プログラミングの習熟過程の検証

Verification of the Mastery Process of Programming

木　下　和　也
Kazuya KINOSHITA

和文要旨

プログラミングを学ぶ学生の学習観，行動，理解度に着目し，習熟過程のモデル構築とシミュレーションを行った。

先行研究から，大学生は失敗に対する柔軟的な態度があり思考過程を重視する傾向にあり，このような傾向は，実はプログラミングの学習に適しているといえる。この学習観をもとに授業の改善を行い，その効果確認のため理解度のアンケートを実施してきた。この授業で実施したアンケート結果，出席管理データ，期末試験の成績に基づき，学生の習熟に要する時間，授業の出席を諦めて離脱する傾向を，メンタルモデルとして導き出し，システムダイナミックスによるプログラミングの習熟過程を再現することができた。

英文要旨

In this paper, I performed the model construction and the simulation of the mastery process paying attention to the study view of the student who studies programming, action, and degree of comprehension. In precedence research, it is drawn that university students have a flexible attitude against failure, and tend to think a thinking process as inportant. In fact, such tendencies are suitable for study of programming. The lesson has been improved based on this study view, and the questionnaire of degree of comprehension has been carried out for that effect check.

Based on a questionnaire result, attendance data, etc. from students in this lesson, I performed the simulation about mastery of the student using system dybamics.

和文キーワード：プログラミング　学習観　習熟度　理解度　システムダイナミックス
英文キーワード：Programming　Study view　Skill level　Degree of comprehension　System dynamics

目　次

はじめに

多くの学生は真面目に授業に出席しており，プログラミングが簡単ではないだろうという意識を持って受講している。であれば，その後の出席率の低下から途中で授業を辞める原因を特定し，できる限り最後まで受講させ，目標とする知識とスキルを身につけさせることが教員の重要な任務となる。しかし，現実の問題として毎回の授業での理解度の個人差は大きく，50名ほどの受講生全員に対して同様の理解を得られることは不可能である。

人間の行動にはメンタルな要素が多く，定量的にその因果関係を導き出し，即時に授業改善に活かすことは難しい。本稿では，先行研究から得られた大学生の学習観に基づき，授業の改善を試み，その効果をアンケートにより検証してみた。さらに，その結果から受講する学生のメンタルモデルを推測し，システムダイナミックスによるシミュレーションを実施し，学生の習熟過程を再現することができた。このシミュレーションモデルのパラメータを調整することで，新たな授業の改善点を探ることができ，プログラミングの学習効果を高める上で参考とすることができる。

1　学生の学習観とプログラミング

大学でのプログラミングの学習と指導を考える上で非常に興味深い先行研究として，堀野・市川[4]では「失敗に対する柔軟的態度」と「思考過程の重視」の2側面，および，「自己充実的達成動機」と「競争的達成動機」の2側面から結果を導いている。

前者の2側面については相関が高く，さらに，結果として図1のような分布が得られている。したがって，その相関の高さとこのグラフから，失敗に対する柔軟的な態度があり思考過程を重視する傾向の学生が多数を占めていることといえる。このような傾向は，実はプログラミングの学習に適しているといえる。

実は，プログラミングに関連のある学習観としてはPapart[1]の先行研究の言葉が印象的である。そこでは，「プログラミングにおけるデバッグという体験に基づく学習が失敗を恐れず，誤りを次第に直しながら，完全なものに近づけていけばよいという柔軟な学習観を育てる」と主張している。

