Department of Computer Science

CASC Analysis Project

Final Report

Jiajun Chen

Table of Contents

Introduction

This is the final report of the CADE ATP System Competition (CASC) Analysis project. Throughout this project, a Java application –a CASC resulting data analysis software – is specified, designed and implemented. In the following part of this report, the Unified Modeling Language (UML) documentations will be used to describe the details of this project.

Background information and requirements elicitation

Problem Statement

The CADE ATP System Competition (CASC) is a yearly competition of fully automated theorem proving (ATP) for classical first order logic. The competition organizers ask the competitive ATP systems to try to solve all the problems that are chosen from the Thousands of Problems for Theorem Provers (TPTP) Problem Library within a limited time. This competition is effective for evaluating the performance of the participated ATP systems in terms of whether the system can solve the problem or not, and the runtime for the problem solved by the system. Every year, the outcome of the competition results in a huge amount of data. It is hard for the committee to make an intuitive decision or a logical decision about who is the winner via that enormous size of data. In order to help the competition organizers to speculate the winner of the competition and to get more information about the result, it is necessary to implement a

program to analyze the resulting data. This project will explore all the possibilities to present the result of the competition more effectively.

Requirements

Following are the functional requirements that describe the interactions between the system and its environment independent from implementation, and the non-functional requirements that are not directly related to functional behavior.

Functional Requirements

- The user should be able to input the URL address
- The user should be able to select a local excel file
- The user should be able to view the data from previous competitions by entering the year number
- The software must download the data from the website and store them into a local excel file.
- The program must read excel documents which goes through the data in the excel file and imports the desired information to the Java program.
- The program must check if there are summary rows in the excel documents
- The program must show the desired information to the user
- The program must have an GUI
- The program should print the result of one sheet to the user at a time
- The program should be able to ask the user if the user wants to delete the downloaded file when the program is closing.

Non-functional Requirements

- The program should check if there is an available Internet access
- The program should check if the data has been successfully saved to the local file or not
- The program should check if the local file exists or not
- The program should be packaged into a runnable jar file
- The program should check if the format of the data fits the program or not
- The program should check if the user reaches the first sheet of the workbook
- The program should check if the user reaches the last sheet of the workbook
- The program should check if the year number that the user entered is legal or not

Scenarios

As M. Carroll said in 1995, "A narrative description of what people do and experience as they try to make use of computer systems and applications (Carroll, 1995)," scenarios are resourceful for requirements elicitation. Each scenario is a concrete, focused, informal description of a single feature of the system used by a single actor. The following presents three different scenarios for using the CASC resulting data analysis application.

Scenario I: User enters the URL address and wants to get the result of the competition

1. User opens the program
2. User presses the “Open URL” button to enter URL address to get data from the website
3. User enters the URL and presses the OK button
4. The program tries to access the website; it will return error message if the data or the website is not available
5. The program downloads the data from the website and saves the data into a local excel file
6. The program reads the local excel file and imports its data
7. The program selects and calculates the desired information from the data
8. The program saves the results into a local text file
9. The program prints the results from the text file into the GUI
10. User presses the exit button
11. The program deletes the output text file
12. The program will ask whether the user needs to delete the local data file or not

Scenario II: User wants to open a local excel file and wants to get the result of the competition

1. User opens the program
2. User presses the “Open Local File” button to open a local excel file from the computer
3. The program pops up a file selector

4. User selects a local excel file
5. The program tries to read the selected file; it will return error message if the data is not available or the format of the table does not pairs with the program
6. The program reads the local excel file and imports its data
7. The program selects and calculates the desired information from the data
8. The program saves the results into a local text file
9. The program prints the results from the text file into the GUI
10. User presses the exit button
11. The program deletes the output text file
12. The program will ask whether the user needs to delete the local data file or not

Scenario III: User enters the year number and wants to get the result from one of the previous competition

1. User opens the program
2. User presses the “Previous Competition” button to enter year number to get the data of a previous competition from the website
3. User enters the year number and presses the OK button
4. The program will first check if the year number that the user just entered is in the available domain or not
5. Once the year number is available, the program searches and gets the relevant URL address in a method that has the entire available URL built-in.

6. The program tries to access the website; it will return error message if the data or the website is not available
7. The program downloads the data from the website and saves the data into a local excel file
8. The program reads the local excel file and imports its data
9. The program selects and calculates the desired information from the data
10. The program saves the results into a local text file
11. The program prints the results from the text file into the GUI
12. User presses the exit button
13. The program deletes the output text file
14. The program will ask whether the user needs to delete the local data file or not

Use Case Diagram

A use case diagram shows a set of cases and actors and their relationships. Mostly a use case diagram is based on the scenarios. Use case diagrams present the static use view of a system. They are very useful in organizing and modeling the

architecture of a program.

