Web技術による出欠管理とファカルティディベロップメントのためのシステム構築について*1

吉澤 康介*2
Kousuke YOSHIZAWA
関東学園大学 経済学部
群馬県太田市藤阿久町 200
z-yoshi@ue.kanto-gakuen.ac.jp

三宅 修平*3
Shuhei MIYAKE
東京情報大学 総合情報学部
千葉市若葉区谷当町 1200-2
miyake@rsch.tuis.ac.jp

安斉 公士*4
Koushi ANZAI
関東学園大学 経済学部
群馬県太田市藤阿久町 200
z-anzaik@ue.kanto-gakuen.ac.jp

学生の出席状況を教員側が十分に把握することは授業を運営して行く上で極めて重要である。本論では、筆者らが開発した、一般教室において一枚一枚異なる使い捨てのパスワードを印刷した出席カードを配り、授業後に感想と共にそのパスワードを入力することで出席を取るシステムを紹介する。開発の焦点は、どこにでもある普通の環境を想定する一方、一般教員に使ってもらいやすいインターフェースを取り入れた点にある。また、限られた労力を有効に活用するために、XML による仕様記述とプログラムの自動生成を取り入れた開発手法を採用している。

1. はじめに

様々な授業において、学生の出席状況を教員側が十分に把握することは授業を運営して行く上で極めて重要である。出席状況を把握する方法としては、一般に教員が学生一人一人に出席カードを配り、それに学生の氏名、学生番号を記入させ、それを毎回集め、教員が整理するという方法が多用されていると思われる。しかし出欠カードの整理は教員にとって、たいへん手間のかかる作業となる。

また、当該授業において学生が授業内容をどの程度理解し、その授業をどう評価しているのか、という情報を十分把握することは、教員が授業を改善して行くために必要不可欠である。これらの情報を集める方法としては、先に述べた出席表の裏面に、個々の学生の感想等を記入させる方法が考えられるが、これも学生に対して記入を義務化しないと書かない場合が多いし、データの整理にも大変な時間を要することになる。

このようなことから、筆者らはすでに情報リテラシ教育等で、授業中に学生全員にネットワークに接続された端末が与えられている環境における出欠管理とファカルティディベロップメントのためのシステムを構築し、その有効性を示している。[1]

本論では、授業中にネットワークに接続されていない環境における学生の出席状況の確認と授業改善のためのデータを収集する、というシステムを構築し、その有効性について報告する。

2. 基本的なアイデアと実現への問題点

基本的なアイデアとしては、授業ごとかつ各学生ごとに、一枚一枚異なる使い捨てのパスワードを印刷した出席カードを配る。パスワードには個々の授業を識別する情報が含まれており、このパスワードと授業の感想やコメント等を授業後にWebページから入力させ出席とするものである。パスワードを入力する際に感想等も入力させることから、学生の理解度、要望を掌握する手掛かりとすることができるようになっている。

アイデアとしては単純であるが、様々な制約の中で現実的に運用可能なシステムを構築するためには、解決すべき問題が多い。

利用可能な環境の制約

もし特別な設備を導入できるのであれば、きわめて利便性の高いシステムを構築できる。例えば、学生証の非接触型 IC カード化と各教室入り口への読

*1 An Attendance Management System Using Web Technology For Faculty Development.
*2 Faculty of Economics, Kanto-Gakuen Univ.
*3 Faculty of Informatics, Tokyo Joho Univ.
*4 Faculty of Economics, Kanto-Gakuen Univ.

み取りゲートの設置、あるいは学生証へのバーコード印刷と各教室入り口へのPOS端末的リーダーの設置などが考えられる。

しかし、こういったシステムの設置には莫大な金銭的投資が必要である一方、システムがはたして期待した教育効果を発揮できるか否かという点で大きなリスクがある。

さらに適用先が、そのような設備を導入できる場所に限定されてしまうという問題がある。

また、特定の設備を前提としたシステムには、その設備が足枷となって、システムの改善や新しい試みができなくなるという「設備の陳腐化」の危険性がつきまとう。

筆者らとしては、むしろ**「どこにでもある普通の環境」**で稼動するシステムを開発することを重視したい。

「どこにでもある普通の環境」とは、具体的には、

- Web閲覧可能な学生用クライアントPC(いわゆる「パソコン実習室」のPCまたは学生の個人所有のPCのどちらでも可とする)
- 教員が管理するPCサーバー(当該組織のネットワーク運用ポリシーの制約で、教員が学内にサーバーを設置できない場合は、学外にサーバーを設置する)