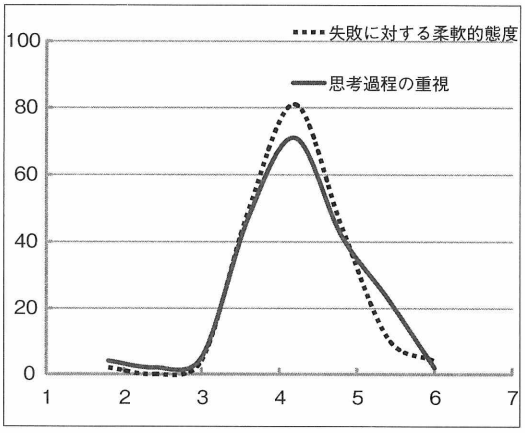

図1　失敗に対する柔軟的態度と思考過程の重視の分布（横軸は調査結果のスコアで1～7点，縦軸は被験者数で対象は194名）
出所：堀野・市川[4] p5-6に加筆

例えるなら，とにかく問題を解くことができれば理解しなくてもよいという態度とは逆であり，時間がかかっても何とか理解したいという学生が多いともいえるからである。そうなると，指導する教員の立場としては，この学習観を理解し，正解するまで待つ，理解するまで待つといった対応で臨むべきであろう。しかし，現実問題として授業時間を学生の理解待ちで遅らせることはできないため，ここに教育工学的検証と実践が求められることとなる。

基本的学習観と達成動機に対しては優位な相関を持っていない。

この結果から，堀野・市川[4]は「学習において失敗を柔軟に捉え，考える過程を重視する態度は，自己の充実を求める動機付けと深い関連があり，これらの態度は競争心に支えられた学習態度とはほぼ独立なものと考えられる」としている。

筆者のプログラミングの授業では，名古屋市内のシステム開発企業の協力を得て，そこに勤務する本学卒業生や社長にプログラミングの役割について講演を依頼している。また，受講生の一部は課外学習を通じて実際に授業で利用するアプリケーションの開発も行っており，この内容を紹介することも授業内容としている。これらの講演や報告，および実際に学生が作成が

開発したアプリケーションを利用することで，プログラミングと社会におけるその役割の理解，または職業としてのエンジニアへの興味を持たせる一助としている。

その影響もあり，2009年7月に授業で実施したアンケートでは45人中の24人が情報関連企業に就職したいと考えており，一方で授業当初経営学部で学ぶ学生が情報関連企業に就職することは関係がないと考えていた学生が12人いたが，最終的には0人となっている。授業を通してプログラマー，システムエンジニア，プロジェクトマネージャといった職業についても話をしており，なぜプログラミングを学ぶのかといった目的意識を持たせることも重要視している。

このことは，授業において自己実現達成動機に影響を与え，授業出席への意欲を高めるとともに，その結果として「失敗に対する柔軟的態度」と「思考過程の重視」の2側面にも影響した可能性がある。

なぜなら，学生の学習観が上述のような動機によって支配されているのであれば，誰かと競争するわけでもなく自己実現により知識とスキルを向上させることが大きな意欲増進に繋がるからである。したがって，その自己実現の目標を大きく掲げることが方策のひとつであると考えられる。

2　繰り返しによる理解の向上

授業で学ぶコンピュータ言語は，アルゴリズムの理解をほぼ要しないHTMLのようなマークアップ言語と，目的とする処理を完遂するためのアルゴリズムを記述するC言語のようなプログラミング言語とに大きく分けることができる。HTMLの特徴は文字や画像の配置及びリンクを基本として，タグの使い方を理解するだけで多様なWebページ作成はできる。しかしながら，プログラミング言語を理解するためには，言語特有の書き方だけではなく，さまざまな処理のアルゴリズムを理解しなければならない。多くの場合，プログラミングを不得意とする学生はこのアルゴリズムを理解できていない。

HTMLとプログラミング言語を比較すればわかるように，1行ごとに画面表示として結果を返すHTMLに対して，プログラミング言語は複数行に渡って構成されるループ処理や，サブルーチン，関数ごとに意味のある処理がなされるため，処理の流れを理解する必要がある。

このようなアルゴリズムの理解には，適性の有無が大きく関わってくるが，初級プログラミングにおいては，練習によって理解度を上げることができるという仮説を立て，反復練習を取り込んだ多言語によるプログラミング学習を展開した。多言語によるプログラミング学習は，同じアルゴリズムを複数の言語で繰り返し練習することである。

