Universal Extraction Tool

Project Engineer: Turki AlHarbi

Major: Computer Science

Project Advisor/Sponsor: Dr. Donald Roberts

Date: 4/21/2019

University of Evansville

ABSTRACT:

Any organization has many different data inputs. They receive mail, invoices, and many different types of documents. The project aims to help those organizations manage their data by providing a tool to extract useful information. The tool is intended to extract data from any document. It needs to be able to take text files and allow the user to identify patterns. After a pattern is identified the tool uses this pattern to extract data from any document that has the same pattern.

Acknowledgments:

Special thanks for my professor and mentor Dr. Donald Roberts for his continuous guidance. He helped create this project by providing insight to choose the best design, commenting on important aspects, and making suggestions that made the project look professional.

Special thanks for my professor and mentor Dr. Deborah Hwang for her continuous supervision over this project. She helped create this project by providing us with the tools to organize and structure it.

Special thanks for my professor and mentor Dr. Robert Morse for directing me toward this project. He helped create this project by suggesting that a project in this area would suit me.

INTRODUCTION:

In the developer's work as a database designer, he often runs into many unstructured documents that contain useful information. He often had to study those documents and create parsers to extract useful information. This is a process that is redundant and wastes a good amount of time. This motivated the idea of creating a tool that would make this process easier and universal for any type of document.

To solve this problem, we notice in every text document there are patterns that can be found. Those patterns tell us where to find useful information. For example, in this document, you can clearly identify where the title is, the name of the project engineer, where the introduction paragraph starts and much other useful information. This project allows the user to identify those patterns by drawing rectangles around useful information on the text file.

Moreover, they can specify which pieces of data go together and export the parsed information to a database. In other words, the user specifies a grouping of parsed values. The tool then creates new instances for each set of values that define this group and stores them in a structured format.

PROBLEM STATEMENT:

Organizations create and receive many documents. These documents contain important pieces of information, but they are not structured. For example, an organization might receive invoices from a vendor. These invoices contain useful information such as items received and prices. If this organization wants to know the total amount of money they paid this vendor, someone must go back and read all those documents or structure them in a way that a program can analyze them. If they wanted to link this information to other information that they have, they would have to structure all their data in a uniform format. In other words, they would have to use a database to store their data and insert the information from those invoices into the database.

Structuring unstructured documents is time-consuming. Back to our invoice example, a solution might be creating a program that can understand that invoice structure and insert the needed information to a database, but it will not be a universal solution. If another vendor sent invoices that have a different format the first solution will not work. Also, creating a specific solution for every new text structure can be time-consuming. Thus, a universal extraction tool is needed.

SPECIFICATIONS AND REQUIREMENTS:

The program should allow users to create parsers for documents. They do so by identifying patterns in unstructured text files to extract useful data. Allow the user to define entities that correspond to database tables. Those entities are defined by what parsed data go into them and what entities they are linked to. Created parsers should work for any document of the same format.

The program should allow the user to select a list of documents and a parser. The program then should use the parser to parse out all the information. After the information is parsed it should be stored in the database.

The program should be written in C#. The interface should be a WPF that allows the user to load/create parsers, define entities, identify text patterns, and select a list of documents to be parsed.

DESIGN APPROACH:

The project consists of three parts. The first part is the interface. It is responsible for loading example text files, creating new parsers, parsing, and getting a connection string to a database from the user. The second part is parsing data from documents. It parses the text and creates entities with the needed values from the text. And the last part is managing the database. It establishes a connection to the database, creates tables if needed and inserts values into them.

The interface follows the Model–View–ViewModel (MVVM) design pattern. The model is the parser in this project. The view contains the graphics-related functions. The ViewModel is an intermediate that transfers signals between the model and the view.

The view is responsible for allowing the user to draw on the loaded text file. Determine what type of drawing was made and receive the needed information. It then translates the coordinates of the drawing into string indexes. It then forwards that information to the ViewModel.

The ViewModel holds an instance of the parser. It receives messages from the view and updates the parser. It also forwards the changes that the parser made to itself to the view to update taps and lists for the user.

The parser is a structure of rectangles following the composite design pattern. The whole text file is a base rectangle and the user draw concrete rectangles inside of it to highlight data. Each rectangle has a position relative to its parent. An enclosed list of rectangles that represent rectangles inside of it. A remainder rectangle that represents the remaining portion of text that is below this rectangle. An Entity name that this rectangle represents and an entity id if the rectangle had already been inserted to the database.

A rectangle can extract data by parsing all the enclosed rectangles, and the remainder rectangle. If it has an entity name, the rectangle creates a database manager instance and inserts

the data. If its parent has an entity name it grabs the parent's id and sends it to the database manager to be used as its parent in the database. A rectangle parses a text file by asking all its enclosed rectangles to parse and the remainder rectangle also.

There are three concrete types of rectangles. A value rectangle which is the most straight forward. It has a name and value. It overrides the base rectangle's parse function and finds its value from its relative position to the parent. It can copy itself by creating a new instance with the same name.

