プログラミング教育における学習コンテキストの獲得

正　司　哲　朗*

Acquisition and Analysis of Learning Context for Programming Education

Tetsuo SHOJI

要　旨

近年、高等教育機関の学習環境の多くは、コース管理システムや学習管理システムなどのWebベースのe-learningシステムを導入している。しかしながら、現在、一般的に利用されているe-learningシステムは、プログラミングやシミュレーションといった実習を行うには機能が不十分である。なぜならば、これらの実習を対象にした場合、e-learningシステムの最大の特徴である学習履歴が残らないからである。本研究では、プログラミング実習を対象に、学習者の履歴情報を取得し、解析することにより、学習者を支援することを目的とする。このため、本研究では、プログラミング実習において、入力情報、端末出力情報、姿勢情報の学習コンテキストを獲得した。獲得されたコンテキストを用いてモデル化を行い、支援すべきシーンの抽出を試みた。実験結果として、モデルの評価値の高い時間帯の中にはプログラミング誤りが生じているシーンが含まれていたことが確認された。

Ⅰ．はじめに

近年、高等教育機関の学習環境の多くは、コース管理システム（CMS）や学習管理システム（LMS）などのWebベースのe-learningシステムを導入している。e-learningシステムを利用する利点は、時空間の制約なしに学習でき、学習者の学習履歴情報が蓄積され、学習者間、もしくは学習者／講師間で手軽にコミュニケーションが出来る点にある。また、このようなe-learningシステムは、情報の共有が容易に実現可能である[1)]。

しかしながら、現在、研究開発されているe-learningシステムは、プログラミング実習を行うには機能が不十分である。なぜならば、このような実習をe-learningシステムを使って利用する場合には、実習課題や説明をシステムに掲載し、学習者がそれらを見ながら、実習課題を解き、課題を提出するという流れが一般的である[2)]。また、最終的に作成されたプログラムの評価が重視される傾向にある。このため、e-learningシステムの最大の特徴である学習履歴（学習者の実習時の入力履歴、操作履歴など）の蓄積が残らないという問題が生じる。このことは、学習者がどのような学習過程で、課題提出までに至ったのかを講師は知ることが出来ず、実習を遂行する上で的確な支援を行えないことを意味している。

2009年９月17日受理　*社会学部現代社会学科講師

そこで本研究の目的は、上記のような問題を改善させるために、プログラミング時における学習者の学習履歴を蓄積し、それを解析することで学習者に対して支援すべきタイミングはどのような場面で起こるのかを見つけ出すことである。

ここでは、プログラミング時における学習履歴を学習コンテキストと呼ぶことにする。本研究で獲得される学習コンテキストは、ソースコードの入力情報、端末の出力情報、さらに学習者の姿勢情報である。

Ⅱ．関連研究

これまでにプログラミング学習を支援する研究が多くなされてきた。例えば、プログラミング学習をe-learningで利用できるシステムである[3)]。これは、授業スライドと問題解答を利用した学習であり、学習者の学習履歴は、スライドのページや節番号などが記録される。また、記述式問題を学習者に解答させることにより、その解答を形態素解析し、解答の評価に利用しているものである。

また､学習者の習得度に応じて､自動的に問題を提示するシステムが提案されている[4)]。これは、各問題に設定された難易度と、受講者の答案プログラムを正誤判定し、評価して得られる値をもとに、受講者の習得度を求めている。

さらに、過去の他の学習者の演習履歴に基づき各演習課題に設定された達成度をもとに、学習者の理解状況と学習意欲に応じた演習課題を出題する手法が提案されている[5)]。

いずれの手法においても、プログラムの課題に達成度、難易度を設定し、学習者の理解状況に応じて、自動的に問題を出題するものであり、プログラミング中における学習者の支援を目的としたものではない。