これまで，筆者は担当するプログラミングの授業において，複雑なアルゴリズムを理解できない学生に対して，よりわかりやすい教材や課題を作成したり，オフィスアワーを拡大するなどの行動で対処してきたが，大きな効果を得たという実感はなかった。また，内容のレベルを下げて，対応したこともあったが，この場合，適性のあると思われる学生が授業中に退屈してしまい，結果として，このような習熟度の高い学生に別途課題を与えるなどのケアに追われる結果となった。

過去にこのような経験と背景があり，できる限り多くの学生が理解でき，さらに適性があり習熟度の高い学生が退屈せずより関心を持って授業に出席できる授業構成やコンテンツを模索した結果，複数言語により同一アルゴリズムの繰り返し学習に行き着いた(注1)。

学習時間の配分としては14回の授業のうち，前半7回をフローチャートとBASICによって展開し，残りの4回をC言語，3回をPerlによって展開した。3度にわたって言語は変わるが，同じアルゴリズムでのプログラミングを学習させるため，最終的には多くの学生が理解し，同時に適性が高いと思われる学生にも言語を替えることで退屈せずに最後まで積極的に学べるものと考えた。

この授業での理解度を受講生45人を対象に調査したところ，図2の結果を得た。ただし，アンケートは0から10までの理解度を，学生自身の実感として記入させたので，主観的なものである。

またグラフは平均値を元に作成しているの

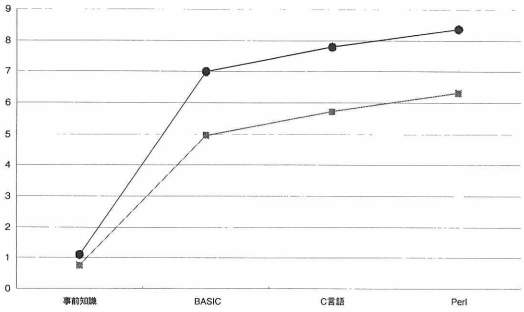

図2　繰り返し学習による理解度の推移(2009年7月調査　上は時間外学習ありと応えた学生の平均値推移,下は時間外学習がないと応えた学生の平均値推移)

で，回を重ねるごとに理解度は上昇傾向にあるが，一部ごく少数の学生は，回を重ねるごとに理解度が下がっているケースもあった。無記名アンケートであるため，理由はわからないが，欠席により授業内容が理解できなくなったり，言語が変わることで，混乱したなどの可能性もある。

時間外学習している学生としていない学生との比較においては，どちらも事前知識はほとんどない状態からスタートし，最終的にはある程度の理解に達しているといえるが，時間外学習を行っている学生の方が明らかに理解度が高い。さらに，最終的な理解度の平均値においては，標準偏差が小さく，ほぼ同じような理解度に達していることがわかった。それに対して時間外学習をしていない学生の標準偏差は比較的大きく，授業だけで理解できる学生とそうではない学生の差が見て取れる。

したがって，時間外学習の有無に関わらず，反復練習による理解度の向上には効果があるが，時間外学習を行わなければその効果が薄れることがわかる。

この方法を導入して以降，それ以前と比較すると学期半ばで授業に来なくなった学生が明らかに減少している。効果としては，繰り返すことで理解が深まるだけではなく，むしろ，わからなくなっても同一学期内に再度学ぶことができることで，あきらめずに再度取り組む意識を持たせ，継続的に授業に出席し，最終的には繰り返し学習の効果で理解できたものと思われる。その意味ではこの繰り返し学習は，授業に継続して出席させるための方策と捉えるべきであろう。

3　理解度による検証

3.1　理解度と習熟度

ここでは，理解度と習熟度という2つの単語を用いるが，その違いを定義しておく。

理解度はあくまでも理解がどのくらいできたかを示すものであって，その知識を用いてその知識の範囲で解くことが可能な応用問題を解けるかどうかは含まない。

プログラミングのアルゴリズムを例にとると，バブルソートのアルゴリズムにおいて昇順にソートするプログラムは理解できたが，降順にソートするプログラムは作れるかどうかというようなことで説明できる。

この例は単純なケースであるが，実際，昇順のソートが理解できたという学生のうち何人かは降順にソートするプログラムは作れない。また作れるにしても瞬時に作る学生もいれば時間がかかる学生もいる。このような場合，理解はできているが習熟していないということになる。本稿においては理解度と習熟度をそのように使い分けて議論を進める。

