Bulletin of Aichi Univ. of Education, 62（Educational Sciences）, pp. 191 − 197, March, 2013

# プログラマ視点からの教育ソフトウェア開発について

松永 豊

情報教育講座

# About Development of the Educational Software from a Programmer's Viewpoint

Yutaka MATSUNAGA

*Department of Information Sciences, Aichi University of Education, Kariya 448-8542, Japan*

## 1. はじめに

プログラミングを行ったことがある人なら容易に想像はつくかもしれないが、設計の段階で運用時のトラブルを予測することは極めて難しい。トラブルの元になる動作の予測ともなれば、難易度はさらに高くなるかもしれない。

周知の通りコンピュータはただの箱であり、プログラムによって全ての動作がコントロールされている。埋め込まれたプログラム（アルゴリズム）が動作の理由である以上、「書いていない未来」は基本的には起こらない。簡単な例えで言えば、ゲームにおける隠れキャラは、当然、プログラマが準備しているからこそ発生する。偶然（バグなどで）発生するなどということはほとんど皆無である。

すなわち、プログラムとは未来を書き記したもの、もっと丁寧に言うなら未来の可能性を全て書き記したものであり、プログラミングとは基本的には未来を予測する作業であるといえる。ソフトを操作する人間がどのような振る舞いをするかをあらかじめ予測し、先回りしてレールを敷かなければならない。

さて、現在、e-Learningなどが盛んに叫ばれており、教育現場でコンピュータを使う機会は格段に増加している。e-Learningにせよ、教育教材にせよ、開発者サイドは当然のことながら使用者（先生や生徒など）の行動を予測してソフトを開発する必要がある。また、異なる権限を持つ複数のユーザ管理をする必要があるかもしれない。（最低限、先生と生徒の2つの権限が必要となることは極めて多い）

このような問題は教育系ソフトに限らず、スタンドアロン系のソフト以外では普通に発生する問題ではあるが、意識しながら作成しないと思わぬところで足元を掬われる。また、教育系ソフトならではの問題が存在しないわけではない。

これらを踏まえ、本論文では教育におけるソフトウェア開発においてはどのようなことを注意すべきか考察する。また、実践例をいくつか紹介し、効果を検証する。

## 2. 教育系ソフト

ここで、教育系ソフトの特徴について考えてみる。ただし、あえて「教育系」と付ける必要もないような一般的なソフトに当てはまる項目も含まれることを断っておく。また、コンピュータの操作そのものを学習対象とした場合（たとえばタイピングソフトなど）は特有の項目が入るので、ここでは触れずに別の場所で述べることにする。

- 使用目的は教育のためである
- 優れた教育系ソフトとはコンテンツが優れている、指導法が優れている、など教育としての質が優れているソフトを指すのが普通である。
- もともとコンピュータを使わずに（紙と鉛筆等で）指導していたものをコンピュータ上で実現したソフトである、などという場合も多い。
- コンピュータを使うことが目的ではないことが多く、コンピュータを使うことによるデメリットが含まれることすらある。ただし、それを上回るメリットが存在する場合、教育系ソフトは存在価値がある。
- 複数の人間が一つのシステムを使う場合、ネットワーク（一般的にはインターネット）が利用されることが多い。
- 指導者、学習者、という関係を持つソフトの場合、少数の指導者が多数の学習者を指導することが多い。
- 指導者、学習者、という関係を持つソフトの場

合、一般的には指導者側のほうが高い操作権限を持つ。
- 反復して使用されることが多い。ただし、操作の慣れを期待するものではない。
- 一日当たりの使用時間は少なくても、長期利用されることが多い。

このように、教育系ソフトウェアの特徴は

教育的価値 ＞ プログラミング的価値

であることは疑いようがなく、それが間違っているわけではない。しかしながら(それゆえ)、教育としての本質以外の部分がおざなりにされているソフトが存在することも事実である。

本論文では、敢えて教育的本質（コンテンツや指導法といった直接的な個所）以外の部分に焦点を当て、優れたソフトとは何かについて論じたい。

## 3. インターフェイスについて

使用者の未来を予測する上で重要なファクタとしてインターフェイス（I/F）がある。インターフェイスとは人間とコンピュータのやりとりの部分であり、大雑把にいえば見た目である。直接、本論文の趣旨とは無関係に思うかもしれないが、人間の行動を決定する多くの部分はインターフェイスにあると言っても過言ではない。

