プログラミング学習のための理解度モデルを用いた自動出題システムの提案

2010SE210 杉浦啓太 2010SE241 寺内雄基

指導教員：蜂巣吉成

1 はじめに

プログラミング言語の学習においてプログラミングの問題を繰り返し学習することは効果的な方法である [1]．しかし，講義でのプログラミングに関する演習時間には限りがあるので，学習者が解くことができる問題数が限定されてしまう．学習者が講義による学習で十分に理解することができなかった場合やより理解を深める場合にドリル形式のコースウェアでの自主学習方法がある．ドリル形式のコースウェアとは多くの問題に触れることができる学習支援システムである．コースウェアの利点は学習者が集まる必要はなく，それぞれ自由な場所や時間に利用できる点である．しかし問題の出題において，学習者の理解を考慮せず同じような問題を出し続けるという欠点がある．

既存の出題システムでは学習者の理解度に応じて難易度を変化させて出題するものがあるが [2]，これは学習者の苦手な分野を考慮しない．学習者のプログラミング能力を上げるためには苦手を克服させることが重要となる．また，学習者の解答の正誤判定のみで理解度を計算するので，正解者は皆同じ理解度となる問題点がある．

本研究の目的は，学習者の理解状況を把握し，学習者に合った問題を出題するシステムを提案することである．学習者の苦手分野を把握するために理解度モデルを用い，理解度モデルと問題の対応を明確化する．学習者の解答時間や正解に至るまでの誤答回数などを理解度の算出に考慮することで，学習者の苦手な箇所を把握し，その箇所を克服させるような出題が可能になる．学習者の理解している問題を出し続けるのはモチベーションの低下に繋がる．学習者に合った問題を出すことは学習の効率を良くすると考えられる．

2 関連研究

AEGIS は学習者の解いた問題により理解度を算出し，理解度に応じた難易度の問題を出題するシステムである [2]．学習者が自分の理解度よりも低い難易度の問題を正解することと，自分の理解度よりも高い難易度の問題を間違えることは自然であると考えられる．AEGIS では，そのような試行のデータは理解度の評価に使用しない．学習者の理解度より低い難易度の問題を正解しなかった時，学習者の理解度より高い難易度の問題を正解した時，AEGIS は学習者の理解度を過大，過小評価していると判断して理解度の再評価をする．これにより，自己理解度にあわせた難易度を出題するシステムを構築している．

MILES は学習者の理解度モデルに基づいて適応的に練習問題の推薦を行うシステムであり，C 言語に対する理解度をモデル化することによって学習者の学習状態を細かに把握し適切な問題を出題する [3]．理解度をモデル化するのを実現するのに C 言語の知識の単位として考えられるものを成績要素とし，その成績要素の包含関係を考慮して理解度モデルを作成している．理解度モデルから学習状況を判断し問題を推薦するシステムになっている．

QA サイクル実行システムとは，受講者への問題の提示から答案プログラムの評価までの一連の流れである．QA サイクル実行システムの中で受講者の習得度を計算し，その結果から習得度に応じた問題を提示する [1]．問題の難易度の上げ下げを受講者の習得度に合わせて教師がコントロールすることが可能になる．習得度の計算は問題に設定された配点の合計となる．

AEGIS では難易度を考慮して出題しているが，苦手分野を考慮していない，正解に至るまでの過程を理解度の判断に考慮していないという問題点が挙げられる．本研究では解答時間や誤答回数を理解度に考慮することや，学習者の理解状況を分野別に把握して出題することを考える．

MILES では全文記述の問題形式で出題しているが，理解度モデルと問題のどの部分と対応しているかが必ずしも明確ではない．本研究では理解度モデルと問題との対応のために最適な出題形式を考えた．

QA サイクル実行システムでは問題の解答を 1 箇所間違えると習得度は大きく変動することもあり，学習者の習得度を正確に把握することが難しい．本研究では，学習者の理解度の計算を加点減点方式を用いて行うことにより安定して理解度を測定できるようにした．

