Remote Package Dependency Analysis

Operational Concept Document

Instructor- Dr. Jim Fawcett

Syracuse University

Bo Qiu (SUID 726133617)

December 5, 2018

Contents

[**1. Executive Summary** 5](#_Toc531816888)

[**2. Introduction** 6](#_Toc531816889)

[*2.1 Application obligations* 6](#_Toc531816890)

[*2.2 Organizing principles* 6](#_Toc531816891)

[*2.3 Key architectural ideas* 6](#_Toc531816892)

[**3. Uses** 7](#_Toc531816893)

[*3.1 Users* 7](#_Toc531816894)

[*3.2 Uses of useful information for people* 7](#_Toc531816895)

[3.2.1 Students and programmers 7](#_Toc531816896)

[3.2.2 Teaching assistants and professors 7](#_Toc531816897)

[3.2.3 Key performance indicators(KPI) managers 8](#_Toc531816898)

[**4. Partitions** 9](#_Toc531816899)

[*4.1 Tokenizer* 9](#_Toc531816900)

[*4.2 SemiExpression* 10](#_Toc531816901)

[*4.3 TypeTable* 10](#_Toc531816902)

[*4.4 TypeAnalysis* 10](#_Toc531816903)

[*4.5 DepAnalysis* 10](#_Toc531816904)

[*4.6 StrongComponent* 11](#_Toc531816905)

[*4.7 Display* 11](#_Toc531816906)

[*4.8 Tester* 11](#_Toc531816907)

[*4.9 Comm* 11](#_Toc531816908)

[*4.10 Server* 11](#_Toc531816909)

[*4.11 Client* 12](#_Toc531816910)

[**5. Application Activities** 13](#_Toc531816911)

[**6. Critical Issues** 15](#_Toc531816912)

[*6.1 quote* 15](#_Toc531816913)

[*6.2 Ignore termination* 15](#_Toc531816914)

[*6.3 GetTok()* 15](#_Toc531816915)

[*6.4 Toker invariant:* 16](#_Toc531816916)

[*6.5 Typetable extract method* 16](#_Toc531816917)

[*6.6 Dependency relationship* 16](#_Toc531816918)

[*6.7 The message pass overload* 17](#_Toc531816919)

[*6.8 The temporary files* 17](#_Toc531816920)

[*6.9 GUI performance* 17](#_Toc531816921)

[**7. Prototyping** 18](#_Toc531816922)

[**8. Reference** 23](#_Toc531816923)

# **1. Executive Summary**

The purpose of this document is showing a design opinion for Remote Package Dependency Analysis, called Operational Concept Document (OCD). For further analysis of a specified folder which contains many code files, it is very important for programmers to understand and see clearly the connection and dependency between all files. It is obviously convenient that when programmers test the whole solution and understand clearly whole structure by using the Remote Package Dependency Analysis. The Remote Package Dependency Analysis has two endpoint, server and client. The client runs the GUI, react the user’s click and shows results, while the server runs the receiver to receive the message passed from client, analysis the requirement by using type analysis, typetable, dependency analysis and strong, and then reply the results to client. With such function, the users could analysis the specified files in remote server. For the further development, we are attempting to add “upload files” function, so that the user could upload files and save to the server.

The users of Remote Package Dependency Analysis only include source code files. However, The users of for Remote Package Dependency Analysis include students, programmers, teaching assistants, professors, key performance indicators(KPI) managers. For each different user, the Remote Package Dependency Analysis could be extended in some different way to meet the different requirements.

The Remote Package Dependency Analysis is constructed with many packages, with respect to Tokenizer, SemiExpression, TypeTable, TypeAnalysis, DepAnalysis, StrongComponent, Display, Comm, Server, Client and Tester. There will be Detailed explanation for these packages in Section 5.

The critical issues include quote, ignore termination and GetTok() and so on. There will be a detailed discussion in Section 6.

# **2. Introduction**

## *2.1 Application obligations*

Computer programming has been developing for almost half a century. From the initial “hello world” to million lines of codes, the volume of a software is much bigger and bigger. As in a huge system, the content becomes much more complex in adding more and more functions. The whole system just like hundred blocks and there are chains connect each other, if a block shakes, other connected blocks also shake. That’s introduce us to learn the way how to design a good structure for big system. Furthermore, as the appearance of The Remote Package Dependency Analysis, the user could easily to understand the structure of connection between files and files, so that the user could easily understand the whole system. And the method of remote analysis facilities the user’s operation and the GUI shows results more clearly.

## *2.2 Organizing principles*

The all projects are designed in C#, with the help of .Net Framework, Windows Communication Framework and Windows Presentation Framework in the Visual Studio 2017 Enterprise environment. And the steps of organizing principles follow like this: many functions to class, small class to big package, individual package to whole solution. The origination principle follows the small part to whole system, because it is easy to understand small part but its hard to understand whole system, and it is more convenient for developer to understand the whole system and extend the program.

