CS4624 – Multimedia, Hypertext, and Information Access
Virginia Tech, Blacksburg, VA
5/08/2014

Detecting and correcting coding errors in ETD-db database

Client: Zhiwu Xie (zhiwuxie@vt.edu)

Created by:
Ryan Mestre
Luke Schmader

Table of Contents

Executive Summary

Detecting and correcting coding errors in the Electronic Thesis and Dissertations Database, or ETDdbErrors for short, is a Java based application created primarily for XML file editing and debugging. ETDdbErrors was inspired by the need to fix XML errors in the Virginia Tech Thesis and Dissertation Library Database, where many documents required significant editing to be valid XML and still had visual display bugs. This application's purpose is to permit the user to easily debug and edit XML files so that they can be correctly displayed.

ETDdbErrors is Java based so that it can be used across a wider array of platforms and systems. The application uses a configuration file, a parser, an output file, a log file, and an input file. The user can edit the configuration file to specify what kind of errors he is encountering, and what solution options he wants for each find of error. Users may specify any number of solutions, and if there is more than 1, the program will display the context of the found error, and prompt you for a decision on which solution to implement. The application also automatically removes control characters that are illegal in XML. This allows users of ETDdbErrors to quickly and efficiently edit and fix problematic XML files.

ETDdbErrors also has a secondary version, called VT Thesis and Dissertation Database Parser, or VTTDP for short. This is an older version of

ETDdbErrors that was made specifically for the VT Thesis and Dissertation Database. This application is not interactive and not usable beyond fixing the errors specific to the VT Thesis and Dissertation Database, but provides a faster and more targeted fix to the problems encountered there.

User Manual

In order to effectively use ETD Database Fixer you must understand how to manage the configuration file. The manual will guide you through set up of ETD Database Fixer and instruct you on the correct use and syntax required to adequately use this application.

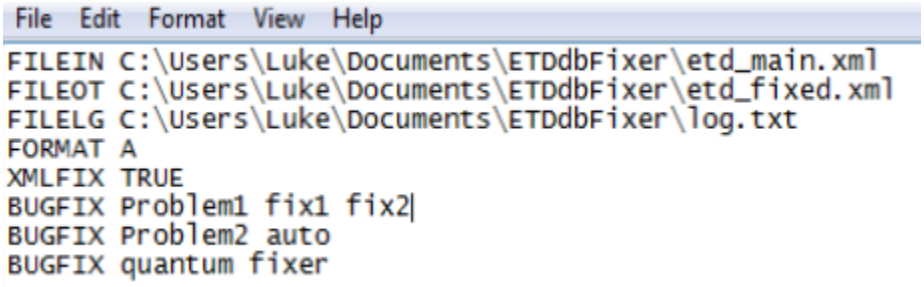
Set Up

1. Place xmlFixer in a desired work directory.
2. Create xmlFixerConfig.txt and place it in the same directory as xmlFixer.
3. Ensure you have a working Java JDK or JRE installed, you can download here http://www.oracle.com/technetwork/java/javase/downloads/index.html?ssSourceSiteId=otnjp
4. Open your command line at the directory of xmlFixer
5. Compile xmlFixer using the “javac xmlFixer” command, this should create xmlFixer.class

Configuration File

The most important part of the program is the configuration file. Here you can specify what changes and fixes you want to apply to your file. The configuration file is a simple list of commands that xmlFixer reads and adjusts its

functionality accordingly. An example of a configuration file and explanation of each command follows.



Commands

All commands should be in uppercase.

* - all commands with * are required

FILEIN*

This command is the input file that XMLFixer will edit.

Syntax - FILEIN (FilePath)

Example- FILEIN C:\Users\JohnSmith\Documents\XMLFixer\input.xml

FILEOUT*

This command is the output file XMLFixer will write to. If this file exists it will be overwritten, elsewise it will be created.

Syntaxt - FILEOUT (FilePath)

Example- FILEOUT C:\Users\JohnSmith\Documents\XMLFixer\output.xml

FILEELG*

This command is the log file where XMLFixer will record changes it makes. If this file exists it will be overwritten, elsewise it will be created.

