開発することになる。

図11に示すように、マスター系システムで5本、トランザクション系で6本のプログラムから構成している。

1. マスター系システム
   ①社員マスター　②部門マスター　③取引先マスター
   ④商品構成マスター　⑤工程マスター

2. トランザクション系システム
   ①受注管理システム　②発注管理システム
   ③生産指示システム　④生産報告システム
   ⑤在庫管理システム　⑥経営管理システム

図11 A社の業務システムの構成

マスター系のプログラムはトランザクション系と比較して処理が単純なため、学習者はマスター系のシステムから作成しスキルを高めていくようにしている。

データベースのテーブル群とそれらの関係は図12に示すとおりである。テーブルの構造は学習者に提示するものであり、学習者が主体的に検討するようにはしていない。

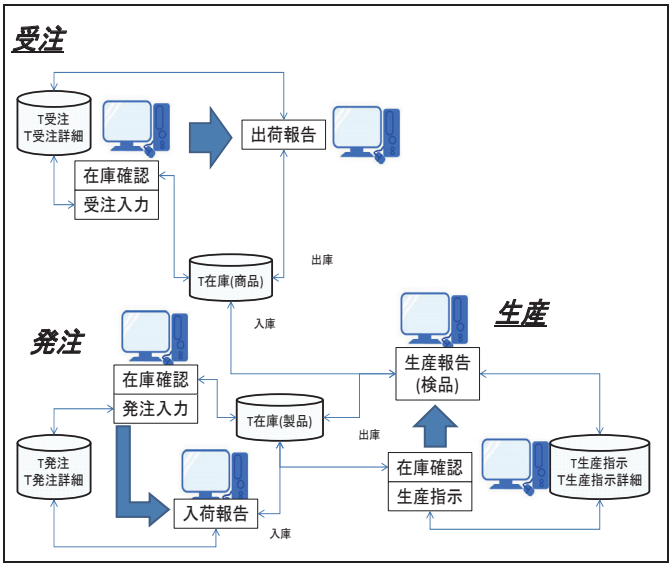

図 12　テーブル間の関係

(1)オブジェクト指向プログラミング

オブジェクト指向プログラミングとは相互にメッセージを送り合うオブジェクトの集まりとしてプログラムを構成する技法とされている(5)。その開発効率の高さから現代のプログラムでは欠かせない存在となっている。オブジェク指向言語には、C++言語, Java 言語など多数あるが、われわれは Microsoft 社が提供している C#言語を採用している。理由は IT 業界で最も利用されている言語のひとつであり言語仕様が洗練されていること、また、同じ Microsoft 社の SQLServer との親和性が高く開発の生産性が高いからである。

オブジェクト指向の概念とクラスの理解は UML と共に、学習者には難解であり挫折の要因になっていると思われる。しかし、オブジェクト指向プログラミングは、「データに対する処理はデータそのものに付随したものである」というオブジェクト指向がもつ本来の概念に基づけば理解は容易になる。すなわち、「正規化されたデータベースのテーブルごとにクラスを作成する」というルールを定めておけばそれに沿って学習者は戸惑うことなくクラスを作成することができる。

テーブルへの基本操作は、①テーブルからのデータ呼び出し、②データの追加、③呼び出されたデータの修正、④呼び出されたデータの削除、の 4 つからなる。これらをクラスにメソッド(関数)として用意しておけば良い。図 13 には社員テーブルに対応する社員クラスのコード例を示している。

社員マスターの画面と「追加」ボタンをクリックしたときの、社員クラスを用いたプログラム例を図 14 と図 15 に示している。クラスを使った処理を用いることで学習者は、オブジェクト指向プログラミングの本質はモジュール(クラス)の再利用によって効率的なプログラムが実現されていることを理解できるようになる。

(2) プログラミングの教授法

われわれの教授法では、学習者はまず、マスター画面の作成を通じて、基本文法を復習しながら、クラスの使い方を学び、オブジェクト指向がもつ効率的なプログラムの作成方法を実践的に学ぶことになる。また、マスター系には複数のシステムがあるが、基本パターンは同じである。このため、学習者にとっては類似プログラムを繰り返し作成することになるためスキルの確実な習得が期待できる。