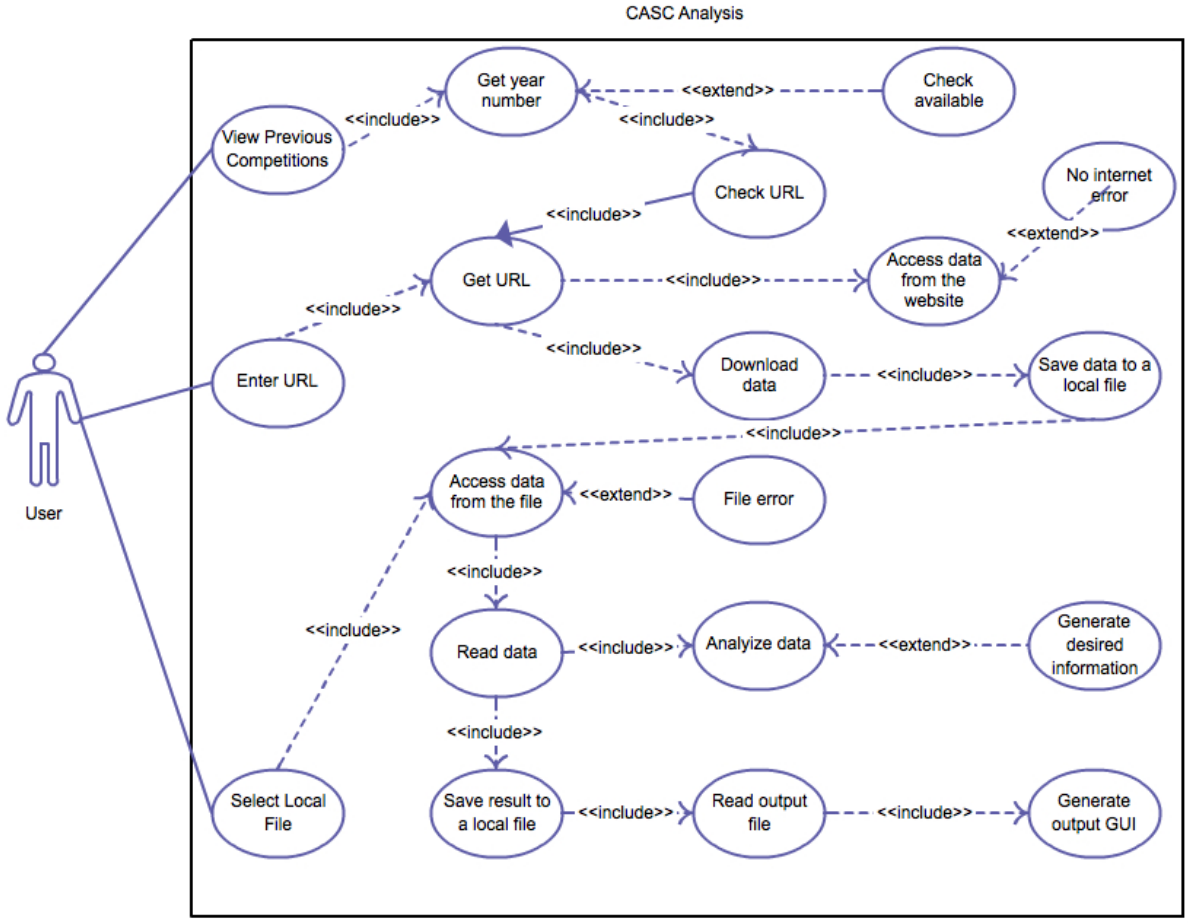

Figure 1

Figure 1 shows the use case diagram of the CASC resulting data analysis application based on the three scenarios above. The user is the only actor in this application, which is the role that humans play when interacting with the application. From the use case diagram, the program can be separated into two parts: Getting resulting data and analyzing resulting data. The user has three different triggers to get resulting data. It is easy to see that the section for analyzing data and printing desired information stays the same in all situations.

Design

- ### Step 1: Finding all the information needed based on the requirements

Since the application can be separated into two parts: getting resulting data and analyzing resulting data along with printing desired information, the implementation of this application can also be separated into two parts.

The first approach after some tables with resulting data have been downloaded as the samples, is reading the table through the Java program. A library called jxl, which stands for Java Excel API, has been found useful. The Java Excel API is a mature, open source Java API enabling developers to read, write, and modify Excel spreadsheets dynamically. All the resulting data in the samples now become readable and the Java program is able to grab the resulting data from the Excel file. In some sheets, summary rows are available. These summary rows can speed up the application. The application can directly read the summary data instead of counting and calculating all the data by itself. Since the summary rows might not exist in some sheets, the application should be able to determine whether the summary rows exist in the current sheet or not, and should be able to handle any situations.

The second approach is for the part of getting resulting data from the website. In order to access the online resulting data and download them into a local file, the Java HTML Parser (jsoup) is proved to be necessary. Jsoup is a Java library for working with HTML. It analyzes the HTML source page of a website and grabs information based on the HTML tags. Jsoup will try to access the URL that the user entered and save the relevant data to a local file.

The following approach, which can be stated as another case of the second approach, is to get resulting data from a local file. A file selector GUI will allow the user to select a local file that has the resulting data and import the data to the program.