を想定する。

あえて「どこにでもある普通の環境」を制約とすることには、次のような利点がある。

- 低コスト。特別な投資は不要。個々の教員レベルでの試行実施が可能である。
- 事実上どこでも利用できる。
- 仮に将来的に何らかの設備投資をするとしても、その時に備えてシステム的なノウハウ、運用上のノウハウを蓄積することができる。

このような点を考慮して、筆者らは「どこにでもある普通の環境」を前提としてシステムを設計することとした。

教員の協力

このシステムは、広く一般の教員(場合によっては職員を含む。以下同様)に使ってもらうことが前提である。

したがって、教員に特別なスキルを要求したり、利用手順が煩雑であってはならない。

現時点において一般に期待できる教員のミニマムのスキル・レベルとしては、

- ブラウザによるWeb閲覧
- メールの送受
- ワードプロセッサによる文書作成

程度であると考えられるから、この範囲で操作可能なシステムを設計する必要がある。

また次のような手間を最小にしなければならない。

- 授業の基本データ入力の手間
- 出席カード印刷と切り離しの手間
- 出席の集計の手間
- 感想の整理の手間
- 出席や感想の閲覧に要する操作の手間

実際にやってみると大した労力を要さない作業でも、「煩雑である」という印象を一度でも与えてしまうと、それだけでシステムの利用に消極的な対応を示す教員も存在すると考えられる。逆に、「楽そうである」、「その割りに役に立ちそうである」という印象を与えることに成功すると、協力してくれる可能性が高い。

大学という組織においては、一部の教員の試みを、強制力を持って他の教員に試行させることは馴染まないので、この一見きわめて曖昧かつ情緒的な要素が、実はシステムの成否の鍵を握っている。

ここで改めて指摘しておきたいのが、**不特定多数のユーザーが利用するシステムを設計する場合、技術的な要素と同様に心理的な要素が大きい**、という点である。一般ユーザーから見たときのシステム利用に対する心理的な敷居をできるだけ低くするように留意しなければならない。

様々なニーズへの対応

筆者ら内部での議論、あるいは他の教職員との議論を通じて、次のような様々なニーズが出てきた。

- 【教室の設備】通常の机と黒板の講義科目、デスクトップPCを前にした実習授業、情報コンセント(or 無線LAN)と持ち込みNotePC

のある授業等へそれぞれ対応したい。

- 【多様な時間割】通常の週1コマ授業、集中講義、2コマ連続講義等に対応したい。
- 【感想やアンケート】出席だけでなく感想を書かせたい、感想に最小/最大字数制限を設けたい、アンケートを実施したい、小テストを実施したい。
- 【出欠管理の厳正化】遅刻と早退を防ぐために二枚のカードを配りたい。逆に一枚で構わない。
- 【出欠カードと授業の対応】パスワードに、科目、日時の情報を含めたい。逆に含めないでどの授業にも使えるようにしたい。
- 【情報公開とプライバシー】出席状況は個人のプライバシー情報だから本人以外には公開しないようにしたい。逆に積極的に公開して本人の自覚を促したい。中間的立場として本人以外の出席状況は匿名で表示したい。
- 【〆切と例外処理】出席登録に〆切を設けたい。登録を忘れた学生に例外処理をしたい。

これらのすべてに最初から対応することは、現実には開発労力の点から困難であることから、当面はコア機能に絞って開発することとした。

しかしながら、上に挙げた以外にも今後現れてくるであろう様々なニーズに対応できるように、システムを随時改修することを想定した方が、システムの有用性を高めるという観点からは望ましい。

だがこれは、きわめてやっかいな問題を引き起こす。

すなわち、実運用の供しているシステムであるにも拘わらず、システム開発における分析、設計などの上流工程への手戻りが常時発生するということを意味するからである。手戻りに伴う再コーディングやテストの手間を何とかして軽減しないと、システムが破綻してしまう可能性がある。

