2005

Towards More Comprehensive Information Retrieval Systems: Entity Extraction Using XSLT

Chris A. McManigal

University of North Florida

TOWARDS MORE COMPREHENSIVE INFORMATION RETRIEVAL SYSTEMS: ENTITY EXTRACTION USING XSLT

by

Chris A. McManigal

A project submitted to the Department of Computer and Information Sciences in partial fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES

December, 2005

The project "Towards More Comprehensive Information Retrieval Systems: Entity Extraction Using XSLT" submitted by Chris A. McManigal in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been

Approved by the project committee: Date

Signature Deleted

1/3/2006

Sherif A. Elfayoumy, Ph.D.
Project Director

Signature Deleted

1/3/06

Judith L. Solano, Ph.D.
Chairperson of the Department

Signature Deleted

1/3/06

Charles N. Winton, Ph.D.
Graduate Director

ACKNOWLEDGEMENT

I would like to thank Dr. Sherif Elfayoumy for the time and assistance necessary to complete this project. Without his direction, I would have been unable to see the big picture due to dwelling on the details. His guidance has been indispensable.

I would also like to thank Dr. William Klostermeyer, Dr. Roger Eggen, and Justin Gaudry for their commitment to instruction and dedication to their students. These three educators were vital to my interest in and understanding of the computer sciences.

Finally, I would like to thank my wife Angela and my son Benton, without whom none of my accomplishments would be worthwhile or even possible.

CONTENTS

FIGURES

ABSTRACT

One problem that exists in today's document management arena is the issue of retrieving information from electronic documents such as images, Microsoft Office documents, and e-mail. Specific data entities must be extracted from these documents so that the data can be searched and queried. This study presents a unique approach to extracting these entities: using Extensible Stylesheet Language Transformations (XSLT) to match patterns in text. Because XSLT is processed at run time, new XSLT templates can be created and used without having to recompile and redeploy the application. The specific implementation addressed in this project extracts entities from an image file. The data in the image file is converted to Extensible Markup Language (XML) text via optical character recognition (OCR), and then this XML text is transformed into an organized, well-formed XML output file using an XSLT template. We show this approach can accurately retrieve the correct data and this method can be extended to other electronic document sources.

Chapter 1

INTRODUCTION

An estimated 90% of all business information is held on paper [Jasper02]. Even with the computer renaissance that has been taking place over the past twenty years, companies still store information on paper. Forms are filled out by hand or typewritten and kept in duplicate, triplicate, or even more. Many companies store copies of these forms in boxes kept in storage rooms because either relevant laws or the risk of litigation requires these documents are kept. More than four billion pages that must be archived are produced in the United States alone each year [Lyman03]. As more and more companies move towards a policy of using a paperless environment, these companies must decide what to do with the many years worth of data contained in those boxes. One idea is to simply leave them on paper and stored in boxes as they are now. However, there are multiple problems with this solution:

- the paper and ink, and therefore image quality, deteriorate over time

- a records request requires someone to know where to look, and therefore, someone must maintain a catalog of the information
- the amount of space wasted on storage is expensive
- a reasonable way to aggregate the data in not available

A second alternative is to use today's technology and scan the documents onto magnetic or optical storage. This process requires manual indexing of the images as they are scanned and stored in a database. Entities from the document can also be manually recorded in the database, but manual indexing and entity extraction are both extremely time consuming and error prone, which can lead to increased costs to the organization.

Another problem exists whether the information is left on paper or transferred to electronic media: this data is not currently searchable by most information retrieval systems. The technology to access information from images is still in its infancy, so current systems are designed to search only textual data content [Weglarz04]. They analyze a text document, and indexes are created and stored in a database along with a copy of the original document. A keyword query is performed against the index, and matching documents are

returned. Unfortunately, these information retrieval systems ignore the vast amounts of data contained on paper and in electronic images.

Even if methods are used to retrieve information from these sources so an information retrieval system can analyze and index them, a second problem exists: false positives. Because queries are limited to keywords, matches are returned regardless of their relevance. For example, if a search is performed in an address book document for a person named "King," the system will return hits for those named "King" as well as those who live on "Martin Luther King Blvd." Obviously, this problem makes searching for "King" on "Martin Luther King Blvd" difficult, as a record will be returned for everyone living on "Martin Luther King Blvd." The information retrieval systems could be made more useful by limiting searches to specific entities.

This project presents a solution to some of the problems associated with retrieving information from electronic documents. First, by using conversion adapters, information will be retrieved from unstructured data formats such as images, Microsoft Office documents, or e-mail and converted into semi-structured data in the form of an Extensible

Markup Language (XML) document. Specific entities will be extracted using Extensible Stylesheet Language Transformations (XSLT) to improve the quality of searches. Finally, this semi-structured data within the XML document could be converted to structured data by storing the data in a database that can be queried using any of the extracted entities. Although this implementation is limited to a single image document (see section 3.1.2), this project presents a new approach to entity extraction and information retrieval that can be successfully replicated in other areas. Specifically, the project demonstrates an XSLT document that is processed at run time can be used to provide rules for entity extraction so the application does not have to be recompiled and redeployed.

Chapter 2

BACKGROUND

2.1 Motivation

The motivation for this project comes from witnessing the conversion of paper to electronic storage in the workplace. A nearby public school system stores transcript data from the early 1900s to present. The school system has used an electronic student information system for the past five years, so the current data is already maintained in a database. However, the past data is kept on paper transcripts in boxes that take up several storerooms. Each box has several records inside sorted by year of graduation and student's last name. No index exists except for a page on the top of each box listing the contents inside (e.g., "2003-04 Benton - Brown").

These records must be retained indefinitely because they provide information including proof of graduation, immunizations, and birth date. A project was started approximately two years ago to scan all of these forms onto both optical storage and microfiche; this redundant storage

provides safety in the case of building fire or natural disaster. Unfortunately, the only indexing that takes place is recording the student's name with the digital files. These records contain unknown amounts of useful data, but at present, manual searching provides the only way to access the data. The electronic extraction of entities and the importing of this data into a database would make both record searching and data mining possible.

2.2 Unstructured, Structured, and Semi-structured Data

Two main classifications of data exist: unstructured and structured data. Unstructured data can be broken into bitmap objects, such as images, video, or audio files, and textual data like Microsoft Office documents or email [Weglarz04]. Documents and files from these two categories definitely contain data, but there is no conceptual or data type definition included. Therefore, human intervention must occur to provide metadata; for instance, text can be tagged with keywords that can be used for searching.

Structured data typically refers to data that is strongly typed and stored as records in a relational database [Sanchez04]. The value of structured data comes not only

from its rigid definitions, but also from the contextual meaning that is provided [Crampton04]. For example, a date is simply a value, but within structural data, this date can be related to a phone call or a meeting. Completeness provides a second benefit for structured data; the data model explicitly requires certain data be included, and therefore, unlike unstructured data, information will not be missing. The rigid data definition, as well as the completeness of the information, ensures the data can be easily searched.

In the gray area between unstructured and structured data lies semi-structured data. Semi-structured data provides more than simple text, often by the use of tagging. Tagging offers both data type and contextual definitions without requiring completeness. XML is one of the most common ways to store and present semi-structured data [Li01]. XML offers the flexibility to define semantics for the data elements by using user-defined tags. Because of the use of these tags, languages such as the XML Path Language (XPath) and the XML Query Language (XQuery) help navigate an XML document tree and can be used to query information from XML documents. This ability provides some structure to the data without the rigidity required of structured data.

2.3 Extensible Stylesheet Language Transformations

The primary focus of the project is on the development of Extensible Stylesheet Language Transformation (XSLT) templates. These templates are used to transform one XML document into another format that is more useful to the user. XSLT templates use the XPath language to traverse and inspect nodes in the source XML document's tree. Upon finding matching nodes, instructions are processed to transform the input data into output.

XSLT is a declarative programming language; instead of telling the computer "how" to do something, the developer tells the computer "what" to do, much like Structured Query Language (SQL) [Kay03]. This characteristic decreases programming time and effort, but the developer must work with the XML processor to ensure the proper results are obtained.

Applications exist today that are developed solely in a procedural programming language, but by using XSLT, the application can be made more robust, as well as more flexible. If written strictly in a procedural language, the conversions from electronic document to XML would have to

be hard coded. Any new forms would require new functions, and the program would have to be rebuilt and redeployed. XSLT, on the other hand, is a scripting language, and therefore, the code is processed at run time. Because of this characteristic, the user can create and add additional templates throughout the lifetime of the application without needing the source code for rebuilding. Though processing the XSLT templates at run time results in somewhat worse performance than compiled programs, the ability to add functionality without recompiling is beneficial to the end user.

Chapter 3

CONVERSION FROM ELECTRONIC DOCUMENT TO XML

3.1 Introduction

This solution implements a Microsoft Windows-based application that allows the user to select electronic documents to be converted to XML. The author chose to use Microsoft Visual C# .NET as the development environment because this choice uses an object-oriented design paradigm, simplifies the graphical user interface (GUI) development, and ensures platform independence with the .NET framework.

3.1.1 User Interaction

The user interacts with the program via multiple GUIs. The main form prompts the user for the image file(s) to be converted and the template to be used. The output results are displayed to the user. The user can manually edit these results as well as create new XSLT templates. All actions produce new GUIs to guide the user through the required steps.

3.1.2 Scope

Although the abstract design of this solution provides support for multiple forms of unstructured data, the scope of this implementation is limited to a single image form, a scanned high school transcript. Additionally, this application extracts specific entities that may be relevant only to this form.

One feature of the solution is an add-on not defined in the project proposal, and therefore, this feature is limited in functionality. The ability for the user to create XSLT templates works solely on the previously mentioned transcript and only with certain fields. The point of this add-on is to demonstrate the flexibility of the application. One of the benefits of using XSLT is the fact that the program does not have to be recompiled every time a different image format is used; instead a new XSLT template is used that is processed at runtime. With the limited template builder, we show a GUI can be designed to allow users who are not fluent in XSLT to be able to define new rules and extract new entities.

3.2 Program Modules

Because modularity in programming ensures efficient development as well as simpler maintenance, this application is composed of several distinct modules as shown in Figure 1.

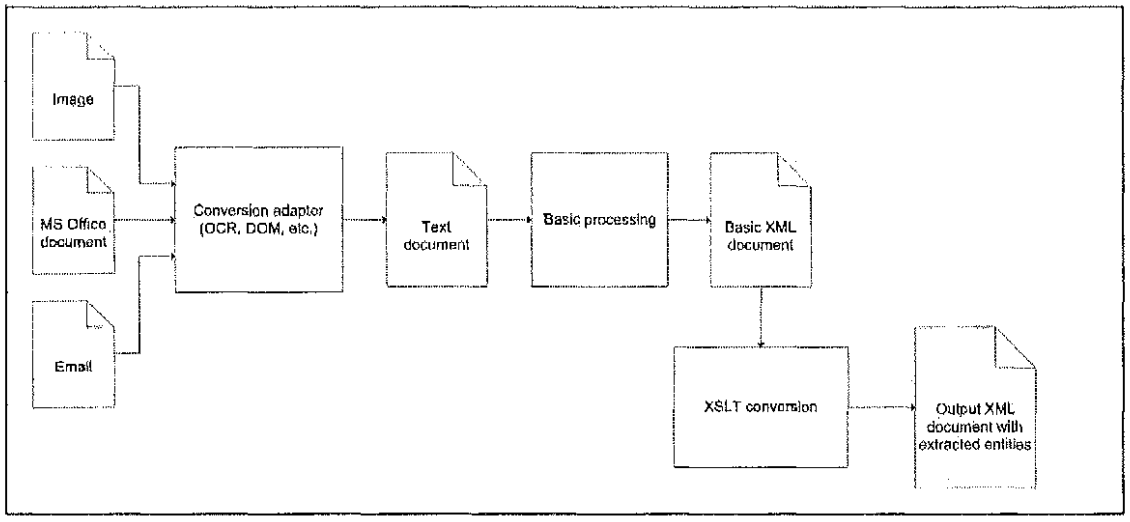


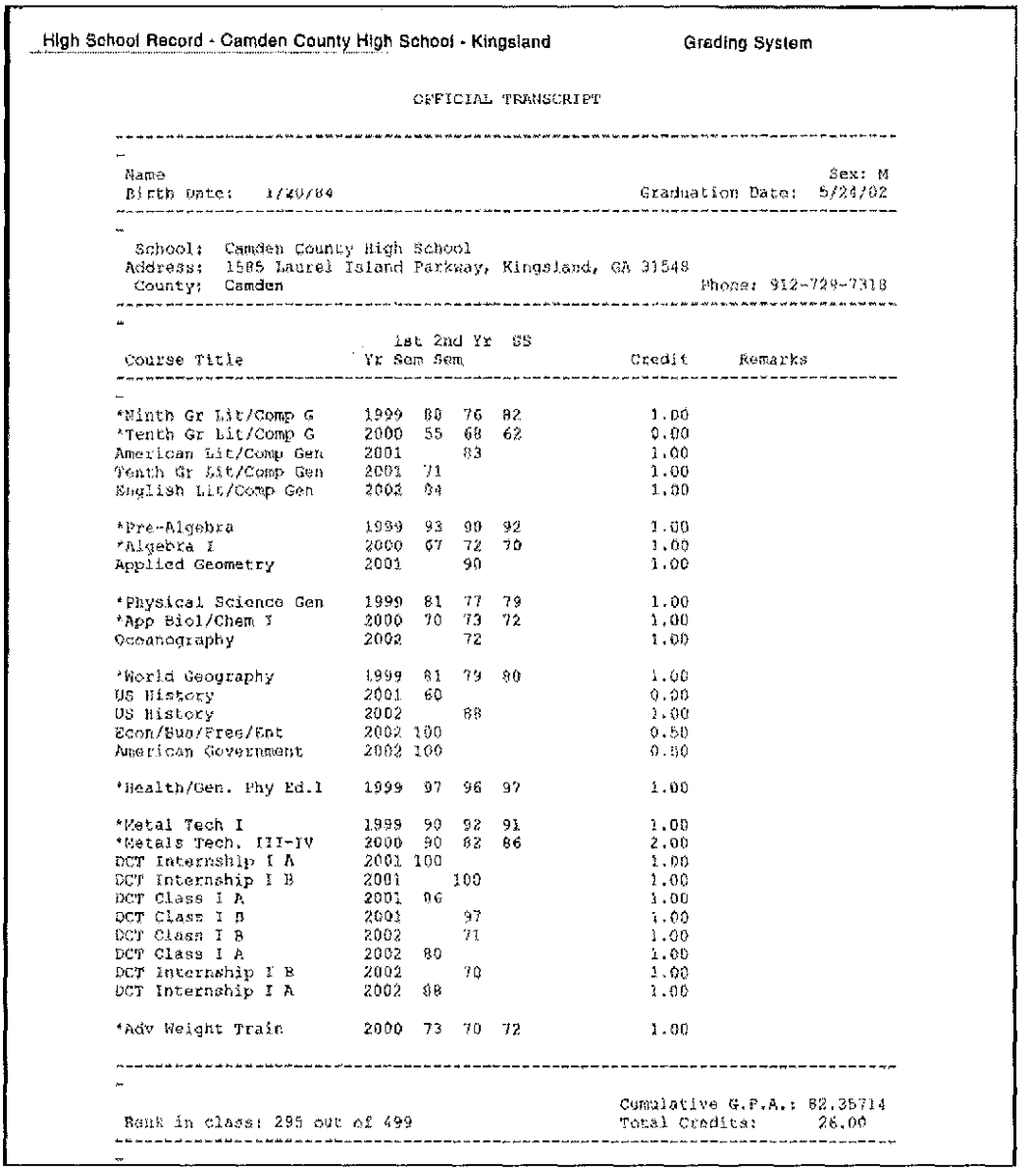
Figure 1: Overview of Project Functionality

The transformation from an electronic document to an XML document requires multiple steps. These same steps are required for any of the multiple electronic document types: retrieve information from the input document as text, process the text document into a basic XML file, transform this basic file using XSLT into a better organized XML file while extracting all desired entities, and then finally,

combine the XML files with others into a larger XML file that can be entered into a database.

3.2.1 Input File

For this implementation, the input file must be a scanned image file saved in the tagged image file format (TIFF). Figure 2 shows a sample transcript for which the application was designed to extract a number of entities.