- **Step2: Class Diagram**

After finish analyzing the requirements and gathering all the information needed, a class diagram has been initialized. Class diagram shows a set of classes, interfaces, and their relationships. They are the most commonly used and the most helpful diagrams in object-oriented implementation.

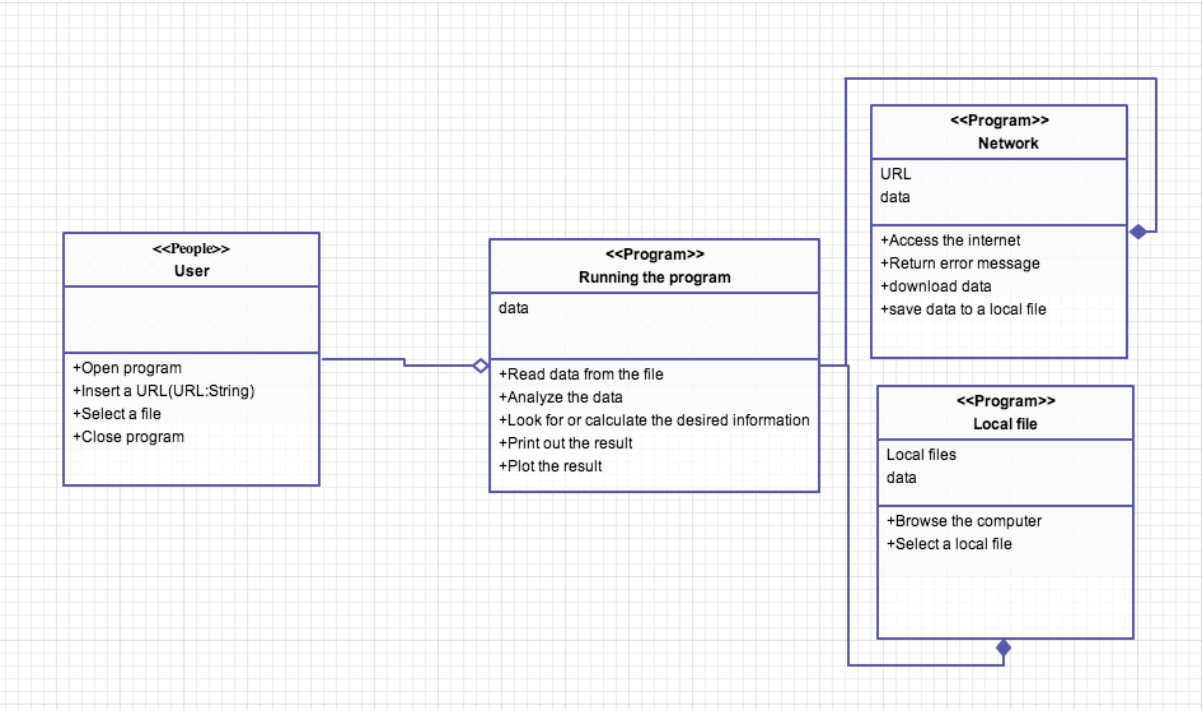

Figure 2

Figure 2 is the class diagram for the CASC resulting data analysis application in its first version. It basically relies on the relevant information and the approaches. This is an immature class diagram for the application because it is a sketch of the design and it does not have any interface or subclasses. The upper right section is a class that gets the resulting data from the Internet. It has some methods that access the data from the website using the Java html parser and download the resulting data into a local file. The lower right section is a class that allows the user to browse his computer, to selects a

local file and to imports the data from that file. The core class that analyzes the imported data and generates the desired result is placing at the center.