本研究の目的は、プログラミング中における学習履歴情報を獲得し、支援すべきタイミングを抽出することであり、これらの従来手法の目的とは異なる。

Ⅲ．学習コンテキストの獲得とモデル化

1．プログラミング入力情報の獲得

プログラミング時において、学習者がキーボードを利用して入力した情報を逐次記録する。しかしながら、プログラミング以外のキーボード情報を記録すると、それがノイズとなり、後の学習コンテキストを解析する上で影響を与える。これを改善させるために、本研究では、学習コンテキスト取得エディタを開発し､エディタ内で入力された文字情報を時間情報とともに記録する。更に、プログラミングの途中経過も視覚的に把握できるように、1行単位でプログラミングコードを記録していく。

```
CharTime:1197446748813,16,33,2
CharTime:1197446749011,91,33,2
CharTime:1197446791160,10,34,2
LineTime:1197446791160
#include<studio.h>
int main(void)
{
------------
CharTime:1197446792019,32,35,3
CharTime:1197446792483,32,36,3
FileWriteTime:1197446942595,ファイル名
LineTime:1197446942595
```

図１　コード入力情報の獲得例

図１に本研究で獲得した入力情報の例を示す。図１における記録形式は以下の通りである。CharTimeは文字入力であることを示し、CharTime:時間情報、文字コード、列番号、行番号を意味している。LineTimeは、改行時における時間情報を記録している。さらに改行後には、それまでに入力されたコードを記録している。

また、FileWriteTimeは、ファイルが書き込まれた情報であり、FileWriteTime:時間情報、書き込みファイル名を意味している。

本研究では、獲得された入力情報から以下のような特徴を抽出する。

特徴１　時間t_nにおける入力文字総数I_{t_n}

アルファベット、記号が入力されれば、プラスとしてカウントとし、Backspaceキー、Delキーは、マイナスとして、式（1）のようにカウントする。

$$I_{t_n} = \sum_{t=0}^{t_n} K_t \tag{1}$$

ただし、K_tは、t時間において、入力された文字・記号によって、式（2）に示す値をとるものとする。

$$K_t = \begin{cases} +1 & (\text{アルファベット、記号キー}) \\ -1 & (\text{BackSapce キー、Del キー}) \end{cases} \tag{2}$$

特徴２　時間t_nにおける入力時間間隔T_{t_n}

t_n時間において、文字が入力されたとすると、入力時間間隔T_{t_n}は式（3）のようにして求める

$$T_{t_n} = t_n - t_{n-1} \tag{3}$$

特徴 1 では、プログラミングする際の入力ミスや試行錯誤を視覚的に見ることが出来る。すなわち、外面的な特徴である。また、**特徴 2** では、次の文字が入力されるまでの時間が長ければ、学習者がどのようなコードを入力すべきかを思考していると考えられ、内面的な特徴を表現していると言える。

2．端末出力情報の獲得

C言語などのコンパイラ言語を用いた実習の場合、コードを入力した後、必ずコンパイルを行う必要がある。コンパイル時に出力されるエラー情報は学習者にとっても、支援する側にとっても有益な情報である。このため、端末に表示されたエラー情報、および実行した結果を出力した情報を時間とともに記録する。

図 2 に本研究で獲得した端末出力情報の例を示す。図 2 の例では、端末上で入力した文字情報とともに時間情報が記録され、その結果、出力されたメッセージも時間情報とともに記録している。

```
g 1199865077
c 1199865077
c 1199865080
  1199865081
j 1199865081
o 1199865082
h 1199865082
o 1199865085
4 1199865086
- 1199865086
1 1199865087
. 1199865087
c 1199865088
  1199865088
joho4-1.c:1:19: 1199865088
studio.h: そのようなファイルやディレクトリはありません
```

図 2　端末出力情報の獲得例

上記の例では、学習者が端末上からgcc joho4-1.cと入力しコンパイルをしたところ、出力エラーが表示され、「ヘッダーファイルstudio.hというファイルやディレクトリがない」ということが分かる。正しくは、C言語の標準ヘッダーファイルはstdio.hであるため、学習者が入力ミスをし

たことがこの端末出力情報から読み取ることができる。

本研究では、獲得された端末出力情報を、後述する学習コンテキストのモデルから得られた評価値の高いシーンを検証するために利用する。

3．姿勢情報の獲得

本研究で姿勢情報を獲得する目的は、姿勢情報がプログラミング時になんらかの影響を与えるかどうかを調べることにあるため、詳細な姿勢情報を必要としない。このため、学習者の姿勢情報は、座席に圧力分布センサーを設置し、座面にかかる圧力分布から学習者の姿勢変化を獲得することにする。図３に本研究で用いた圧力分布センサーから取得された圧力分布を可視化したものを示す。図３（左）は、圧力分布を画像で表したものであり、赤い色部分は、周りと比べて圧力がかかっている領域である。また、図３（右）は、圧力分布を３次元表示したものである。