本システムではこれに対して、**XML による仕様記述とプログラムの自動生成**という方法で対処することを試みた。

3. コア機能システムの運用

前節で述べた考察に基づき、筆者らは出席管理システム "kagami2" [1]を開発中である。既に2002年5月中旬にコア機能の部分を完成させ、筆者ら以外にも一部の先生方に実際の授業で運用していただいている。

コア機能で実装済みなのは、以下に示すように出席管理システムとして最低限備えているべき部分である。

- 使い捨てパスワードを印刷した出席カードを作成する機能(授業毎に各学生1枚、パスワード中に科目と授業日時の情報が含まれる)
- 学生がそのカードを用いて出席登録をする機能
- 出席登録時に感想を登録する機能(最低字数制限あり)
- 学生が自分の出欠状況を確認する機能
- 教員が科目ごとの出欠状況を確認する機能

以下、このコア機能の概要を実際の操作手順に沿って解説する。

教員の事前作業

教員は、当該授業が始まる前に(教室にPCとプリンタがあるなら授業中でも構わない)、図1の教員用総合メニューのWebページにアクセスする。

このページは、パスワードで保護されており、学

図1 教員用総合メニュー

[1] この名称には、授業を写す「鏡」となって欲しいという期待が込め

生はアクセスできないようになっている。

続いて、自分を選択する。図2のように科目を選択するページとなる。[2]

各教員の担当科目は、事前に事務方から入手してデータベースに登録済みである。なお、このシステムの一般教員への説明に際して、全教科のデータが登録済みなので自分でいちいちデータを登録しなくてよいという点に、かなり関心を示した教員が多かった。些細な事ではあるが「手間の軽減」という点で一定の訴求力があったと思われる。

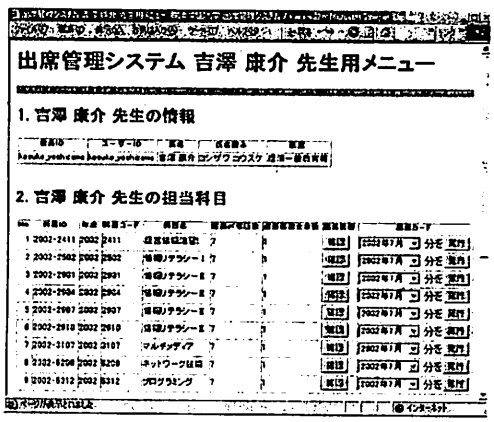

図2 個別の教員用メニュー

続いて、該当科目の出席カード【発行】ボタンをクリックする。いわゆる「万年カレンダー」が内蔵されているので、当月のカレンダーが表示される。

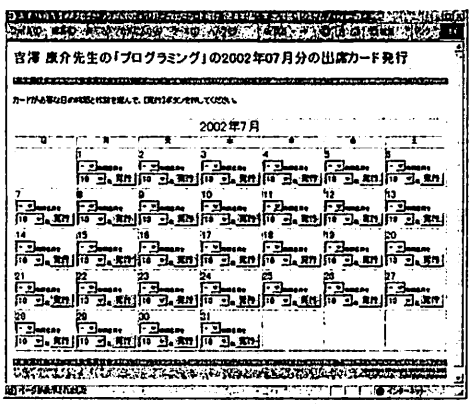

図3 カード発行用カレンダー

ここで、時間と枚数を選択して【発行】ボタンをクリックする。(科目が決まれば、授業の曜日と時間が特定される場合が多いと思われるが、補講や集中講義、あるいは何らかの理由で授業時間や曜日が一定しないケースを想定して、あえてユーザーに選択させるインターフェースを採用している。)

続いて、図4のように印刷すべきカードがPDF形式で表示される。これを表示するためにはブラウザにAdobeのAcrobat Reader[3]のプラグインが組み込まれていることが必要である。(Acrobat Reader自体はかなり普及しているので、運用上特に大きな障害とはならないであろうと思われる。)