インターフェイスの善し悪しを考える上で、簡単な例を考えてみる。たとえば、以下のような見た目のソフトがあったと考える。

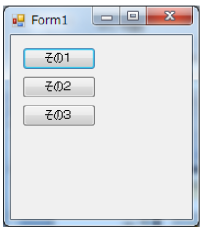


図1（a）I/F 1a

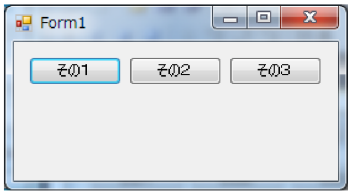


図1（b）I/F 1b

図1（a）、図1（b）のどちらの場合も「その1」「その2」「その3」の順にクリックされる可能性が高い。これは、アプリケーション上のボタン等が、一般的には上から下、左から右の順に並べられることが多いからであり、キャプションが1、2、3となっていれば昇順に選択される可能性が高いからである。

では、必ず1→2→3の順番でなければならない場合はどうだろうか。図1のままでも順にクリックしてくれることを期待してよいのだろうか。この「ねばならない」場合に「何となく上から下」や「何となく数字順」では危険すぎる。

そこで次の段階として若干説明を付けることにする。たとえば、以下のようなインターフェイスに改造したとする。

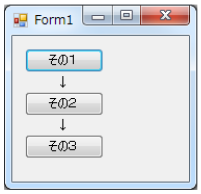


図2（a）I/F 2a

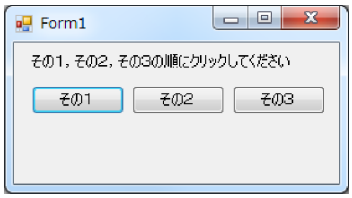


図2（b）I/F 2b

図2（a）、図2（b）は説明に匹敵する記号や文を入れただけだが、図1と比べればプログラマの意図する順番にクリックしてもらえる確率は上がる。しかしながらこの場合も突然「その2」をクリックしてしまうユーザがいたとして全てユーザの責任と断ずることができるだろうか。つい、間違って押してしまうということはないのだろうか。

最後の段階としてユーザの権限を奪ってしまう方法を考える。たとえば、以下のようなインターフェイスに改造したとする。

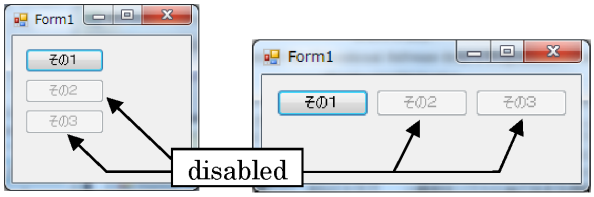


図3（a）I/F 3a　　図3（b）I/F 3b

図3（a）、図3（b）のインターフェイスでは、最初から「その1」しかクリックできない。「その2」「その3」は押せないボタンになっている。「その1」をクリックした後にしかるべきタイミングで「その2」のボタンだけをクリック可能にすれば、説明文を書くまでもなくプログラマの意図する順番通りユーザはクリックしてくれるだろう。（むしろ、その順以外に押されることは完全になくなる。）

以上の結果、図1～図3の場合、図3が最もインターフェイスとして優れていると考えることができる。単純なことではあるが、これは非常に重要なことである。

さて、ではプログラミングという観点から複雑度を考えてみよう。容易に想像できると思われるが、図1が最も簡単なプログラムであり、図2、図3の順に複雑度が増していく。すなわち、優れたインターフェイスにはコストが掛かることを意味する。

ここで話を教育ソフトのことに戻そう。先にも触れたように、教育ソフトを作る場合、最も重要となることは、コンテンツが優れているとか指導手順が優れているなど、「教育」に関係する部分であることは言うまでもない。これは、そもそもコンピュータを使うか使わないか以前の問題であり、教育的価値の重要度

がトップに来ることは疑いようがない。もともと紙で行っていたものをコンピュータ上で行わせることで、記録（保管）や集計が容易になる、程度の理由でもコンピュータ上で行わせるメリットはあるため、「とにかくコンピュータ上で指導する」ことが目的になる場合すらありうる。