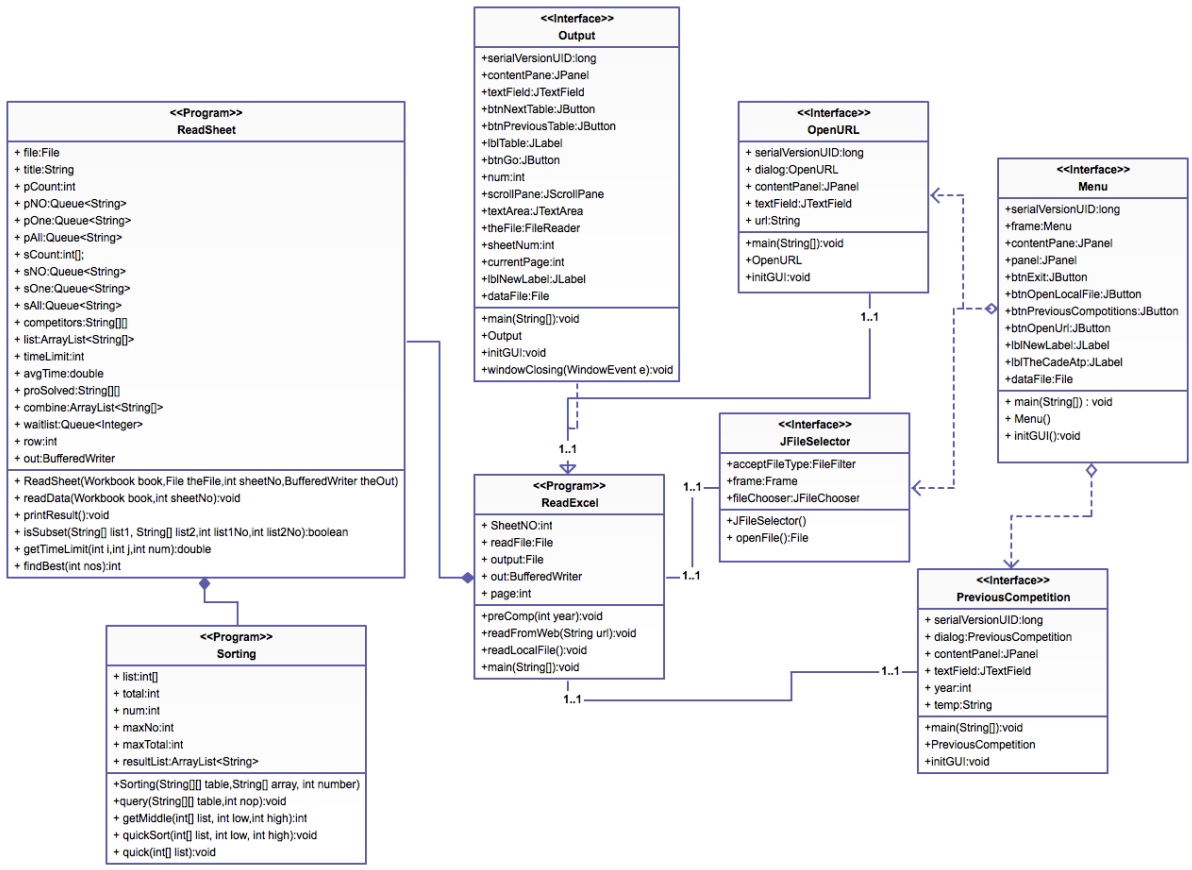

Figure 3

Figure 3 is the final class diagram; it was evolved from the first version. The key improvements are the implementation of the user interface and the specialization of the analyzing part. The application starts with the Menu GUI, and it goes to different GUI based on the selection of the different data importing methods by the user. Each data importing method calls the relevant methods in the ReadExcel class and imports the resulting data to the program. The ReadExcel class is the main pivot of this program; it has several methods that can access the resulting data and passes the data to the ReadSheet class. Most of the data analyzing the result generating happen in the ReadSheet class. The Sorting class is used as a functional class for sorting the number of

problems that the competitors solved in each group with different time limit; it returns the sorting results back to the ReadSheet class. After finishing generating the desired information in the ReadSheet class, the ReadExcel class calls the Output GUI and shows the desired information to the user.

- **Step 3: Sequence diagram**

Sequence diagram is a kind of diagram that shows the interaction among the objects and their relationships. The purpose of the sequence diagram is to present the sequence of interaction among the objects in a time based view. The scope of a sequence diagram includes all the interactions for a single use case. Following are three sequence diagrams of the application based on the three use cases.