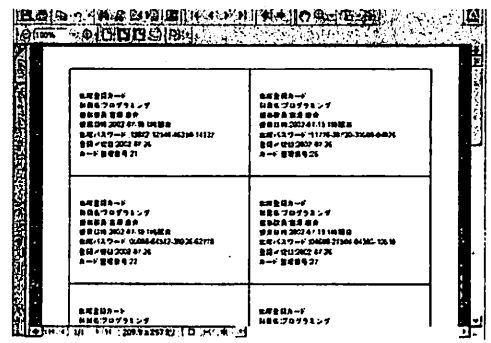

図4 出席カード

ここで、PDF形式を採用しているのは、正確にA4用紙のサイズに印刷するためである。

実は、図4のカードは市販の名刺カード用紙のサイズに正確に合わせてある。したがって、名刺カード用紙のコスト高さえ厭わなければ、カードの切り離しの手間は大幅に軽減されることになる。これも「教員の手間を軽減する」という印象を与えるための工夫の一環である。

また、一般のA4用紙に印刷する場合でも、カードのサイズが固定されることは、裁断の手間を軽減することに役立つ。

A4用紙を鋏で裁断することは、一見時間を要する作業のように思われるが、筆者らの主観ではA4用紙10枚程度(カード100枚程度)なら実際にやってみると大したことではない。

られている。

[2] 図には表示されていないが、ここで出席登録時の感想の最低文字数を指定することもできる。将来的には、個々の授業の「出欠ポリシー」を登録できるようにする予定である。

[3] Acrobat Readerについては、http://www.adobe.co.jp/ を参照されたい。

図4から分かるように、カードには20桁からなる個別の使い捨てパスワードが印刷されている。このパスワードは32bitの乱数と個々のカードを唯一識別するカード固有の32bitの番号を混合したものである。

また、カードに不足があれば、何回でも追加印刷して構わない。その科目のその授業で何枚のカードを印刷したのかは、カード整理番号として表示されている。

以上述べてきたように、カードの裁断を除いて、この時点までは単にメニューをクリックするだけで、出席カードを生成できるようになっている。

授業の際には、このカードを一人一枚厳守で配布することになる。この手間が、おそらくこのシステムを利用することで教員にかかる最大の負担となる。

いまのところ、一人一枚を担保する手段としては、教員(または教育アシスタントや事務職員)が手渡しする以外の有効な方法は見当たらない。

学生の人数が増えた場合カードの配布に問題が生じる恐れがあるが、少なくとも、60〜80名のクラスでは特に大きな混乱は生じていない。ただしこれ以上の人数のクラスではこのシステムを運用していないので何とも言えない。

学生の作業

カードを受け取った学生は登録の〆切までに図5に示す学生用総合メニューにアクセスすることになる。

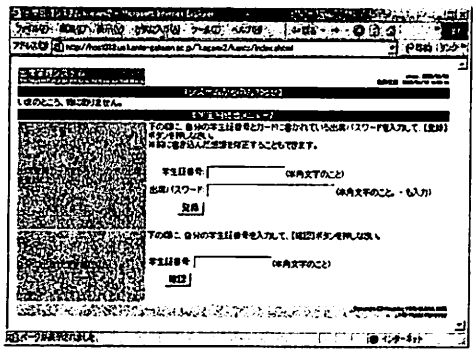

図5 学生用総合メニュー

自分の学生証番号と出席パスワードを正しく入力

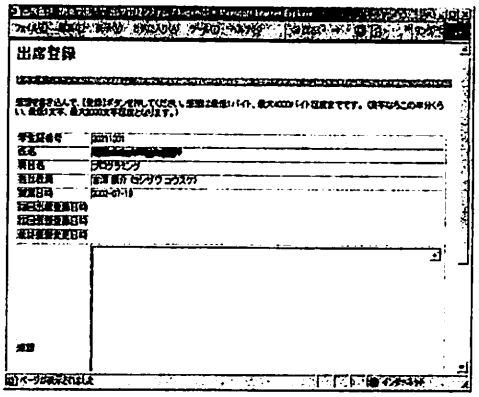

図6 出席と感想の登録

すれば、感想登録のページが表示される。

もちろん、一度使用してしまったパスワードは、他の学生が使うことはできない。〆切内に正しいパスワードと最初に出席を登録した学生の学生証番号を入力し直せば、既に登録済みの感想を「修正」する画面となる。これ以外の場合はエラーとなる。