AEGIS では難易度，MILES では理解度と分野，QA サイクルシステムでは理解度を考慮して出題しているが，難易度と分野と理解度の 3 つを考慮した出題ではない．本研究では，理解度モデルを用いて学習者の理解状況を把握する．理解度モデルを構文要素と概念要素に分けて理解度を細かく測定することにより，学習者への最適な問題の出題を可能とする．理解度モデルを用いて理解度を把握し，その点数に応じて最適な難易度の問題を出題する．

3 理解度モデルを用いた出題方法の提案

3.1 出題方法の概要

学習者に効果的な学習を補助するための出題方法を提案する．効果的な問題を出題する上で必要なのは，学習者の理解状況を把握することである．理解状況というのは，学習者がある分野を，どの程度理解しているかを示すものである．学習者の理解状況を把握することで，学習者の理解

していない箇所を理解させることが可能になる．学習者の理解状況を把握する手段として理解度モデルを用いる．問題と理解度モデルを対応させ，学習者の解答をもとに理解状況を数値化した理解度を算出する．本研究が提案する出題方法の主な流れは，まず始めに学習意図を一通り網羅できる問題を出題し，学習者の理解度を算出した後，理解度の低い箇所の問題を出題させるという方法である．

今回，理解度モデルと問題との対応付けが比較的容易な空欄補充問題を採用する．

3.2 理解度モデルの概要

プログラミング言語の問題を出題する際，学習者に学んで欲しい学習意図を考えて出題する．この学習意図を用いて学習者の理解状況を把握する理解度モデルを作成した．まず，学習意図の決め方について説明する．学習意図とは学習者の理解すべき要素であり，教科書 [4] を参考に決めるのが良いと判断した．抽出した学習意図を，変数，条件分岐，ループ，構造体など 10 の分野に分けて理解度モデルを作成した．これにより，学習者の理解状況を把握しやすくした．この考え方は MILES と同様であるが，本研究ではさらに学習意図を整理する際，構文要素と概念要素の 2 つに分けた．プログラミング言語の文法に関する学習意図は構文要素に，プログラムの処理や流れに関する学習意図は概念要素に分けた．この理解度モデルを用いることにより，学習者がどこを理解していて，どこを理解できていないのかを細かく測定できる．

今回，学習モデルの関係性を考慮して理解度モデルの全体図を作成した．この全体図を用いて理解度の計算をすると，学習意図同士の関係が複雑であり理解度を正しく把握することが困難である．そこで，単元別に分けて理解度モデルを作成することにした結果，理解度モデルは木構造を用いたモデルとなった．学習意図はさらにいくつかの学習意図から構成されることがあり，階層構造になる．例えば，for 文という構文要素の学習意図はさらに，初期化式，条件，増減から構成される．

3.3 構文要素と概念要素

構文要素と概念要素の点数の決め方について説明する．理解度モデルでこの 2 つのノードは図 1 にあるように，根の 1 つ下の階層にある．さらにそれぞれの学習意図が構文要素と概念要素に分かれて下の階層にある．構文要素・概念要素はその下の階層にある葉の点数により決められる．下の階層が上の階層に与える配点の重みは大学の授業での出題頻度を基に考えた．それぞれの下の階層にある葉を理解すればその上にある階層のノードも理解していると判断できるようになっている．葉の点数は問題を解くことにより変動する．問題の空欄箇所と対応する学習意図の葉の点数が変動する．

問題を解くことにより，構文要素と概念要素に与える影響は次の 3 通りである．

1. 構文要素に影響する学習意図のみが対応
   (a) 終端文字 (構文要素)
   a[i]=’ \0 ’
   (b) アロー演算子 (構文要素)
   f -> age
2. 概念要素に影響する学習意図のみが対応
   (a) 初期値 (概念要素)
   ans=1;
   while(i >0){
   ans=ans+i;
   }
3. 構文要素と概念要素に影響する学習意図の両方に対応
   (a) if 文 (構文要素) と入れ子 (概念要素)
   if(a>0){
   if (b>0){