## *2.3 Key architectural ideas*

The key ideas to design Remote Package Dependency Analysis is to extract the words and convert them to tokers, and then by using semiexpression, the tokers combines to specified lines of outputs. With the help of typeanalysis, the output of semiexpression construct the many divisions of output. The typetable is obtained according to typeanalysis, and dependency analysis is based on typetable. Finally, we get the strong component by using dependency analysis and under the help of WCF, we could operate remotely. The GUI design of client is also one of important process in the whole program, with the high performance efficient of GUI, the users could get better experience when using the program.

# **3. Uses**

The users of Remote Package Dependency Analysis only include source code files. However, The users of for Remote Package Dependency Analysis include students, programmers, teaching assistants, professors, key performance indicators(KPI) managers. For each different user, the Remote Package Dependency Analysis could be extended in some different way to meet the different requirements.

## *3.1 Users*

The users of Remote Package Dependency Analysis only include source code files, because the code files are imported to the program and being analyzed, the code files could be anywhere, however the entrance of program is the folder, and in this folder, all files, include sub-folder, will being scanned and the code files will be found. In this situation, the files could be analyzed as a group and so we could get the useful information of connection between folder and folder or files and files. The uses of Remote Package Dependency Analysis could be extended for different requirement with the difference group of users.

## *3.2 Uses of useful information for people*

### 3.2.1 Students and programmers

For students and programmers, the Remote Package Dependency Analysis helps them to future the architecture of program and help them understand the structure of big system. The sub-function, type analysis, helps students and programmers to code more and avoid coding some complex function to let the code being more beautiful.

### 3.2.2 Teaching assistants and professors

When the teaching assistants and professors reviewing homework and grading them, they need to understand hundreds of homework, it doesn’t make sense. By using the Remote Package Dependency Analysis, they could easily know the structure of homework and decide to increase their grades of decrease their grades, it makes sense. Also, by using type analysis, if there exits the global function in class or global variables, the GUI will show that and tell the professors and teaching assistants that the students’ structure is not good, so the grades should be cut down a little. All in all, Remote Package Dependency Analysis helps teaching assistants and professors to grade.

### 3.2.3 Key performance indicators(KPI) managers

The assessment of effectiveness for a software engineer in a company shows worker’s working ability, also called key performance indicators (KPI). The KPI is the real effective output of software for thousands of lines of codes, excludes comments, repeated grammars and redundant lines. The salary of a software engineer could be increased or decreased according to his KPI. It is important for managements in a company, however we need to build a more complex system in the foundation of Remote Package Dependency Analysis which could automatically erase comments, repeated grammars and redundant lines. And the more complex program could automatically erase repeated function so that to truly analyze whole work load of a software programmer.