教員による出欠・感想の確認

授業後、適当な期間を置いて図2の個別の教員用メニューから学生の出欠状況と感想一覧を確認することになる。

以上で、一連の出欠管理の手順が終了したことになる。

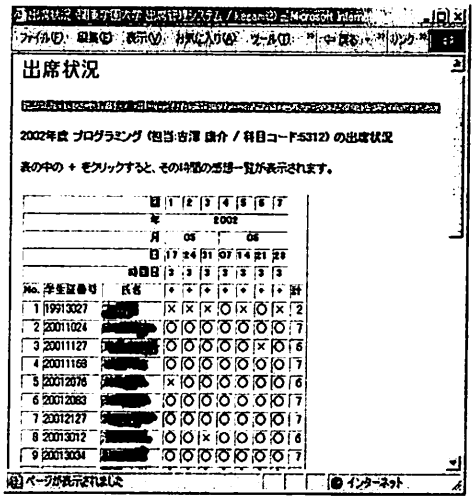

図7 出席状況

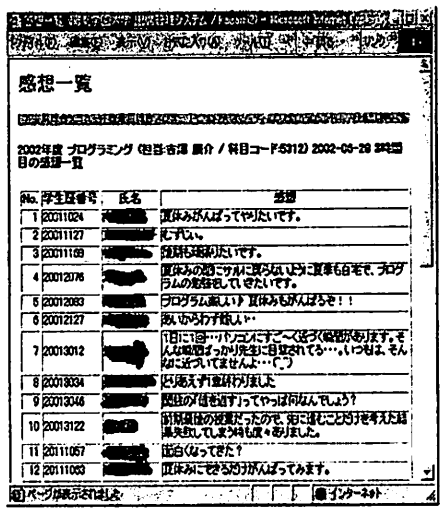
感想一覧

図8 感想一覧

4. システムの内部構造

全体の概要

システムは、PHP[2][3]をシステム記述言語とした、概略を表1に示すようないわゆる3層データベース[5]として実装されている。

項目	バージョン等
OS	Linux (Turbo Linux 7 Server[8])
RDBMS	PostgreSQL 7.1.2[4][5]
Web サーバー	Apache 1.3.20[7]
PHP	PHP 4.1.2
PDF ライブラリ	pdflib-4.0.2[6]
XSLT プロセッサ	Sablotron 0.90[9][11]

表1 システムの概要

なお、表中の PDF ライブラリ pdflib というのは、PDF ファイルを生成するためのフリーの API ライブラリである。本システムでは、PDF 形式のデータ生成にこの pdflib を用いている。

XSLT プロセッサについては後述する。

いかにして省力化するか

第1節で述べたように、このシステムは様々なニーズに応えるために、頻繁に仕様変更、それもシステムの根幹に係わる変更が頻発することが予想される。

その変更労力をできるだけ省力化するためには、

- 仕様記述を一箇所に集中させ
- プログラムを自動生成する

という手法を採用することが考えられる。

しかしシステム全体に渡ってこの方針を貫くことは容易ではない。決定的な仕様記述言語はいまだ存在しない。

そこで筆者らは、仕様から比較的容易に機械的なプログラム生成が可能であるデータベースのスキーマの部分に集中して、この手法を適用することを試みた。

XML による仕様記述

```
<database name="kagami2"> ……データベース全体
  <attr_prototypes> ……属性辞書
    <attr_prototype name="valid"> ……一つの属性
      <type>char(1)</type>……SQL の型
      <constraints> ……制約の集合
        <constraint>not null</constraint> ……個々の制約
      </constraints>
      <alias>有効</alias> ……分かりやすい別名
      <rem>この tuple の状態を示すフラグ。</rem> …人間のための解説
      <alts>……値の選択肢の集合
        <alt> ……選択肢その1
          <value>1</value> ……値
          <rem>有効</rem>……分かりやすい別名
        </alt>
        <alt>
          <value>0</value>
          <rem>無効</rem>
        </alt>
      </alts>
    </attr_prototype> ……一つの属性定義の終了
    以下、属性定義が並ぶ
  </attr_prototypes>……属性辞書の終了
  <tables>……テーブルの集合
    <table name="student">……一つのテーブル
      <alias>学籍テーブル</alias> ……分かりやすい別名
      <rem>全学生の一覧表。</rem> ……人間のための解説
      <attrs> ……属性の集合
        <attr>valid</attr>……属性その1
        <attr>studentId</attr>……属性その2
        中略
      </attrs>
      <pkeys> ……主キーの集合
        <pkey>studentId</pkey>……主キーその1
      </pkeys>
    </table>……一つのテーブルの終了
    以下、他のテーブルの定義が並ぶ
</database> ……データベース全体の終了
```