An ID rectangle is used only for the star rectangle. A star rectangle is a portion of the text file that has a variable length. It must have a pattern that repeats and an end identifier. A repeating pattern is a base rectangle that will repeat until an end identifier is reached. The end identifier is a base rectangle that must contain an ID rectangle. An ID rectangle has a name and can determine if at its relative position the contained string is the identifier.

The last part of the system is the database manager. It establishes a connection to the database. It inserts data that it receives from rectangles to the database. The insert function can either be an insert without a relationship to other tables or an insert with relation to a specific table with an id to a row in that table.

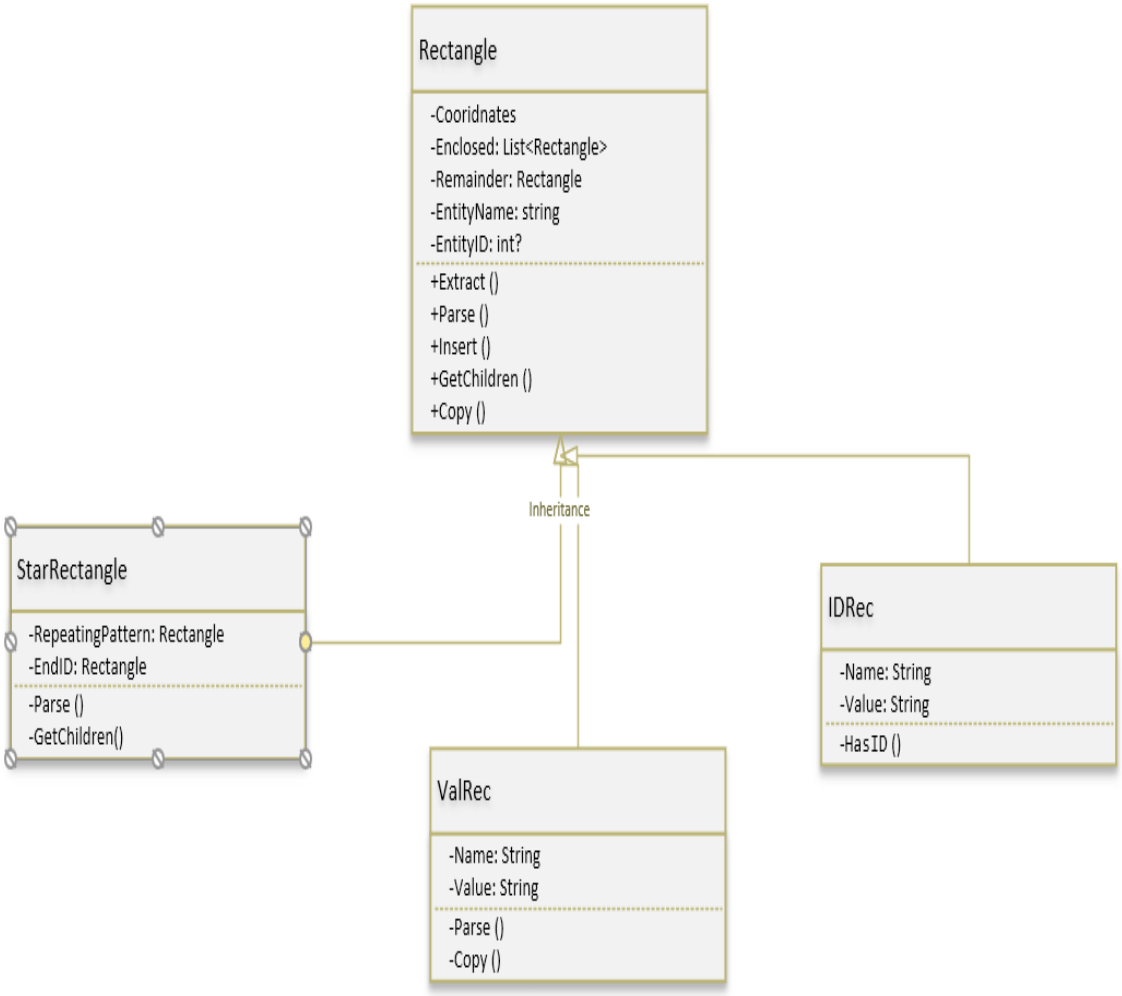

Figure 1 Parser Design

THIRD-PARTY SUBSYSTEMS:

The system relies on MySQL for creating a default database. “MySQL is the world's most popular open source database. Whether you are a fast growing web property, technology ISV or large enterprise, MySQL can cost-effectively help you deliver high performance, scalable database applications.” [1].

DESIGN TRADEOFFS:

Initially, the plan was to use formal languages to parse. The parser would consist of a sequential list of identifiers and attributes that represent static identifiers and named values. This was very successful in parsing data reliably and covered many more cases. The problem with it is that it is very difficult to work with. It would require a user that is very familiar with formal languages. Even with a good understanding, it would take a significant portion of time to build a parser. Thus, drawing rectangles was much more preferred. It covers most cases and can be used by any person with a simple understanding of their organization.