1 と 2 の場合，空欄と対応する学習意図の点数が変動する．3 の場合，両方とも同じ点数が割り振られる．

3.4 理解度の把握の仕方

次に理解度の把握の仕方について説明する．

例えば C 言語の繰り返しは while 文，for 文，do-while 文で記述することができるが，繰り返しにおける概念としては繰り返しにおける変数の初期値，繰り返しの条件，繰り返しにおける変数の更新などが挙げられる．これらを区別して表現できるように，繰り返しは構文要素と概念要素から構成されるように図 1 を作成した．

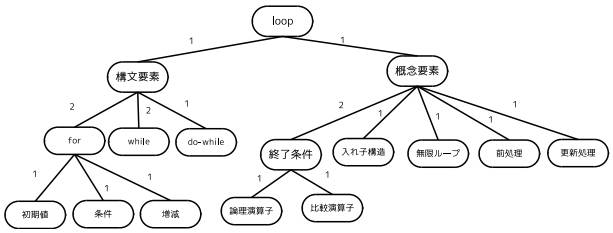

図 1 繰り返しの理解度モデル

例えば，for(e1 ; e2 ; e3) において，e1 は図 1 の構文要素の for 文の初期化式と概念要素の繰り返しにおける変数の初期値に対応する．次に，while 文の階乗計算を行うソースコードを考える．

while 文の階乗計算を行うソースコード

```
fact=1; i=1;
while( i <= n) {
  fact = fact * i;
  i++;
}
```

以上のソースコードでの，while 文による階乗計算の fact = 1; i=1; は概念要素の繰り返しにおける変数の初期値に対応する．これにより，while 文の初期化式の点数が

高く，繰り返しにおける変数の初期値の点数が低い場合は，while 文の文法は理解しているが，繰り返しの初期値は理解していないことがわかるようになると考えられる．

繰り返しという分野で論理式について学習すると，条件分岐における論理式の理解が深まると考えられるので，論理式は繰り返しの条件と条件分岐の条件から共有される．

3.5 理解度の算出方法

理解度の算出方法は，絶対評価での評価方法を用いる．各ノードの満点を 100 点，各葉ノードの初期値を満点の半分とし 50 点に設定する．学習者の解答が正解だった場合，空欄箇所に対応する理解度モデルの葉ノードの値が問題の難易度，誤答回数，経過時間に応じ上昇する．一昨年の我々の研究室の研究 [5] でも誤答回数や解答時間を考慮した点数づけであったが，一問一問点数を決める場合，理解度が大きく変動する可能性がある．また，解答数と正解数の割合で点数を決める場合，問題数が増えるにつれ変動が少なくなり，問題を解く中で理解が深まった場合でも理解度がほとんど上昇しないので，加点減点方式を使う．解答の正解時には，次の計算式が適用される．

(理解度)=(現理解度による点数)+(問題の難易度による点数)+(経過時間に対応する点数)-(誤答回数に対応する点数)

1. 問題の難易度による配点は，高難易度 15 点，標準難易度 10 点，低難易度 5 点とする．
2. 誤答回数に対応する点数は，1 度間違える毎に 1 点 (1 問に対する解答回数は最大で 10 回) 減点とする．
3. 解答時間に対応する点数は，3 分以内で 3 点，5 分以内 2 点，7 分以内で 1 点 (1 問の制限時間は 10 分) とする．

1 つの問題で同じ学習意図の空欄が複数出現する場合，1 つ目の空欄が正解ならば，それ以降の空欄も正解できると考えて，2 つ目の空欄の配点は半分に，3 つ目の空欄はさらに半分とする．

2 回目以降での正解は，間違いであることが分かってからの解答なので，1 回目での正解よりは理解度が低いと考えた．解答回数以内で答えられない，または制限時間以内に答えられなかった場合は，難易度に関わらず 5 点減少するという方法を採用した．

理解度が 40 点以下の場合には低難度の問題，70 点以上で高難度の問題を出題するようにし，学習者の難易度に合わない問題を出し続けることにならないように点数を決めた．

難易度の決定方法については，問題を解く場合，学習者はある程度プログラムのソースコードを読まなければならないので，問題の難易度は空欄毎に設定せず，ソースコード毎に設定する．難易度の基準としては，処理の複雑さを主に考える．