本研究では圧力分布センサーから得られた圧力分布情報をもとに、t_n時間における圧力分布の変化量D_{t_n}を式（4）から求める。

$$D_{t_n} = \sum_{i=0}^{m}\sum_{j=0}^{n}\left|P_{t_n}(i,j) - P_{t_{n-1}}(i,j)\right| \qquad (4)$$

ただし、m、nは圧力分布センサーのセンサー数を示し、$P_{t_n}(i、j)$は、t_n時間におけるi、j番目の圧力分布値を示している。

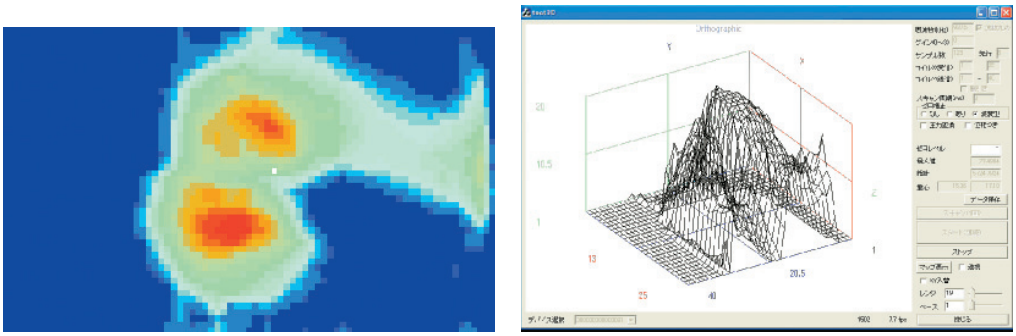

図３　座面にかかる圧力分布の例

4．学習コンテキストのモデル化

本研究の目的は、プログラミング演習において、ある時間区間αにおいて、支援すべきタイミングを抽出することである。それには、獲得された学習コンテキストを基に、評価モデルを構築する必要がある。前節で得られた特徴をもとに、以下のような評価モデルを構築する。

まず、特徴１の入力文字総数は、t_n時間の前後の時間区間$\alpha/2$における分散$\sigma_{I_{t_n}}(\alpha)$を式（1）から求める。分散$\sigma_{I_{t_n}}(\alpha)$は、入力されていない状態、もしくは削除キーとアルファベット・記号キーが繰り返し入力されると、時間区間αにおいて小さくなる。一方で、$\sigma_{I_{t_n}}(\alpha)$は、アルファベット・記号キーが連続して入力されると、時間区間αにおいて大きくなる。ただし、文字が連続で削除される場合にも、$\sigma_{I_{t_n}}(\alpha)$は高くなるので、式（5）ように定義する。

$$\sigma_{I_{t_n}}(\alpha)=\begin{cases}+\ \sigma_{I_{t_n}}(\alpha) & \text{（文字数が正方向に増加）}\\ -\ \sigma_{I_{t_n}}(\alpha) & \text{（文字数が負方向に増加）}\end{cases} \tag{5}$$

また、特徴２の入力時間間隔は、t_n時間の前後の時間区間 α/2における平均入力時間間隔$\overline{T}_{t_n}(\alpha)$を式（3）から求める。同様に、姿勢情報に関しても、ある時間t_nの前後の時間区間 α/2における平均変化量$\overline{D}_{t_n}(\alpha)$を式（4）から求める。

上記の特徴を利用して、ある時間t_nにおける評価値を式（6）のように定義する。

$$E_{t_n}=\overline{T}_{t_n}(\alpha)+\overline{D}_{t_n}(\alpha)-\sigma_{I_n}(\alpha) \tag{6}$$

ただし、$0\le\overline{T}_{t_n}(\alpha)\le 1 \quad 0\le\overline{D}_{t_n}(\alpha)\le 1 \quad \left|\sigma_{I_n}(\alpha)\right|<1$ であり、それぞれ正規化された値である。以上のことから、評価値E_{t_n}が高い時間は、次のような特徴を持つシーンである。