Sequence diagram 1

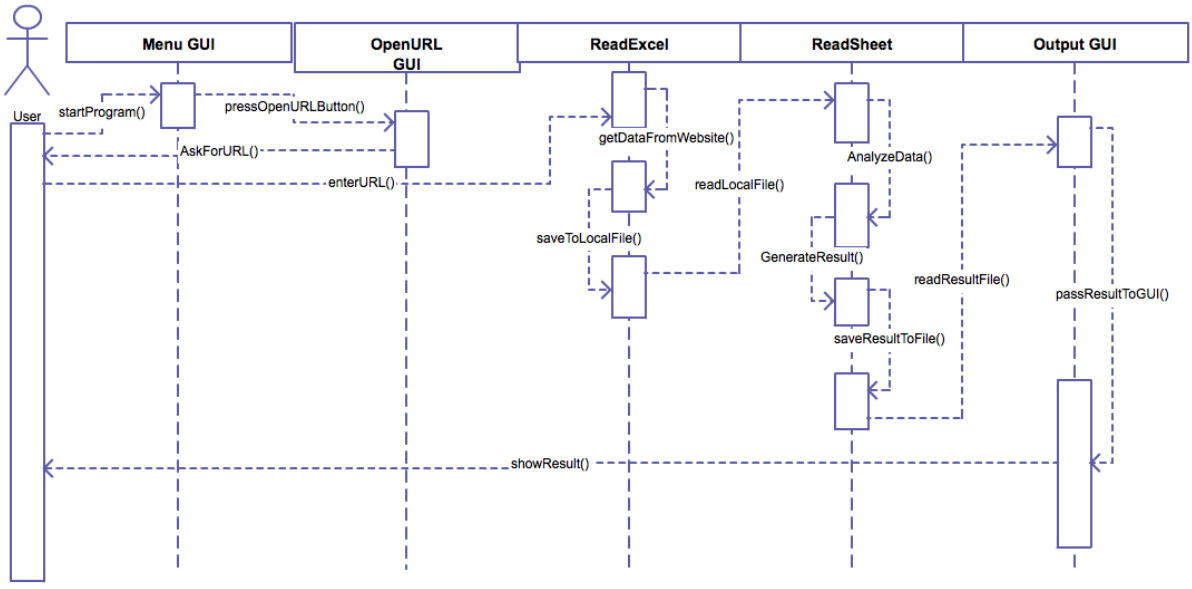

Figure 4

Sequence diagram 2

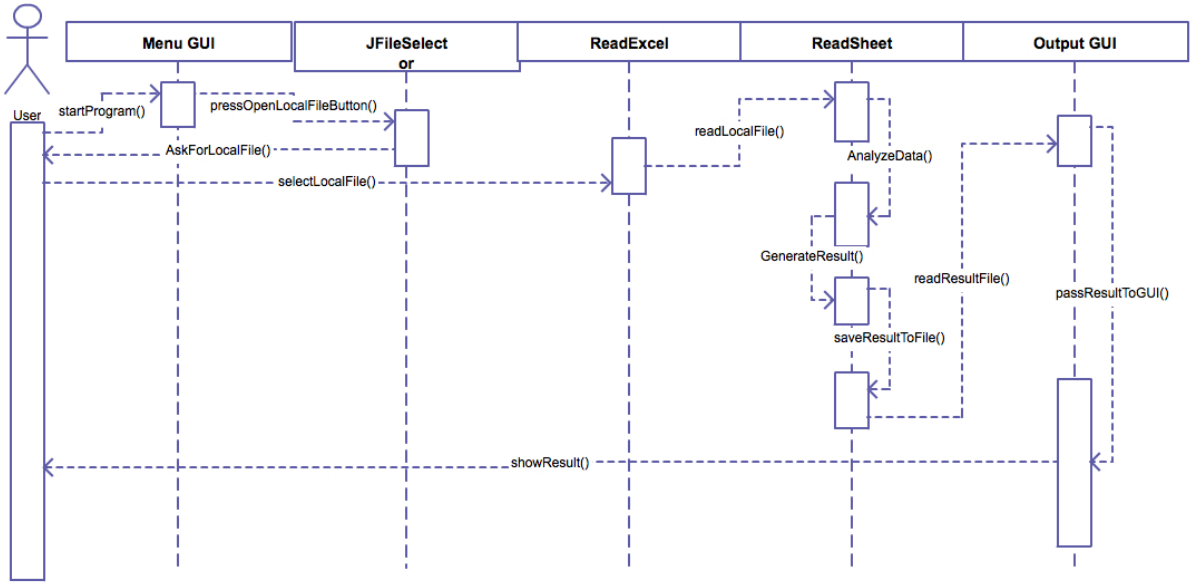

Figure 5

Sequence diagram 3

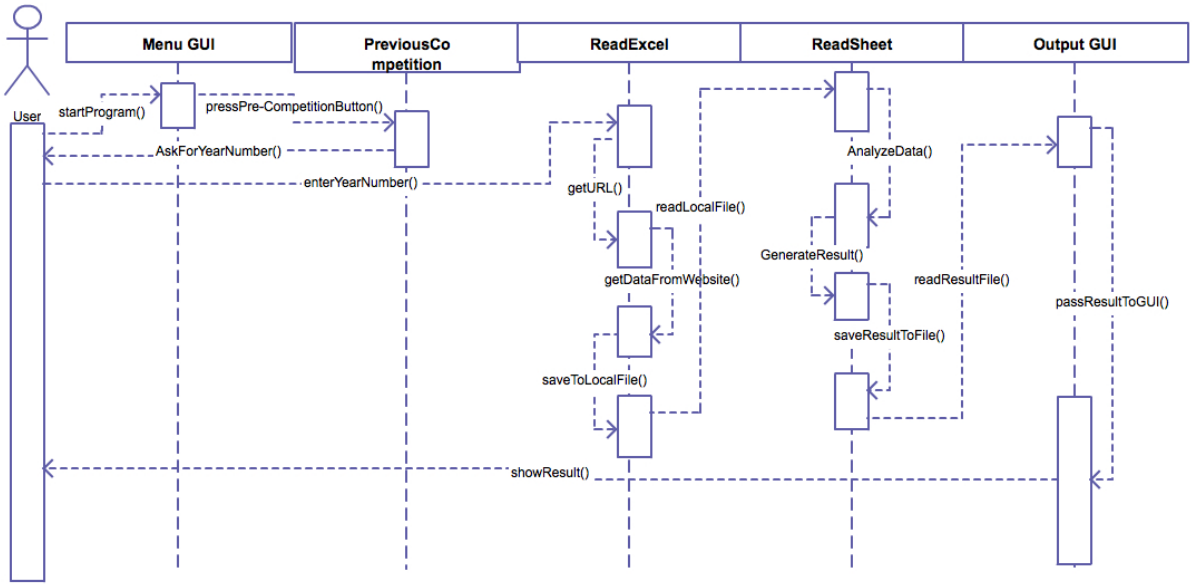

Figure 6

Figure 4,5,6 are the sequence diagrams for use case I, II, III, respectively. They basically show the interactions between actors and objects in the sequential order that those interactions occur in all use cases.