High School Record - Camden County High School - Kingsland Grading System

OFFICIAL TRANSCRIPT

Name Sex: M
Birth Date: 1/20/84 Graduation Date: 5/24/02

School: Camden County High School
Address: 1585 Laurel Island Parkway, Kingsland, GA 31548
County: Camden Phone: 912-729-7318

Course Title	Yr	1st Sem	2nd Sem	Yr SS	Credit	Remarks
*Ninth Gr Lit/Comp G	1999	80	76	82	1.00	
*Tenth Gr Lit/Comp G	2000	55	68	62	0.00	
American Lit/Comp Gen	2001		83		1.00	
Tenth Gr Lit/Comp Gen	2001	71			1.00	
English Lit/Comp Gen	2002	84			1.00	
*Pre-Algebra	1999	93	90	92	1.00	
*Algebra I	2000	67	72	70	1.00	
Applied Geometry	2001		90		1.00	
*Physical Science Gen	1999	81	77	79	1.00	
*App Biol/Chem I	2000	70	73	72	1.00	
Oceanography	2002		72		1.00	
*World Geography	1999	81	79	80	1.00	
US History	2001	60			0.00	
US History	2002		88		1.00	
Econ/Bus/Free/Ent	2002	100			0.50	
American Government	2002	100			0.50	
*Health/Gen. Phy Ed.1	1999	97	96	97	1.00	
*Metal Tech I	1999	90	92	91	1.00	
*Metals Tech. III-IV	2000	90	82	86	2.00	
DCT Internship I A	2001	100			1.00	
DCT Internship I B	2001		100		1.00	
DCT Class I A	2001	96			1.00	
DCT Class I B	2001		97		1.00	
DCT Class I B	2002		71		1.00	
DCT Class I A	2002	80			1.00	
DCT Internship I B	2002		70		1.00	
DCT Internship I A	2002	88			1.00	
*Adv Weight Train	2000	73	70	72	1.00	

Cumulative G.P.A.: 82.35714
Rank in class: 295 out of 499 Total Credits: 26.00

Figure 2: Sample Input Transcript File

Although much information is available in the input image, this program will only extract the student's name, gender,

birth date, graduation date, grade point average, rank in class, and total number of credits earned.

3.2.2 Conversion Adapter

For the given input image file, the first step of the process is performing Optical Character Recognition (OCR). The OCR process retrieves the text information from the original image, and the output is a simple text file. Microsoft Office 2003 contains the Microsoft Office Document Imaging Object Model, which contains OCR functions that can be called from within a Visual C# application. Although the OCR problem is interesting, this problem has been thoroughly researched and will not be discussed.

3.2.3 Processing of OCR Output

The next step requires some basic processing to occur on the OCR's output file. XSLT requires an XML file as input because XSLT transforms one XML file into another XML file. The text file, at a minimum, must have some basic XML tags placed in the document so an XSLT template can be used for processing. This application takes the text output from the OCR step and attaches an XML declaration as a header. Next,

the text is surrounded by a single XML node tag pair: <text> and </text>. Finally, the XSLT template file name provided by the user is used to create the XSL stylesheet declaration that directs the XML processor to the appropriate template. The result is a basic, well-formed XML document with one <text> node as shown:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl"
     href="C:\MS\ImageToXMLSource\Data\test2.xslt"?>
<text>

. . .

</text>
```

3.2.4 XSLT Templates

The main effort of the project development was the creation of the XSLT templates. Each form type that has been scanned into an image has a different XSLT template. A library of XSLT templates could be developed, with a different template representing different form types. The user can then choose the image to be converted along with the proper form; this library could increase the program's functionality while making the addition of forms quite simple. The key feature is multiple forms can be added

without having to recompile, and therefore redeploy, the application.

3.2.4.1 Predefined Template

The predefined template is packaged with the application. This template contains pattern-matching code written for the specific form type used in this implementation. Specifically, the template uses a colon (':') for the delimiter, and the program retrieves only those entities mentioned in section 3.2.1. The predefined template recursively searches the text for the output data until no text remains.

3.2.4.2 User-created Templates

Included with the application is a template builder. This feature allows the user to define additional templates, thus adding flexibility to the overall concept of this solution. The user is provided a text box with proper XSLT headers and a root template, which are both required parts of an XSLT document. Optionally, the user can invoke an interface that allows rules to be defined graphically, thus providing accessibility to those unfamiliar with XSLT. The

key feature of the template editor is the XSLT code is automatically generated after the user defines the rules visually.

The interface asks the user to provide a name for the document's root element as well as a name for the specific relevant data. The interface then prompts the user to define a prefix to the data, and the user must indicate whether the data string is terminated by length or by a suffix. After the rule has been saved, the program automatically generates the XSLT code and displays this code in the text box for the user to see. Finally, the user is prompted to save the file for future use. More details on the interface are provided in the User Manual in Appendix A.

3.2.5 XSL Transformation and Entity Extraction

When the default XSLT template is used, the string inside the <text> node of the basic XML file is passed to the extraction template along with the delimiter. The XML processor navigates the single-node tree and searches for the delimiter within the text. When the delimiter is found, the substring before the colon is pattern-matched against

all possible words that precede the colon in the image. When the correct match is found, instructions are processed to locate the string after the delimiter and to mark that string as one piece of the required data.

For instance, when the program locates a colon preceded by "Sex," the begin tag <gender> is written. The XSLT then instructs the processor to take the substring two characters in length immediately following the delimiter. The white space, if any, is removed, and the resulting string is output. Finally, the closing tag </gender> is written. After retrieving the data, the remaining text following the delimiter is recursively passed to the extraction template so the next item can be located.

Data such as the student's name or birth date can be broken into multiple parts. In these cases, the full string is passed to a template that subdivides the string based on specific delimiters found in that string. With a birth or graduation date, for example, a forward slash ('/') separates the month, day, and year fields. Using this delimiter, the date is parsed into three separate children of the output <date> node.

When the processor encounters a prefix that is not required in the default XSLT template, the recursive call is made immediately with no further instructions required. The process repeats until there are no more delimiters found. Processing of the user-created templates is similar except only certain data fields have been implemented. This add-on feature was created to show templates can be created graphically, and therefore, full functionality does not currently exist.

3.2.6 XML Output File

The file output by the XSL Transformation contains an XML header, a <transcript> root node, and several data nodes as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<transcript>
  <studentname>
    <last></last>
    <first></first>
    <middle></middle>
  </studentname>
  <gender></gender>
  <birthdate>
    <month></month>
    <day></day>
    <year></year>
  </birthdate>
  <graddate>
    <month></month>
    <day></day>
    <year></year>
```

```
  </graddate>
  <gpa></gpa>
  <class>
    <rank></rank>
    <total></total>
  </class>
  <credits></credits>
</transcript>
```

An XML reader can easily navigate the well-formed XML output file in order to determine element names, their values, and even the node's depth within the tree. This information is then displayed to the user via the main interface, thus providing data verification and editing capabilities. Additionally, several individual XML files can be traversed by an XML reader and then combined into a single output file with an XML writer.

3.2.6.1 Editing Data

After completion of the conversion process from an image file to an XML document, the program displays all retrieved data to the user. The user can select any converted file in order to verify data against the input file and to edit the data retrieved. A new form is displayed that shows each entity extracted from OCR text and a thumbnail of the original image. The user can cross-check any of the

entities and make corrections directly in the text box. When saved, any edits are written to the XML output file as well as the main GUI screen for the image in question. The most typical cause of errors is the OCR process, which will be discussed further in section 4.2.

3.2.6.2 Exporting Multiple Data to a Single File

As discussed above, the XSL Transformation process produces a single XML output file for each input image file. Although these individual files might be adequate for some users, the program also provides the capability to combine these multiple output files into a single XML document. This output simply inserts all of the root <transcript> nodes from each document as children in a <collection> node as follows:

```
<collection>
  <transcript>
    +<studentname>
    <gender>
    +<birthdate>
    +<graddate>
    <gpa>
    +<class>
    <credits>
  </transcript>
  +<transcript>
  +<transcript>
</collection>
```

Using XML files instead of directly inputting the data into a database provides flexibility to the user as they may prefer to work with XML files based on other applications that may be already in use. Also, the XML files offer some degree of adaptability in what type of data is stored compared to the rigid rules some databases place on data type and format. Finally, most databases today readily accept XML as input in a variety of ways, thus the user maintains this option if desired.

Chapter 4

TESTING

4.1 Overview

In order to justify any application, we must be able to show the benefits gained from the program outweigh the development costs. When discussing entity extraction from images, we must be able to quantify the accuracy of the program. After all, if the user must spend an inordinate amount of time with manual edits after having executed the program on several files, then the cost of the problem may be prohibitive. The primary cause of errors in this application is due to the OCR process; these errors will be discussed as well as the increase in accuracy provided by some simple solutions.

4.2 Discussion of OCR Errors

Several factors influence the quality of the text derived from an image during the OCR process including scan resolution, paper quality, typeface clarity, typographical and formatting complexities, and linguistic differences

[Haigh96]. For instance, the transcripts in this study have been stored for years in boxes; stains, dust, and ink smears can all be inadvertently translated into various characters, like a speck being resolved as an accent mark. In this analysis, specific errors showed up multiple times, and each was handled individually.

The possible choices for the gender of the student include "M" for male and "F" for female. However, many times (see section 4.4) the male code was incorrectly interpreted as an "N," and the female code translated to "P." Another example is when "G.P.A." is rendered as "C.P.A." These types of errors affect the program in different ways. In the first case, the actual data is inaccurate; the second example shows where an incorrect prefix could prevent the XSLT template from locating the proper data because there would be no pattern match, therefore missing data would be the result.

Using the editing form discussed in section 3.2.6.1, these errors could easily be fixed. If the user sees a gender of "P," he can inspect the image and determine the correct character and replace it. Similarly, if a piece of data is missing, the data can be manually entered after having

viewed the form. However, if a high percentage of the images produce these types of errors, then the user spends too much time making corrections to make the application beneficial. In the cases where many consistent errors were noticed, corrections were hard coded by using simple replace techniques when writing the OCR result to the basic XML file. Though this substitution limits the abstractness of the application, the solution is within reason when considering the scope of this project.

4.3 Special Characters

The input transcript images create another common error that will halt the XML processor. Many of the class names in the course history portion of the transcript include an ampersand ('&') character; for example, a class has the title "Debate & Public Speaking." The ampersand is a special character in XML, informing the processor of an entity reference where a code should be replaced by a symbol [Kay03]. The XML processor throws an exception when this symbol is encountered out of context, like with the class titles. In this case, the ampersand is replaced by the string "and" prior to writing the basic XML file.

Alternatively, the ampersand could be replaced with "&amp:" and the processor will know the ampersand is a literal.

4.4 Empirical Results

The accuracy of the extracted entities is the basis of testing for this application. This study consists of a sample of 130 randomly selected transcripts from one year's records, which is approximately 25% of the 499 available. Each transformation extracts 14 possible entities, so there are 1820 total entities to be checked for accuracy.

The transcripts were first checked with no corrections made; this experiment simulated the worst-case scenario. The total number of errors equaled 247 for an accuracy of 86.4%. The most common problem was the OCR translation of "C.P.A." for "G.P.A."; this error prevented other fields from being evaluated, because the XML processor stopped navigating the file when an invalid prefix was found.

After using a replacement routine to correct the previous error, the test was repeated. This test run produced 118 total errors for an accuracy of 93.5%.

The next error correction replaced the string " :" with ":." Though this change seems minor, simply removing a white space character preceding the colon, the consequences were great due to the fact the colon is used as the delimiter in the default template. The results of this change produced only 42 errors and an accuracy of 97.7%.

A third correction consisted of replacing the gender codes of "N" and "H" with "M" and "P" with "F." Although "H" and "P" were nowhere near as prevalent as "N," there was no justifiable reason to leave them unchanged when they were obviously wrong. This final test run generated 9 errors and an accuracy of 99.5%.

As can be seen with the above results, a few relatively minor corrections to the OCR output can dramatically improve the accuracy of the entity extraction. Replacing the three most common recognition errors reduced the number of errors by roughly 96%, and the overall accuracy of the entity extraction increased 13.1% to nearly perfect.

Chapter 5

CONCLUSIONS AND FUTURE ENHANCEMENTS

5.1 Conclusions

This project demonstrates one solution to the problem of retrieving information from data stored in images. By extracting entities from the input image files, the program makes the information contained in these images available for searching and querying. Though the scope of this application is limited to a specific input form type, the general solution is applicable to a wide variety of documents. This solution is more comprehensive than other information retrieval systems because of the way the information is extracted: using XSL Transformations. Because XSLT is processed at runtime, many different forms can be converted to XML without having to recompile and redeploy the application. Additionally, by employing an XSLT template editor that works graphically, users not trained in XSLT or XML are still able to utilize the program. These users can visually add rules that define the location of entities within a document. The template editor uses these rules to automatically generate the well-formed

XSLT code necessary for the application and allows the user to save the file for use in the program.

5.2 Future Enhancements

Several enhancements could be made to this program, and these enhancements can be broken into two categories: improving accuracy and making the application more general.

In order to improve accuracy, more work should be done with the available OCR features to eliminate some of the interpretation errors mentioned in section 4.2. By reducing these errors, fewer hard-coded replacements as well as manual edits would need to occur.

Secondly, the program could be designed to be more general in form. The current instance works well with image files of high school transcripts, but a more general application would extract information from other image layouts, thus adding depth. Specifically, the current program is designed to pull specific entities from within a <text> node in an XML document, but a more general application might let the user specify which entities to extract and use these entity names throughout the program in places like the data field

editor. The XSLT template editor could use this design feature as well. Finally, breadth could be added to the project by retrieving information from other data sources like Microsoft Office documents and e-mail.

REFERENCES

[Crampton04]
Crampton, Dennis, "How to…deal with structured and unstructured data," Applications (January 2004).

[Haigh96]
Haigh, Susan, "Optical Character Recognition (OCR) as a Digitization Technology," Network Notes #37, National Library of Canada (November 1996).

[Jasper02]
Jasper, Kim, "AIIM Industry White Paper on Records, Document and Enterprise Content Management for the Public Sector," IBM and Project Consult, Hamburg, 2002.

[Li01]
Li, Q., and Moon, B., "Indexing and querying XML data for regular path expressions," Proceedings of 27th International Conference on Very Large Data Bases (September 2001).

[Lyman03]
Lyman, Peter, and Varian, Hal R, "How Much Information? 2003," http://www.sims.berkeley.edu/research/projects/how-much-info-2003/.

[Kay03]
Kay, Michael. XSLT Programmer's Reference. Wiley Publishing, Indianapolis, Indiana, 2003.

[Sanchez04]
Sanchez, J. A., Proal, C., and Maldonado-Naude, F., "Supporting structured, semi-structured and unstructured data in digital libraries," Proceedings of the Mexican International Conference on Computer Science (ENC 04, Colima, Mexico).

[Weglarz04]
Weglarz, Geoffery, "Two Worlds of Data - Structured and Unstructured," DM Review Magazine (September 2004).

APPENDIX A

PROGRAM DEMONSTRATION

This application is built around four distinct user interfaces; an example and description of each follows.

The ImageToXML opening program screen provides the user the ability to select the file(s) from which the entities will be extracted and the XSLT template to be used. Additionally, the user can launch the new template editor and join multiple XML files into one.