理解度モデルの根の部分，その単元の理解を高めること，苦手分野を克服させるような出題を目的にするので，理解度モデルの葉ノードの理解度の低い問題の類似問題を出題することを優先する．算出方法に絶対評価を用いた．相対評価での算出方法では他のノードとの比較で理解度の低さを捉える．しかしこの算出方法では，学習意図をどの程度理解しているかがわからないので，どの段階で難易度の高い問題を出題するべきかが不明確になってしまうという問題点があるので，絶対評価を用いた．

4 学習プロセス

学習者の理解できていない問題を理解させる方法について考えた．まず，構文要素と概念要素とで覚え方が異なるということに注目した．構文要素とは構文，つまりどうやって決められた文法に則って書いているかというものである．概念要素とはプログラムの処理，流れを理解できているかということである．効果的に学習させるためにはこの 2 つを分けるべきと考えた．

構文要素は．プログラムの書き方を覚えているかどうかである．学習者に問題を与え，答えられなかった場合は解答から書き方を学ぶ．次の問題でもその構文要素を含む空欄箇所に答えが書くことができない場合，難易度を下げた問題を出題する．同じソースコード内にその構文を 2 つ用意し，片側を空欄にする．これにより，書き方を見ながら真似をして書くことにより構文を覚えさせるという方法をとる．

概念要素は，プログラムの処理や流れのことである．標準問題を解くことができない場合は，比較的単純な処理をする問題を解かせることから始めることにより，理解できるような仕組みにする．始めに難易度の低い問題から出題しない理由は 2 つ挙げられる．1 つ目は，標準問題から始めることにより，容易な問題を解かせるという手間を省きモチベーションの低下を防ぐことである．学習者が解くことができない問題を解かせることは学習者のモチベーションの低下につながるので，標準問題でも解くことができない場合は 2 問ほどで難易度を下げるようにした．2 つ目は，大学の講義などで十分な理解を得られなかった，またはより理解を深める場合には自主学習用のコースウェアとしてのシステムであるので，標準問題から出題した場合の方が学習として効率的であると考えた．

5 評価

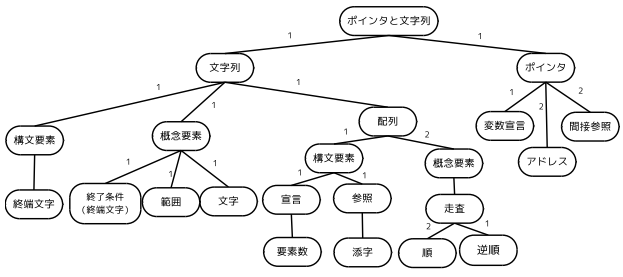

図 2 ポインタと文字列の理解度モデル

出題システムのプロトタイプを実装し，学生 10 人ほどを

対象にテストを行った．この評価実験の目的は，学習者が理解している箇所，理解していない箇所を把握し，理解度の低い箇所を補うことが出来ているのか，システムを使用して効率よく学習できているのかを評価することにある．

評価実験では，文字列とポインタの単元の問題を出題した．学習者には，理解度を測るための問題を 4 問と最後の確認用に 2 問の問題，理解度を測る問題と確認用の問題の間に問題を数問解いてもらった．A グループにはシステムを使った出題方法を用いて，理解度モデルの頂点の学習意図全てが 85 点以上，または 5 問出題した後確認テストへ進むようにした．B グループにはシステムを使わずに用意した問題を一様乱数に当てはめて選ばれた問題を決められた順番に 5 問出題した．図 3 は理解度モデルの根の部分，文字列とポインタについて，A グループと B グループとの点数の変化を示した図である．

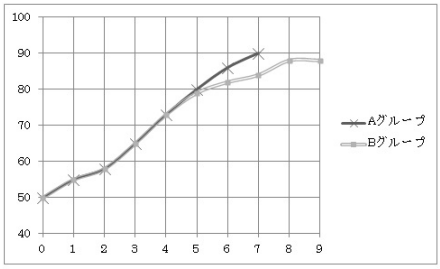

図 3 A グループと B グループの理解度の変化の比較

A グループ，B グループともに始めの 4 問は同じ問題なので，理解度モデルの点数にはほとんど差は無かったが 5 問目以降に差が表れた．A グループでは 5 問目以降も順調に理解度の上昇が見られるが，B グループではその伸びが緩やかである．また，9 問目では変化が見られなかった．