- 文字が入力されていない、削除を繰り返している、もしくは削除、入力を繰り返している。
- 入力時間間隔が長い。
- 姿勢が変化している。

本実験では、このような特徴をもつシーンを支援すべきタイミングであると仮定し、実際に獲得されたコンテキストを基に、評価値を求め検証を試みる。

Ⅳ．実験と結果

１．実験環境

本研究で実験した環境を図４に示す。ソースコードの入力情報の獲得には、Java1.5を用いて、図５に示すような専用エディタを開発した。また、端末の出力情報を獲得するために、BSDライセンスのフリーソフトウェアであるttyrec-1.0.8をベースに改良を加えた。

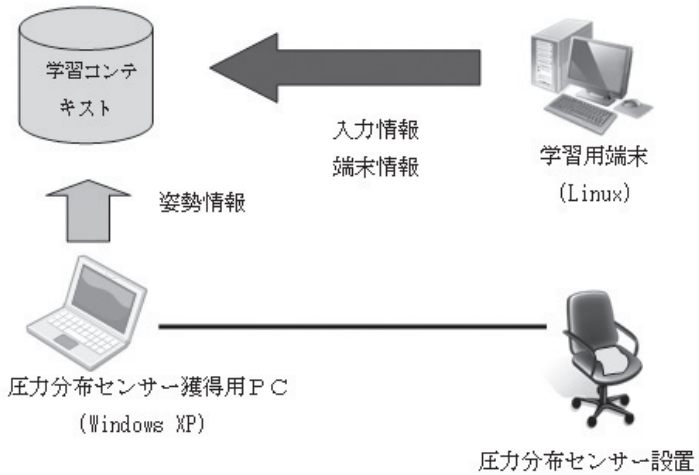

図４　実験環境

さらに、姿勢情報を獲得するために、図６のような圧力分布センサーを使用した。このセンサーは、電磁誘導タイプであり、横方向にm=40個、縦方向にn=25個のセンサーが取り付けられている。このセンサーを学習者の座席に取り付け、姿勢情報を獲得した。

プログラミング演習は、Linux端末を利用して行った。演習内容はC言語の初級である。また、C言語のコンパイラはgccを利用した。本実験では、2008年１月９日と2008年１月16日の２日間にわたって、学習者２名のコンテキストを獲得した。

図５　コンテキスト獲得用エディタ

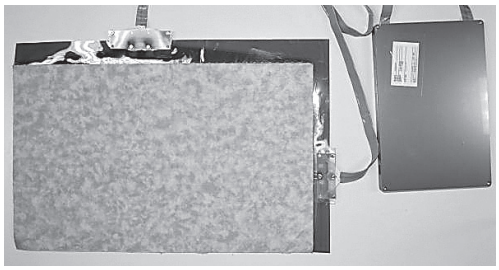

図６　本実験で用いた圧力分布センサー

２．実験結果

学習者Aにおいて、学習コンテキストを獲得し、式（6）に基づき評価値Eを求めると図７のようになった。このときのプログラミング作成課題は、１から10まで順番に数字を表示するものであった。

図７において最も評価値の高かった時間帯（図中の円部分）周辺において、獲得した端末情報の時間と照らし合わせてみると、コンパイル時において、「７行目: error: 文法エラー が "i" の前にあります。」というメッセージが表示されていた。このときの入力情報を確認すると、「printf("\%d ¥n"、i)」となっており、「;」が抜けていたことがわかった。

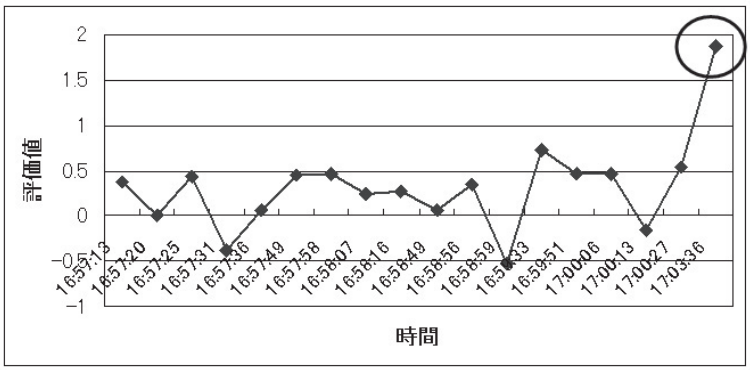

図７　学習者Aにおける学習コンテキストの評価値

また、このときの文字総数の分散、入力間隔の平均、姿勢の平均変化量の割合を調べたところ、図８のようになった。この図からは、プログラムのエラー出力に対して、姿勢の平均変化量、および入力間隔の平均がともに高かったことがわかる。