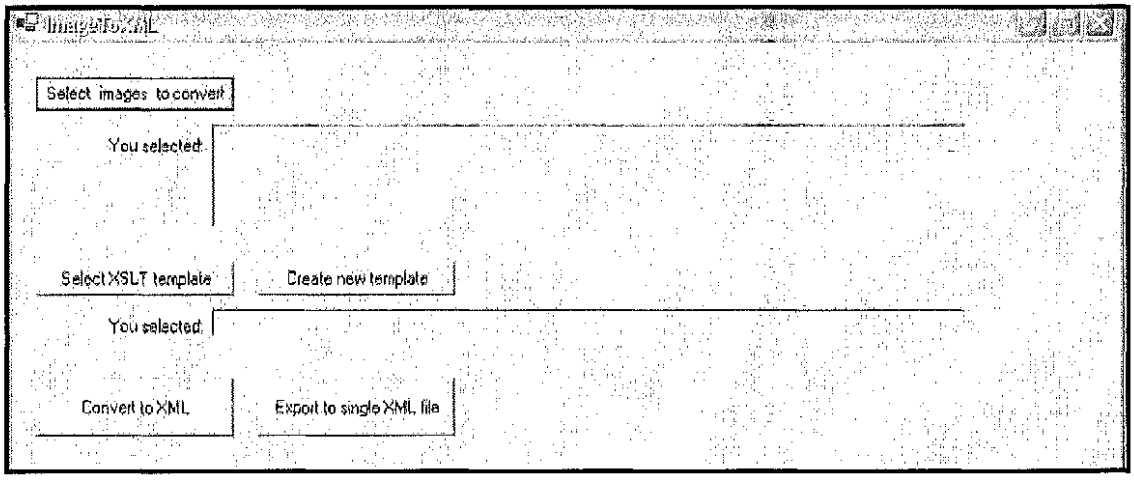


Figure 3: Main Program Screen

Buttons:

"Select images to convert" - opens a Windows Open File Dialog box for selection of input image documents

"Select XSLT template" - opens a Windows Open File Dialog box for selection of XSLT template

"Create new template" - opens the template editor for creating and saving new XSLT templates

"Convert to XML" - starts the processing of all selected files using the XSLT template specified

"Export to single XML file" - exports the individual XML files for each transcript into a single output XML file

After clicking the "Convert to XML" button, the main screen displays the extracted entities in a table as shown in Figure 4:

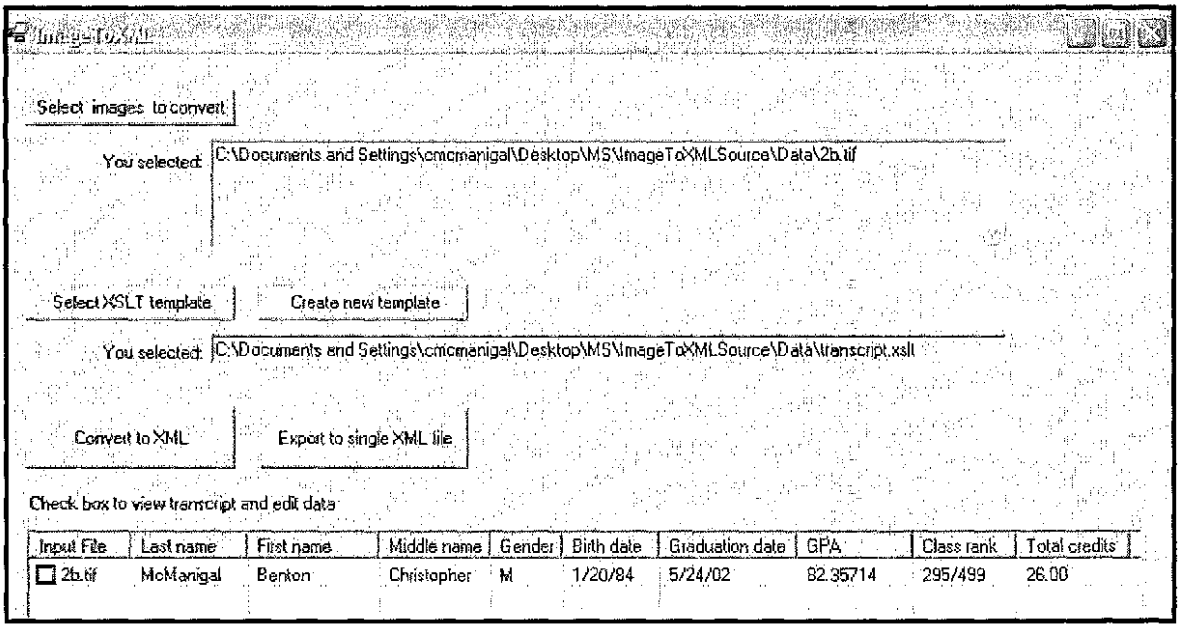


Figure 4: Main Program Screen after Conversion

By placing a check in the box to the left of the student's name, the user causes the data editor window to open as shown in Figure 5:

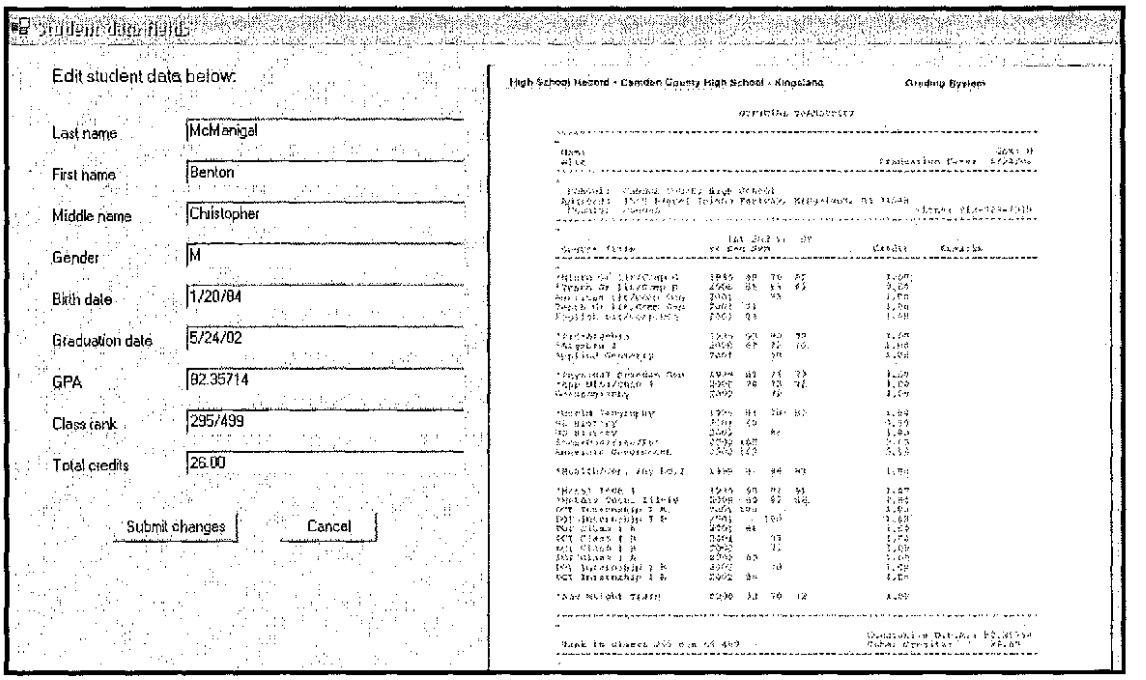


Figure 5: Data Editor Screen

The data editor screen displays each entity in an editable text box. Also, a thumbnail of the related image file is shown. The user can make changes to the entities and save these changes by clicking the "Submit changes" button. Alternatively, the user can cancel back to the main screen.

When the user clicks the "Create new template" button, the template editor is displayed as in Figure 6:

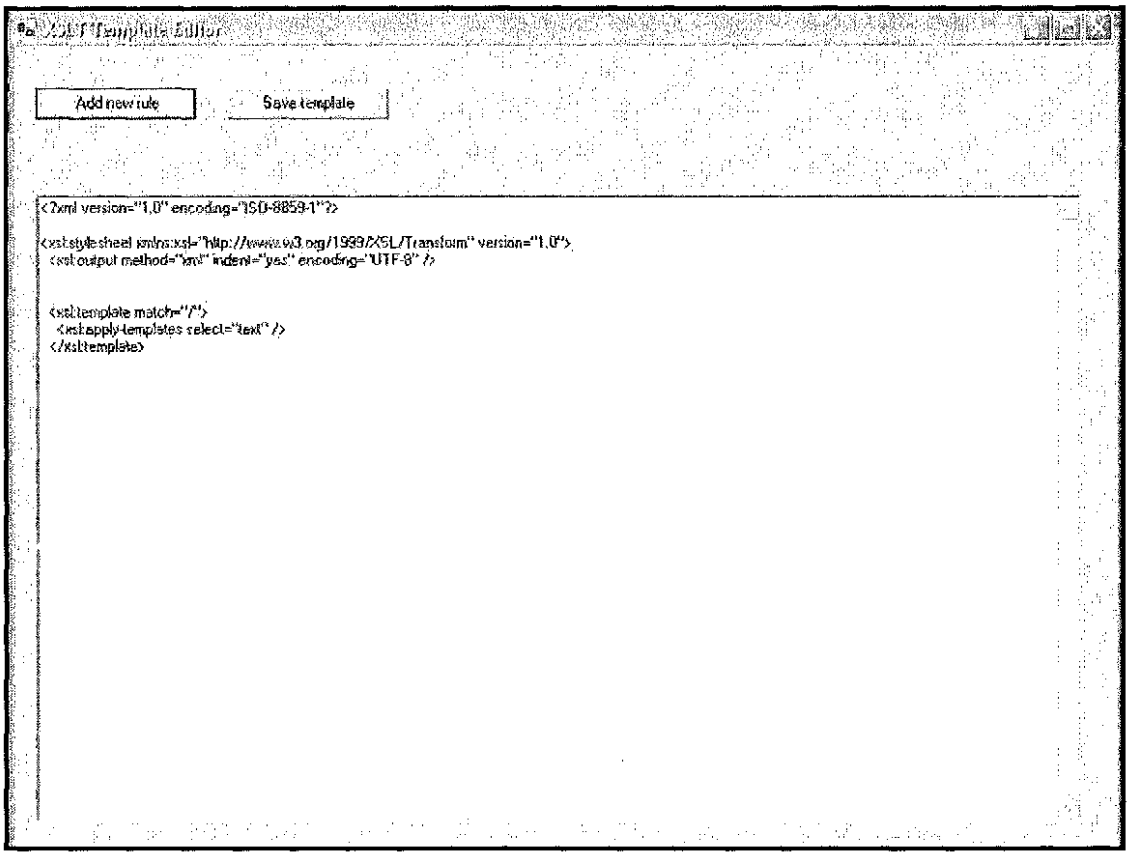


Figure 6: XSLT Template Editor

If experienced with XSLT, the user can create a new template directly in the text box shown. The default header and root template are included. Alternatively, the user can click the "Add rule" button to open the graphical rules editor as shown in Figure 7:

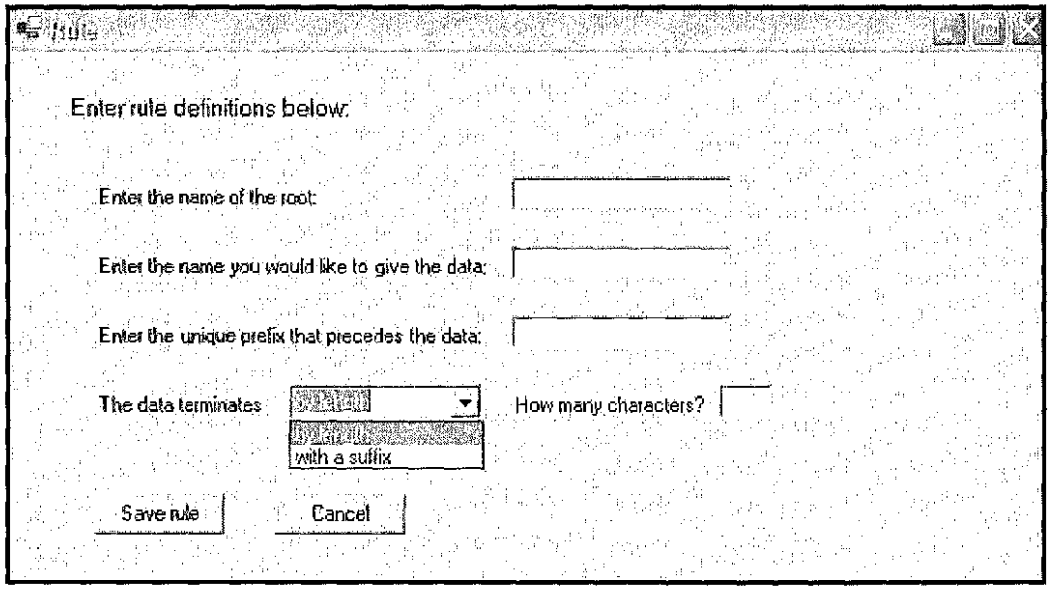


Figure 7: Graphical Rules Editor

One rule can be added at a time, so after having saved the first rule, the user can open the graphical rules editor again. The user must first enter the root element's name; for instance, the user might enter "transcript" when dealing with high school transcripts. The user must next name the entity that needs to be extracted, like "lname" or "gender." The next field requires the unique prefix for the

data in the original document; the user must enter the string that comes immediately before the desired data. Finally, the user determines how the data is terminated, either by a certain number of characters or with a suffix string. When the user clicks "Save rule," the XSLT source code is automatically generated. For example, in Figures 8 and 9, the user has chosen to add a rule to extract an entity called "gender" that is preceded by the string "Sex:" and is terminated after two characters:

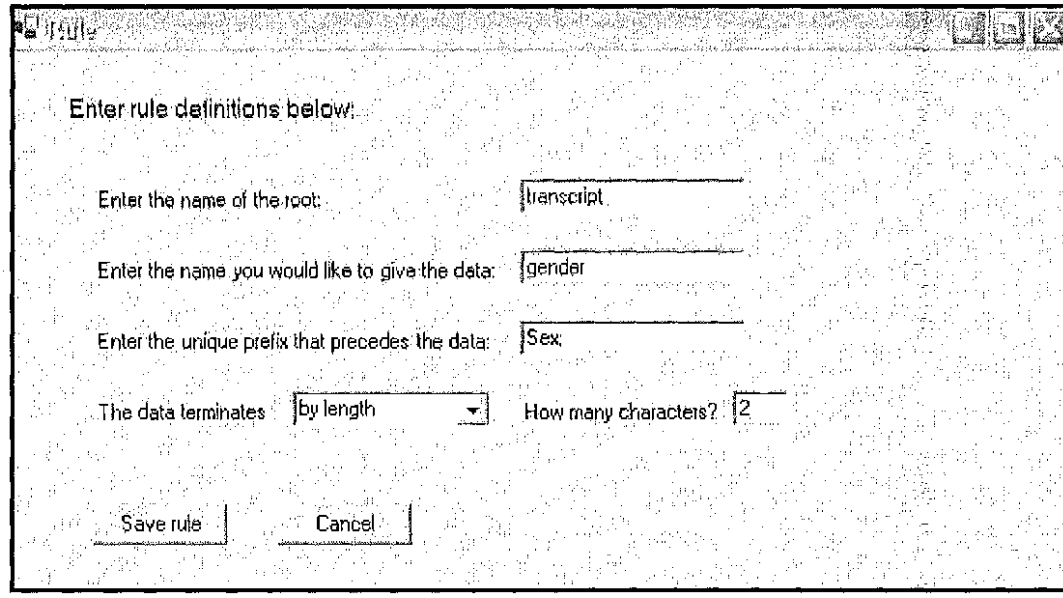


Figure 8: Sample Rule

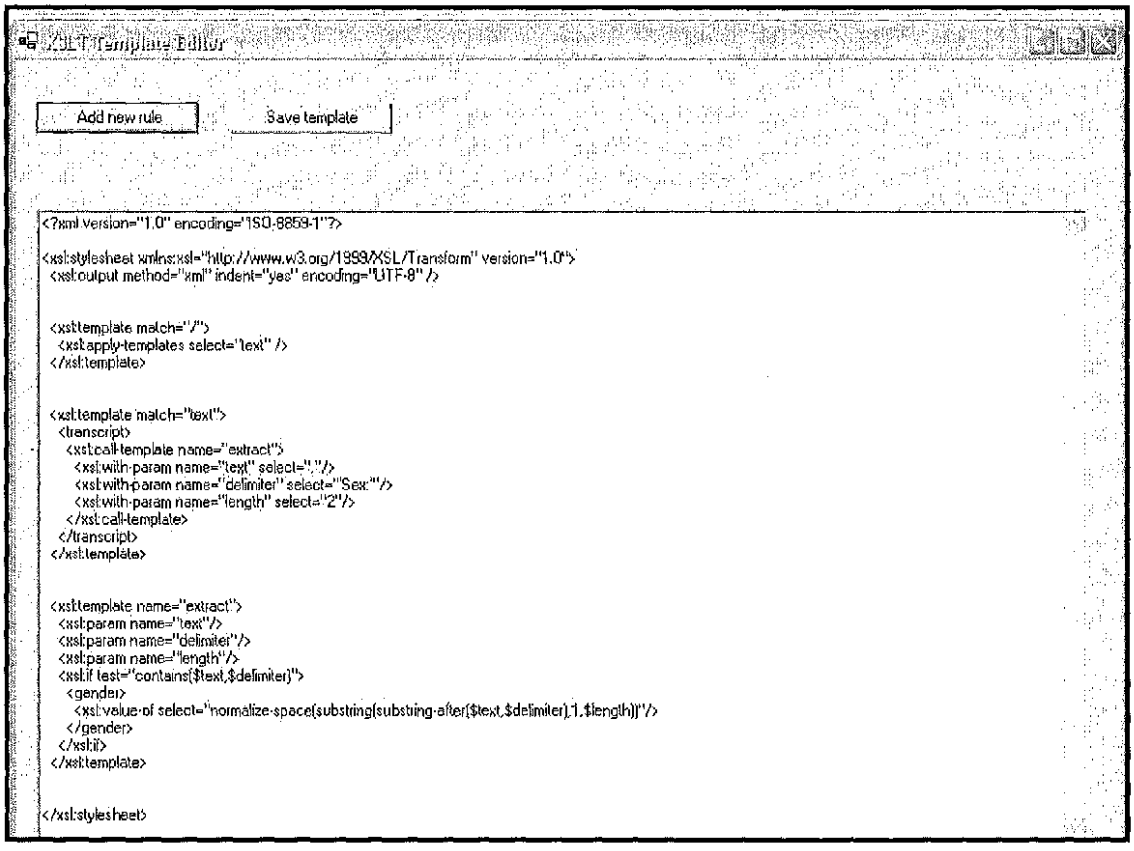


Figure 9: Sample Generated XSLT Code

The "Save template" button opens a Windows Save File Dialog box so the user can save the template for future use.