このような観点（理由）から考えれば、「優れた教育ソフト」が必ずしも「優れたインターフェイス」であるわけではないことが分かる。現時点で一定以上の評価を得られた「優れた教育ソフト」はたくさんあるが、インターフェイス造りにコストが掛かる以上、多少の手抜きはやむを得ないとも考えられる。

## 4. 使用者権限について

e-Learning系システムの特徴として、一つのシステムを複数の人間が同時に使うことが多い。そのため、権限を意識した構成にする必要がある。ここでは使用者権限について考察する。

一般的に「教師／先生／指導者」と「生徒／学生／学習者／受講生／児童」の関係を考えれば容易に想像ができるように、生徒より教師のほうに権限を持たせる構成になる。この場合、生徒が可能な操作は教師も可能だが、教師しかできない操作や教師にしか見ることができない情報は当然ありうる。

そこで、「教師」「生徒」それぞれの権限について詳しく考えてみる。実際にはこの2つに加えて「管理者」権限を用意する場合が多い。まず、「管理者」権限であるが、これはすべての操作を行うことができる権利と考えればよい。次に「教師」の権限についてであるが、教師が単独の場合と複数の場合で若干権限が異なる。単独の場合は、システムが有するほぼすべての操作権限を与えても問題ない。すなわち、「管理者」権限と同レベルで問題ない。具体的には、生徒全員のID管理、生徒全員の記録の閲覧、生徒の代理操作、問題文の変更、などがあげられる。（図4参照）

教師が複数の場合は、一部教師ごとのプライベートな権限があってもよい。たとえば複数の教員の平均点で成績を付ける場合など、他の教師の評価に影響されないように敢えて途中経過を見せないほうがよい場合もありうる。また、個別指導のスケジュール管理なども本人のみが操作できるほうが良いかもしれない。さらに、担当する学生が異なる場合や、一部だけ重複する場合などがあるかも知れない。（図5参照）

最後に「生徒」権限であるが、原則として個人のデータしか操作できない。ただし、一部のデータは公開する（他の生徒も見ることができる）ようにすることもありうる。たとえば、相互評価をさせる場合などがこれに当たると考えられる。

次に「教師用ソフト（教師権限）」と「生徒用ソフト（生徒権限）」をどのように実現するか、どのようなシステムにするかについて考える。このようなシステムを構成する場合、主に2通りの方法が考えられる。

- 教師用ソフトと生徒用ソフトを完全に分ける
- ログイン時のIDにより権限が及ぶ範囲を決定させる

どちらの方法もメリット・デメリットがあるが、それぞれの特徴について検討してみよう。まず、教師用のソフトと生徒用のソフトを完全に分ける場合である。広い意味では、ウェブページにおいて生徒用URLと教師用URLが異なる場合も含めてもよいかもしれない。この場合、生徒には教師用のソフト（教師用のページ）を使わせないことになるので教師専用の（特