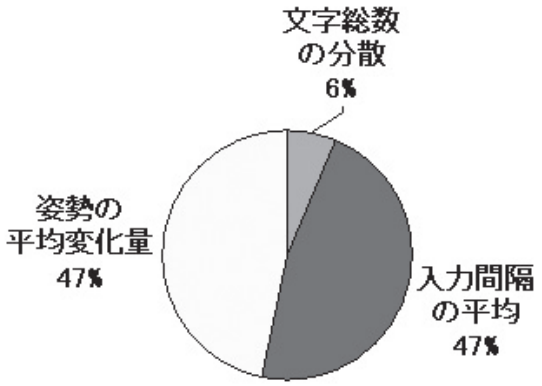

図８　文字総数の分散、入力間隔の平均、姿勢の平均変化量の割合

さらに、学習者Aにおいて、学習コンテキストを獲得し、式（6）に基づき評価値Eを求めると図９のようになった。このときのプログラミング作成課題は、５回ループさせ、メッセージを表示されるものであった。

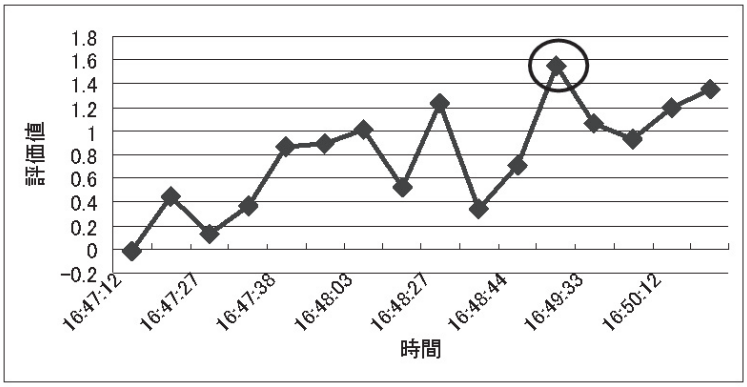

図９　学習者Aにおける学習コンテキストの評価値

このときに、評価値が高い値を示していた時間帯（図中の円部分）では、コンパイル時のエラー出力はされていなかった。また、評価値の高いシーンにおいて、文字総数の分散、入力間隔の平均、姿勢の平均変化量の割合を図10に示す。図10では、図８に比べて、文字総数の分散が増えており、入力間隔の平均時間が減少している。姿勢の平均変化量には違いがないことがわかる。すなわち、この時間帯では、プログラムを作成するにあたり、入力と削除を繰り返し行っており、学習者は課題を解くために、思考錯誤をしていた可能性がある。

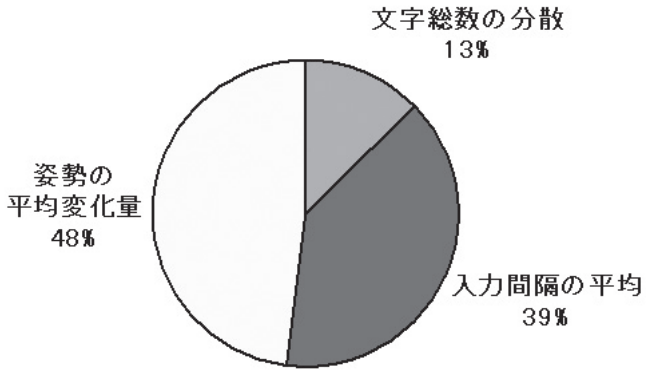

図10　文字総数の分散、入力間隔の平均、姿勢の平均変化量の割合

次に、同様に、学習者Bにおいて、学習コンテキストを獲得し、式（6）に基づき評価値Eを求めると図11のようになった。このときのプログラミング作成課題は、素数を求めるものであった。

図11において最も評価値の高かった時間帯（図中の円部分）周辺において、獲得した端末情報の時間と照らし合わせてみると、プログラミングをコンパイルしており、「error: invalid value in assignment.At top level:error: 文法エラー が "return" の前にあります」のようなメッセージが表示されていた。これは、「代入の左辺が不適切」という意味である。このときの入力情報を確認すると、「if(a%i=0) break;」と記述しており、評価式が間違っていることがわかる。さらに、このとき、学習者は、この意味がわからなかったため、実習の講師にアドバイスをもらっていた時間でもあった。