Finally, the user can click the "Export to single XML file" button to open a Windows Save File Dialog box. All XML files generated in the conversion process can be combined into a single XML file, and then this file can be used as needed by other applications.

APPENDIX B

SOURCE CODE LISTINGS

```
//*********************************************************************
//    Title:            ImageToXML:Form1.cs
//    Author:           Chris A. McManigal
//    Date:       11/01/2005
//    Version:    1.0
//*********************************************************************

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Text;
using System.IO;
using System.Xml;
using IXOCR;

namespace ImageToXML
{
      /// <remarks>
      /// interface that allows user to select files used in
      /// processing; displays data retrieved from image
      /// </remarks>
      public class Form1 : System.Windows.Forms.Form
      {
            #region class variables

            /// <summary>
            /// Required designer variable.
            /// </summary>
            private System.ComponentModel.Container components = null;

            //
            // user-generated class variables
            //
            private System.Windows.Forms.Button button1;
            private System.Windows.Forms.Button button2;
            private System.Windows.Forms.OpenFileDialog
                  openFileDialog1;
            private System.Windows.Forms.TextBox textBox1;
            private System.Windows.Forms.Label label1;
            private System.Windows.Forms.Button button3;
            private System.Windows.Forms.TextBox textBox2;
            private System.Windows.Forms.Label label2;
```

```
private System.Windows.Forms.OpenFileDialog
      openFileDialog2;
private System.Windows.Forms.ListView listView1;
private System.Windows.Forms.ColumnHeader columnHeader1;
private System.Windows.Forms.ColumnHeader columnHeader2;
private System.Windows.Forms.ColumnHeader columnHeader3;
private System.Windows.Forms.ColumnHeader columnHeader4;
private System.Windows.Forms.ColumnHeader columnHeader5;
private System.Windows.Forms.ColumnHeader columnHeader6;
private System.Windows.Forms.ColumnHeader columnHeader7;
private System.Windows.Forms.ColumnHeader columnHeader8;
private System.Windows.Forms.ColumnHeader columnHeader9;
private System.Windows.Forms.ColumnHeader columnHeader10;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Button button5;

private string[] filesChosen;
private string templateName;
private int numFiles;
private TextGenerator myGenerator;
private Extractor myExtractor;
private DataMod[] myDataMod;
private string[] basicXML;
private string[] transformXML;
private ListViewItem[] lvi;
private Form2 editForm;
private Form3 progressForm;
private Form4 templateForm;
private System.Windows.Forms.Button button7;
private System.Windows.Forms.SaveFileDialog
      saveFileDialog1;
private bool initialLoad = true;

#endregion

#region constructors

/// <summary>
/// default constructor
/// </summary>
public Form1()
{
      //
      // Required for Windows Form Designer support
      //
      InitializeComponent();
}

#endregion

/// <summary>
/// cleans up any resources being used
/// </summary>
protected override void Dispose( bool disposing )
{
      if( disposing )
      {
```

```
            if (components != null)
            {
                  components.Dispose();
            }
      }
      base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// required method for Designer support - do not modify
/// the contents of this method with the code editor
/// </summary>
private void InitializeComponent()
{
      this.button1 = new System.Windows.Forms.Button();
      this.openFileDialog1 = new
            System.Windows.Forms.OpenFileDialog();
      this.textBox1 = new System.Windows.Forms.TextBox();
      this.label1 = new System.Windows.Forms.Label();
      this.button2 = new System.Windows.Forms.Button();
      this.button3 = new System.Windows.Forms.Button();
      this.textBox2 = new System.Windows.Forms.TextBox();
      this.label2 = new System.Windows.Forms.Label();
      this.openFileDialog2 = new
            System.Windows.Forms.OpenFileDialog();
      this.listView1 = new System.Windows.Forms.ListView();
      this.columnHeader1 = new
            System.Windows.Forms.ColumnHeader();
      this.columnHeader2 = new
            System.Windows.Forms.ColumnHeader();
      this.columnHeader3 = new
            System.Windows.Forms.ColumnHeader();
      this.columnHeader4 = new
            System.Windows.Forms.ColumnHeader();
      this.columnHeader5 = new
            System.Windows.Forms.ColumnHeader();
      this.columnHeader6 = new
            System.Windows.Forms.ColumnHeader();
      this.columnHeader7 = new
            System.Windows.Forms.ColumnHeader();
      this.columnHeader8 = new
            System.Windows.Forms.ColumnHeader();
      this.columnHeader9 = new
            System.Windows.Forms.ColumnHeader();
      this.columnHeader10 = new
            System.Windows.Forms.ColumnHeader();
      this.label3 = new System.Windows.Forms.Label();
      this.button5 = new System.Windows.Forms.Button();
      this.button7 = new System.Windows.Forms.Button();
      this.saveFileDialog1 = new
            System.Windows.Forms.SaveFileDialog();
      this.SuspendLayout();
      //
      // button1
      //
      this.button1.Location = new
```

```
      System.Drawing.Point(16, 24);
this.button1.Name = "button1";
this.button1.Size = new System.Drawing.Size(136, 23);
this.button1.TabIndex = 2;
this.button1.Text = "Select images to convert:";
this.button1.Click += new
      System.EventHandler(this.button1_Click);
//
// openFileDialog1
//
this.openFileDialog1.Filter = "Tagged image file
      (*.tif; *.tiff)|*.tif;*.tiff|All files
      (*.*)|*.*";
this.openFileDialog1.Multiselect = true;
this.openFileDialog1.Title = "Select image files:";
//
// textBox1
//
this.textBox1.Location = new
      System.Drawing.Point(136, 56);
this.textBox1.Multiline = true;
this.textBox1.Name = "textBox1";
this.textBox1.ReadOnly = true;
this.textBox1.ScrollBars =
      System.Windows.Forms.ScrollBars.Both;
this.textBox1.Size = new
      System.Drawing.Size(520, 72);
this.textBox1.TabIndex = 3;
this.textBox1.Text = "";
//
// label1
//
this.label1.Location = new
      System.Drawing.Point(32, 64);
this.label1.Name = "label1";
this.label1.TabIndex = 4;
this.label1.Text = "You selected:";
this.label1.TextAlign =
      System.Drawing.ContentAlignment.TopRight;
//
// button2
//
this.button2.Location = new
      System.Drawing.Point(16, 232);
this.button2.Name = "button2";
this.button2.Size = new System.Drawing.Size(136, 40);
this.button2.TabIndex = 5;
this.button2.Text = "Convert to XML";
this.button2.Click += new
      System.EventHandler(this.button2_Click);
//
// button3
//
this.button3.Location = new
      System.Drawing.Point(16, 152);
this.button3.Name = "button3";
this.button3.Size = new System.Drawing.Size(136, 23);
```

```
this.button3.TabIndex = 6;
this.button3.Text = "Select XSLT template";
this.button3.Click += new
      System.EventHandler(this.button3_Click);
//
// textBox2
//
this.textBox2.Location = new
      System.Drawing.Point(136, 184);
this.textBox2.Name = "textBox2";
this.textBox2.ReadOnly = true;
this.textBox2.Size = new
      System.Drawing.Size(520, 20);
this.textBox2.TabIndex = 7;
this.textBox2.Text = "";
//
// label2
//
this.label2.Location = new
      System.Drawing.Point(32, 184);
this.label2.Name = "label2";
this.label2.TabIndex = 8;
this.label2.Text = "You selected:";
this.label2.TextAlign =
      System.Drawing.ContentAlignment.MiddleRight;
//
// openFileDialog2
//
this.openFileDialog2.Filter = "XSLT templates (*.xsl;
      *.xslt)|*.xsl; *.xslt;";
this.openFileDialog2.Title = "Select XSLT template:";
//
// listView1
//
this.listView1.CheckBoxes = true;
this.listView1.Columns.AddRange(new
      System.Windows.Forms.ColumnHeader[] {

            this.columnHeader1,
            this.columnHeader2,
            this.columnHeader3,
            this.columnHeader4,
            this.columnHeader5,
            this.columnHeader6,
            this.columnHeader7,
            this.columnHeader8,
            this.columnHeader9,
            this.columnHeader10});
this.listView1.FullRowSelect = true;
this.listView1.GridLines = true;
this.listView1.Location = new
      System.Drawing.Point(16, 312);
this.listView1.MultiSelect = false;
this.listView1.Name = "listView1";
this.listView1.Size = new
      System.Drawing.Size(728, 192);
this.listView1.TabIndex = 11;
```

```
this.listView1.View =
      System.Windows.Forms.View.Details;
this.listView1.Visible = false;
this.listView1.SelectedIndexChanged += new
      System.EventHandler(
      this.listView1_SelectedIndexChanged);
this.listView1.ItemCheck += new
      System.Windows.Forms.ItemCheckEventHandler(
      this.listView1_ItemCheck);
//
// columnHeader1
//
this.columnHeader1.Text = "Input File";
this.columnHeader1.Width = 66;
//
// columnHeader2
//
this.columnHeader2.Text = "Last name";
this.columnHeader2.Width = 76;
//
// columnHeader3
//
this.columnHeader3.Text = "First name";
this.columnHeader3.Width = 86;
//
// columnHeader4
//
this.columnHeader4.Text = "Middle name";
this.columnHeader4.Width = 72;
//
// columnHeader5
//
this.columnHeader5.Text = "Gender";
this.columnHeader5.Width = 47;
//
// columnHeader6
//
this.columnHeader6.Text = "Birth date";
this.columnHeader6.Width = 63;
//
// columnHeader7
//
this.columnHeader7.Text = "Graduation date";
this.columnHeader7.Width = 90;
//
// columnHeader8
//
this.columnHeader8.Text = "GPA";
this.columnHeader8.Width = 75;
//
// columnHeader9
//
this.columnHeader9.Text = "Class rank";
this.columnHeader9.Width = 68;
//
// columnHeader10
//
```

```
this.columnHeader10.Text = "Total credits";
this.columnHeader10.Width = 74;
//
// label3
//
this.label3.Location = new
      System.Drawing.Point(16, 288);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(216, 23);
this.label3.TabIndex = 15;
this.label3.Text = "Check box to view transcript and
      edit data";
this.label3.Visible = false;
//
// button5
//
this.button5.Location = new
      System.Drawing.Point(168, 232);
this.button5.Name = "button5";
this.button5.Size = new System.Drawing.Size(136, 40);
this.button5.TabIndex = 16;
this.button5.Text = "Export to single XML file";
this.button5.Click += new
      System.EventHandler(this.button5_Click);
//
// button7
//
this.button7.Location = new
      System.Drawing.Point(168, 152);
this.button7.Name = "button7";
this.button7.Size = new System.Drawing.Size(136, 23);
this.button7.TabIndex = 23;
this.button7.Text = "Create new template";
this.button7.Click += new
      System.EventHandler(this.button7_Click);
//
// saveFileDialog1
//
this.saveFileDialog1.Filter = "XML files
      (*.xml)|*.xml;";
//
// Form1
//
this.AutoScaleBaseSize = new
      System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(848, 710);
this.Controls.Add(this.button7);
this.Controls.Add(this.button5);
this.Controls.Add(this.label3);
this.Controls.Add(this.listView1);
this.Controls.Add(this.label2);
this.Controls.Add(this.textBox2);
this.Controls.Add(this.textBox1);
this.Controls.Add(this.button3);
this.Controls.Add(this.button2);
this.Controls.Add(this.label1);
this.Controls.Add(this.button1);
```

```
            this.Name = "Form1";
            this.Text = "ImageToXML";
            this.WindowState =
                  System.Windows.Forms.FormWindowState.Maximized;
            this.ResumeLayout(false);

      }
      #endregion

      /// <summary>
      /// the main entry point for the application
      /// </summary>
      [STAThread]
      static void Main()
      {
            Application.Run(new Form1());
      }

      #region public methods

      /// <summary>
      /// updates the ListView after the user makes changes
      /// </summary>
      public void updateForm()
      {
            for (int i = 0; i < this.filesChosen.Length; i++)
            {
                  this.updateLV (i, this.initialLoad);
            }

            this.listView1.Refresh();
      }
      #endregion

      #region form control code

      /// <summary>
      /// selects image files
      /// </summary>
      private void button1_Click(object sender,
            System.EventArgs e)
      {
            string fnames = "";

            openFileDialog1.ShowDialog();

            //
            // gets filenames for display and use
            //
            if( (numFiles = openFileDialog1.FileNames.Length) > 0
}
            {
                  filesChosen = new string[numFiles];
                  int i = 0;

                  foreach( string filename in
                        openFileDialog1.FileNames )
```

```
            {
                  filesChosen[i] = filename;
                  fnames += filesChosen[i] +
                        System.Environment.NewLine;
                  i++;
            }
      }

      textBox1.Text = fnames;
}

/// <summary>
/// starts conversion from image to XML
/// </summary>
private void button2_Click(object sender,
      System.EventArgs e)
{
      int numSteps = 3;
      int numFiles;

      if (filesChosen == null)
      {
            MessageBox.Show("Please select an image file",
                  "Warning", MessageBoxButtons.OK,
                  MessageBoxIcon.Warning);
      }
      else if (templateName == null)
      {
            MessageBox.Show("Please select a template",
                  "Warning", MessageBoxButtons.OK,
                  MessageBoxIcon.Warning);
      }
      else
      {
            this.initialLoad = true;
            numFiles = filesChosen.Length;
            progressForm = new Form3(numFiles * numSteps);

            //
            // creates a TextGenerator to convert images to
            // basic XML files
            //
            myGenerator = new TextGenerator (filesChosen);
            basicXML = new string[numFiles];

            //
            // converts one file at a time to basic XML
            //
            for (int i = 0; i < numFiles; i++)
            {
                  progressForm.step(filesChosen[i]);
                  basicXML[i] = myGenerator.getBasicXML(i,
                        templateName);
            }

            //
            // creates an Extractor to extract entities
```

```
                // from the basic XML file
                // using the given XSLT template
                //
                myExtractor = new Extractor (basicXML,
                      templateName);
                transformXML = new string[numFiles];
                progressForm.setTitle ("Extracting entities . .
                      .");

                //
                // extracts entities, and transforms a basic
                // XML file to final XML file
                //
                for (int i = 0; i < numFiles; i++)
                {
                      progressForm.step(filesChosen[i]);
                      transformXML[i] =
                            myExtractor.applyTemplate(i);

                }

                //
                // creates a DataMod object for each XML file
                // which
                // provides accessors and mutators to the
                // student data
                //
                lvi = new ListViewItem[numFiles];
                myDataMod = new DataMod[numFiles];
                progressForm.setTitle ("Generating XML . . .");

                for (int i = 0; i < numFiles; i++)
                {
                      progressForm.step(filesChosen[i]);

                      myDataMod[i] = new DataMod
                            (transformXML[i]);

                      this.updateLV (i, this.initialLoad);

                }

                this.label3.Visible = true;
                this.listView1.Visible = true;

                this.initialLoad = false;
          }
    }

    /// <summary>
    /// selects XSLT template
    /// </summary>
    private void button3_Click(object sender,
          System.EventArgs e)
    {
          string fname = "";
```

```
        openFileDialog2.ShowDialog();

        //
        // gets filename for display and use
        //
        if( openFileDialog2.FileNames.Length > 0 )
        {
              foreach( string filename in
                    openFileDialog2.FileNames )
              {
                    fname += filename;
              }
        }

        templateName = fname;
        textBox2.Text = fname;
}

/// <summary>
/// determines if ListView item is checked, and opens edit
/// form if checked
/// </summary>
private void listView1_ItemCheck(object sender,
        System.Windows.Forms.ItemCheckEventArgs e)
{
        if(e.NewValue == CheckState.Checked)
        {
              for(int i=0; i< this.filesChosen.Length; i++)
              {
                    if(   i != e.Index)
                          listView1.Items[i].Checked = false;
              }
              editForm = new Form2(this, myDataMod[e.Index]);
              editForm.Show();
        }
}

/// <summary>
/// writes individual XML files to one
/// </summary>
private void button5_Click(object sender,
        System.EventArgs e)
{
        Stream myStream;
        XmlTextWriter writer;

        if(saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
              if((myStream = saveFileDialog1.OpenFile()) !=
                    null)
              {
                    writer = new XmlTextWriter(myStream,
                          Encoding.UTF8);
                    writer.Formatting = Formatting.Indented;

                    writer.WriteStartDocument();
                    writer.WriteStartElement("collection");
```

```
                    // write individual transcript data for
                    // each file
                    for(int i=0; i< this.filesChosen.Length;
                          i++)
                    {

                    this.myDataMod[i].writeElements(writer);

                    writer.WriteWhitespace(Environment.NewLin
                          e + Environment.NewLine);
                    }

                    writer.WriteEndElement();
                    writer.WriteEndDocument();

                    writer.Flush();
                    writer.Close();

                    myStream.Close();

                    MessageBox.Show("File creation
                          completed", "Finished");
              }
        }
}

/// <summary>