# **4. Partitions**

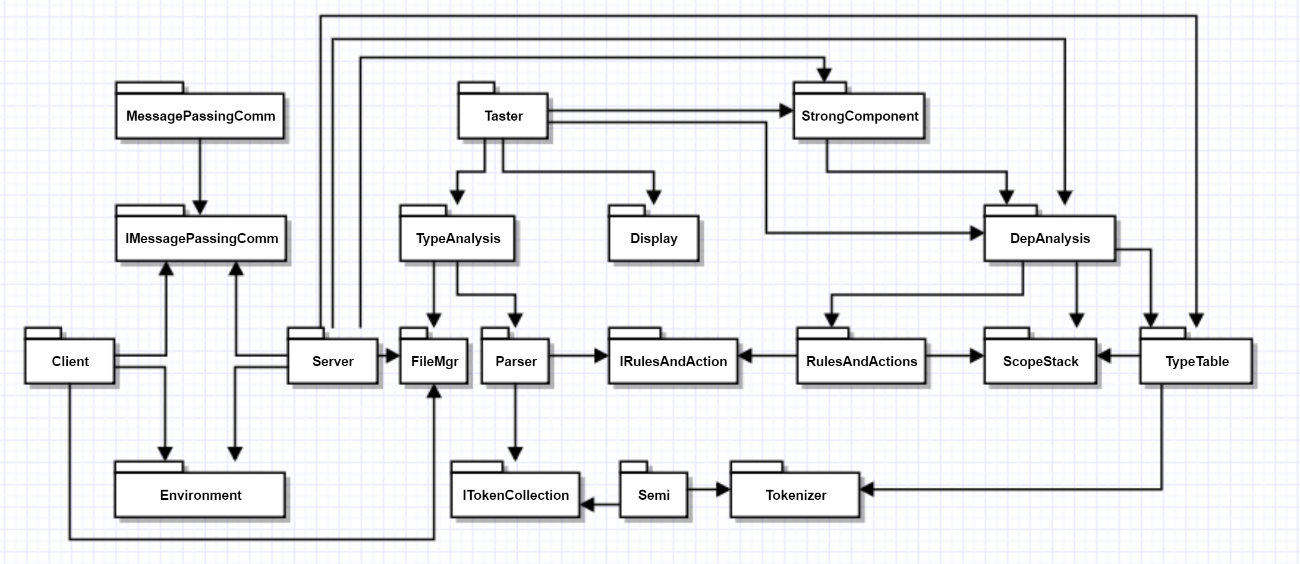


Figure 1: Package diagram

## *4.1 Tokenizer*

This package is responsible for opening target files, tokenizing every symbol with respectively tokens. Each symbol is endowed with its own properties. And in this package, there are two interfaces, ITokenState and ITokenSource. ITokenSource is the interface between source files and program, it facilities the reading string stream. The ITokenSource is a classification tool, which endows every symbol a state and this interface is implements by the derived classes.

Toker class implements the State Pattern with the derived state classes collect the below token types:

1. Alphanumeric tokens

2. Punctuator tokens

3. Special one and two-character tokens with defaults that may be changed by calling two special classes

4. Single-line comments returned as a single token, e.g., //

5. Multi-line comments returned as a single token, e.g., /\* ... \*/

6. Quoted strings

## *4.2 SemiExpression*

The input of this package is the output from Tokenizer package. The main function of this package is outputing the useful strings. By calling back the method GetTok(), this package sorts out tokens, and according to the dictionary, which should be outputted and which should be ignored. The SemiExpression terminates a token collection after extracting any of the single character tokens: semicolon, open brace, closed brace. Extracting newline if a 'using' or '#' is the first token on that line.

## *4.3 TypeTable*

The typetable provides a container that stores type information needed for dependency analysis. It is a table that stores class, structure, enum, interface, alias and delegate. By recalling the tokers and typeanalysis, to match the every toker with the codes in files, when the toker is matched, the results will return and the typetable will add a new list of this kind and show the file names and namespace.

## *4.4 TypeAnalysis*

Typeanalysis finds all the types defined in each of a collection of C# source files. It does this by building rules to detect type definitions - classes, structs, enums, and aliases and so on. It recalls semiexpression and parser, to divide every toker into right class, structure, enum, interface, alias and delegate. When parsing the files, it also records begin lines, end lines, types, complex and so on. With the help of typeanalysis, the sketch of whole files is clearly and better for users’ understanding.

## *4.5 DepAnalysis*

The dependency analysis finds, for each file in a specified collection, all other files from the collection on which they depend. File A depends on file B, if and only if, it uses the name of any type defined in file B. It might do that by calling a method of a type or by inheriting the type. Note that this intentionally does not record dpedndencies of a file on files outsied the file set, e.g., language and platform libraries. It shows the dependency between files and files. Once a class, structure, enum, interface, alias and delegate are defined in file A, and are used in file B, so we can define that file B depend on file A. The depanalysis package analyze the whole selected folder and create a dictionary to store the analysis results.

## *4.6 StrongComponent*

A strong component is the largest set of files that are all mutually dependent. That is, all the files which can be reached from any other file in the set by following direct or transitive dependency links. The term 'Strong Component' comes from the theory of directed graphs. There are many algorithms for finding the strong component in our program, but here we use the Tarjan Algorithm, by using this algorithm, first we need create a graph based on dependency analysis, tracking every node in the graph. Finally, we store the result in a list for convenient recalling and showing.

## *4.7 Display*

The display class here we only use in the tester, because there are so many packages. in every package, there exists different type of variable. The display class was only used in tester for convenient show the results.

## *4.8 Tester*

This is a package for testing the combination of many packages. As for above packages, there also be a test-sub inside each package. However, for testing whole program, it is necessary to create an extra Test package to check the stability of whole system.

## *4.9 Comm*

The Comm package implements asynchronous message passing communication using the Windows Communication Foundation Framework (WCF), which provides a well-engineered set of communication functionalities wrapping sockets and windows IPC. The channel is created in this class to connect the sender and receiver so that the message could pass from another port. In the server, it creates a channel, so that the message from each other could be received by another.

## *4.10 Server*

A package residing on a remote machine that exposes an HTTP endpoint for Comm Channel connections. The Server implements typetable, dependency analysis and strong component and return the results message to the client.

## *4.11 Client*

A package, based on Windows Presentation Foundation (WPF), residing on the local machine. This package provides facilities for connecting a channel to the remote Server. This package provides the capabilitiy for sending requests messages for each of the functionalities, and for receiving messages with the results, and displaying the resulting information. By getting back the reply message, the client refresh the GUI and post the new received message for users, it facilities the operations of user and quick response.