A社のアプリケーションシステムは大規模であるためチームを組み分担して開発を進める方法も考えられる。しかし、われわれの教授法では、学習者はグループワークではなく一人ですべてのプログラムを作成することを求めている。理由は、繰り返し類似のプログラムを作成することで基本パターンを習得することを重視しているからである。

```
class Cls 社員
 {
   DB db = new DB();
   public Cls 社員()
  {
  }
  public void 呼び出す(ds 米菓 ds)
  {
    ds.M 社員.Clear();
  string sql = " SELECT * FROM M 社員";
    db.adM 社員.SelectCommand = new
    OdbcCommand(sql, db.odbcConnection1);
    db.adM 社員.Fill(ds);
  }
  public void 追加する(DataRow row, ds 米菓 ds)
  {
       ds.M 社員.Rows.Add(row);
       db.adM 社員.Update(ds);
  }
  public void 修正する(DataRow row, ds 米菓 ds)
  {
    for (int j = 1; j <row.Table.Columns.Count; j++)
         ds.M 社員.Rows[0][j] = row[j];
    db.adM 社員.Update(ds);
  }
  public void 削除する(ds 米菓 ds)
  {
    ds.M 社員.Rows[0].Delete();
    db.adM 社員.Update(ds);
  }

 }
```

図 13　社員クラスのプログラム例

また、学習者が途中で挫折しないように、ソースプログラムはすべて学習者に開示するようにしている。理由は、学習者のプログラミングスキルを高めることに学習の目的があるからである。開示するからこそ、プログラムを作成意欲が減退しないと考えている。

マスター系のプログラムを作成した後、トランザクション系のプログラムの作成を行っていくが、学習方法はマス

ター系と同じてある。

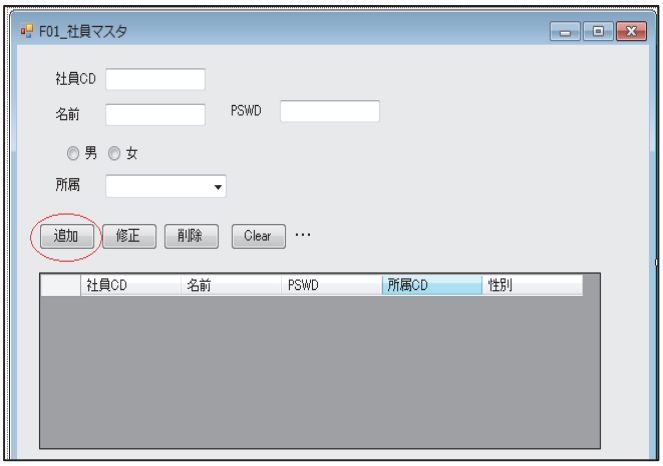

図 14　社員マスターの画面例

```
DataRow row を取得する()
 {
     DataRow row = ds.M 社員.NewRow();
     row["社員 CD"] = txt 社員 CD.Text;
     row["名前"] = txt 名前.Text;
     if (rdo 男.Checked)
         row["性別"] = "男";
     else
         row["性別"] = "女";
     row["所属 CD"] = cmb 所属.Text;
     row["PSWD"] = txtPSWD.Text;
     return row;
}
private void btn 追加_Click(object sender,
                                        EventArgs e)
 {
   if (データチェック(true))
   {
     DataRow row = row を取得する();
     実社員.追加する(row, ds);
     lblM.Text = "追加しました";
     lblM.ForeColor = Color.Blue;
   }
}
```

図 15　「追加」ボタンのクリック時のプログラム例

プログラミング教授法の評価

静岡大学情報学部では 3 年後期から研究室配属が行われる。3 年後期では、学生は指導教員のもとで卒業研究の予備演習（研究室単位で行われるゼミ）を行い、4 年生は 1 年間をかけて卒業研究を行う体制が採られている。