3.2　理解度のアンケート

次に毎回の授業の理解度と授業の継続およびプログラミングスキルについての検証を行いたい。この検証には2010年の秋学期に行った授業アンケートを用いる。

この授業では毎回の授業で理解度のアンケートを行った。まず，そのアンケートに関して説明したい。アンケートは毎回の授業終了直前に行っている。質問内容は図4に示されている内容である。このアンケートは携帯電話の画面を利用して実施しているため，画面が小さく，アンケートの選択肢の文章を長く書くことはできない。そのため，できるだけ簡潔な文章且つ理解度のイメージが伝わるように心がけて作成した。しかし，学生の性格の差，たとえば遠慮がちであったり，逆に自信過剰であったりすると，結果が逆転する可能性もあるので，そうならないように，また，文章から受けるイメージの差異が大きくなることを避けるため，毎回のアンケート実施時に理解度のイメージを補足説明している。なお，対象となる学生は春学期から継続して受講している学生が73%と大半であり，

『処理2(20110111)』

質問.1
本日の理解度
◎ よく理解できた　あとは忘れないようにおぼえるだけ
◎ たぶん理解できていると思うが、復習してきちんと理解を確かめる
◎ 大体理解できたが、完全ではない。復習が必要と自覚している
◎ なんとなくわかるのだが、理解できているとはいえない
◎ ほとんど理解できないか、まったく理解できない

送信する
リセット
戻る

図3　アンケートの選択肢(注2)

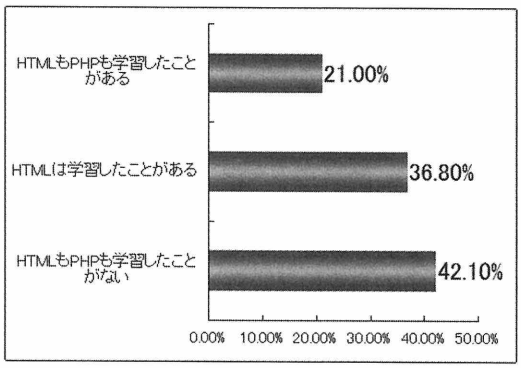

図4　Webプログラミングの事前知識
(2010年10月調査)

彼らは春学期末試験の合格者である。

3.3　授業の進め方

春学期の受講者または同様の知識とスキルを持っている学生を対象としているが，新規に秋学期から受講することも可能であるため，その対策を行っている。具体的には，授業の1回目から3回目は春学期の復習という名目で，基本的なアルゴリズムの整理を行っている。秋学期から新規に学習する内容はその続きに相当するアルゴリズムとそれらを利用したWebプログラミングである。また，図4の調査からは，Webプログラミングに必要な基礎知識の既習状況がわかる。80%ほどの受講生はPHPの知識がない。この数字には春学期から受講している学生の大半が含まれる。そのため，秋学期から受講した学生であっても，Webプログラミングについてのスタートラインは同じであり，受講できるように配慮している。

4回目の授業からは各種ソートのアルゴリズムの理解と，バブルソート・プログラムの実装をベースにPerl，C言語によって学習させた。図5はこの4回目の授業以降の理解度を集計した結果である。このグラフを作成する当たり，学生によって理解度の解釈が異なる点に注意した。前述のように学生は多くの場合，多少遠慮がちに自己評価をする傾向が見られる。しかし，授業についていけないほど難しいと感じない限り，理解が低いとは自己評価しないようである。そのため，授業についていけないほど理解度が低いものと，その中間層の3つに分類してグラフ化した。すなわち，上述したアンケート項

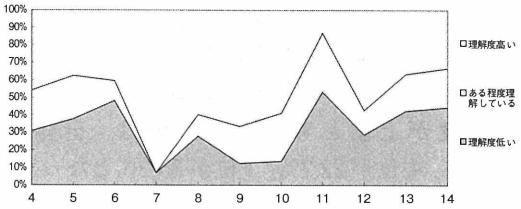

図5　理解度の推移（第4回目以降2010年10月より2011年1月までの調査）

目について「よく理解できた」および「多分理解できた」を理解度が高いと判断，「なんとなくわかる」と「ほとんど理解できない」を理解度が低い，「大体理解できた」を中程度の理解としてその推移をグラフ化した。