Result:

The project works as a proof of concept. It successfully allows the user to load an example document to create the parser. Users can highlight information on the document by drawing rectangles. They specify the type of rectangle wither it was a value, variable size rectangle, or a new entity. The information is then parsed and stored into a local MySQL database.

Entity mapping is very minimal. This makes the project do only half the work. The data is stored in a more structured form. After that, the user needs to export the data in the database into a better-structured database.

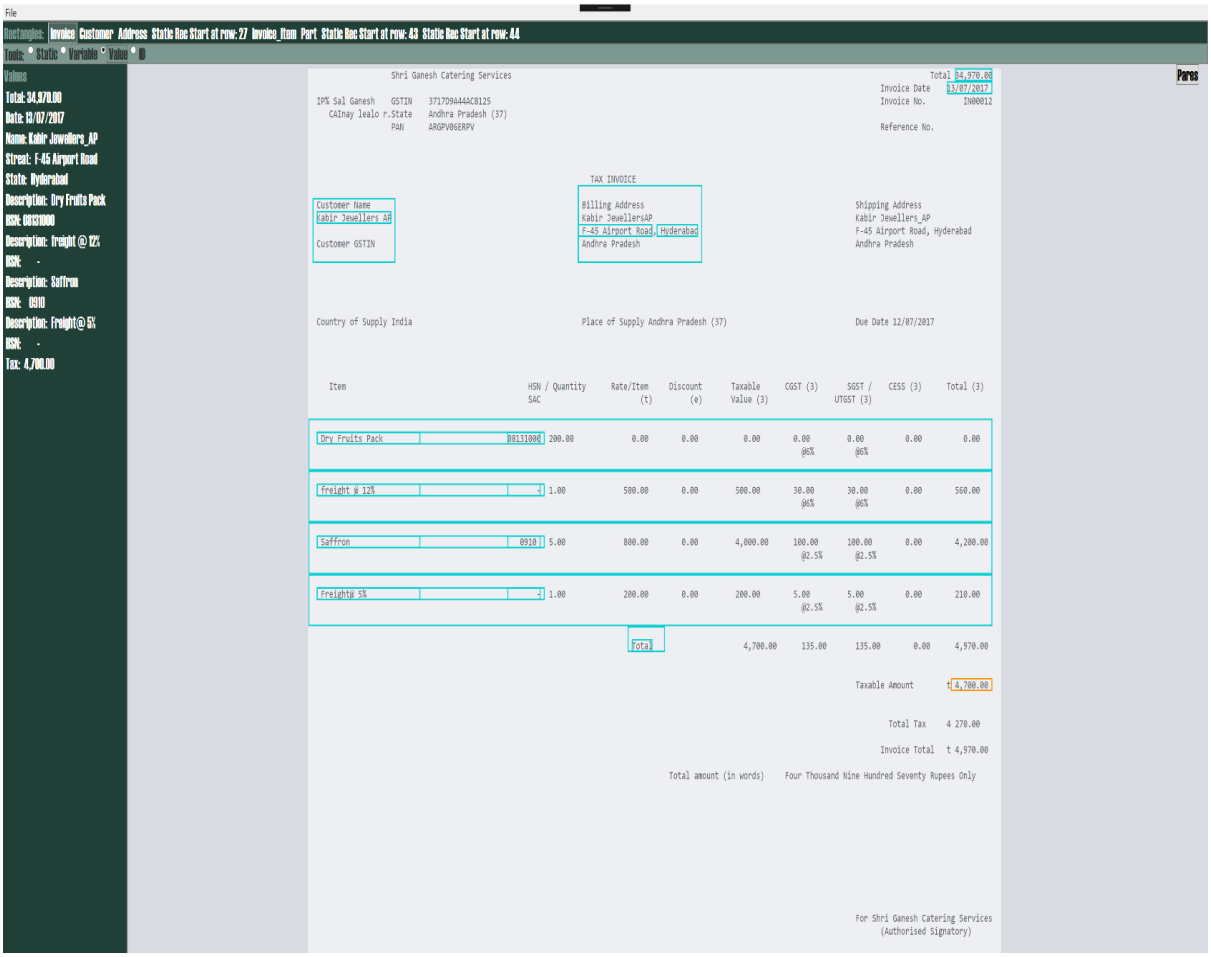

Figure 2 A use case example

CONCLUSION:

The project has a lot of potential. Creating a parser using this tool takes minutes. It will allow big organizations to transform their old unstructured data in an acceptable period. The project still needs further development. Currently, parsers are not stored which makes it really a proof of concept. Users would not go through the trouble of making the parser every time. More importantly, better entity mapping needs to be implemented. Mapping the customer's database exactly would save a lot of time. They would not have to go through the trouble of structuring their unstructured documents and then restructuring the new structure.

REFERENCES:

1. "MySQL Editions." *MySQL*, www.mysql.com/products/.

Biography:

Turki AlHarbi is computer science senior at the University of Evansville. He works at Stark Engineering as a software engineer and his main responsibility is the company's infrastructure. This implies that a huge amount of his time is spent designing databases and in many cases creating solutions to automate data input to those databases. He plans to continue working there after graduation. He will be responsible for developing software solutions for the company and its customers along with developing software products.