Syntax - FILEELG (FilePath)

Example- FILEELG C:\Users\JohnSmith\Documents\XMLFixer\error.txt

FORMAT

This command specifies the encoding of the input file. Defaults to UTF-8 if unspecified.

Syntax - FORMAT (option)

Options are:

A - Specifies UTF-8 encoding

B - Specifies ISO-8859-1 encoding

C - Specifies ANSI encoding

Example - FORMAT C

XMLFIX

This command specifies whether the XMLFIX setting is on or off. If it is on then XMLFixer will automatically detect illegal XML characters and prompt you to fix them. This defaults to TRUE.

Syntax - XMLFIX (option)

Options are:

TRUE - turns XMLFIX on

FALSE - turns XMLFIX off

Example - XMLFIX TRUE

BUGFIX

This command specifies a sequence of characters to be considered errors if detected. When XMLFixer finds one of these sequences, it responds according to the options you give it. If a single option is specified, XMLFixer will automatically implement the chosen fix, if multiple are specified, it will prompt you for a decision.

Syntax - BUGFIX (error) (option1) (option2) (option3) ….

Error - the string that, if found, will prompt this bugfix

Options are:

String - any sequence of characters, this option will tell the program to let you replace Error with this sequence of characters

! - this option tell the program to delete error if chosen.

Example - BUGFIX lo*p loop lop !

QUIT

This command exits the programs, all changes made before QUIT is executed remain.

Syntax - QUIT

Example- QUIT

Running the program

Once set up and the configuration file have been completed, open the command line in the directory of xmlFixer.class and enter the command “java xmlFixer”. This will run the program, it will prompt you for input. Follow the instructions until editing is finished.

Developer's Manual

The final version of our program is in xmlFixer.java. It is the interactive program that allows users to define their own errors and submit changes for errors that are detected. The program's main function reads the configuration file (xmlFixerConfig.txt) and is what will detect the errors and read the input from the user. If the option is selected, it is also what will detect any illegal characters for XML files and allow the user to enter changes. The helper function called "caseEvaluation" determines how user defined errors are handled. As of now, if there is only one suggestion on how to fix a user-defined error, a replacement will automatically be made for the error.

Inventory Of Files:

ErrorFinder.java:

This file contains the source code for the program that assists in the identification of patterns found associated with each error.

FileParser.java:

This file contains the source code for the non-interactive program that fixes errors found in the file based off of observed patterns found for each error.

xmlFixer.java:

This file contains the source code for the interactive application that allows users to manually choose the corrections at each error found in the file.

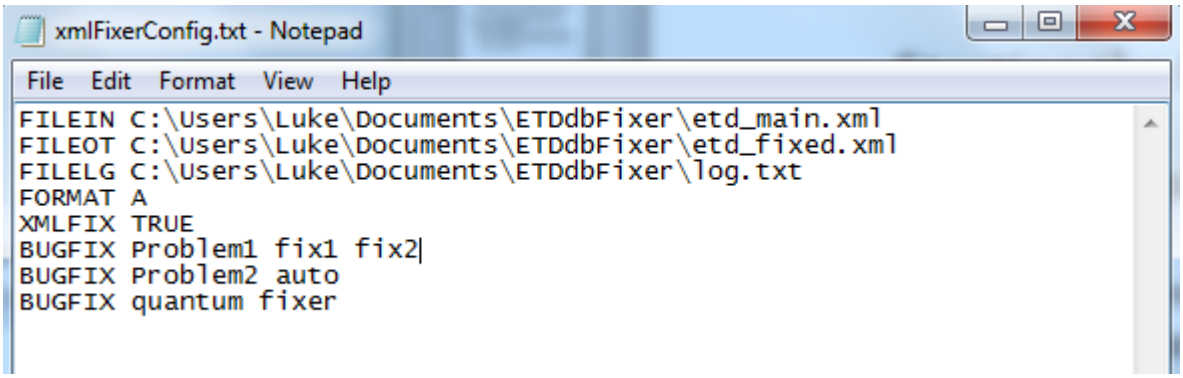
xmlFixerConfig.txt

This is the configuration file for xmlFixer. It specifies the input file, output file, log file, format of the input file, whether or not to detect control characters, and user-defined errors.

log.txt

This file contains descriptions of all the changes made by xmlFixer. Each line includes what line was changed, what the error was, and what it was replaced by.