図9 XML によるデータベースの仕様記述

まずデータベースの構造を XML で記述する。

図9に実際の仕様記述を説明のために簡略化したものを示す。なおタグは筆者らが独自に定めたものである。

この図から分かるように、タグとその階層構造は、標準的なリレーショナルデータベースの考え方に沿ったものであり、おおむね以下のような構造を成している。

- データベースは属性辞書とテーブルの集合からなる。
- 個々の属性には、型、制約、取り得る値の範囲などが含まれる。
- テーブルには属性の集合、主キーなどが含まれる。

プログラムの自動生成

このXMLによる仕様記述から、このシステムでは図10に示すような様々なコンポーネントを自動生成している。

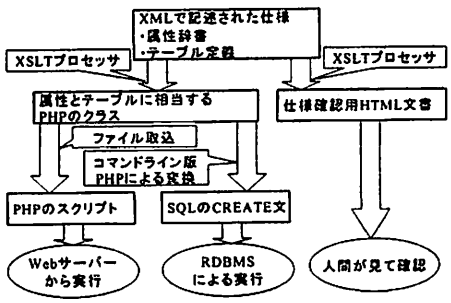

図10 プログラム自動生成の流れ

このような集中的な仕様記述によって、例えば学生証番号の桁数を変更するといった修正が、速やかにシステム全体に行き渡り、矛盾の発生を回避できることになる。

自動生成される主たる成果物は、データベースのテーブルにアクセスするためのPHPのクラスオブジェクトと、RDBMSに与えるテーブル生成のためのSQL文である。このうち、SQL文の生成例を図11に示す。

また、人間にとって分かりやすいドキュメントを自動生成することも重要であり、システムの開発やメンテナンスの手間を軽減することに寄与している。(図12)

```
— student(学籍テーブル) の削除
DROP TABLE student;
— student(学籍テーブル) の生成
CREATE TABLE student (
  valid char(1) not null, — 有効
  studentId varchar(32), — 学生証番号
  userId varchar(32), — ユーザーID
  shimei varchar(128), — 氏名
  shimeiYomi varchar(128), — 氏名読み
  shimeiRomaji varchar(128), — 氏名ローマ字
  seibetsu char(1) not null, — 性別
  shozoku varchar(128), — 所属
  EMail1 varchar(128), — E-Mail1
  EMail2 varchar(128), — E-Mail2
  EMailPrivate1 varchar(128), — E-Mail個人用1
  EMailPrivate2 varchar(128), — E-Mail個人用2
  Web1 varchar(128), — Web1
  Web2 varchar(128), — Web2
  WebPrivate1 varchar(128), — Web個人用1
  WebPrivate2 varchar(128), — Web個人用2
  privacyPassword varchar(32), — プライバシー・パスワード
  PRIMARY KEY(studentId)
);
```

図11 生成されたSQL文

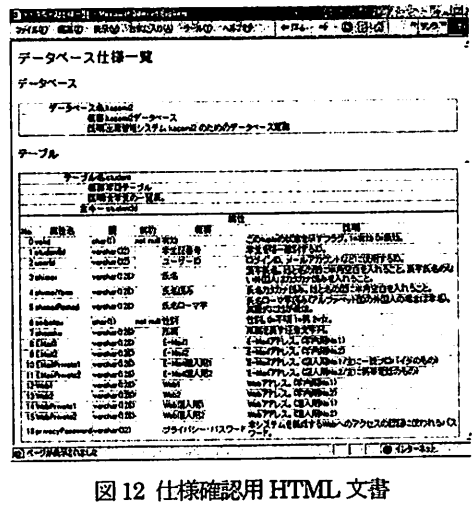

図12 仕様確認用HTML文書

XSLTプロセッサの問題点

XMLから各種プログラムの自動生成には当初はすべてXSLTプロセッサ[4]を用いる予定であったが、XSLTはプログラミング言語として記述能力が低く、また変換用スタイルシートも煩雑になる傾向があっ