われわれの研究室では、3 年生は週一回 3 時間程度行うゼミにおいて、本稿で紹介した教授法を 2 年前から実践している。

また、4 年次では 3 年後期に習得したプログミングスキルを実践する場として数年前から PBL（プロジェクトベース学習）を行っている。4 年生の学生にとっては PBL が卒業研究となる PBL の目的は、実企業を対象にした業務改善に取り組む過程で学生が大学で学んだ知識を深化させ知識の応用力を高めることにある。

卒業研究は、浜松市内の町工場を対象に IT を使った業務改善がテーマになる。教員の指導を受けながら学生は町工場の経営分析と業務分析を行い、導出した経営課題を解決するためにアプリケーションシステムを開発するプロジェクトに取り組むことになる。システムの導入がテーマではなく、IT を使った業務改善をテーマにしているので、改善活動の成果を定量的に評価することを課している。

3 年次に行っているプログラム学習法が有効であることは、第 1 に、図 11 に提示した 5 つのマスター以外の新たなマスター作成づくりや管理者に対して提供する閲覧システムを作成する課題を与えた場合でも、学生は自力で作成できていること、第 2 に、オブジェクト指向プログラミングにおいては典型的な演習課題である「ジャンケンゲーム」や「自動販売機」の課題を与えた場合でもクラスの作成を当たり前のように行っていること、第 3 に、4 年次に行っている PBL では実企業を対象にしているため、実際の業務の流れは複雑である。また、ユーザ側の情報リテラシーレベルもバラツキがあるのでシステムの開発をケース教材のように段階的に開発することはできない。そのような障害があっても、学生は各社ごとに設定した要件定義に沿って 3 年次に習得した基本パターンをベースにそれらのカスタマイズと組み合わせによってプログラムの作成ができている。これらの事実は、提案した学習法が有効であることを示している。

実企業の業務システムが複雑であったとしてもそれは作成するプログラムの複雑さを意味しているわけではない。比較的規模の小さな組織の業務システムを対象にする限り、プログラムの作成は基本パターンのカスタマイズとと組み合わせによって作成することが可能である。実際、PBL においては、学生は、知識やスキルが乏しいためにプログラムづくりに苦労するよりも、要件定義の変更が頻繁に起きることへの対応や、システムの導入にあたりユーザ側の積極的な協力が得られないことへの対応に苦労することのほうが多い。逆に、仕様変更が頻繁しそれへの対応を行う過程は、データ中心アプローチとオブジェクト指向アプローチの有用性を再認識する機会になっている。

IT 業界において専門知識をもった SE が開発するシステムでさえ、成功する割合は 3 割程度と言われている (6)。

PBL においてシステム導入に失敗するケースのほとんどは、組織上の問題と業務の例外処理への対応が十分にされていないことにあり、プログラムの作成スキルにあるわけではない。導入できなかった企業においても企業側からは学生が作成したシステムを実際に使って初めて自社の IT 化の姿を具体的に描くことができたという評価が得られている。われわれが数年前から行っている産学連携の PBL が持続している理由である。

おわりに

本稿では、学生がプログラミングを学び、その延長線上に実企業の業務アプリケーションを開発するスキルを習得するための教授法を提案した。

その特長は、第 1 に、学習者のプログミングに対する意欲を高めることを最も重視していることである。そのためにビジネスゲームを用いて経営者の模擬体験を行い業務の流れを理解させている。また、ケース教材の活用によって、業務システムにおける IT の必要性を認識させ、さらにシステム開発の要件について主体的に検討できる能力の育成を図っている。プログラミング学習は、学習の途中で挫折しやすいので高い興味と強い関心を持続させることが何よりも必要である。

第 2 の特長は、伝統的に行われているコンピュータ言語の教育が文法中心の教授法になっているのに対して、われわれの教授法では、処理の基本パターンの習得に重点を置いていることである。実際のシステム開発において必要となる文法知識は全体のごく一部に過ぎない。例えば、クラスについても多重継承や抽象クラスなどを扱う高度な知識は求めず、代わりに基本パターンの習得を重視している。