次にポインタと, 配列の点数の変化を比較した．最終的な理解度はどちらもほぼ同じであったが，その過程では違いが見られた．始めの 4 問を解いた時点での配列の理解度が 72 点，ポインタの理解度が 69 点であり，ポインタの理解度の伸びが配列と比べて悪かった．学習者の解答を見てみると間接演算子をつけるかつけないかでの間違いが多く，平均回答数もポインタの方が多かった．解答回数が多いので，かかった時間も長くなり時間による加点も配列より少なかった．配列の問題は確認テストを除いた 7 問中 3 問で平均回答数は 1.4 回，残りの 4 問がポインタを含む問題で平均回答数が 2.3 回であった．誤答回数がポインタの方が多いので, 理解度の伸びに変化が表れたと考えられる．

6 考察

B グループの学習者はいくつかの学習意図の理解度が上がり切ってしまい全体の理解度はあまり上がっていかないという状況が生まれた．一方 A グループの学習者は，理解度の低い箇所の理解度が上がっていくことにより，全体の理解度は順調に上がっていくことになった．すでに理解していると判断できた問題を出題しないことにより，出題する問題数を削減できた．学習者の理解度の低い問題を出題するということは，学習者の理解度を効率よく上げることの手助けになったと考えられる．

今回の評価実験では配列よりポインタの方が苦手である学習者がほとんどであった．A グループではポインタの問題を多く出題することにより，苦手を補うことが出来た．しかし，問題の後半でも間接演算子での間違いがあった．1 度目での解答での正解は理解していると考えられるが，2 度目 3 度目での正解は理解がそこまで高いとは考えにくい．誤答回数による点数の減点の方法を見直したい．ポインタのほうが出来が悪かったが，ポインタと配列の理解度を見るとあまり変化が見られなかった．今回の評価実験では誤答 1 回に対し 1 点減点していたが，この方法では点数の変化があまりない．そこで，標準問題の解答時，1 回目なら 10 点，2 回目なら 5 点，3 回目なら 2.5 点といったように半減させていく方法に見直しをする必要があるのではないのかと考えた．

7 おわりに

本研究では，学習者の理解状況に合わせた自動出題の方法について提案した．学習者の理解度の把握に理解度モデルを用いた．問題の空欄箇所と理解度モデルの葉の学習意図と対応付けし，理解度を把握する．出題方法の流れとしては理解度を把握するための問題セットを解いてもらい，学習者の理解度の低い学習意図の問題を出題するという流れである．評価実験を行い，学習者の理解度の低い問題を出すことで理解度を効率よく伸ばすことが確認できた．

今後の課題としては，より学習者の理解度を正確に把握できるように問題の配点や問題と学習意図の対応の妥当性の確認が必要である．

参考文献

[1] 中島 秀樹，高橋 直久，細川 宜秀，“プログラミング学習のための QA サイクル”， 電子情報通信学会論文誌． Vol. J88-D-I， No.2， pp. 439-450， Feb. 2005

[2] 菅沼 明，峯 恒憲，正代 隆義，“学生の理解度と問題の難易度を動的に評価する練習問題自動生成システム (学習支援)”，情報処理学会論文誌，Vol. 46，No.7，pp.1810-1818，Jul.2005

[3] 小芦 勇介，玉木 久夫，“理解度モデルに基づく C 言語学習支援システムの設計と実装および試用”，情報処理学会研究報告. コンピュータと教育研究会報告. Vol. 2013-CE-119，No.1，Mar.2013

[4] 柴田 望洋，「新版 明解 C 言語 入門編」，ソフトバンククリエイティブ株式会社，2004

[5] 加藤 拓也，森 拓矢，沖 良太，“理解状況度に着目した空欄補充問題によるプログラミング学習支援システムの提案”，2011 年度南山大学卒業論文.