/// opens template editor
/// </summary>
private void button7_Click(object sender,
      System.EventArgs e)
{
      templateForm = new Form4();
      templateForm.Show();
}

private void textBox1_TextChanged(object sender,
      System.EventArgs e){}

private void label3_Click(object sender,
      System.EventArgs e){}

private void listView1_SelectedIndexChanged(object sender,
      System.EventArgs e){}

#endregion

#region private methods - helpers
/// <summary>
/// retrieves the input file name from the full path name
/// </summary>
/// <param name="longName">the full path name to the input
/// file</param>
/// <returns>the name of the input file</returns>
private string getFilename (string longName)
{
```

```
            int count;
            string[] fileParts;
            string shortName = "";

            fileParts = longName.Split('\\');
            count = fileParts.Length;
            shortName += fileParts[count - 1];

            return shortName;
        }

        /// <summary>
        /// inserts data into the ListView control
        /// </summary>
        /// <param name="index">the index of the item to
        /// modify</param>
        /// <param name="initialLoad">whether this call is initial
        /// or a subsequent edit</param>
        private void updateLV (int index, bool initialLoad)
        {
            if (!initialLoad)
            {
                // removes data that has been modified
                lvi[index].Remove();
            }

            // inserts new data into ListViewItem
            lvi[index] =
            listView1.Items.Add(getFilename(filesChosen[index]));

        lvi[index].SubItems.Add(myDataMod[index].getLastName());
        lvi[index].SubItems.Add(myDataMod[index].getFirstName());
        lvi[index].SubItems.Add(myDataMod[index].getMiddleName());
        lvi[index].SubItems.Add(myDataMod[index].getGender());
        lvi[index].SubItems.Add(myDataMod[index].getBirthDate());
        lvi[index].SubItems.Add(myDataMod[index].getGradDate());
        lvi[index].SubItems.Add(myDataMod[index].getGPA());
        lvi[index].SubItems.Add(myDataMod[index].getClassRank());
        lvi[index].SubItems.Add(myDataMod[index].getCredits());
        }

        #endregion
    }
}
```

```
//*********************************************************************
//    Title:            ImageToXML:Form2.cs
//    Author:           Chris A. McManigal
//    Date:       11/01/2005
//    Version:    1.0
//*********************************************************************

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Drawing.Imaging;

namespace ImageToXML
{
      /// <remarks>
      /// displays editable data fields as well as thumbnail of image
      /// </remarks>
      public class Form2 : System.Windows.Forms.Form
      {
            #region class variables

            private System.Windows.Forms.TextBox textBox1;
            private System.Windows.Forms.TextBox textBox2;
            private System.Windows.Forms.TextBox textBox3;
            private System.Windows.Forms.TextBox textBox4;
            private System.Windows.Forms.TextBox textBox5;
            private System.Windows.Forms.TextBox textBox6;
            private System.Windows.Forms.TextBox textBox7;
            private System.Windows.Forms.TextBox textBox8;
            private System.Windows.Forms.TextBox textBox9;
            private System.Windows.Forms.Button button1;
            private System.Windows.Forms.Button button2;
            private System.Windows.Forms.Label label1;
            private System.Windows.Forms.Label label2;
            private System.Windows.Forms.Label label3;
            private System.Windows.Forms.Label label4;
            private System.Windows.Forms.Label label5;
            private System.Windows.Forms.Label label6;
            private System.Windows.Forms.Label label7;
            private System.Windows.Forms.Label label8;
            private System.Windows.Forms.Label label9;
            private System.Windows.Forms.Label label10;

            private DataMod myDataMod;
            private System.Windows.Forms.PictureBox pictureBox1;
            private Bitmap transcript;
            private Form1 parent;

            /// <summary>
            /// required designer variable
            /// </summary>
            private System.ComponentModel.Container components = null;

            #endregion
```

```
#region constructors
/// <summary>
/// default constructor
/// </summary>
public Form2()
{
      //
      // required for Windows Form Designer support
      //
      InitializeComponent();
}

/// <summary>
/// overridden constructor to handle Form1 and DataMod
/// </summary>
/// <param name="parent">parent form that called this child
/// form</param>
/// <param name="dm1">class that stores the data for this
/// file</param>
public Form2 (Form1 parent, DataMod dm1)
{
      InitializeComponent();

      this.parent = parent;
      myDataMod = dm1;
}

#endregion

/// <summary>
/// cleans up any resources being used
/// </summary>
protected override void Dispose( bool disposing )
{
      if( disposing )
      {
            if(components != null)
            {
                  components.Dispose();
            }
      }
      base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// required method for Designer support - do not modify
/// the contents of this method with the code editor
/// </summary>
private void InitializeComponent()
{
      this.textBox1 = new System.Windows.Forms.TextBox();
      this.textBox2 = new System.Windows.Forms.TextBox();
      this.textBox3 = new System.Windows.Forms.TextBox();
      this.textBox4 = new System.Windows.Forms.TextBox();
      this.textBox5 = new System.Windows.Forms.TextBox();
      this.textBox6 = new System.Windows.Forms.TextBox();
```

```
this.textBox7 = new System.Windows.Forms.TextBox();
this.textBox8 = new System.Windows.Forms.TextBox();
this.textBox9 = new System.Windows.Forms.TextBox();
this.button1 = new System.Windows.Forms.Button();
this.button2 = new System.Windows.Forms.Button();
this.label1 = new System.Windows.Forms.Label();
this.label2 = new System.Windows.Forms.Label();
this.label3 = new System.Windows.Forms.Label();
this.label4 = new System.Windows.Forms.Label();
this.label5 = new System.Windows.Forms.Label();
this.label6 = new System.Windows.Forms.Label();
this.label7 = new System.Windows.Forms.Label();
this.label8 = new System.Windows.Forms.Label();
this.label9 = new System.Windows.Forms.Label();
this.label10 = new System.Windows.Forms.Label();
this.pictureBox1 = new
      System.Windows.Forms.PictureBox();
this.SuspendLayout();
//
// textBox1
//
this.textBox1.Location = new
      System.Drawing.Point(136, 56);
this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(216,
      20);
this.textBox1.TabIndex = 0;
this.textBox1.Text = "";
//
// textBox2
//
this.textBox2.Location = new
      System.Drawing.Point(136, 88);
this.textBox2.Name = "textBox2";
this.textBox2.Size = new System.Drawing.Size(216,
      20);
this.textBox2.TabIndex = 1;
this.textBox2.Text = "";
//
// textBox3
//
this.textBox3.Location = new
      System.Drawing.Point(136, 120);
this.textBox3.Name = "textBox3";
this.textBox3.Size = new System.Drawing.Size(216,
      20);
this.textBox3.TabIndex = 2;
this.textBox3.Text = "";
//
// textBox4
//
this.textBox4.Location = new
      System.Drawing.Point(136, 152);
this.textBox4.Name = "textBox4";
this.textBox4.Size = new System.Drawing.Size(216,
      20);
this.textBox4.TabIndex = 3;
```

```
this.textBox4.Text = "";
//
// textBox5
//
this.textBox5.Location = new
      System.Drawing.Point(136, 184);
this.textBox5.Name = "textBox5";
this.textBox5.Size = new System.Drawing.Size(216,
      20);
this.textBox5.TabIndex = 4;
this.textBox5.Text = "";
//
// textBox6
//
this.textBox6.Location = new
      System.Drawing.Point(136, 216);
this.textBox6.Name = "textBox6";
this.textBox6.Size = new System.Drawing.Size(216,
      20);
this.textBox6.TabIndex = 5;
this.textBox6.Text = "";
//
// textBox7
//
this.textBox7.Location = new
      System.Drawing.Point(136, 248);
this.textBox7.Name = "textBox7";
this.textBox7.Size = new System.Drawing.Size(216,
      20);
this.textBox7.TabIndex = 6;
this.textBox7.Text = "";
//
// textBox8
//
this.textBox8.Location = new
      System.Drawing.Point(136, 280);
this.textBox8.Name = "textBox8";
this.textBox8.Size = new System.Drawing.Size(216,
      20);
this.textBox8.TabIndex = 7;
this.textBox8.Text = "";
//
// textBox9
//
this.textBox9.Location = new
      System.Drawing.Point(136, 312);
this.textBox9.Name = "textBox9";
this.textBox9.Size = new System.Drawing.Size(216,
      20);
this.textBox9.TabIndex = 8;
this.textBox9.Text = "";
//
// button1
//
this.button1.Location = new System.Drawing.Point(80,
      360);
this.button1.Name = "button1";
```

```
this.button1.Size = new System.Drawing.Size(96, 23);
this.button1.TabIndex = 9;
this.button1.Text = "Submit changes";
this.button1.Click += new
      System.EventHandler(this.button1_Click);
//
// button2
//
this.button2.Location = new System.Drawing.Point(208,
      360);
this.button2.Name = "button2";
this.button2.TabIndex = 10;
this.button2.Text = "Cancel";
this.button2.Click += new
      System.EventHandler(this.button2_Click);
//
// label1
//
this.label1.Location = new System.Drawing.Point(32,
      56);
this.label1.Name = "label1";
this.label1.TabIndex = 11;
this.label1.Text = "Last name";
this.label1.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// label2
//
this.label2.Location = new System.Drawing.Point(32,
      88);
this.label2.Name = "label2";
this.label2.TabIndex = 12;
this.label2.Text = "First name";
this.label2.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// label3
//
this.label3.Location = new System.Drawing.Point(32,
      120);
this.label3.Name = "label3";
this.label3.TabIndex = 13;
this.label3.Text = "Middle name";
this.label3.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// label4
//
this.label4.Location = new System.Drawing.Point(32,
      152);
this.label4.Name = "label4";
this.label4.TabIndex = 14;
this.label4.Text = "Gender";
this.label4.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// label5
```

```
//
this.label5.Location = new System.Drawing.Point(32,
      184);
this.label5.Name = "label5";
this.label5.TabIndex = 15;
this.label5.Text = "Birth date";
this.label5.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// label6
//
this.label6.Location = new System.Drawing.Point(32,
      216);
this.label6.Name = "label6";
this.label6.TabIndex = 16;
this.label6.Text = "Graduation date";
this.label6.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// label7
//
this.label7.Location = new System.Drawing.Point(32,
      248);
this.label7.Name = "label7";
this.label7.TabIndex = 17;
this.label7.Text = "GPA";
this.label7.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// label8
//
this.label8.Location = new System.Drawing.Point(32,
      280);
this.label8.Name = "label8";
this.label8.TabIndex = 18;
this.label8.Text = "Class rank";
this.label8.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// label9
//
this.label9.Location = new System.Drawing.Point(32,
      312);
this.label9.Name = "label9";
this.label9.TabIndex = 19;
this.label9.Text = "Total credits";
this.label9.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// label10
//
this.label10.Font = new
      System.Drawing.Font("Microsoft Sans Serif",
      10F, System.Drawing.FontStyle.Regular,
      System.Drawing.GraphicsUnit.Point,
      ((System.Byte)(0)));
this.label10.Location = new
```

```
            System.Drawing.Point(32, 16);
        this.label10.Name = "label10";
        this.label10.Size = new System.Drawing.Size(160, 23);
        this.label10.TabIndex = 20;
        this.label10.Text = "Edit student data below:";
        //
        // pictureBox1
        //
        this.pictureBox1.Location = new
            System.Drawing.Point(368, 16);
        this.pictureBox1.Name = "pictureBox1";
        this.pictureBox1.Size = new System.Drawing.Size(360,
            416);
        this.pictureBox1.SizeMode =
        System.Windows.Forms.PictureBoxSizeMode.AutoSize;
        this.pictureBox1.TabIndex = 21;
        this.pictureBox1.TabStop = false;
        //
        // Form2
        //
        this.AutoScaleBaseSize = new System.Drawing.Size(5,
            13);
        this.ClientSize = new System.Drawing.Size(1016, 734);
        this.Controls.Add(this.pictureBox1);
        this.Controls.Add(this.label10);
        this.Controls.Add(this.label9);
        this.Controls.Add(this.label8);
        this.Controls.Add(this.label7);
        this.Controls.Add(this.label6);
        this.Controls.Add(this.label5);
        this.Controls.Add(this.label4);
        this.Controls.Add(this.label3);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.button2);
        this.Controls.Add(this.button1);
        this.Controls.Add(this.textBox9);
        this.Controls.Add(this.textBox8);
        this.Controls.Add(this.textBox7);
        this.Controls.Add(this.textBox6);
        this.Controls.Add(this.textBox5);
        this.Controls.Add(this.textBox4);
        this.Controls.Add(this.textBox3);
        this.Controls.Add(this.textBox2);
        this.Controls.Add(this.textBox1);
        this.Name = "Form2";
        this.Text = "Student data fields";
        this.WindowState =
            System.Windows.Forms.FormWindowState.Maximized;
        this.Load += new
            System.EventHandler(this.Form2_Load);
        this.ResumeLayout(false);

}
#endregion

#region form control code
```

```
/// <summary>
/// loads the form with current data
/// </summary>
private void Form2_Load(object sender, System.EventArgs e)
{
      string filename = "";

      textBox1.Text += myDataMod.getLastName();
      textBox2.Text += myDataMod.getFirstName();
      textBox3.Text += myDataMod.getMiddleName();
      textBox4.Text += myDataMod.getGender();
      textBox5.Text += myDataMod.getBirthDate();
      textBox6.Text += myDataMod.getGradDate();
      textBox7.Text += myDataMod.getGPA();
      textBox8.Text += myDataMod.getClassRank();
      textBox9.Text += myDataMod.getCredits();

      filename = myDataMod.getFilename().Replace(".xml",
            ".tif");
      transcript = new Bitmap (filename);
      Image.GetThumbnailImageAbort callback = new
      Image.GetThumbnailImageAbort(ThumbnailCallback);
      pictureBox1.Image =
            transcript.GetThumbnailImage(480,660,
            callback,IntPtr.Zero);
}

/// <summary>
/// button that submits edited info to DataMod for writing
/// to file
/// </summary>
private void button1_Click(object sender,
      System.EventArgs e)
{
      myDataMod.setLastName(textBox1.Text);
      myDataMod.setFirstName(textBox2.Text);
      myDataMod.setMiddleName(textBox3.Text);
      myDataMod.setGender(textBox4.Text);
      myDataMod.setBirthDate(textBox5.Text);
      myDataMod.setGradDate(textBox6.Text);
      myDataMod.setGPA(textBox7.Text);
      myDataMod.setClassRank(textBox8.Text);
      myDataMod.setCredits(textBox9.Text);
      myDataMod.updateXML();

      this.parent.updateForm();
      this.Close();
//    MessageBox.Show("Changes submitted","Result");
}

/// <summary>
/// cancel button that closes the form
/// </summary>
private void button2_Click(object sender,
      System.EventArgs e)
{
```

```
            this.Close();
        }

        /// <summary>
        /// method required as a parameter by GetThumbnailImage()
        ///   method
        /// </summary>
        private bool ThumbnailCallback()
        {
            return false;
        }

        #endregion
    }
}
```

```
//*********************************************************************
//    Title:            ImageToXML:Form3.cs
//    Author:           Chris A. McManigal
//    Date:       11/01/2005
//    Version:    1.0
//*********************************************************************

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace ImageToXML
{
      /// <remarks>
      /// popup progress bar to inform user that processing is
      /// occurring
      /// </remarks>
      public class Form3 : System.Windows.Forms.Form
      {
            #region class variables

            private System.Windows.Forms.ProgressBar progressBar1;
            private System.Windows.Forms.Label label1;
            /// <summary>
            /// required designer variable
            /// </summary>
            private System.ComponentModel.Container components = null;

            #endregion

            #region constructors
            /// <summary>
            /// default constructor
            /// </summary>
            public Form3()
            {
                  //
                  // Required for Windows Form Designer support
                  //
                  InitializeComponent();
            }

            /// <summary>
            /// overridden constructor to handle DataMod
            /// </summary>
            /// <param name="maxSize">max size of the progress
            /// bar</param>
            public Form3 (int maxSize)
            {
                  InitializeComponent();

                  this.progressBar1.Maximum = maxSize;
                  this.progressBar1.Step = 1;
                  this.Show();
            }
```

```
#endregion


/// <summary>
/// cleans up any resources being used
/// </summary>
protected override void Dispose( bool disposing )
{
      if( disposing )
      {
            if(components != null)
            {
                  components.Dispose();
            }
      }
      base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// required method for Designer support - do not modify
/// the contents of this method with the code editor
/// </summary>
private void InitializeComponent()
{
      this.progressBar1 = new
            System.Windows.Forms.ProgressBar();
      this.label1 = new System.Windows.Forms.Label();
      this.SuspendLayout();
      //
      // progressBar1
      //
      this.progressBar1.Location = new
            System.Drawing.Point(16, 64);
      this.progressBar1.Name = "progressBar1";
      this.progressBar1.Size = new System.Drawing.Size(384,
            23);
      this.progressBar1.TabIndex = 0;
      //
      // label1
      //
      this.label1.Location = new System.Drawing.Point(24,
            16);
      this.label1.Name = "label1";
      this.label1.Size = new System.Drawing.Size(376, 40);
      this.label1.TabIndex = 1;
      this.label1.TextAlign =