第 3 の特長は、演習課題では、アルゴリズムを考えることよりもプログラミングに慣れることを重視している点である。類似のプログラムを繰り返し作成することがプログラム学習には効果がある。

4 の特長は、ステップ数の多い大規模なシステムであってもチームを組み分担して作成するのではなく、学習者がひとりですべてを作成することを求めていることである。チームを組んだ場合、分担した部分しか知識は習得できないし、責任の所在があいまいになり、プログラミングの苦手な学習者は他の学習者に依存するという弊害が生じやすくなる。

第 5 に、指導する側にとっては負荷が小さいという特長がある。従来の教授法と比較して学習者の意欲を高める工夫がされているのでプログラム解答例となるソースコードを学習者に提示しても不正な行為をすることはないと考えている。実際、われわれの実践例では、プログラムコードの標準化がされているので学生同士が相互に教えあうことが活発に行われており、それが教員の負荷を下げ、指導しやすいものにしている。

業務システムを対象にする限り、プログラムの作成自体に工夫や独創性が求められることは少ない。求められるのは IT を使って業務改善をどのように行うのかの要件定義の部分である。

プログラム作成にあたっては標準化と基本パターンの利用を重点的に指導することにより簡単にプログラムを作成できるようになる。しかし、一方でそのことがモチベーションを維持した能動的な学習につながらない側面があるとの指摘もあるであろう。しかし、その点に関しては、ビジネスゲームとケース教材の活用が主体的に学ぶ動機づけを担保するものであることを示した。

本稿で提案した教授法の欠点として、基本パターンの習得を重視した学習では異なるアプリケーションの開発能力は習得できないという懸念が考えられる。この点に関しては、プログラムづくりには、ものづくりの現場における熟練工の技に似た側面があると考えている。柔道や狂言の世界では古来から守破離の教えが現代に受け継がれている。すなわち、初学者は「型」を無批判に覚える段階から始め、その次に型の応用を学ぶ段階へと進む。そして、その実践の連続と繰り返しが独自の世界を切り開くというものである。その意味で、本稿で提示した教授法は「型」の習得の段階に相当するものである。

提案した教授法は大学教育の枠を越え、社会人を対象にした教育プログラムとしても利用できると考えている。さまざまな分野で女性の社会参画が求められているにもかかわらず、育児や介護など家庭の事情で社会参加ができない人も多い。在宅ワークはその打開策として注目を集めている。また、都市と地方の経済格差が拡大しており、地方では条件に合う働き場所がなかなか見つからないのが現状である。プログラム作成は PC と開発環境を用意できれば自宅でもできる仕事であり、地理的な場所と時間を問わない。プログラミングスキルを習得する機会ができれば女性の社会参画の裾野が広がっていくことが期待できる。実際、われわれは大学発ベンチャー企業を設立しその取り組みを始めている。

参考文献

1）Keller, J.M. and Suzuki, K.: Use of the ARCSmotivationmodel in courseware design(Chapter16), Instructional designs formicrocomputer courseware. Lawrence Erlbaum Associates, U.S.A.(1988).
2）田中宏和，「ビジネスゲーム　経営のしくみ」, 静岡学術出版, 2014.
3）小林 修,「初等プログラミング教育の方法」, 日本教育工学雑誌 17(2), 105-116, 1993-11-20.
4）佐藤雅彦，村上泉，山本晃士，菅俊一，石川将也，佐藤匠:「コンピュータサイエンスにおける教育原理」，平成 14 年度ハイテク・リサーチ・センター報告書, pp. 488-501 (2004).
5）出井 秀行，「C＃プログラミング入門―「オブジェクト指向」の「プログラミング手法」を基礎から解説」, 工学社, 2011.
6）「成功率は 31.1％第 2 回 プロジェクト実態調査（対象 800 社）」日経コンピュータ 2008 年 12 月 1 日号.