Step 4: State diagram

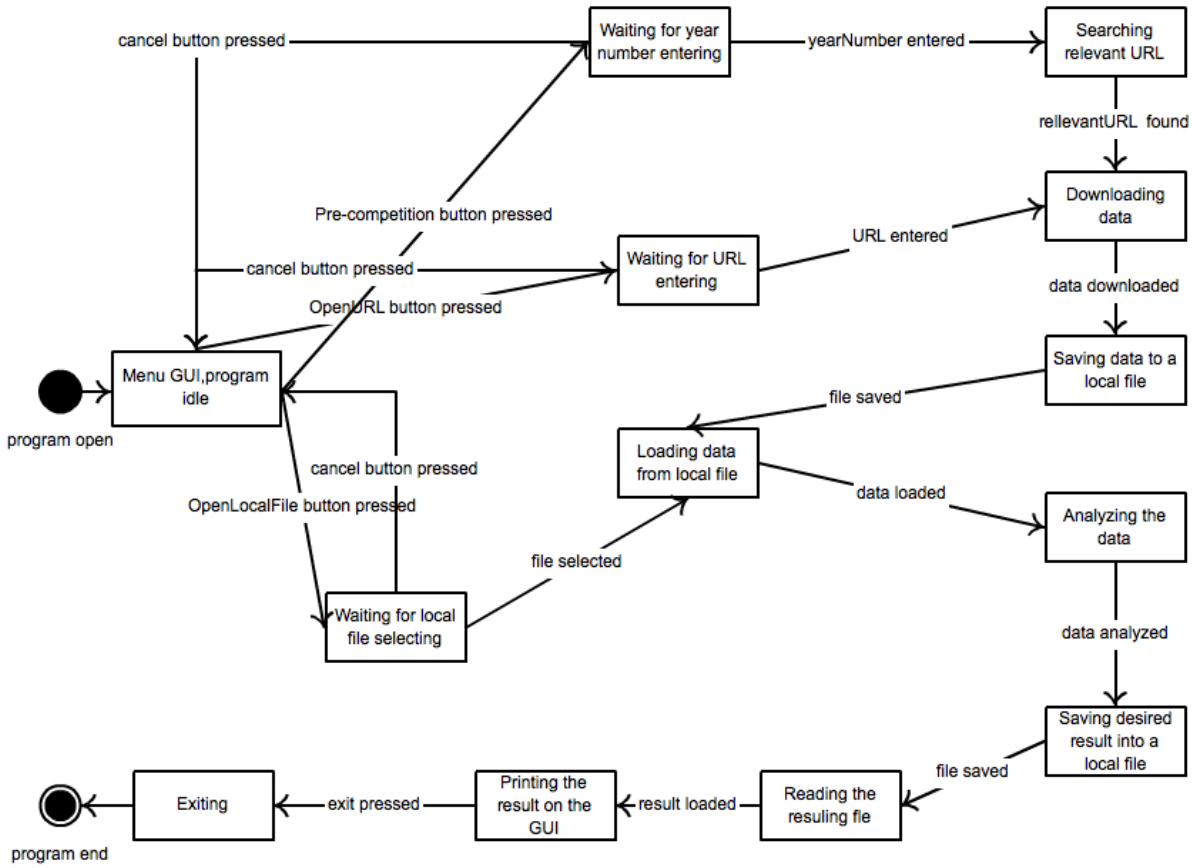

Figure 7

Figure 7 is the state diagram for the CASC resulting data analysis application; it describes the dynamic behavior of the program. It shows how the program evolves and changes state in response to a stimulus. Figure 7 shows the changes of the state of the program from the "program open" state to the "program end" state; it is also a complete flow of running this application.