このグラフから受講生の理解度の推移を読み解いていきたい。第4回目から第6回目までは授業が進むたびに理解度が下がっている。これは授業が進むに連れて難度が上がっていることが主たる要因である。第7回目に理解度が高い学生が多い理由はHTMLの基本構造と記述方法がテーマであり，これまでのようなアルゴリズムに内容がまったく入っていなかったからである。また図4の事前知識調査からも明白なように，HTMLの学習経験者が36.8%含まれていることも理由である。第8回目の授業の理解度が若干下がっているのは，授業の難度の問題というよりも，サーバトラブルのため，受講生に混乱が生じたことが原因であると考えられる。第9回目にあたっては，第8回目にサーバトラブルでで確認できなかった実習結果を各自で確認できたことと，第8回目のアンケート結果と一部学生とのコミュニケーションにより直

接わかりにくい部分を聞き出し，それを補足する資料を作成，配布したことが功を奏した可能性があり，理解度が若干上昇した。それ以降，授業内容が進むに連れて理解度は徐々に下がっていくのは，難度が上がっていくためである。ただし，難度が上がり理解度が下がることと，授業についていけなくなることはまったく別の問題であり，自主的な時間外学習や課題を解くことで習熟度が上がり，翌週までには前回内容の理解はできている学生が多いと考えられる。それを示す根拠が授業期間最終日のアンケート（図6）と，期末試験の結果である。

図6は，授業期間最終日のアンケートにおいて，その日ではなく，授業期間を通しての理解度を調査した結果である。このグラフから，最終的にどのくらい理解しているかを判断した。理解度が上位と考えられるのは，全体の44.25%，中位が22.2%，下位が33.2%となった。実は試験の結果はこれよりも上位にシフトしており，授業終了後試験までに復習した結果，または遠慮がちにアンケートに答えたとも考えられる。当日の授業は理解度が低くても，それまでの内容を理解していると実感している学生は，理解度を高く自己評価している。したがって，このグラフは，毎回の授業での理解度とは異なり，主観的ではあるが学生が自覚する習熟度であると判断できる。実際，期末試験の評価は，このグラフに近い分布であり，継続して出席した学生の習熟度は日ごと高まるものと考えられる。

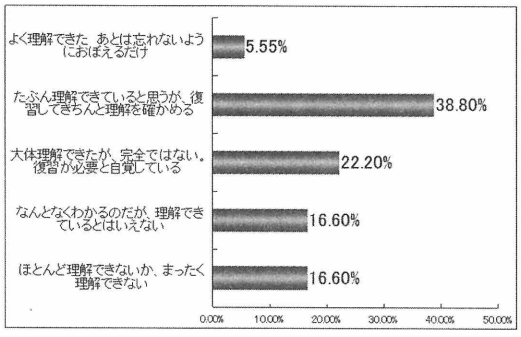

図6　全体的な理解度の分布（2011年1月調査）

4　習熟度と学習の継続モデル

学生が最後まで離脱することなく授業に出席し，中位レベル以上の習熟度を獲得することを目的とする場合，授業の改善がどのように影響するかを考えたい。

4.1　習熟過程モデルの構築

これまでに述べた授業での改善は，大学生の学習観を失敗に対して柔軟的であり，思考過程を重視するものとして学習内容を計画し，学習意欲を維持するために社会におけるプログラミングの役割を卒業生や企業者の協力によって伝えることで，成し遂げられている。この効果は，授業期間を通じて自己実現のため理解しようと努力する意欲に影響し，習熟度を高める効果があると考えられる。また，繰り返し学習によって，当初は理解できていなかった内容を繰り返すことで理解を高め，習熟度が上がる効果が見られた。この効果は特に苦手意識を持つ学生が授業の初期の段階で離脱していくことを防いでいるものと考えられる。