# **5. Application Activities**

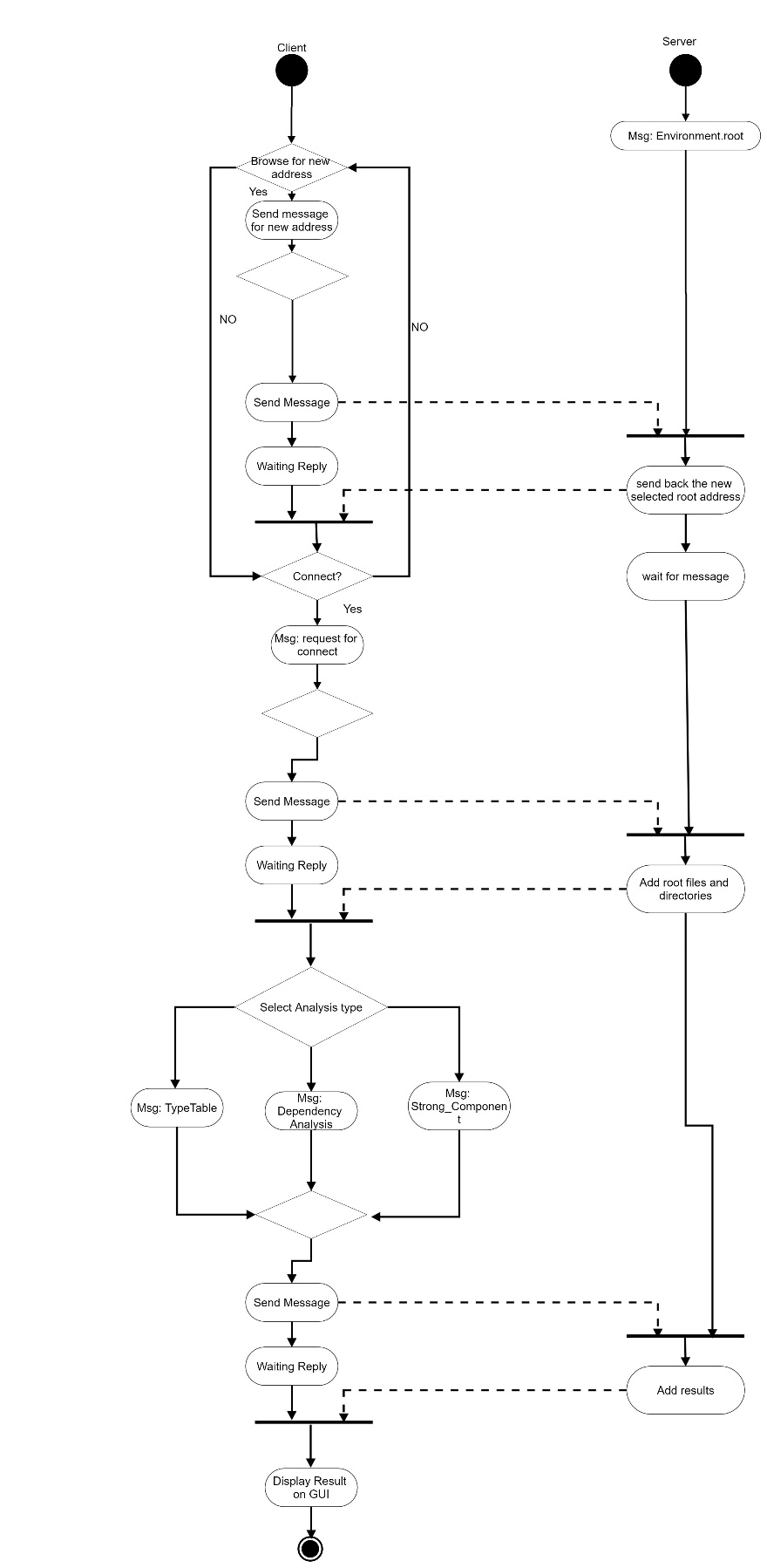


Figure 2: Activities diagram

The sketch of activities diagram shows the system working mechanism. This is the main working flow of Remote Package Dependency Analysis. At the first, if the users press the Browse button and choose a new folder, the root address will update, if the user don’t choose new address, the root address will remain default or the last stored value. Then the user should connect to the server, if not, the user can’t go to the next function. After the client connects to the server, the user could run three types of analysis in the server, and the server will return back the results to client, so that client could show the message to the user.

# **6. Critical Issues**

## *6.1 quote*

It is obviously to see that when meet double quote and single quote, as the class we define, we have two classes which is DoubleQuoteState and SingleQuoteState. There is a way if the code recognizes one quote and the next is not quote, the codes return the tokens of single quote. And if the next is also quote, the codes return the tokens of double quote. There are two ways in designing structure of classes of DoubleQuoteState and SingleQuoteState. The relationship between these two classes is parallel while another is progressive.