            System.Drawing.ContentAlignment.MiddleLeft;
      //
      // Form3
      //
      this.AutoScaleBaseSize = new System.Drawing.Size(5,
            13);
      this.ClientSize = new System.Drawing.Size(416, 102);
      this.ControlBox = false;
      this.Controls.Add(this.label1);
```

```
            this.Controls.Add(this.progressBar1);
            this.MaximizeBox = false;
            this.MinimizeBox = false;
            this.Name = "Form3";
            this.StartPosition =
            System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Progress - Performing OCR . . .";
            this.ResumeLayout(false);

        }
        #endregion

        #region form control code

        /// <summary>
        /// public method to advance progress bar based on
        /// processing status
        /// </summary>
        /// <param name="filename">the file in use</param>
        public void step(string filename)
        {
            this.label1.Text = "File: " + filename;
            this.progressBar1.PerformStep();

            if (this.progressBar1.Value ==
                this.progressBar1.Maximum)
            {
                this.Close();
            }
        }

        /// <summary>
        /// sets label to inform user of status
        /// </summary>
        /// <param name="action">the current action being
        /// performed</param>
        public void setTitle (string action)
        {
            this.Text = "Progress - " + action;
        }

        #endregion
    }
}
```

```
//*********************************************************************
//    Title:            ImageToXML:Form4.cs
//    Author:           Chris A. McManigal
//    Date:       11/01/2005
//    Version:    1.0
//*********************************************************************

using System;
using System.IO;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace ImageToXML
{
      /// <summary>
      /// provides methods for creating and saving XSLT text
      /// </summary>
      public class Form4 : System.Windows.Forms.Form
      {
            #region class variables

            private System.Windows.Forms.Button button1;
            private System.Windows.Forms.TextBox textBox1;
            private System.Windows.Forms.Button button2;
            /// <summary>
            /// Required designer variable.
            /// </summary>
            private System.ComponentModel.Container components = null;
            private System.Windows.Forms.SaveFileDialog
                  saveFileDialog1;
            private string xsltBeg = "";
            private string nl = Environment.NewLine;
            private string ind1 = "  ";
            private string ind2 = "    ";
            private string ind3 = "      ";
            private string ind4 = "        ";
            private string ind5 = "          ";
            private Form5 ruleForm;
            private string rootElement = "";
            private string element = "";
            private string prefix = "";
            private string numChars = "";
            private string suffix = "";
            private int ruleCount = 0;
            private bool isLength;
            private string rootElTemp;
            private string[] rule = new string[25]; // defalut to max
                  //25 rules

            #endregion

            #region constructors
            /// <summary>
            /// default constructor
            /// </summary>
```

```
public Form4()
{
	//
	// Required for Windows Form Designer support
	//
	InitializeComponent();

	this.templateInit();
	//
	// add code for rest of rules
	//
	this.textBox1.Text = this.xsltBeg;
}
#endregion

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
	if( disposing )
	{
		if(components != null)
		{
			components.Dispose();
		}
	}
	base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
	this.button1 = new System.Windows.Forms.Button();
	this.textBox1 = new System.Windows.Forms.TextBox();
	this.button2 = new System.Windows.Forms.Button();
	this.saveFileDialog1 = new
		System.Windows.Forms.SaveFileDialog();
	this.SuspendLayout();
	//
	// button1
	//
	this.button1.Location = new System.Drawing.Point(16,
32);
	this.button1.Name = "button1";
	this.button1.Size = new System.Drawing.Size(120, 23);
	this.button1.TabIndex = 0;
	this.button1.Text = "Add new rule";
	this.button1.Click += new
		System.EventHandler(this.button1_Click);
	//
	// textBox1
	//
```

```
            this.textBox1.Location = new System.Drawing.Point(16,
                  112);
            this.textBox1.Multiline = true;
            this.textBox1.Name = "textBox1";
            this.textBox1.ScrollBars =
                  System.Windows.Forms.ScrollBars.Both;
            this.textBox1.Size = new System.Drawing.Size(784,
                  472);
            this.textBox1.TabIndex = 1;
            this.textBox1.Text = "";
            //
            // button2
            //
            this.button2.Location = new System.Drawing.Point(160,
                  32);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(120, 23);
            this.button2.TabIndex = 2;
            this.button2.Text = "Save template";
            this.button2.Click += new
                  System.EventHandler(this.button2_Click);
            //
            // saveFileDialog1
            //
            this.saveFileDialog1.Filter = "XSLT templates
                  (*.xslt)|*.xslt;";
            this.saveFileDialog1.Title = "Save XSLT template
                  as:";
            //
            // Form4
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5,
                  13);
            this.ClientSize = new System.Drawing.Size(824, 606);
            this.Controls.Add(this.button2);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.button1);
            this.Name = "Form4";
            this.Text = "XSLT Template Editor";
            this.ResumeLayout(false);

      }
      #endregion

      #region form control code

      /// <summary>
      /// opens the Rule form to get user input
      /// </summary>
      private void button1_Click(object sender, System.EventArgs
            e)
      {
                  ruleForm = new Form5(this);
                  ruleForm.Show();
      }

      /// <summary>
```

```
/// saves thes text to file and closes the form
/// </summary>
private void button2_Click(object sender, System.EventArgs
      e)
{
      Stream myStream;
      StreamWriter sw;

      if(saveFileDialog1.ShowDialog() == DialogResult.OK)
      {
            if((myStream = saveFileDialog1.OpenFile()) !=
                  null)
            {
                  sw = new StreamWriter(myStream);
                  sw.Write(this.textBox1.Text);
                  sw.Close();
                  myStream.Close();
            }
      }
      this.Close();
}
#endregion

#region public methods

/// <summary>
/// mutator for the name of the root of the data
/// </summary>
/// <param name="root">the name of the root node</param>
public void setRoot (string root)
{
      this.rootElement = root;
}

/// <summary>
/// mutator for the element name of the data
/// </summary>
/// <param name="element">the element name</param>
public void setElement (string element)
{
      this.element = element;
}

/// <summary>
/// mutator for the prefix to the data
/// </summary>
/// <param name="prefix">the string that precedes the
/// data</param>
public void setPrefix (string prefix)
{
      this.prefix = prefix;
}

/// <summary>
///   mutator for the length of the data
/// </summary>
/// <param name="numChars">length of the data</param>
```

```
public void setNumChars (string numChars)
{
      this.numChars = numChars;
      this.isLength = true;
}

/// <summary>
/// mutator for the suffix of the data
/// </summary>
/// <param name="suffix">string after the data</param>
public void setSuffix (string suffix)
{
      this.suffix = suffix;
      this.isLength = false;
}

/// <summary>
/// calls private method to process the new rule
/// </summary>
public void addRule()
{
      this.processRule();
}

/// <summary>
/// accessor to get name of root element
/// </summary>
/// <returns>string containing name of root
/// element</returns>
public string getRoot()
{
      return this.rootElement;
}

#endregion

#region private methods

/// <summary>
/// creates header info for XSLT template
/// </summary>
private void templateInit()
{
      string header, ssBeg, rootTemp;

      header = "<?xml version=\"1.0\" encoding=\"ISO-8859-
            1\"?>";
      ssBeg = "<xsl:stylesheet

      xmlns:xsl=\"http://www.w3.org/1999/XSL/Transform\"
      version=\"1.0\">" + nl + ind1 + "<xsl:output
      method=\"xml\" indent=\"yes\" encoding=\"UTF-8\" />";

      rootTemp = ind1 + "<xsl:template match=\"/\">" + nl +
            ind2 + "<xsl:apply-templates select=\"text\"
            />" + nl + ind1 + "</xsl:template>";
```

```
        this.xsltBeg += header + nl + nl + ssBeg + nl + nl +
              nl + rootTemp;
}

/// <summary>
/// adds new rule to the current XSLT text
/// </summary>
private void processRule()
{
        this.ruleCount++;

        if (ruleCount <=1)
              this.addRootElement();

        this.addRuleTemp();

        this.textBox1.Text = this.xsltBeg + nl + nl + nl +
              this.rootElTemp + nl + nl + nl;

        for (int i = 1; i <= this.ruleCount; i++)
        {
              this.textBox1.Text += this.rule[i] +
                    this.addTestTempEnd() + nl + nl + nl;
        }

        this.textBox1.Text += this.addSSend();
}

/// <summary>
/// adds XSL code for the root element
/// </summary>
private void addRootElement()
{
        string tempBeg, tempEnd, rootBeg, rootEnd, callBeg,
              callEnd, delim, val;

        tempBeg = "<xsl:template match=\"text\">";
        tempEnd = "</xsl:template>";
        rootBeg = "<" + this.rootElement + ">";
        rootEnd = "</" + this.rootElement + ">";
        callBeg = "<xsl:call-template name=\"extract\">" + nl
              + ind4 + "<xsl:with-param name=\"text\"
              select=\".\"/>";
        callEnd = "</xsl:call-template>";
        delim = "<xsl:with-param name=\"delimiter\"
              select=\"'" + this.prefix + "'\"/>";

        if (this.isLength)
              val = "<xsl:with-param name=\"length\"
                    select=\"" + this.numChars + "\"/>";
        else
              val = "<xsl:with-param name=\"suffix\"
                    select=\"'" + this.suffix + "'\"/>";

        this.rootElTemp = ind1 + tempBeg + nl + ind2 +
              rootBeg + nl + ind3 + callBeg + nl + ind4 +
              delim + nl + ind4 + val + nl + ind3 + callEnd +
```

```
            nl + ind2 + rootEnd + nl + ind1 + tempEnd;
}

/// <summary>
/// adds XSL code for the rule template
/// </summary>
private void addRuleTemp()
{
      string tempBeg, testBeg, elemBeg, elemEnd, param,
            rule;
      string tempName = "extract";

      if (this.ruleCount > 1)
      {
            tempName += this.ruleCount;
            this.insertTempCall(tempName);
      }

      tempBeg = "<xsl:template name=\"" + tempName + "\">"
            + nl + ind2 + "<xsl:param name=\"text\"/>" + nl
            + ind2 + "<xsl:param name=\"delimiter\"/>";
      testBeg = "<xsl:if
            test=\"contains($text,$delimiter)\">";
      elemBeg = "<" + this.element + ">";
      elemEnd = "</" + this.element + ">";

      if (this.isLength)
      {
            param = "<xsl:param name=\"length\"/>";
            rule = "<xsl:value-of select=\"normalize-
                  space(substring(substring-
                  after($text,$delimiter),1,$length))\"/>";
      }
      else
      {
            param = "<xsl:param name=\"suffix\"/>";
            rule = "<xsl:value-of select=\"normalize-
                  space(substring-before(substring-
                  after($text,$delimiter),$suffix))\"/>";
      }

      this.rule[this.ruleCount] = ind1 + tempBeg + nl +
            ind2 + param + nl + ind2 + testBeg + nl + ind3
            + elemBeg + nl + ind4 + rule + nl + ind3 +
            elemEnd;
}

/// <summary>
/// adds XSLT code for a template call
/// </summary>
/// <param name="tempName">the name of the template to be
/// called</param>
private void insertTempCall(string tempName)
{
      string insert, callBeg, callEnd, delim, val;

      callBeg = "<xsl:call-template name=\"" + tempName +
```

```
                        "\">" + nl + ind5 + "<xsl:with-param
                        name=\"text\" select=\".\"/>";
                callEnd = "</xsl:call-template>";
                delim = "<xsl:with-param name=\"delimiter\"
                        select=\"'" + this.prefix + "'\"/>";

                if (this.isLength)
                        val = "<xsl:with-param name=\"length\"
                                select=\"" + this.numChars + "\"/>";
                else
                        val = "<xsl:with-param name=\"suffix\"
                                select=\"'" + this.suffix + "'\"/>";

                insert = nl + nl + ind3 + callBeg + nl + ind4 + delim
                        + nl + ind4 + val + nl + ind3 + callEnd;

                this.rule[this.ruleCount - 1] += insert;
        }

        /// <summary>
        /// adds XSLT code to end xsl:if and xsl:template sections
        /// </summary>
        /// <returns>a formatted string containing the end tags for
        /// xsl:if and xsl:template</returns>
        private string addTestTempEnd()
        {
                string testEnd, tempEnd;

                testEnd = "</xsl:if>";
                tempEnd = "</xsl:template>";

                return (nl + ind2 + testEnd + nl + ind1 + tempEnd);
        }


        /// <summary>
        /// adds the XSL stylesheet footer
        /// </summary>
        /// <returns>a string representing the XSL stylesheet
        /// footer</returns>
        private string addSSend()
        {
                return "</xsl:stylesheet>";
        }

        #endregion
    }

}
```

```
//*************************************************************************
//    Title:            ImageToXML:Form5.cs
//    Author:           Chris A. McManigal
//    Date:       11/01/2005
//    Version:    1.0
//*************************************************************************

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace ImageToXML
{
      /// <summary>
      /// provides GUI for user rule building
      /// </summary>
      public class Form5 : System.Windows.Forms.Form
      {
            #region class variables
            private System.Windows.Forms.Label label1;
            private System.Windows.Forms.Label label2;
            private System.Windows.Forms.TextBox textBox1;
            private System.Windows.Forms.Label label3;
            private System.Windows.Forms.TextBox textBox2;
            private System.Windows.Forms.Label label4;
            private System.Windows.Forms.ComboBox comboBox1;
            private System.Windows.Forms.Label label5;
            private System.Windows.Forms.TextBox textBox3;
            private System.Windows.Forms.TextBox textBox4;
            private System.Windows.Forms.Label label6;
            private System.Windows.Forms.Button button1;
            private System.Windows.Forms.Button button2;
            private System.Windows.Forms.TextBox textBox5;
            private System.Windows.Forms.Label label7;

            private Form4 parent;
            private bool isLength;

            /// <summary>
            /// required designer variable
            /// </summary>
            private System.ComponentModel.Container components = null;

            #endregion

            #region constructors
            /// <summary>
            /// default constructor
            /// </summary>
            public Form5()
            {
                  //
                  // required for Windows Form Designer support
                  //
                  InitializeComponent();
```

```
}

/// <summary>
/// overloaded constructor that receives parent info
/// </summary>
/// <param name="parent"></param>
public Form5(Form4 parent)
{
      InitializeComponent();

      this.parent = parent;
      this.textBox5.Text = this.parent.getRoot();
}
#endregion

/// <summary>
/// clean up any resources being used
/// </summary>
protected override void Dispose( bool disposing )
{
      if( disposing )
      {
            if(components != null)
            {
                  components.Dispose();
            }
      }
      base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// required method for Designer support - do not modify
/// the contents of this method with the code editor
/// </summary>
private void InitializeComponent()
{
      this.label1 = new System.Windows.Forms.Label();
      this.label2 = new System.Windows.Forms.Label();
      this.textBox1 = new System.Windows.Forms.TextBox();
      this.label3 = new System.Windows.Forms.Label();
      this.textBox2 = new System.Windows.Forms.TextBox();
      this.label4 = new System.Windows.Forms.Label();
      this.comboBox1 = new System.Windows.Forms.ComboBox();
      this.label5 = new System.Windows.Forms.Label();
      this.textBox3 = new System.Windows.Forms.TextBox();
      this.textBox4 = new System.Windows.Forms.TextBox();
      this.label6 = new System.Windows.Forms.Label();
      this.button1 = new System.Windows.Forms.Button();
      this.button2 = new System.Windows.Forms.Button();
      this.textBox5 = new System.Windows.Forms.TextBox();
      this.label7 = new System.Windows.Forms.Label();
      this.SuspendLayout();
      //
      // label1
      //
      this.label1.Font = new System.Drawing.Font("Microsoft
```

```
      Sans Serif", 10F,
      System.Drawing.FontStyle.Regular,
      System.Drawing.GraphicsUnit.Point,
      ((System.Byte)(0)));
this.label1.Location = new System.Drawing.Point(32,
      24);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(176, 23);
this.label1.TabIndex = 0;
this.label1.Text = "Enter rule definitions below:";
//
// label2
//
this.label2.Location = new System.Drawing.Point(48,
      112);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(240, 23);
this.label2.TabIndex = 1;
this.label2.Text = "Enter the name you would like to
      give the data:";
this.label2.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// textBox1
//
this.textBox1.Location = new
      System.Drawing.Point(288, 112);
this.textBox1.Name = "textBox1";
this.textBox1.Size = new System.Drawing.Size(128,