毎回のアンケート結果と最終的な習熟度の関係から，毎回の授業での理解度がそれほど高くなくても，自主的な学習によって，結果として習熟度を上げている現実があり，上述したような改善や工夫によって自主学習への意欲を高めることが可能であると考えている。

これらの結果を踏まえ，プログラミングの授業における受講生の習熟度過程のシミュレーションモデルを提案する。

受講生は当初そのほとんどが習熟レベルが低い，つまり初心者であるとし，最終的に習熟度が高くなっていくと仮定する。その段階を図6のグラフと同様に3つに分類し，習熟度が低い学生，中位の学生，高い学生とした。また，低い学生と中位の学生は，途中で授業に来なくなる傾向を持たせており，それが離脱係数のグラフ関数によって定義されている。習熟度の低い学生は，時間とともに中位にレベルアップし，注意の学生は高い学生へとレベルアップするが，これに要する時間をそれぞれ設定した。

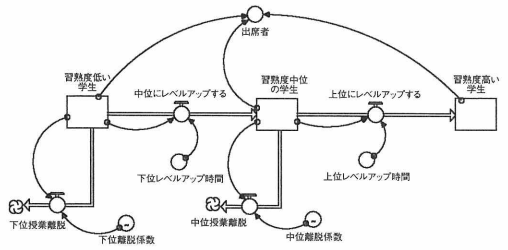

図7　習熟と離脱のSDモデル

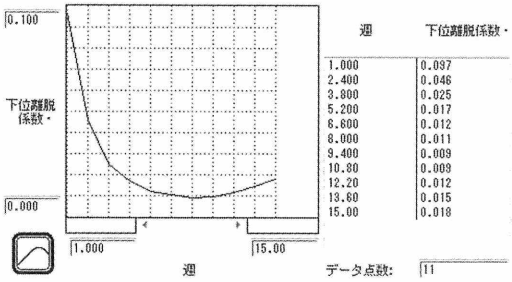

図8　下位レベルでの離脱係数の変化(テーブル関数)

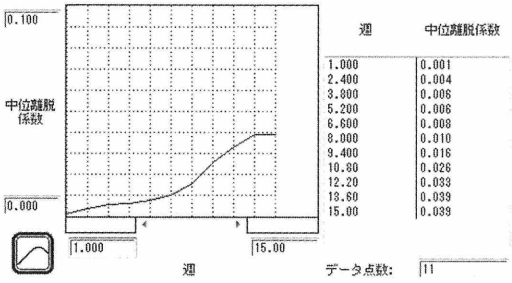

図9　中位レベルでの離脱係数の変化(テーブル関数)

次にこのモデルの方程式群を以下に示す。

習熟度高い学生（t）= 習熟度高い学生（t - dt）+（上位にレベルアップする）* dt
初期値 習熟度高い学生 = 2

上位にレベルアップする = 習熟度中位の学生 / 上位レベルアップ時間

習熟度中位 _ の学生（t）= 習熟度中位の学生（t - dt）+（中位にレベルアップする - 上位にレベルアップする - 中位授業離脱）* dt
初期値 習熟度中位 _ の学生 = 8

中位にレベルアップする = 習熟度低い学生 / 下位レベルアップ時間
上位にレベルアップする = 習熟度中位の学生 / 上位レベルアップ時間

中位授業離脱 = 習熟度中位の学生 * 中位離脱係数

習熟度低い学生（t）= 習熟度低い学生（t - dt）+（- 中位にレベルアップする - 下位授業離脱 * dt
初期値 習熟度低い学生 = 50

中位にレベルアップする = 習熟度低い学生 / 中位レベルアップ時間
下位授業離脱 = 習熟度低い学生 * 下位離脱係数

中位レベルアップ時間 = 12

出席者 = 習熟度低い学生 + 習熟度高い学生 + 習熟度中位の学生

上位レベルアップ時間 = 10

下位離脱係数 = グラフ（TIME）
（1.00, 0.0965）,（2.40, 0.046）,（3.80, 0.025）,（5.20, 0.017）,（6.60, 0.012）,（8.00, 0.0105）,（9.40, 0.009）,（10.8, 0.0095）,（12.2, 0.0115）,（13.6, 0.0145）,（15.0, 0.018）