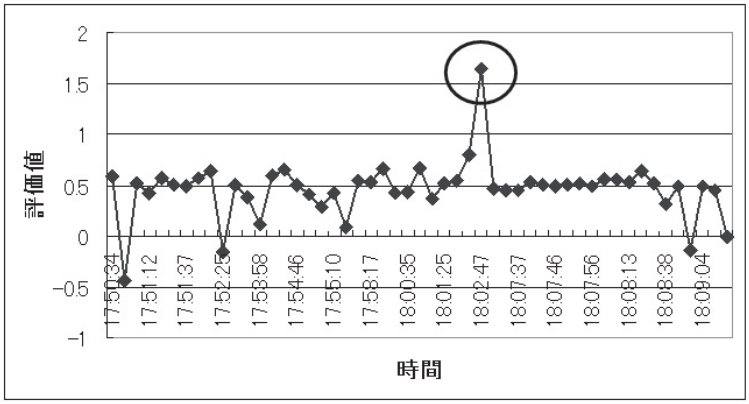

図11　学習者Bにおける学習コンテキストの評価値

また、このときの文字総数の分散、入力間隔の平均、姿勢の平均変化量の割合を調べたところ、図12のようになった。図12からは、この時間帯は講師からアドバイスを聞いている状態であったため、文字総数の分散が低く入力間隔の平均が高いことから、文字をほとんど入力しておらず、姿勢変化が増えていたことがわかる。

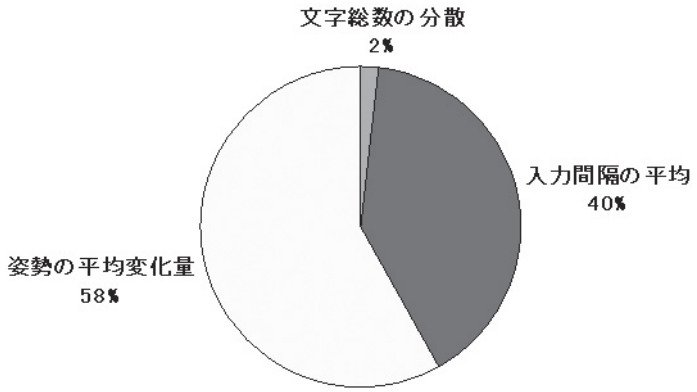

図12　文字総数の分散、入力間隔の平均、姿勢の平均変化量の割合

さらに、学習者Bにおいて、学習コンテキストを獲得し、式（6）に基づき評価値Eを求めると図13のようになった。このときのプログラミング作成課題は、九九の計算を行うものであった。図中の最も評価値の高い時間帯周辺は、獲得した端末情報の時間と照らし合わせてみると、学習者がコンパイルと実行を繰り返していたシーンであった。