      20);
this.textBox1.TabIndex = 2;
this.textBox1.Text = "";
//
// label3
//
this.label3.Location = new System.Drawing.Point(48,
      152);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(240, 23);
this.label3.TabIndex = 3;
this.label3.Text = "Enter the unique prefix that
      precedes the data:";
this.label3.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// textBox2
//
this.textBox2.Location = new
      System.Drawing.Point(288, 152);
this.textBox2.Name = "textBox2";
this.textBox2.Size = new System.Drawing.Size(128,
      20);
this.textBox2.TabIndex = 3;
this.textBox2.Text = "";
//
// label4
//
```

```
    this.label4.Location = new System.Drawing.Point(48,
          192);
    this.label4.Name = "label4";
    this.label4.Size = new System.Drawing.Size(112, 23);
    this.label4.TabIndex = 5;
    this.label4.Text = "The data terminates";
    this.label4.TextAlign =
          System.Drawing.ContentAlignment.MiddleLeft;
    //
    // comboBox1
    //
    this.comboBox1.Items.AddRange(new object[] {

"by length",

"with a suffix"});
    this.comboBox1.Location = new
          System.Drawing.Point(160, 192);
    this.comboBox1.Name = "comboBox1";
    this.comboBox1.Size = new System.Drawing.Size(112,
          21);
    this.comboBox1.TabIndex = 4;
    this.comboBox1.Text = "(Choose one)";
    this.comboBox1.SelectedIndexChanged += new
          System.EventHandler(
          this.comboBox1_SelectedIndexChanged);
    //
    // label5
    //
    this.label5.Location = new System.Drawing.Point(288,
          192);
    this.label5.Name = "label5";
    this.label5.Size = new System.Drawing.Size(120, 23);
    this.label5.TabIndex = 7;
    this.label5.Text = "How many characters?";
    this.label5.TextAlign =
          System.Drawing.ContentAlignment.MiddleLeft;
    this.label5.Visible = false;
    //
    // textBox3
    //
    this.textBox3.Location = new
          System.Drawing.Point(408, 192);
    this.textBox3.Name = "textBox3";
    this.textBox3.Size = new System.Drawing.Size(32, 20);
    this.textBox3.TabIndex = 5;
    this.textBox3.Text = "";
    this.textBox3.Visible = false;
    //
    // textBox4
    //
    this.textBox4.Location = new
          System.Drawing.Point(360, 192);
    this.textBox4.Name = "textBox4";
    this.textBox4.Size = new System.Drawing.Size(128,
          20);
    this.textBox4.TabIndex = 6;
```

```
this.textBox4.Text = "";
this.textBox4.Visible = false;
//
// label6
//
this.label6.Location = new System.Drawing.Point(288,
      192);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(72, 23);
this.label6.TabIndex = 9;
this.label6.Text = "Enter suffix:";
this.label6.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
this.label6.Visible = false;
//
// button1
//
this.button1.Location = new System.Drawing.Point(48,
      256);
this.button1.Name = "button1";
this.button1.TabIndex = 7;
this.button1.Text = "Save rule";
this.button1.Click += new
      System.EventHandler(this.button1_Click);
//
// button2
//
this.button2.Location = new System.Drawing.Point(152,
      256);
this.button2.Name = "button2";
this.button2.TabIndex = 8;
this.button2.Text = "Cancel";
this.button2.Click += new
      System.EventHandler(this.button2_Click);
//
// textBox5
//
this.textBox5.Location = new
      System.Drawing.Point(288, 72);
this.textBox5.Name = "textBox5";
this.textBox5.Size = new System.Drawing.Size(128,
      20);
this.textBox5.TabIndex = 1;
this.textBox5.Text = "";
//
// label7
//
this.label7.Location = new System.Drawing.Point(48,
      72);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(240, 23);
this.label7.TabIndex = 13;
this.label7.Text = "Enter the name of the root:";
this.label7.TextAlign =
      System.Drawing.ContentAlignment.MiddleLeft;
//
// Form5
```

```
        //
        this.AutoScaleBaseSize = new System.Drawing.Size(5,
              13);
        this.ClientSize = new System.Drawing.Size(680, 638);
        this.Controls.Add(this.textBox5);
        this.Controls.Add(this.label7);
        this.Controls.Add(this.button2);
        this.Controls.Add(this.button1);
        this.Controls.Add(this.textBox3);
        this.Controls.Add(this.comboBox1);
        this.Controls.Add(this.label4);
        this.Controls.Add(this.textBox2);
        this.Controls.Add(this.label3);
        this.Controls.Add(this.textBox1);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.label5);
        this.Controls.Add(this.textBox4);
        this.Controls.Add(this.label6);
        this.Name = "Form5";
        this.Text = "Rule";
        this.ResumeLayout(false);

}
#endregion

#region form control code

/// <summary>
/// determines which input box to show on form
/// </summary>
private void comboBox1_SelectedIndexChanged(object sender,
      System.EventArgs e)
{
      if (this.comboBox1.SelectedIndex == 0)
      {
            this.isLength = true;
            this.label6.Visible = false;
            this.textBox4.Visible = false;
            this.label5.Visible = true;
            this.textBox3.Visible = true;
      }
      else
      {
            this.isLength = false;
            this.label5.Visible = false;
            this.textBox3.Visible = false;
            this.label6.Visible = true;
            this.textBox4.Visible = true;
      }
}

/// <summary>
/// retrieves values from form, and sends data to parent
/// form for processing
/// </summary>
private void button1_Click(object sender,
```

```
                System.EventArgs e)
        {
                this.parent.setRoot(this.textBox5.Text);
                this.parent.setElement(this.textBox1.Text);
                this.parent.setPrefix(this.textBox2.Text);

                if (this.isLength)
                        this.parent.setNumChars(this.textBox3.Text);
                else
                        this.parent.setSuffix(this.textBox4.Text);

                this.parent.addRule();
                this.Close();
        }

        /// <summary>
        /// closes the form if the user cancels
        /// </summary>
        private void button2_Click(object sender, System.EventArgs
                e)
        {
                this.Close();
        }

        #endregion
    }
}
```

```
//*********************************************************************
//    Title:            ImageToXML:TextGenerator.cs
//    Author:           Chris A. McManigal
//    Date:       11/01/2005
//    Version:    1.0
//*********************************************************************

using System;
using System.Windows.Forms;
using System.IO;
using IXOCR;

namespace ImageToXML
{
      /// <remarks>
      /// uses OCR to convert input image file into output text file
      /// (basic XML)
      /// </remarks>
      public class TextGenerator
      {
            #region class variables
            //
            // user-generated class variables
            //
            private OCR fileOCR;
            private string[] fileArr;
            private int arrIndex;

            #endregion

            #region constructors
            /// <summary>
            /// default constructor
            /// </summary>
            public TextGenerator() { }

            /// <summary>
            /// contructs TextGenerator by storing the list of input
            /// filenames
            /// </summary>
            /// <param name="filesChosen">array of image filenames
            /// chosen for conversion</param>
            public TextGenerator (string[] filesChosen)
            {
                  int i = 0;
                  fileArr = new string[filesChosen.Length];

                  foreach (string filename in filesChosen)
                  {
                        fileArr[i] = filename;
                        i++;
                  }
            }

            #endregion

            #region public methods
```

```
/// <summary>
/// public method that calls private methods to perform
/// OCR, add XML tags,
/// and write the output to a file
/// </summary>
/// <param name="arrIndex">index indicating which file to
/// use</param>
/// <param name="template">name of the XSLT template
/// chosen</param>
/// <returns>a string containing the name of the output XML
/// file</returns>
public string getBasicXML (int arrIndex, string template)
{
      string text = "";
      string basicXML = "";
      string filename = "";

      this.arrIndex = arrIndex;
      text = performOCR();
      basicXML = addBasicXML (text, template);
      filename = writeToFile (basicXML);

      return filename;
}

#endregion

#region private methods

/// <summary>
/// extracts text from the image using MODI OCR
/// </summary>
/// <returns>a string containing text extracted from the
/// image</returns>
private string performOCR()
{
      string text = "";

      fileOCR = new OCR();
      text += fileOCR.getText (fileArr[arrIndex], false);

      return text;
}

/// <summary>
/// adds required tags for XML format, including reference
/// to XSLT template
/// </summary>
/// <param name="text">text extraced in OCR step</param>
/// <param name="template">name of template chosen</param>
/// <returns>a string containing the text from the image
/// plus XML/XSLT tags</returns>
private string addBasicXML (string text, string template)
{
      string basicXML = "";
      string headerXML = "<?xml version=\"1.0\"
```

```
            encoding=\"ISO-8859-1\"?>";
        string headerXSLT = "";
        string extension = "xslt";
        string XSLTname = "";
        string nl = Environment.NewLine;
        string begNode = "<text>";
        string endNode = "</text>";

        XSLTname += getFilename (extension, template);

        headerXSLT += "<?xml-stylesheet type=\"text/xsl\"
            href=\"" + XSLTname + "\"?>";

        basicXML += headerXML + nl + headerXSLT + nl +
            begNode + nl + text + endNode;

        return basicXML;
}

/// <summary>
/// writes text to file with .xml extension
/// </summary>
/// <param name="basicXML">text from the image plus
/// XML/XSLT tags</param>
/// <returns>a string containing the name of the output XML
/// file</returns>
private string writeToFile (string basicXML)
{
        string filename = "";
        string extensionOld = ".tif";
        string extensionNew = "Temp.xml";
        StreamWriter sw;


        // replaces special character
        basicXML = basicXML.Replace ("&", "and");


        // replaces known OCR errors:
        // ":" for " :"
        // "M" for "N" in gender
        // "G" for "C" in "C.P.A."
        // "in" for "In" in "Rank in class"
        // "out of" for "cut of" in class rank
        // "otal Credits"  for "otal Credits"
        basicXML = basicXML.Replace (" :", ":");
        basicXML = basicXML.Replace ("Sex: N", "Sex: M");
        basicXML = basicXML.Replace ("Sex: H", "Sex: M");
        basicXML = basicXML.Replace ("Sex: P", "Sex: F");
        basicXML = basicXML.Replace ("C.P.A.", "G.P.A.");
        basicXML = basicXML.Replace ("Rank In", "Rank in");
        basicXML = basicXML.Replace ("cut of", "out of");
        basicXML = basicXML.Replace ("otal Credits", "otal
            Credits");

        filename = fileArr[arrIndex].Replace(extensionOld,
            extensionNew);
```

```
            if (File.Exists(filename))
            {
                  sw = new StreamWriter(filename);
            }
            else
            {
                  sw = File.CreateText(filename);
            }

            sw.Write(basicXML);
            sw.Close();

            return filename;
      }


      /// <summary>
      /// generates output name given input filename
      /// </summary>
      /// <param name="inputName">name of input file</param>
      /// <param name="extensionNew">output file
      /// extension</param>
      /// <returns>a string containing the output
      /// filename</returns>
      private string getFilename (string extensionNew, string
            inputName)
      {
            int count;
            string[] fileParts;
            string extensionOld = "";
            string outputName = "";

            fileParts = inputName.Split('.');
            count = fileParts.Length;
            extensionOld += fileParts[count - 1];
            outputName += inputName.Replace(extensionOld,
                  extensionNew);

            return outputName;
      }

      #endregion
}
}
```

```
//*********************************************************************
//    Title:            ImageToXML:Extractor.cs
//    Author:           Chris A. McManigal
//    Date:       11/01/2005
//    Version:    1.0
//*********************************************************************

using System;
using System.IO;
using System.Xml;
using System.Xml.Xsl;

namespace ImageToXML
{
      /// <remarks>
      /// extracts entities form the OCR results by applying an XSLT
      /// transformation to the input XML file
      /// </remarks>
      public class Extractor
      {
            #region class variables
            //
            // user-generated class variables
            //
            private string[] fileArr;
            private string XSLTtemplate = "";

            #endregion

            #region constructors

            /// <summary>
            /// default constructor
            /// </summary>
            public Extractor() {}

            /// <summary>
            /// contructs Extractor by storing the list of basic XML
            /// filenames
            /// </summary>
            /// <param name="basicXML">array of basic XML
            /// filenames</param>
            /// <param name="templateName">filename of the XSLT
            /// template</param>
            public Extractor (string[] basicXML, string templateName)
            {
                  int i = 0;
                  fileArr = new string[basicXML.Length];

                  foreach (string filename in basicXML)
                  {
                        fileArr[i] = filename;
                        i++;
                  }

                  XSLTtemplate += templateName;
            }
```

```
#endregion

#region public methods

/// <summary>
/// public method to extract perform the XSLT
/// transformation
/// </summary>
/// <param name="arrIndex">the index of the file to be
/// transformed</param>
/// <returns>a string representing the output file's
/// name</returns>
public string applyTemplate (int arrIndex)
{
      string filename = "";

      filename += transform (fileArr[arrIndex]);

      this.removeTemps(arrIndex);

      return filename;
}

#endregion

#region private methods

/// <summary>
/// applies the XSLT template to the specified XML file
/// </summary>
/// <param name="XMLfile">the name of the XML file to be
/// transformed</param>
/// <returns>a string representing the output file's
/// name</returns>
private string transform (string XMLfile)
{
      string output = "";
      string extensionOld = "Temp.xml";
      string extensionNew = ".xml";
      XmlDocument doc = new XmlDocument();
      XslTransform trans = new XslTransform();

      doc.Load (XMLfile);
      trans.Load (XSLTtemplate);

      output = XMLfile.Replace(extensionOld, extensionNew);

      trans.Transform (XMLfile, output);

      return output;
}

/// <summary>
/// generates output name given input filename
/// </summary>
/// <param name="inputName">name of input file</param>
```

```
        /// <returns>a string containing the output
        /// filename</returns>
        private string getFilename (string inputName)
        {
              string outputName = "";
              string extension = "Temp.xml";
              string extensionNew = ".xml";

              outputName += inputName.Replace(extension,
                    extensionNew);

              return outputName;
        }

        /// <summary>
        /// removes temp files creeated by TextGenerator
        /// </summary>
        /// <param name="arrIndex">the index of the file to be
        /// deleted</param>
        private void removeTemps(int arrIndex)
        {
              File.Delete(fileArr[arrIndex]);
        }

        #endregion
    }
}
```

```
//*********************************************************************
//    Title:            ImageToXML:DataMod.cs
//    Author:           Chris A. McManigal
//    Date:       11/01/2005
//    Version:    1.0
//*********************************************************************

using System;
using System.Xml;
using System.Windows.Forms; // used for debugging
using System.Text;

namespace ImageToXML
{
      /// <remarks>
      /// reads XML file for display, provides accessors and mutators
      /// for entities, and writes changes to XML file
      /// </remarks>
      public class DataMod
      {
            #region class variables
            //
            // user-generated class variables
            //
            private XmlTextReader reader;
            private XmlTextWriter writer;
            private string filename = "";
            private string lname = "";
            private string fname = "";
            private string mname = "";
            private string gender = "";
            private string bmonth = "";
            private string bday = "";
            private string byear = "";
            private string gmonth = "";
            private string gday = "";
            private string gyear = "";
            private string gpa = "";
            private string classrank = "";
            private string classtotal = "";
            private string credits = "";

            #endregion

            #region constructors

            /// <summary>
            /// default constructor
            /// </summary>
            public DataMod(){ }

            /// <summary>
            /// constructor that makes call to read all elements into
            /// local variables
            /// </summary>
            /// <param name="filename">the XML file name</param>
            public DataMod (string filename)
```

```
{
        this.filename = filename;
        reader = new XmlTextReader (this.filename);
        this.readNodes();
}

#endregion

#region public accessors

/// <summary>
/// gets the filename
/// </summary>
/// <returns>the XML filename</returns>
public string getFilename()
{
        return filename;
}

/// <summary>
/// gets the last name
/// </summary>
/// <returns>the student's last name</returns>
public string getLastName()
{
        return lname;
}

/// <summary>
/// gets the first name
/// </summary>
/// <returns>the student's first name</returns>