中位離脱係数 = グラフ（TIME）
（1.00, 0.001）,（2.40, 0.0035）,（3.80, 0.0055）,（5.20, 0.006）,（6.60, 0.0075）,（8.00, 0.01）,（9.40, 0.0155）,（10.8, 0.0255）,（12.2, 0.033）,（13.6, 0.039）,（15.0, 0.039）

4.2　シミュレーション結果の検証

これらのテーブル関数の分布やレベルアップに要する時間を変えることによって複数の実験を行ったところ，今回の授業アンケートと試験の結果および出席管理データとほぼ同じ結果を得ることができた。その結果を図10から図12にかけて示す。

これらの結果から，繰り返し学習の効果がなければ，授業からの離脱者，特に習熟度の低い学生の離脱者のテーブル関数は全体的に上方にシフトしているものと考えられる。また，上位へのレベルアップ時間はより長い時間となっているであろう。結果として最終的な受講者数は

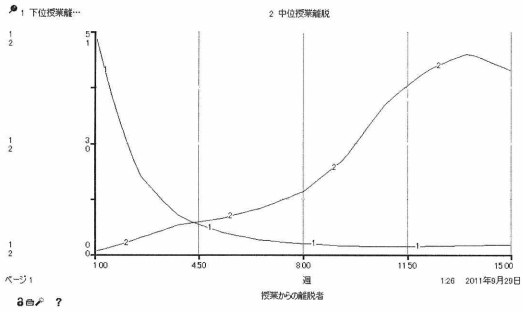

図10　下位および中位での離脱者数推移

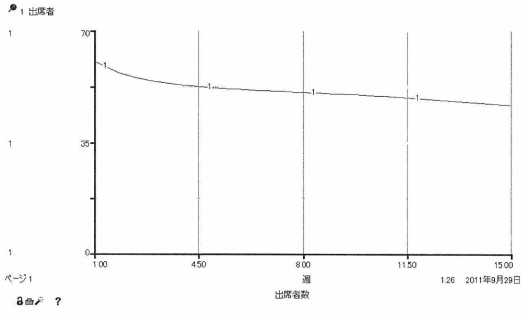

図11　受講者数の推移

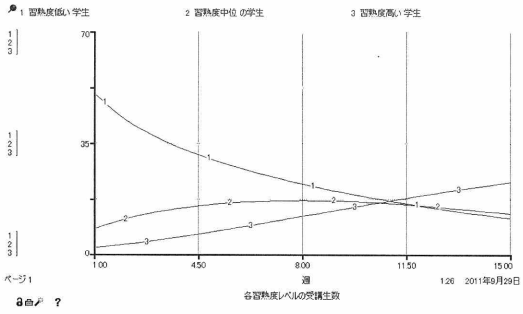

図12　習熟度別受講生数の推移

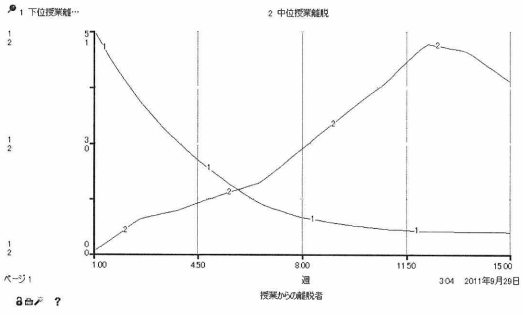

図13　下位および中位での離脱者数推移(授業改善前)

減少し，全体的に習熟度は低くなるであろう。その状態のシミュレーション結果を改善前の状態として，図13から図15に示す。

これらのシミュレーション結果を参考に，より高い習熟度と目指すのであれば，グラフ関数

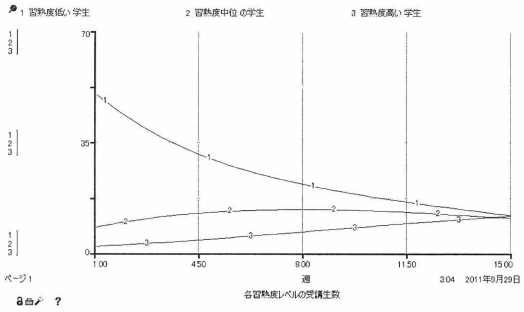

図12　習熟度別受講生数の推移(改善前)