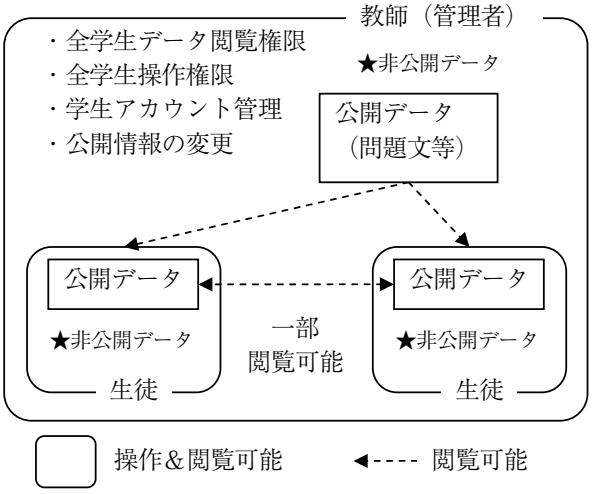


図4　教師が単数の場合

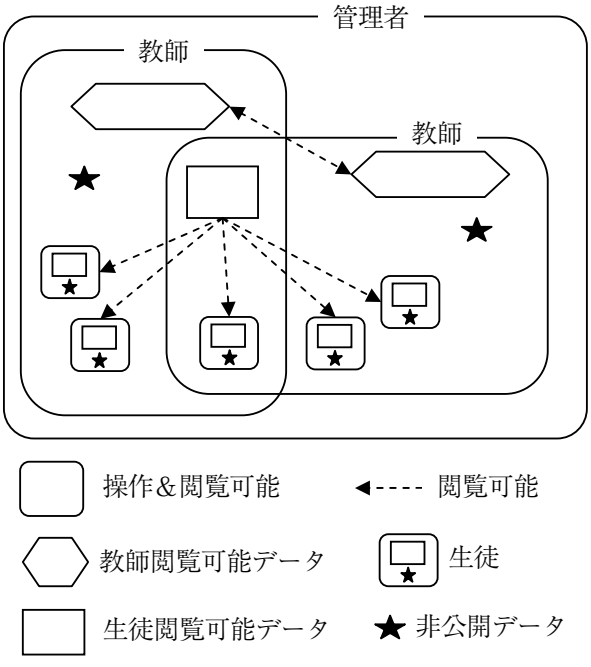


図5　教師が複数の場合

有の）機能を自由に盛り込むことができるというメリットがある。一方で、最低、2種類のソフト（あるいはURL）を管理しなければならず、生徒の前で実演する場合など生徒になりきるときはソフトを変えなければならないという煩雑さを含むことになる。また、教師用のページを生徒に公開できない場合などは教師用URLを秘密にするなどの必要が生ずる。

次にログイン時のIDにより権限が及ぶ範囲を決定させる方法について考えてみよう。これは入口が同じであることを意味する。URLでいうなら教師用と生徒用で共通のURLを用いることになる。この場合のメリットは1つのソフト（あるいはURL）で済むためわかりやすいことがまず挙げられる。しかしながら教師側ソフトをありかを公開しているという点で若干セキュリティが下がるというデメリットがある。特に相手が情報科学等にも興味がある学生の場合、システムの不備を見つけることも広い意味では教育たり得るので慎重にならざるを得ない。

## 5. 長期使用に耐えうる設計について

教育系ソフトは学習者が単独で使う場合もあるが、授業などで補助的に使うことも多い。授業が半年間や年間単位で構成されていることは珍しくないため、長期に渡っての使用に耐えうる仕掛けが必要になる場合もある。また、授業の進行具合とある程度同期させたい場合も多い。

WEBベースのシステムの場合は、サーバサイドで内容を更新するだけで済むこともあるが、生徒のパソコンにインストールさせるタイプのソフトの場合は単純ではない。また、ネットワークを介しての使用の有無でも状況は変わる。そこで、いくつかのタイプに分けて特徴を考えてみよう。

- WEBベース
- インストールタイプ（スタンドアロン系）
- インストールタイプ（ネットワーク使用）

まず、ブラウザを利用するWEBベースのシステムについてだが、この方式の場合はネットワーク（インターネット）に接続して毎回サーバ上のデータを使うことが前提になるので、先ほども述べたように必要に応じてサーバサイドで変更を加えることができる。コンテンツの提供自体をコントロールすれば授業の進行具合と同期をとることも容易い。プログラミングレベルでいえば、WEBベースのサーバプログラム言語は充実しており、当然ながら、ネットワークプログラミングを容易に行うことが可能となる。たとえばPHPなどが有名であり［1］、データベースとの相性も良いため有効な選択肢と考えることができる。ただし、原則ブラウザ依存になるため、ブラウザが抱える諸問題を常に意識しながら作成する必要が生ずる。たとえばブラウザのバックボタンの取り扱いをどうするか、Shiftを押しながらリンククリックされた場合はどうするか（すなわち、新しいウィンドウで表示された場合の対処）、数種類あるブラウザへの対応など単純ではない。

次にインストールタイプ（EXEタイプ）についてだが、ネットワーク使用の有無とは関係なく、インターフェイスの工夫は施しやすい。なぜなら、ブラウザの束縛から逃れられるので、GUI開発環境を使うなどすれば、かなり独自のインターフェイスも実現可能だからである。ブラウザ依存でなくなるということは、たとえば、「○○キーはブラウザでは△△という動作になるため使えない」などの制限からも解放される。