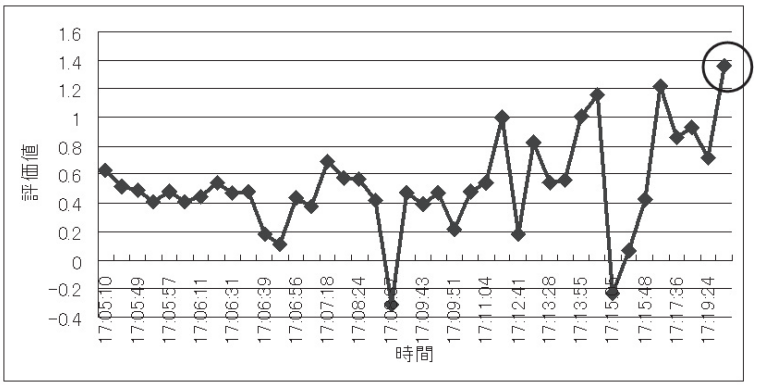

図13　学習者Bにおける学習コンテキストの評価値

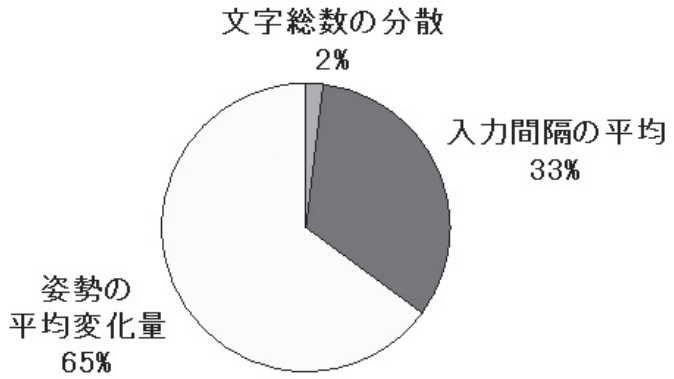

図14　文字総数の分散、入力間隔の平均、姿勢の平均変化量の割合

また、このときの文字総数の分散、入力間隔の平均、姿勢の平均変化量の割合を調べたところ、図14のようになった。図14からは、文字総数の分散が極端に小さく、姿勢の平均変化量が多いことから、コンパイルの出力結果をもとに、プログラムを修正していたため、文字の総数の分散が少なく、また姿勢変化が多くなったのは、学習者が画面に出力されるコンパイル結果の影響を受け、体がモニタ方向に近づけたり、遠ざけたりしたことが要因であると考えられる。

上記の結果を踏まえて、学習者の学習状態を考察してみると、学習者Aは、姿勢の平均変化量より入力間隔の平均の割合が多い傾向がある。また、学習者Bは、入力間隔の平均より、姿勢の平均変化量の割合が多い傾向があることがわかる。

Ⅴ．むすび

本研究では、プログラミング教育において、入力情報、端末出力情報、姿勢情報の学習コンテキストを獲得し、これらの情報を用いてモデル化を行い、支援すべきシーンの抽出を試みた。実験結果として、モデルの評価値の高い時間帯の中には、ソースコードに誤りが含まれていたことが確認できた。しかしながら、評価値の高いシーン全てが、このように明確にわかるものではなかった。

今後の課題としては、上記のようなシーンを詳細に分析する必要がある。それには、プログラムの構造解析もあわせて行う必要がある。関連研究に正解プログラムと解答プログラムから構造パターンを抽出し、構造誤りを検出するものがある[6)]。このような研究も踏まえて、学習コンテキストと構造パターンの関係性を見つけることが挙げられる。

参考文献

1）林敏浩、水野貴規、富永浩之、垂水浩幸、山崎敏範、“学習者が教材を投稿・共有できるe-Learningシステムの開発”、情報処理学会研究報告、CE-94、pp.9-16、2008

2）中尾茂子、安達一寿、北原俊一、新行内康慈、“ブレンディッドラーニングによるプログラミング学習の実践と評価”、日本教育情報学会教育情報研究、Vol.22、No.3、pp.47-56、2006

3）宮地功、林青松、姚華平、吉田幸二、“C言語学習を支援するe-ラーニング”、電子情報通信学会技術研究報告、ET2005-6、pp.33-38、2005

4）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－受講者の習得度に応じた問題自動提示メカニズム－”、電子情報通信学会論文誌D-I、Vol.J88-D-I、No.2、pp.439-450、2005

5）田口浩、糸賀裕弥、毛利公一、山本哲男、島川博光、“個々の学習者の理解状況と学習意欲に合わせたプログラミング教育支援”、情報処理学会論文誌、Vol.48、No.2、pp.958-968、2007

6）宮地恵佑、高橋直久、“構造誤り検出機能を有するアセンブラプログラミング演習支援システムの実現と評価”、電子情報通信学会論文誌D、Vol.J91-D、No.2、pp.280-292、2008

Summary

In a recent, many of learning environments in institutions of higher education have introduced Course Management System（CMS）or Learning Management System（LMS）. However, functions of these systems are insufficient for programming or numeric simulation exercise. Because, when these systems are used for these exercise, learning history（learning contexts）are not accumulated. Therefore, the objective of this paper is to accumulate learning contexts for C-language programming exercise, and is to support learners by analyzing these contexts. In order to accumulate these contexts and support learners, this study acquires learning context from learners. The learning contexts are information of keyboard input, terminal output and learner's posture. To evaluate these contexts, we construct a context model from acquired context. When we applied this model to learning context, we extracted these scenes that showed high value of evaluetion, from these context. As a result, we knew that these scenes were caused by syntax error of C-language. In a word, we could find out scenes with the possibility which should support learners.