**Solution**: To simplify the structure of whole system, here we choose the relationship of parallel, because if we imply the parent relationship between these two classes, the whole structure may seem more complexity. This thought is also applicable to other classes.

## *6.2 Ignore termination*

One of rules we design to terminate an expression is when meeting the semicolon. However, a for loop contains two semicolons and in such a situation we want the expression stops when meet the right parenthesis.

**Solution**: define a new class which has method especially for each special expression.

## *6.3 GetTok()*

In this program, the GetTok() is an important method, to return the string back to the TokenContext, however, the problem will come if the last token is still not be removed and another new string comes in, the result will produce a chaos of expression sorts.

**Solution**: Thus we put GetTok() into every single state classes instead of using one single GetTok() from Toker class, it lighten the burden of classification and will eliminate the error.

## *6.4 Toker invariant:*

To satisfy some special operator like “==”, “++”, “--”, etc., and keep the invariant of input parameter and returned parameter, our peek process and the method of passing parameter should be set as a fixed rule. It is shown as below:

1. Each derived state enters getTok() with its char type at the beginning of the token sourcr.

2. That happens because nextState() method returns the state matching the first chars in the source.

3. GetTok() promises to remove only it’s characters from the token source.

4. It ensures that by peeking into the token source before removing a character.

5. This requires a token source that can be peeked at least two characters deep without removing any of them.

6. Neither .Net nor C++ file streams support more than one-character deep peeks, so we need to build a wrapper around them to provide this capability. That’s what TokenFileSource does.

## *6.5 Typetable extract method*

In the design, we meet that how to extract the typetable, there exists two ways, maybe there will be more ways we even don’t know, the compared string should recall the Toker or Semi.

Solution: the solution is that we choose to recall the toker, because we think that if there is any same toker exist in the files, so that class, delegate or others catalog exists in this file, and toker is much smaller unit than semi, and it may be more convenient to further use.

## *6.6 Dependency relationship*

How to handle if a class in a file recall the interface in another file?

Solution: if the class recall the interface in another file, here we use toker to handle this, so that the interface name appears in another file, so that the file which the class exists depend on the file which the interface exists.

## *6.7 The message pass overload*

Will the message overflows when passing?

Solution: when passing the message, the maximum of volume is 1024 bytes, so that we need store the content dividedly into the arguments to avoid the overflow.

## *6.8 The temporary files*

When running the analysis, the whole files will be imported, included some temporary files, will they influent the results?

Solution: These files is different files, and they exist everywhere, so there is no need to process them.

## *6.9 GUI performance*

The list box is better than text box for showing text? Which is better?

Solution: the text box is specified for showing text however the list box is used for choosing items, so the performance of text box is better than list box.