一方、ソフトの更新に関しては、WEBベースのプログラムに比べて圧倒的に複雑になる。スタンドアロン系のソフトの場合、ネットワークに接続しないため教師サイドで内容を更新することはできない。更新に関しては完全に諦めるしかない場合もあるが、授業進度との同期に関しては方法がないわけではない。最も単純の方法としては「キーワード（パスワード）方式」が挙げられる。たとえば、「授業前半では課題Aを授業後半では課題Bを扱いたい、なおかつ、課題Bを先に見せたくない」という場合などに、キーワードにより操作を限定するやり方である。課題Bを操作（閲覧）可能にするキーワードを適切なタイミングで発表すれば時系列的なコントロールもできる。あるいはデータファイルだけ別途配布するという方法も考えられる。

最後にインストールタイプでなおかつネットワークを利用するソフトについてだが、最も自由度の高いソフトが期待できる。ただし、ネットワークプログラミングに関しては敷居が高くなる。理由はいくつかあるが、たとえば生徒のパソコンにインストールさせるソフト（クライアントソフト）とサーバソフトが異なる動作環境（異なるOS）になる可能性が高い。それに連動して開発環境（プログラミング言語）も異なる可能性が高い。たとえば、図6のような環境は割とよくある例である。

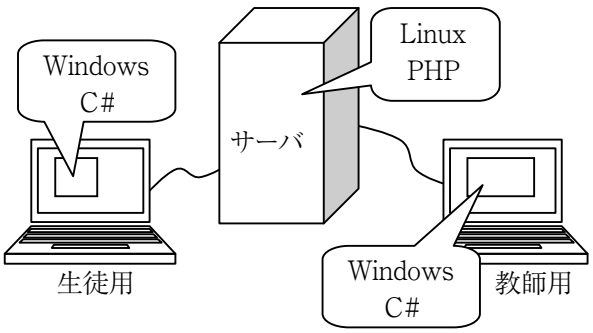


**図6　サーバ・クライアントの構成例**

- クライアント
  - ➢ OS：Windows
  - ➢ 言語：Visual C#
- サーバ
  - ➢ OS：Linux
  - ➢ 言語：PHP

また、メンテナンスを考慮した構造にする場合、さらに工夫が必要である。たとえば、授業の補助ソフトを長期に渡って使用する場合、ソフト自身の改良版(いわゆるバージョンアップ)を再配布したくなる場合がある。無論、旧バージョンをアンインストールしてもらい新バージョンを再インストールしてもらえればよいのだが、その作業自体が教育になりうることはそれほど多くなく、むしろ、パソコン操作が不慣れなものに対して単なる混乱の元であることが多い。そこで全自動的（あるいは半自動的）にバージョンが上がる仕組みを施しておくと極めて効果的である。具体的にはソフトを2段構成にすることである。起動用のプログラムをA、本体をBとすると、Aは主にネットワーク接続に対する仕組みとBを呼び出す仕組みを用意しておく。ただし、Aは通常起動が可能であるのに対し、Bはオプション無しの通常起動では限定的な動作（たとえばバージョンが表示されるだけなど)にしておく。すなわち、Bは専用の起動オプションがなければ起動しないようにしておく。Aはネットワークに接続した際にサーバ上のバージョンを取得する。また、同時にBのバージョンを確認する。バージョンが適切だった場合はBを起動オプション付きで起動し、そのまま本体であるBにプロセスを移す。もし、バージョンが古くなっている場合はBのみ最新バージョンのダウンロードを試み、成功すればプロセスを移し、失敗すればバージョンが古くなっている旨を表示する。

多少複雑な構成にはなるが、一度、この仕組みを完成させておけば基本的にはネットワークを利用する他のソフトにも利用可能なので価値は高い。

## 6. パソコン操作が目的の場合について

前述の通り、e-Learningをはじめとした様々な教育系ソフトは基本的にパソコン操作が本来の目的ではないため、なるべく本質以外の部分で学習者を混乱させるべきではない。逆にいえば、パソコン操作が得意なほうが有利だからという理由で本来の目標でないはずのパソコン操作が目的になってはいけない。例えば計算力向上のために時間を争う算数教材ソフトを作ったとして、マウス操作の上達度でランクが決まるのでマウス操作の練習ばかりする、というのは明らかに本末転倒である。