Configuration File Example

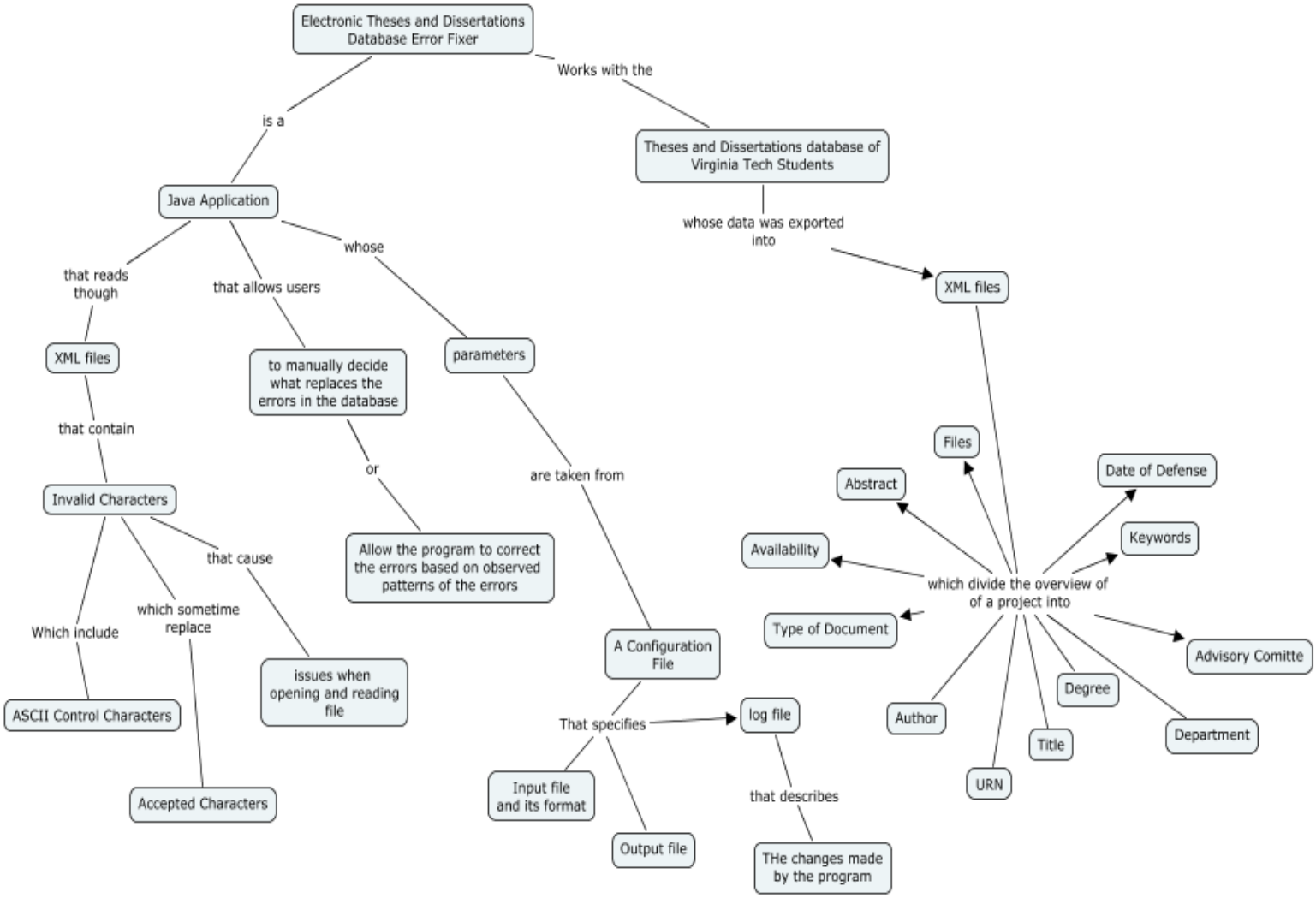


Example Console

```
ERROR IN LINE: 63566 at index: 260 with value: quantum
ries of relativity, quantum mechanics, c
Enter string to replace error or enter "!" to delete it. The choices you defined are:
1. fixer
AUTOMATIC REPLACEMENT: fixer

ERROR IN LINE: 86656 at index: 73 with ascii value: 12
and Wind Tunnel Veri@cation of a Morphin
Enter string to replace error or enter "!" to delete it
```

Concept Map


Electronic Theses and Dissertations Database Error Fixer
Works with the
is a
Theses and Dissertations database of Virginia Tech Students
Java Application
whose data was exported into
that reads though
that allows users
whose
XML files
XML files
to manually decide what replaces the errors in the database
parameters
Files
Abstract
Date of Defense
that contain
or
are taken from
Keywords
Availability
Invalid Characters
Allow the program to correct the errors based on observed patterns of the errors
which divide the overview of of a project into
that cause
Type of Document
Advisory Comitte
Which include
which sometime replace
A Configuration File
Degree
issues when opening and reading file
Author
Department
ASCII Control Characters
That specifies
log file
Title
Input file and its format
URN
that describes
Accepted Characters
Output file
THe changes made by the program

Lessons Learned

This project in its entirety has been a learning process for us. We (Luke Schmader and Ryan Mestre) began this project in a completely different group with a different client. Unfortunately, that client had to cancel the project, and the 6 people in that group had to find a new task. We chose ETDdbErrors and had to move quickly since we were already a 2 weeks behind.

The first step of ETDdbErrors was meeting with our client, Zhiwu Xie. Our initial meeting was rather complicated, since we had to understand the entire project goal in one hour, and were completely unfamiliar with most of the topics and tools of this project. This was made worse because it was clear that Zhiwu was not crystal clear on what he wanted, and that most of the project would be exploring, thinking, and testing in order to together find out what the best approach to take would be.