# **7. Prototyping**

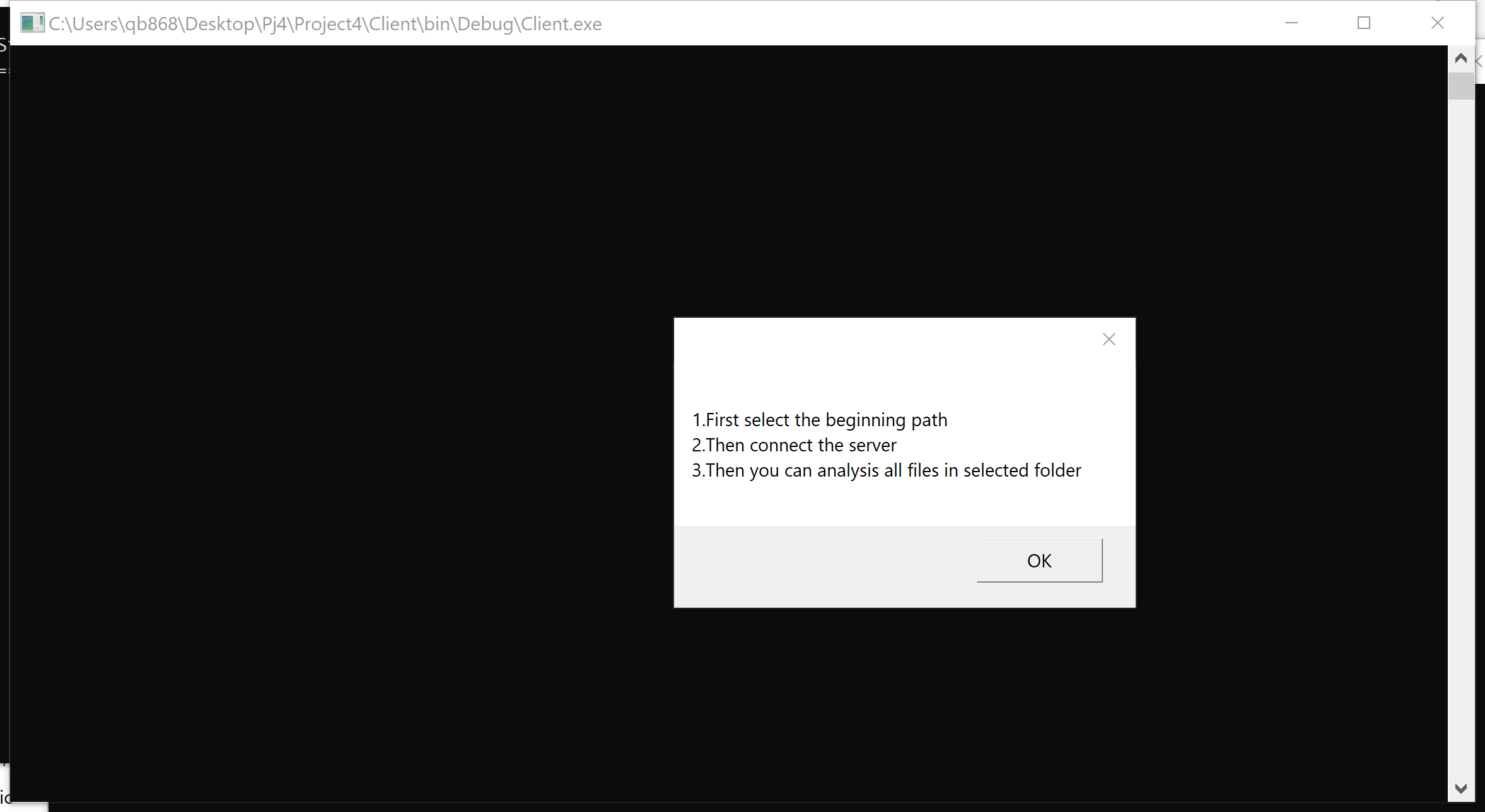


Figure 3: Message box

The beginning of client will pop up a message box, note that the uses of client, and by reading it carefully, user could know how to use the client GUI.