無論、最初からパソコン操作の習得を目的とした教育ソフトはこれには当てはまらない。たとえばキーボード練習ソフトは入力が速く正確にできるようになることが目的なので、反復学習により指に動作を記憶させるなどは完全に正しい行為である。

そこで、コンピュータリテラシ習得が目標のソフトを開発する場合についても議論しておく。コンピュータリテラシとはいろいろあるが、パソコンを使用する際、本来様々なトラブルに遭遇することは周知の事実である。そのため、習うより慣れろ的な教育は一定範囲正しい。これを延長して考えるとあらかじめ様々なトラブルを体験しておいたほうが良いことが分かる。以上の結果より、コンピュータ操作の習得自体が目標の場合、前述の主張とは逆に「インストール操作も教育の一環だから」「複雑でやや不親切な仕組みもパソコン操作の練習になるので」など、割り切ることも可能ということになる。しかしながら、このことを盾に取って手抜きをすることはやはり本意ではないだろう。

## 7. 実践

筆者は過去に大学の授業サポート用にいくつかのソフトを作成している[2] [3] [4」[5]。筆者の所属が情報教育講座であり、コンピュータ操作の習得自体が教育内容である授業やプログラミングの授業も多く担当している。対象が情報コースの学生の場合、将来ソフト開発を行う人材と割り切ればダメなインターフェイスを反面教師的にも紹介できるため、前述の通り、多少手抜きをしても許される、別の言い方をすれば、教育の一部であると割り切ることも可能な立場にある。しかしながら、なるべく本質以外の部分で混乱する可能性（混乱する未来）を排除したソフト開発を心がけている。過去、作成したソフトはさまざまな個所で本論文の提案が生かされているため、その紹介と効果について説明する。

まず、シミュレーションの授業で用いたソフトである［2]。学生所有のノートパソコンを用いた授業ではあるが、受講生は大半が情報コース以外の学生であり、初回のアンケートでも「パソコン操作が苦手」と多数が回答した授業である。授業で用いる学生所有のパソコンはWindowsマシンなので、当然、作成したのはWindowsソフトである。大半はVisual C#で開発した。コンピュータシミュレーションは場合によっては計算能力が極めて要求されるが、非力なノートパソコン上で扱えるシミュレーションということでテーマとして人工生命を取り入れている［6］[7]。学生にとっては聞きなれないテーマであるとは思われるが、コンピュータシミュレーションの本質を体験させるのに手頃であったので積極的に導入した。そのような理由もあり、なるべく本質部分のみが演習できるよう意識し

た。また、パソコン操作に不慣れな学生が多く混じっていたので、よりインターフェイスに拘れるEXEタイプのソフトとした。各ソフトは基本的にはその時間限りの演習補助用なので、ネットワーク接続不要のスタンドアロンタイプとした。(一部、HSP[8]を使ったソフトも配布したが、これはプログラミング教育の一環と言えるのでここでは割愛する。なお、HSPとは初心者向けプログラム開発言語であり、情報コース以外の学生にプログラミングを体験させる理由で利用した。)ソフト自体は授業用ページ経由で毎時間の演習開始時ダウンロードさせたが、一度ダウンロードさせるとその後はネットワーク経由ではコントロールできない。そこで、キーワード方式（パスワード方式）の仕掛けを組み込み、授業中の適切なタイミングでキーワードを発表する方式を採用した。授業後のアンケート調査ではパソコン操作が苦手な学生も特に混乱がなかったと多く回答しており、概ね成功したと思われる。