Implementation

- **Step 1: Grab the resulting data from the website**

The first approach of the implementation is to grab the resulting data from the website and save it to a local file. Since the Java HTML Parser (jsoup) library provides a possibility to access the online data directly from the application, the most important goal of this step is to tell the computer: what kinds of data are needed?

```
<TR>
<TH ALIGN=LEFT BGCOLOR=GREY> <SPAN CLASS ="smallfont">Higher-order Theorems</SPAN>
<TH BGCOLOR=SKYBLUE> <A HREF="http://www.cs.ru.nl/~kuehlwein/">Satallax&#8209;MaLeS</A><BR><SPAN CLASS="xxsmallfont"> 1.2</SPAN>
<TH BGCOLOR=LIME> <A HREF="http://www.satallax.com">Satallax</A><BR><SPAN CLASS="xxsmallfont"> 2.7</SPAN>
<TH BGCOLOR=YELLOW> <A HREF="http://www.cl.cam.ac.uk/research/hvg/Isabelle/">Isabelle</A><BR><SPAN CLASS="xxsmallfont"> 2013</SPAN>
<TH BGCOLOR=WHITE> <EM><A HREF="http://isabelle.in.tum.de/">Isabelle</A><BR><SPAN CLASS="xxsmallfont"> 2012</SPAN></EM>
<TH BGCOLOR=WHITE> <A HREF="http://www.ags.uni-sb.de/~leo/">LEO&#8209;II</A><BR><SPAN CLASS="xxsmallfont"> 1.6.0</SPAN>
<TH BGCOLOR=WHITE> <A HREF="http://gtps.math.cmu.edu/tps.html">TPS</A><BR><SPAN CLASS="xxsmallfont"> 3.120601S1b</SPAN>
<TH BGCOLOR=WHITE> <A HREF="http://www.alumnos.unican.es/ccc66/cocATP.htm">cocATP</A><BR><SPAN CLASS="xxsmallfont"> 0.1.8</SPAN>
<TR><TH ALIGN=LEFT> <A HREF="http://www.cs.miami.edu/~tptp/cgi-bin/SeeTPTP?Category=Problems&Domain=SET&File=SET002^7.p"><TT>SET002^7</TT></A><SUP>*</SUP>
<TD ALIGN=RIGHT> <A HREF="Results/TEQ/Satallax-MaLeS---1.2/SET002^7">0.09</A>
<TD ALIGN=RIGHT> <A HREF="Results/TEQ/Satallax---2.7/SET002^7">0.19</A>
<TD ALIGN=RIGHT> <A HREF="Results/TEQ/Isabelle---2013/SET002^7">42.70</A>
<TD ALIGN=RIGHT> <A HREF="Results/TEQ/Isabelle---2012/SET002^7">45.48</A>
<TD ALIGN=RIGHT> <SUP>+</SUP><A HREF="Results/TEQ/LEO-II---1.6.0/SET002^7">0.28</A>
<TD ALIGN=RIGHT> <A HREF="Results/TEQ/TPS---3.120601S1b/SET002^7">TMO&#8209;Non</A>
<TD ALIGN=RIGHT> <A HREF="Results/TEQ/cocATP---0.1.8/SET002^7">TMO&#8209;Non</A>
```

Figure 8

Figure 8 is a part of the source code of the website that has all the resulting data. It is easily to observed that all the relevant data are stored between the HTML tags in the following format:

```
/*
 *<TABLE>
 * <TR>
 *          <TH>
 *          </TH>
 *
 *          <TD>
 *          </TD>
 * </TR>
 * </TABLE>
 */
```

Therefore, using the jsoup library, the program is able to access the source code of the website, look for those specific HTML tags, and grab the data between the HTML tags back to the program.

Meanwhile, the initialization of a local Excel file is successful by using the Java Excel API (jxl). The data that the jsoup grabs will be entered into the Excel file that the jxl created. It is like the Java Excel API prepares the chocolate box, the Java HTML Parser collects the chocolates, and the program puts all the chocolates into the chocolate box. After finish saving the local Excel file, all the resulting data is ready to be analyzed.

- **Step 2: Another way of data collection**

Besides directly accessing the resulting data by entering the URL address, the program also provides two ways to gather relevant data. The first way is an extension of entering the URL address. Some URL addresses have been stored into the program, so that the user can enter the year number of the competition instead of entering the URL address. The program gets the URL address of the year that the user just inputted and grabs the resulting data like the previous method. Another way to access the resulting data is to read a local Excel file. It is useful when Excel file that has all the resulting data has been saved or downloaded. The program pops up a file selector for the user to browse the computer and to select the local file.

- **Step 3: Analyze the resulting data**

Results for THF (Higher-order Theorems)

Higher-order Theorems	Satallax- 1.2	Satallax 2.7	Isabelle 2013	*Isabelle 2012*	LEO-II 1.6.0	TPS 3.120601S1b	cocATP 0.1.8
SET002^7*	0.09	0.19	42.70	45.48	+0.28	TMO-No	TMO-No
SEV019^5	2.92	8.17	4.04	6.85	+0.03	UNK-Nor	TMO-No
SET583^7*	0.15	3.28	16.19	18.83	+0.19	TMO-No	TMO-No
SYN036^5	1.48	0.19	3.04	6.36	+0.18	TMO-No	TMO-No
SET926^7*	1.17	10.24	91.05	93.01	+0.18	TMO-No	TMO-No
SYO374^5	0.80	0.01	28.35	56.52	+0.07	TMO-No	TMO-No
SYN045^7*	0.14	0.01	16.04	18.83	+1.07	67.25	TMO-No
SYO376^5	0.09	0.01	4.07	62.57	+0.06	16.16	TMO-No
SET907^7*	1.17	10.25	93.62	93.17	+0.39	TMO-No	TMO-No
SEV084^5	4.17	0.20	146.27	TMO-No	+10.38	TMO-No	TMO-No
AGT034^2	2.25	11.07	89.67	86.28	+0.17	TMO-No	TMO-No
SYO387^5	0.14	5.34	111.72	114.10	+0.68	TMO-No	TMO-No
SWV426^2	0.14	8.12	TMO-No	19.66	+0.06	TMO-No	TMO-No
LCL731^5	0.14	0.01	4.58	TMO-No	+0.08	32.88	+120.76
NUM801^1	0.14	12.96	12.88	16.13	+26.55	1.58	TMO-No
SYO173^5	0.14	0.40	6.21	8.57	+0.03	UNK-Nor	TMO-No
SYO064^4.001	0.14	3.39	16.16	18.73	+0.01	3.12	TMO-No