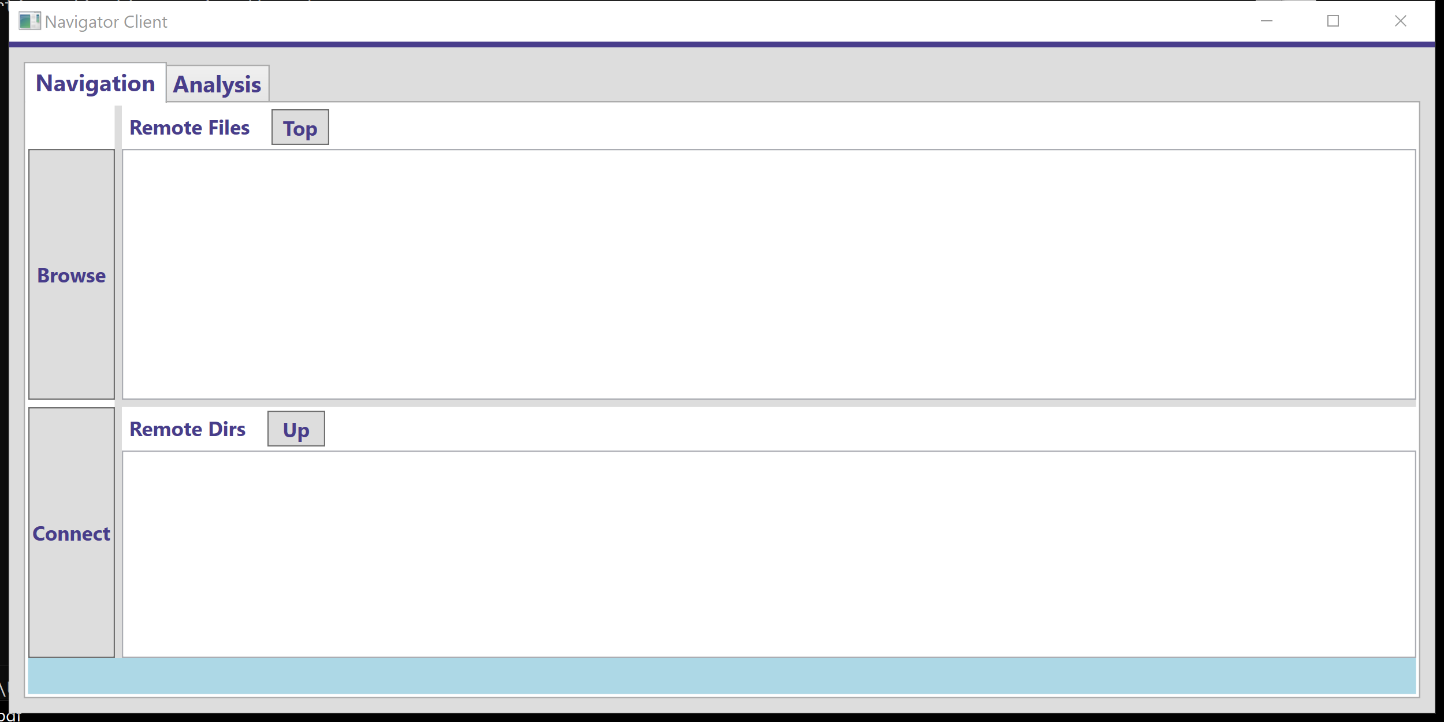


Figure 4: Main Window

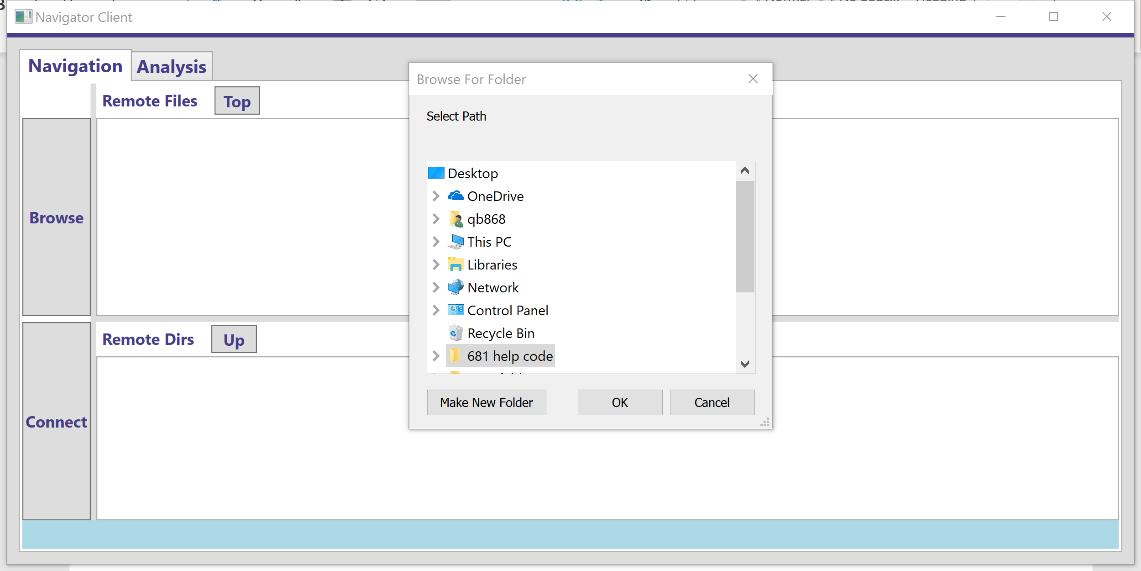
This is one of the tab item in the main window of client. It includes two table items, two labels, two list boxes, four buttons and one text box. The table items let the users to go to another page of this window and use other functions. The Top button let the directory go back to the root address. The Up button let the directory go to upper directory. And the Browse button let the user to choose root folder. The Connect button should be pressed, if not, the text box will tell you that the client is not connecting to server.

Figure 5: Folder dialog

By pressing the Browse button, it pops up a new window and let users to choose a new root folder, if the users don’t choose or close the window, it doesn’t matter, the default root address is ”..\..\..\”, it means the root address is the solution address.