次に、タイピング試験用ソフトについて説明する[5]。これは純粋な教育系ソフトというよりは完全にコンピュータ操作習得用であるが、本論文の趣旨に合致した構成になっているので紹介する。コンピュータリテラシの重要な要素の一つにタッチタイピングがある。最近は、タブレット等、直感的なインターフェイスを売りとするデバイスが多く出回り始めているが、大量の文字や数値を直接操作する場合、現在のところキーボードが最も優れたデバイスであることは疑いようがない。たとえば、WORDやEXCELを最も効果的に使おうと考えた場合はキーボードの使用は不可欠である。キーボード不要の環境が整うのはもうしばらく先であろう。そこで、タッチタイピングの練習成果を確認するソフトを開発した。これは授業でタイピング練習用として用いられているCrew Typing[9]をベースにネットワーク経由で結果等を自動的に集計するシステムである。初期のバージョンでは様々な情報を設定用ファイル（INIファイルと考えればわかりやすい）に入れていたが、教師がコントロールしやすい反面、学生側にとってもハッキングしやすい構成だったので全面的に改造した。(実際にトラブルがあったわけではないが、学生から「いたずら」できる可能性を指摘されたため。)そこで、バージョンアップに耐えうる構成を一部導入した。具体的にはサーバ接続時に自身のバージョンとサーバ利用可能なバージョンを比較し、利用不能なほどバージョンが古くなった場合は再ダウンロードを促すものである（半自動化)。ただし、1. このソフトを使用するのは試験時のみである、2. 試験時のみにしかサーバへの接続許可を出す必要がなかった、3. 練習自体はCrew Typingを用いて可能、などの条件があったため、古くなった旨のメッセージを表示するだけで自動更新機能までは組み込んでいない。

次にプログラミング演習の支援ソフトについて説明する［3］［4]。これは完全にWEBベースのソフトであり、ブラウザ上で使用する前提のソフトである。PukiWiki［10］をベースに作成されており、

- 授業内容のアナウンス
- 自動出欠記録
- レポート提出システム
- レポートコメント機能
- 提出レポート保管庫
- 面接予約システム
- 電子カルテ
- プログラミング授業間の連動
- 教員間スケジュールの連動

など、扱える範囲は多く、すでに長期利用中である。学生、教員ともシステムのURLは共通で、ログイン時のIDにより権限を変える方式を採用した。WEBベースであるので当然ネットワークとの相性は抜群で、使用のたびにサーバ上で細かくバージョンがアップしたことも理由として大きい。ただし、一部、不満も出てきた。具体的にはレポート提出の仕組みやレポートコメント機能がインターフェイスに起因する理由で使いにくく感じることである。そこで、現在、複雑な操作をさせる部分を中心にブラウザ非依存の仕組みを構築しつつある。まだ、学生指導では使ってはいないが、必要に応じて更新を繰り返して使っていきたい。

## 8. まとめ

以上、本研究では、プログラマ視点からの教育ソフトウェア開発について述べた。教育系ソフトの本質的な部分は敢えて触れず、その脇役となる仕組みについて説明した。教育ソフトの本質から離れる脇役の重要性を述べた後、効果的に利用する方法について提案した。具体的には、インターフェイスの工夫について述べたあと、それを生かしたネットワーク利用の方法について提案した。また、過去に筆者が作成したソフトと提案手法の利用方法についても紹介した。

今後も教育の現場でコンピュータを利用する機会はますます増えると考えられるが、なるべく生徒(学生)が学習に集中できるよう細かいところにも気を使ったソフト開発を心がけたい。筆者同様、教育ソフトの開発を手掛けている人や、学生等に教育ソフト開発を指導している先生は多いと思われるが、参考になれば幸いである。

## 参考文献

［1］PHP: http://www.php.gr.jp
［2］松永豊：『自作教材ソフトを用いたシミュレーション演習授業』、愛教大学研究報告61輯（教育科学編）2011
［3］松永豊:『プログラミング演習授業支援システムの開発』、愛教大学研究報告59輯（教育科学編）2009
［4］プログラミング演習授業のための面接予約システムの開発、松永豊、愛教大研究報告58輯（教育科学編）、2008
［5］松永豊：『試験用タイピングソフトの開発と実践』、愛教大学研究報告56輯（教育科学編）2006
［6］有田隆也：『人工生命』、医学出版、2002
［7］星野力：『人工生命の夢と悩み－コンピュータ中の知能と行動の進化』、裳華房、1994
［8］http://hsp.tv
［9］http://www.crew.sfc.keio.ac.jp/projects/2000keyboarding/index.html
［10］PukiWiki: http://pukiwiki.sourceforge.jp

（2012年9月18日受理）