Figure 9

Figure 9 shows a part of a spreadsheet that contains all types of relevant information. Besides the cell on the upper left, all the contents in the first row are the name of the competitors, and all the contents in the first column are the name of the problems. All the numbers in the spreadsheet are the time that the competitor spent on solving that problem. The "TMO"(Time out) exists when the competitor uses more time than the time limit. In case there are some unknown errors happen, the content will be stated as "UNK."

The analysis is based on some questions that are listing in the following:

1. Which problems were solved by NO SYSTEM?
2. Which problems were solved by ONE SYSTEM?
3. Which problems were solved by ALL SYSTEM?
4. Which systems solved NO PROBLEMS?

5. Which systems solved ONE PROBLEM?
6. Which systems solved ALL PROBLEMS?
7. Which systems solved superset of another system?
8. What is the best combination that has the maximum problems solved when the time limit is equally divided?
9. What is the best combination that has the maximum problems solved when the time limit is not equally divided?

In order to solve the first three questions, the program firstly needs to count the valid number of rows in the current spreadsheet. Then the program iterates the whole table and counts how many non-numeric contents are there in the table. The counting results are saved into different variables based on the horizontal iteration or the vertical iteration. After that, the name of all problems will be added into different queues based on the counting result. The name of the problems in those queues answers the question 1 to 3.

Solved/150	119/150	116/150	108/150	98/150	76/150	47/150	14/150
Av. CPU Time	10.42	11.39	54.65	60.54	5.05	25.23	52.05
Solutions	0/150	0/150	0/150	0/150	76/150	0/150	14/150
µEfficiency	505	393	36	21	397	48	20
SOTAC	0.25	0.24	0.25	0.24	0.24	0.32	0.23
New Solved	25/27	25/27	20/27	20/27	14/27	3/27	1/27

Figure 10

When answering the question 4 to 6, the counting results from the vertical iteration are needed. The name of the competitors will be added into different queues based on the counting result. In some spreadsheets, there are some summary rows that have all the counting information (like Figure 10). In case there exists some summary rows in the current spreadsheet, the program can directly get the counting information from the summary rows instead of counting by iterating the whole spreadsheet. The name of the systems in those queues can answer the question 4 to 6.

In order to solve the rest of the questions, the program separates the table into several individual string arrays without null values. Then the program calls a method that justifies whether a system solved superset of another system. The method gets the arrays of two competitors each time. It goes over the two arrays and sees if there a system solved subset of another system.

The last two questions are related to the time limit that is provided by the committee. When the time limit is equally divided, the program counts the number of problems a competitor solved in different combinations and adds the counting result into a 2D array. Here is the pseudo code of this action:

```
for(i=2 to N systems){
        Set time limit to overall limit/I;
        Take top i systems;
        Count the number of problems that a competitor solved;
}
```

After getting the counting results, the program passes these results to a sorting class that uses quick sort methods to place the counting results in order. It is easily to get the best group combination from the sorted results.

When the time limit is not equally separated, the program needs to distribute the time limit based on the following algorithm:

$$Distributed\ time\ for\ competitor\ A = \left(\frac{Avg.time\ of\ A}{Avg.time\ of\ A + Avg.time\ of\ B + \cdots}\right) * Overall\ time\ limit$$

Thus the program needs to calculate the average time of solving problems through either the counting results or the relevant information in the summary rows. Then the program counts the number of problems a competitor solved in different combinations and find out the best group combinations.

Conclusion

Throughout the report, a clear view and structure of the CADE ATP System Competition Analysis Application project has been stated. All the detailed UML documentations are found resourceful for developing this project. The implementation of this program is a great practice for data analyzing and data mining. The design pattern of this project can be easily applied to any situations that need to access online data and analyze locally. Despite all this, there are still a lot of rooms for improvement. The application should have a better GUI and more functions. In conclusion, the CADE ATP System Competition Analysis Application will find itself useful to the committee of the CADE ATP System Competition.

Reference

[1]. Carroll, J. M. (1995). *Scenario-Based Design: Envisioning Work and Technology in System Development.* Wiley.
[2]. Hedley, J. (n.d.). *jsoup*. Retrieved from jsoup: http://jsoup.org/
[3]. Khan, A. (n.d.). *JExcelApi*. Retrieved from JExcelApi: http://jexcelapi.sourceforge.net/