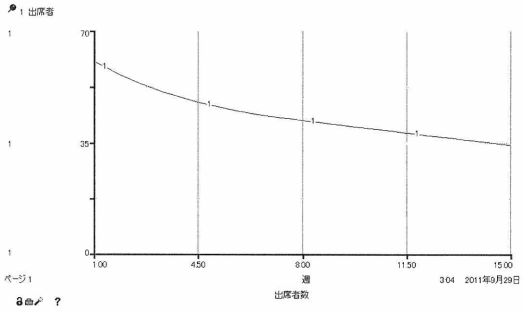

図11　受講者数の推移(授業改善前)

をより下方へシフトするような工夫を見つければよいし，何らかの方法で全体的な意欲向上を図れば，レベルアップ時間がより短くなるであろう。

まとめ

筆者の担当する授業を標本として実施したアンケート，出席管理，成績分布から，プログラミングを学ぶ学生の習熟過程を推測しシミュレーションを行った。

シミュレーション実施過程においてアンケートや期末試験の成績の分布とより近い結果を集めて検討した結果が本稿の提案するシミュレーションモデルである。

シミュレーションの元となった授業以前のアンケートが存在しないため，授業からの離脱者数と成績でしか判断できないが，できる限り授業の離脱者を少なくする方法をとるためには，各レベルの学生の離脱係数の変化を変えるような授業改善を行えばよいと判断できる。このモデルを雛形とし，毎年の同様の授業と比較することで，授業アンケートなどでは拾えない学生の習熟度の変化を捉え，授業の反省点を見つけ

るツールとすることはできるであろう。また，異なる科目においても，その科目に特有の学生の習熟度変化や，それを調整するパラメータが存在しうるので，今後は，他の科目への応用を考え，比較していくことで本領域の研究を進めていきたいと考えている。

注

（注1）この学習方法では，アルゴリズムに関しては春学期に行っていた従来の内容の一部を秋学期に移行し，70％ほどの内容に絞って展開した。しかし，この内容を3回にわたって言語を替えて繰り返すため，学ぶ知識量としては減少していないと考えている。また，秋学期に継続して受講する学生にとっては，年間を通じて同じ分量のアルゴリズムの知識を学べるように秋学期の授業配分も考慮している。

（注2）このアンケートについては，木下［5］において述べられているシステムを利用している。学生の携帯電話（スマートフォン）のWeb画面に表示される質問に対する回答によって集計を行っている。

参考文献

［1］Papert, S."Mindstorms: Children, Computer, and powerfyl ideas", Basic Books, 1980（奥村貴代子訳　マインドストーム－子ども，コンピュータ，そして強力なアイデア」未来社　1980年）

［2］Richmond, B., "Introduction to Systems Thinking", isee systems, 2004（バーシティウェーブ訳「システム思考入門Ⅱ　ビジネス編」カットシステム 2004）

［3］Tom DeMarco, Timothy Lister, "Peopleware: Productive Projects and Teams（Second Edition）",Dorset House, 1999

（松原 友夫，山浦 恒央 訳「ピープルウエア 第2版－ヤル気こそプロジェクト成功の鍵」日経BP社；第2版 2001年）

［4］堀野緑・市川伸一「大学生の基本的学習観の形成要因の考察－心理尺度と面接法による学習者情報と活用－」教育情報研究 1993年 pp.3-10

［5］木下和也「学生による経営学部内Webシステム構築について」愛知学院大学経営管理研究所紀要　第15号 2008年 pp.9-18

［6］土金達男「シミュレーションによるシステムダイナミックス入門」東京電機大学出版局 2005

［7］文部科学省「高等学校学習指導要領解説情報偏」開隆堂　2010