public string getFirstName()
{
        return fname;
}

/// <summary>
/// gets the middle name
/// </summary>
/// <returns>the student's middle name</returns>
public string getMiddleName()
{
        return mname;
}

/// <summary>
/// gets the gender
/// </summary>
/// <returns>the student's gender</returns>
public string getGender()
{
        return gender;
}

/// <summary>
/// gets the birth date
```

```
/// </summary>
/// <returns>the student's birthdate</returns>
public string getBirthDate()
{
      string result = "";

      if (bmonth.Length > 0 && bday.Length > 0 &&
            byear.Length > 0)
            result += (bmonth + "/" + bday + "/" + byear);

      return result;
}

/// <summary>
/// gets the graduation date
/// </summary>
/// <returns>the student's graduation date</returns>
public string getGradDate()
{
      string result = "";

      if (gmonth.Length > 0 && gday.Length > 0 &&
            gyear.Length > 0)
            result += (gmonth + "/" + gday + "/" + gyear);

      return result;
}

/// <summary>
/// gets the GPA
/// </summary>
/// <returns>the student's GPA</returns>
public string getGPA()
{
      return gpa;
}

/// <summary>
/// gets the class rank
/// </summary>
/// <returns>the student's class rank</returns>
public string getClassRank()
{
      string result = "";

      if (classrank.Length > 0 && classtotal.Length > 0)
            result += (classrank + "/" + classtotal);

      return result;
}

/// <summary>
/// gets the number of credits
/// </summary>
/// <returns>the student's total credits</returns>
public string getCredits()
{
```

```
        return credits;
    }

    #endregion

    #region public mutators

    /// <summary>
    /// sets the last name
    /// </summary>
    /// <param name="newLname">user-edited last name</param>
    public void setLastName (string newLname)
    {
        lname = newLname;
    }

    /// <summary>
    /// sets the first name
    /// </summary>
    /// <param name="newFname">user-edited first name</param>
    public void setFirstName (string newFname)
    {
        fname = newFname;
    }

    /// <summary>
    /// sets the middle name
    /// </summary>
    /// <param name="newMname">user-edited middle name</param>
    public void setMiddleName (string newMname)
    {
        mname = newMname;
    }

    /// <summary>
    /// sets the gender
    /// </summary>
    /// <param name="newGender">user-edited gender</param>
    public void setGender (string newGender)
    {
        gender = newGender;
    }

    /// <summary>
    /// sets the birth date
    /// </summary>
    /// <param name="newBdate">user-edited birth date</param>
    public void setBirthDate (string newBdate)
    {
        string[] fileParts = newBdate.Split('/');

        if (fileParts.Length == 3)
        {
            bmonth = fileParts[0];
            bday = fileParts[1];
            byear = fileParts[2];
        }
```

```
}

/// <summary>
/// sets the graduation date
/// </summary>
/// <param name="newGdate">user-edited graduation
/// date</param>
public void setGradDate (string newGdate)
{
      string[] fileParts = newGdate.Split('/');

      if (fileParts.Length == 3)
      {
            gmonth = fileParts[0];
            gday = fileParts[1];
            gyear = fileParts[2];
      }
}

/// <summary>
/// sets the GPA
/// </summary>
/// <param name="newGPA">user-edited GPA</param>
public void setGPA (string newGPA)
{
      gpa = newGPA;
}

/// <summary>
/// sets the class rank and class total
/// </summary>
/// <param name="newRank">user-edited class rank and class
/// total</param>
public void setClassRank (string newRank)
{
      string[] fileParts = newRank.Split('/');

      if (fileParts.Length == 2)
      {
            classrank = fileParts[0];
            classtotal = fileParts[1];
      }
}

/// <summary>
/// sets the number of credits
/// </summary>
/// <param name="newCredits">user-edited number of
/// credits</param>
public void setCredits (string newCredits)
{
      credits = newCredits;
}

/// <summary>
/// writes the local variables to file
/// </summary>
```

```
public void updateXML()
{
      writer = new XmlTextWriter(this.filename,
            Encoding.UTF8);
      writer.Formatting = Formatting.Indented;

      writer.WriteStartDocument();
      this.writeElements(writer);
      writer.WriteEndDocument();

      writer.Flush();
      writer.Close();
}

/// <summary>
/// writes all the data elements and their values to XML
/// file
/// </summary>
/// <param name="writer">the XML text writer used to send
/// the data to file</param>
public void writeElements(XmlTextWriter writer)
{
      writer.WriteStartElement("transcript");

      writer.WriteStartElement("studentname");
      writer.WriteElementString("last", lname);
      writer.WriteElementString("first", fname);
      writer.WriteElementString("middle", mname);
      writer.WriteEndElement();

      writer.WriteElementString("gender", gender);

      writer.WriteStartElement("birthdate");
      writer.WriteElementString("month", bmonth);
      writer.WriteElementString("day", bday);
      writer.WriteElementString("year", byear);
      writer.WriteEndElement();

      writer.WriteStartElement("graddate");
      writer.WriteElementString("month", gmonth);
      writer.WriteElementString("day", gday);
      writer.WriteElementString("year", gyear);
      writer.WriteEndElement();

      writer.WriteElementString("gpa", gpa);

      writer.WriteStartElement("class");
      writer.WriteElementString("rank", classrank);
      writer.WriteElementString("total", classtotal);
      writer.WriteEndElement();

      writer.WriteElementString("credits", credits);

      writer.WriteEndElement();
}

#endregion
```

```
#region private methods
/// <summary>
/// walks through node tree storing entities in local
/// variables
/// </summary>
private void readNodes ()
{
      int childCount = 0;

      this.reader.MoveToContent();

      while (this.reader.Read())
      {
         if (this.reader.NodeType ==
            XmlNodeType.Element)
         {
           switch (this.reader.LocalName)
           {
             case "studentname":
               while (childCount < 3 && this.reader.Read())
               {
                  if(this.reader.NodeType ==
                        XmlNodeType.Element)
                  {
                     switch(this.reader.LocalName)
                     {
                        case "last":
                        lname += this.reader.ReadString();
                        childCount++;
                        break;
                        case "first":
                        fname += this.reader.ReadString();
                        childCount++;
                        break;
                        case "middle":
                        mname += this.reader.ReadString();
                        childCount++;
                        break;
                        default:
                        break;
                     }
                  }
               }

               childCount = 0;
               break;

               case "gender":
                 gender += this.reader.ReadString();
                 break;
               case "birthdate":
                 while (childCount < 3 && this.reader.Read())
                 {
                    if (this.reader.NodeType ==
                          XmlNodeType.Element)
                    {
```

```
            switch (this.reader.LocalName)
            {
               case "month":
               bmonth += this.reader.ReadString();
               childCount++;
               break;
               case "day":
               bday += this.reader.ReadString();
               childCount++;
               break;
               case "year":
               byear += this.reader.ReadString();
               childCount++;
               break;
               default:
               break;
            }
         }
      }
      childCount = 0;
      break;
   case "graddate":
      while (childCount < 3 && this.reader.Read())
      {
         if (this.reader.NodeType ==
               XmlNodeType.Element)
         {
            switch (this.reader.LocalName)
            {
               case "month":
               gmonth += this.reader.ReadString();
               childCount++;
               break;
               case "day":
               gday += this.reader.ReadString();
               childCount++;
               break;
               case "year":
               gyear += this.reader.ReadString();
               childCount++;
               break;
               default:
               break;
            }
         }
      }
      childCount = 0;
      break;
   case "gpa":
         gpa += this.reader.ReadString();
         break;
   case "class":
      while (childCount < 2 && this.reader.Read())
      {
         if (this.reader.NodeType ==
               XmlNodeType.Element)
         {
```

```
                                switch (this.reader.LocalName)
                                {
                                    case "rank":
                                    classrank +=
                                          this.reader.ReadString();
                                    childCount++;
                                    break;
                                    case "total":
                                    classtotal +=
                                          this.reader.ReadString();
                                    childCount++;
                                    break;
                                    default:
                                    break;
                                }
                            }
                        }
                        childCount = 0;
                        break;
                    case "credits":
                        credits += this.reader.ReadString();
                        break;
                    default:
                        break;
                }
            }
        }

            this.reader.Close();
        }

        #endregion
    }
}
```

```
//*******************************************************
// Default XSLT template
//*******************************************************

<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:output method="xml" indent="yes" encoding="UTF-8" />



  <xsl:template match="/">
    <xsl:apply-templates select="text" />
  </xsl:template>



  <xsl:template match="text">

    <transcript>
      <xsl:call-template name="extract">
        <xsl:with-param name="text" select="."/>
        <xsl:with-param name="delimiter" select="'&#58;'"/>
      </xsl:call-template>
    </transcript>

  </xsl:template>



  <xsl:template name="extract">
    <xsl:param name="text"/>
    <xsl:param name="delimiter"/>

    <xsl:choose>
      <xsl:when test="contains($text,$delimiter)">


        <xsl:if test="substring(substring-before($text,$delimiter),
                  (string-length(substring-before($text,$delimiter))
                  - string-length('School')) + 1)='School'
             or substring(substring-before($text,$delimiter),
                  (string-length(substring-before($text,$delimiter))
                  - string-length('Address')) + 1)='Address'
             or substring(substring-before($text,$delimiter),
                  (string-length(substring-before($text,$delimiter))
                  - string-length('County')) + 1)='County'
             or substring(substring-before($text,$delimiter),
                  (string-length(substring-before($text,$delimiter))
                  -  string-length('Phone')) + 1)='Phone'
             ">
          <xsl:call-template name="extract">
            <xsl:with-param name="text" select="substring-
                  after($text,$delimiter)"/>
            <xsl:with-param name="delimiter" select="$delimiter"/>
```

```
  </xsl:call-template>
</xsl:if>


<xsl:if test="substring(substring-before($text,$delimiter),
    (string-length(substring-before($text,$delimiter))
    - string-length('Name')) + 1)='Name'">
  <studentname>
    <xsl:call-template name="getname">
    <xsl:with-param name="fullname" select="normalize-
          space(substring-after($text,$delimiter))"/>
    </xsl:call-template>
  </studentname>

  <xsl:call-template name="extract">
    <xsl:with-param name="text" select="substring-
          after($text,$delimiter)"/>
    <xsl:with-param name="delimiter" select="$delimiter"/>
  </xsl:call-template>
</xsl:if>


<xsl:if test="substring(substring-before($text,$delimiter),
    (string-length(substring-before($text,$delimiter))
    - string-length('Sex')) + 1)='Sex'">
  <gender>
    <xsl:value-of select="normalize-space(substring(substring-
          after($text,$delimiter),1,2))"/>
  </gender>

  <xsl:call-template name="extract">
    <xsl:with-param name="text" select="substring-
          after($text,$delimiter)"/>
    <xsl:with-param name="delimiter" select="$delimiter"/>
  </xsl:call-template>
</xsl:if>


<xsl:if test="substring(substring-before($text,$delimiter),
    (string-length(substring-before($text,$delimiter))
    - string-length('Birth Date')) + 1)='Birth Date'">
  <birthdate>
    <xsl:call-template name="getdate">
      <xsl:with-param name="fulldate"
          select="normalize-space(substring(substring-
                after($text,$delimiter),1,9))"/>
    </xsl:call-template>
  </birthdate>

  <xsl:call-template name="extract">
    <xsl:with-param name="text" select="substring-
          after($text,$delimiter)"/>
    <xsl:with-param name="delimiter" select="$delimiter"/>
  </xsl:call-template>
</xsl:if>
```

```
<xsl:if test="substring(substring-before($text,$delimiter),
    (string-length(substring-before($text,$delimiter))
    - string-length('Graduation Date')) + 1)='Graduation
    Date'">
  <graddate>
    <xsl:call-template name="getdate">
      <xsl:with-param name="fulldate"
          select="normalize-space(substring(substring-
                after($text,$delimiter),1,9))"/>
    </xsl:call-template>
  </graddate>

  <xsl:call-template name="extract">
    <xsl:with-param name="text" select="substring-
          after($text,$delimiter)"/>
    <xsl:with-param name="delimiter" select="$delimiter"/>
  </xsl:call-template>
</xsl:if>


<xsl:if test="substring(substring-before($text,$delimiter),
    (string-length(substring-before($text,$delimiter))
    - string-length('G.P.A.')) + 1)='G.P.A.'">
  <gpa>
    <xsl:value-of select="normalize-space(substring(substring-
          after($text,$delimiter),1,9))"/>
  </gpa>

  <xsl:call-template name="extract">
    <xsl:with-param name="text" select="substring-
          after($text,$delimiter)"/>
    <xsl:with-param name="delimiter" select="$delimiter"/>
  </xsl:call-template>
</xsl:if>


<xsl:if test="substring(substring-before($text,$delimiter),
    (string-length(substring-before($text,$delimiter))
    - string-length('class')) + 1)='class'">
  <class>
    <xsl:call-template name="getclassrank">
      <xsl:with-param name="classrank"
        select="normalize-space(substring(substring-
          after($text,$delimiter),1,15))"/>

    </xsl:call-template>
  </class>

  <xsl:call-template name="extract">
    <xsl:with-param name="text" select="substring-
          after($text,$delimiter)"/>
    <xsl:with-param name="delimiter" select="$delimiter"/>
  </xsl:call-template>
</xsl:if>
```

```
      <xsl:if test="substring(substring-before($text,$delimiter),
          (string-length(substring-before($text,$delimiter))
          - string-length('Credits')) + 1)='Credits'">
        <credits>
          <xsl:value-of select="normalize-space(substring(substring-
                after($text,$delimiter),1,6))"/>
        </credits>

        <xsl:call-template name="extract">
          <xsl:with-param name="text" select="substring-
                after($text,$delimiter)"/>
          <xsl:with-param name="delimiter" select="$delimiter"/>
        </xsl:call-template>
      </xsl:if>

    </xsl:when>

    <xsl:otherwise/>

  </xsl:choose>
</xsl:template>



<xsl:template name="getname">
  <xsl:param name="fullname"/>
  <xsl:variable name="comma" select="'&#44;'"/>
  <xsl:variable name="space" select="'&#32;'"/>

  <last><xsl:value-of select="normalize-space(substring-
    before($fullname,$comma))"/></last>
  <first><xsl:value-of select="normalize-space(substring-
    before(normalize-space(substring-
          after($fullname,$comma)),$space))"/></first>
  <middle><xsl:value-of select="normalize-space(substring-
    before(substring-after(normalize-space(substring-
          after($fullname,$comma)),$space),$space))"/></middle>

</xsl:template>



<xsl:template name="getclassrank">
  <xsl:param name="classrank"/>
  <xsl:variable name="sep" select="'out of'"/>
  <xsl:variable name="space" select="'&#32;'"/>

  <rank><xsl:value-of select="normalize-space(substring-
    before($classrank,$sep))"/></rank>
  <total><xsl:value-of select="normalize-space(substring(normalize-
    space(substring-after($classrank,$sep)),1,3))"/></total>

</xsl:template>
```

```
  <xsl:template name="getdate">
    <xsl:param name="fulldate"/>
    <xsl:variable name="delimiter" select="'&#47;'"/>

    <month><xsl:value-of select="normalize-space(substring-
      before($fulldate,$delimiter))"/></month>
    <day><xsl:value-of select="normalize-space(substring-
      before(substring-
            after($fulldate,$delimiter),$delimiter))"/></day>
    <year><xsl:value-of select="normalize-space(substring-
      after(substring-
            after($fulldate,$delimiter),$delimiter))"/></year>

  </xsl:template>

</xsl:stylesheet>
```

VITA

Chris A. McManigal has a Bachelor of Mechanical Engineering degree from the Georgia Institute of Technology, 1991, and expects to receive a Master of Science in Computer and Information Sciences from the University of North Florida, December 2005. Dr. Sherif Elfayoumy served as Chris' project director. After having served five years as a nuclear-trained submarine officer in the U.S. Navy and two years as a missile systems engineer for Lockheed Martin Missiles and Space, Chris is currently employed as a student information systems specialist with Camden County Schools in Kingsland, Georgia. He works primarily with multiple proprietary student information systems using dBASE IV, Microsoft SQL Server, and Sybase databases and their associated SQL querying utilities. He also uses Visual C#, Java, and ASP to develop web applications for the school system.