[4] XSLTとは、端的に言えば、あるXML文書を他のXMLないしHTML文書(あるいは他の任意形式)に変換する仕組みである。スタイルシートと呼ばれるその変換ルール自体がXMLで記述されている。スタイルシートを解釈して変換を実施するのが、XSLTプロセッサである。[阿川]

たので、やむを得ずSQL文の生成の部分でPHPによる変換を使用している。

こういったプログラムの自動生成は必然的にかなり複雑なテキスト処理を伴う。XSLTを使用することで「すべてをXMLの枠内で処理」となる点は評価できるが、現時点ではテキスト処理に長けたスクリプト言語に優位性があるものと考えられる。

5. まとめ

教員は何を得たか

もちろん、出席状況が容易に確認できる意味も大きいが、多くの教員にとって図8の感想一覧が得られることが最大の利点であろう。わずか一行ないし数行の感想であるが、学生との間でコミュニケーションが図れる意義は大きい。例えば、授業に関する不満が書かれた場合、それに対して教員が誠実に対応することによって、教員と学生との間での信頼関係が醸成され、学生の授業に対する満足度が上がっていったことが明白にわかる例もある。

学生は何を得たか

そのうら返しの関係が学生にも成り立つ。「質問はありますか」という教員の問いに教室が静まり返るのは日常茶飯事であるが、発言しやすい機会さえあれば、学生も意見を述べるのである。

その一方で、出席管理システムの是非については、おおむね好意的な学生が多いものの、否定的な意見を持つ者もいるようである。特にパスワードが長いという点と、〆切内に登録し忘れた場合への不満が大きいと考えられる。[5]

システム開発者は何を得たか

このシステムの開発における焦点は、教室にコンピューターが無い環境で使い物になるシステムを作れるか、一般教員と学生に使ってもらえるシステム作れるか、そして、とにかく限られた労力で実用的に使えるシステムを開発できるかという三点であった。これらの点に関しては、いずれも一定の成果を納め、肯定的な解答を得たものと考える。

今後の展望

以上の総括を踏まえ、教員・学生からの要望・不満をできるだけ吸収するようにシステムの改良を続けていきたい。一般的な使い勝手の向上はもちろんのこと、本年度後期には、i モード等の携帯電話の利用、感想だけでなく簡易アンケート処理(小テスト)を行う機能の搭載などを試みる予定である。

謝辞

システムの開発にあたっては、関東学園大学、東京情報大学の多数の教職員の皆様に筆者らの議論に参加していただいた。特に出欠管理に使い捨てパスワードを利用するというアイデアは、関東学園大学の小川浩助教授との議論の中で生まれて来たものである。また同大学の瀧上豊教授、中空齋雅教授には、学内有志の研究会においてこのシステムを紹介する機会をご提供いただいた。さらに瀧上豊教授、二神常爾講師の両先生には実際にシステムを使用してもらい貴重なご意見、データをご提供いただいた。これらの皆様方にこの場を借りて深く謝意を表したい。

参考文献

[1] 吉澤康介, 長江和子, 三宅修平, 安斉公士, 木村昌史, 「3層型データベースを用いた出欠管理システム」, 平成12年度第13回情報処理教育研究集会講演論文集, pp357-360, 2000

[2] http://www.php.net/

[3] 山田祥寛, 「今日からつかえるPHP4サンプル集」, 秀和システム, 2002

[4] http://www.postgresql.org/

[5] 石井達夫, 「PostgreSQL 完全攻略ガイド」, 技術評論社, 1999

[6] http://www.pdflib.com/

[7] http://www.apache.org/

[8] http://www.turbolinux.co.jp/

[9] http://www.gingerall.com/charlie/ga/act/index.act

[10] http://www.xml.org/

[11] Steven Holzner, (株)クイック訳, 「XSLT 実践ガイド」, アスキー, 2000

[5] システムを利用中の一部の先生に、出席管理システムに関する学生アンケートを実施して頂いたのであるが、結果が報告されたのが本稿〆切の二日前であった。このアンケート結果やその他の学生・教員のシステムに対する評価については別の機会に報告することとしたい。