After our initial meeting we devised a timeline so that we could better manage our time and handle the project responsibly. The timeline was as follows:

- 1st Milestone, Feb 26th: Create Linux VM, configure database software, and set up database.
- MIDTERM PRESENTATION (March 3rd)
- 2nd Milestone, March 12th: Identify all errors present in database.
- 3rd Milestone, March 26th: Fix error related to non-latin alphabet characters
- 4th Milestone, April 12th: Fix other unknown errors.

- 4.5 Milestone, Program to fix Database errors is complete. Client is reviewing code. If code is adequate then we may modify program as stated in 5th milestone. Elsewise, reevaluate Milstone 2,3,4.
- 5th Milestone, April 26th: Modify program to be usable for other applications
- Final Presentation (May 6th)

With our timeline and objectives we got to work, and everything was going great all the way until the midterm presentation. A few days after the midterm presentation we were very ahead of schedule and had reached 4.5 milestone.

This is where our trouble began, around March 25 we contacted our client with our current status, and asking for his feedback so that we could progress to the final stages of the project. He never answered. We emailed him frequently for around 4 weeks , he constantly said he would get back to us and never did, or simply did not answer.

After speaking with the professor, we managed to get into contact with him, and were able to progress, even though our time and ability to improve the program was severely hindered. This is why the biggest lesson we learned was to make sure you have open channels of communication with your client, and understand what he wants from the beginning. If these aspects were better handled, we might of created a much better program.

Acknowledgements

Ryan Mestre
Developer/Designer
mryan8@vt.edu

Luke Schmader
Developer/Designer
lukes4@vt.edu

Zhiwu Xie
Client
zhiwuxie@vt.edu

Ed Fox
Professor/ Advisor
fox@vt.edu

References

Author. 2013. Electronic Theses and Dissertations at VT. (November 2013). Retrieved May 6, 2014 from http://scholar.lib.vt.edu/theses/browse/.