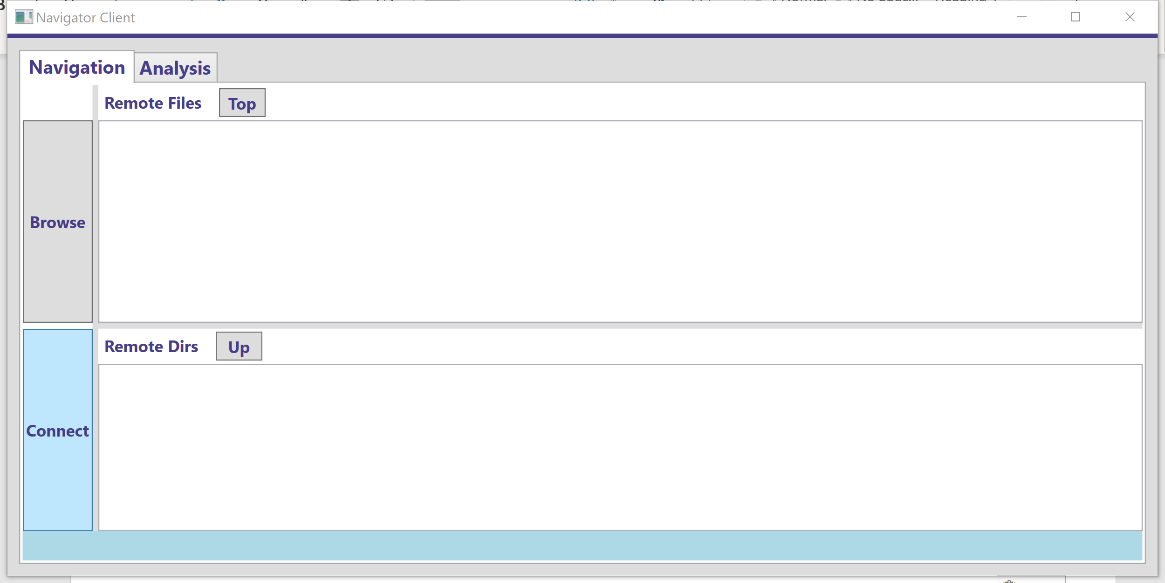


Figure 6: Connect First

The users need to connect to the server first, if not, they can’t use other function in this window. And Textbox below in light blue will also tell users to connect the server first.

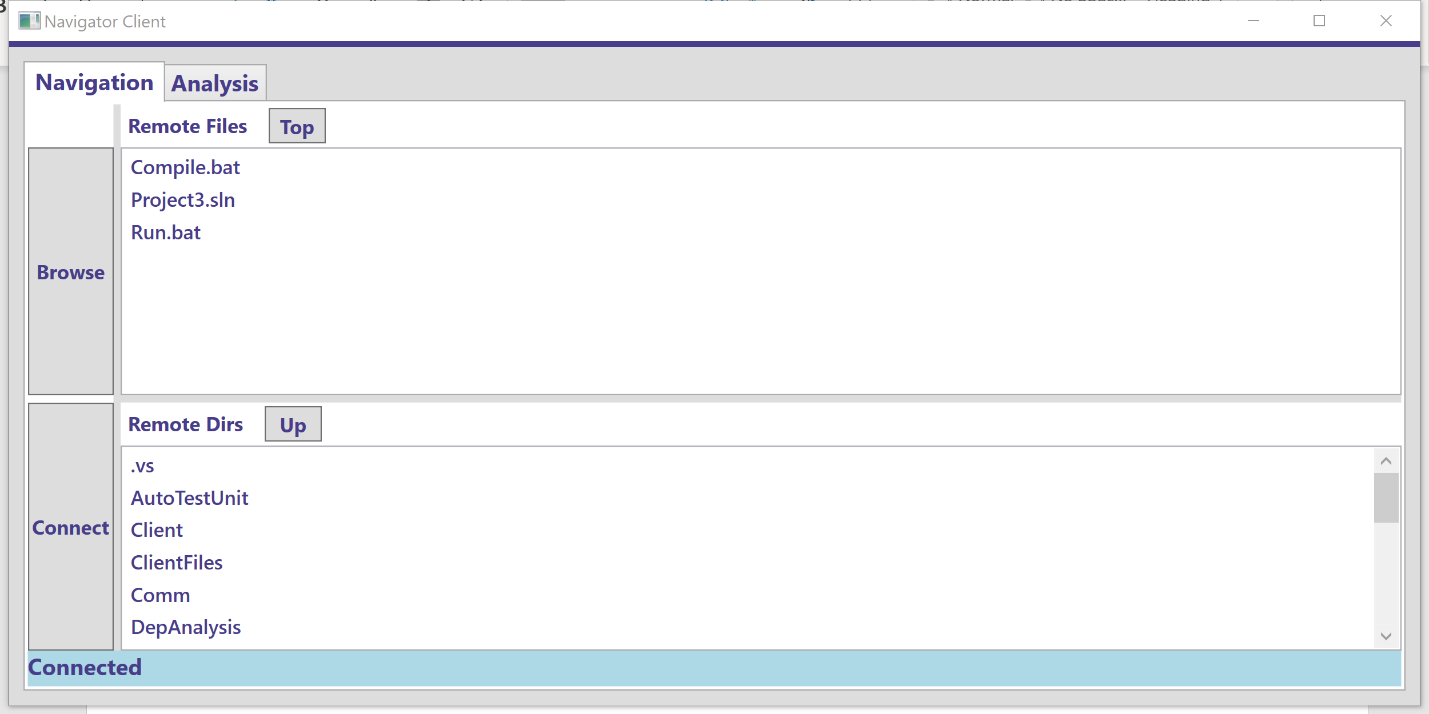


Figure 7: Show the files and directories

After connecting to the server, two list boxes will refresh the texts and show the message getting from the server, the message contains folders information and files information under the root address. The users could also change the root address again by pressing Browse button to choose a new address.

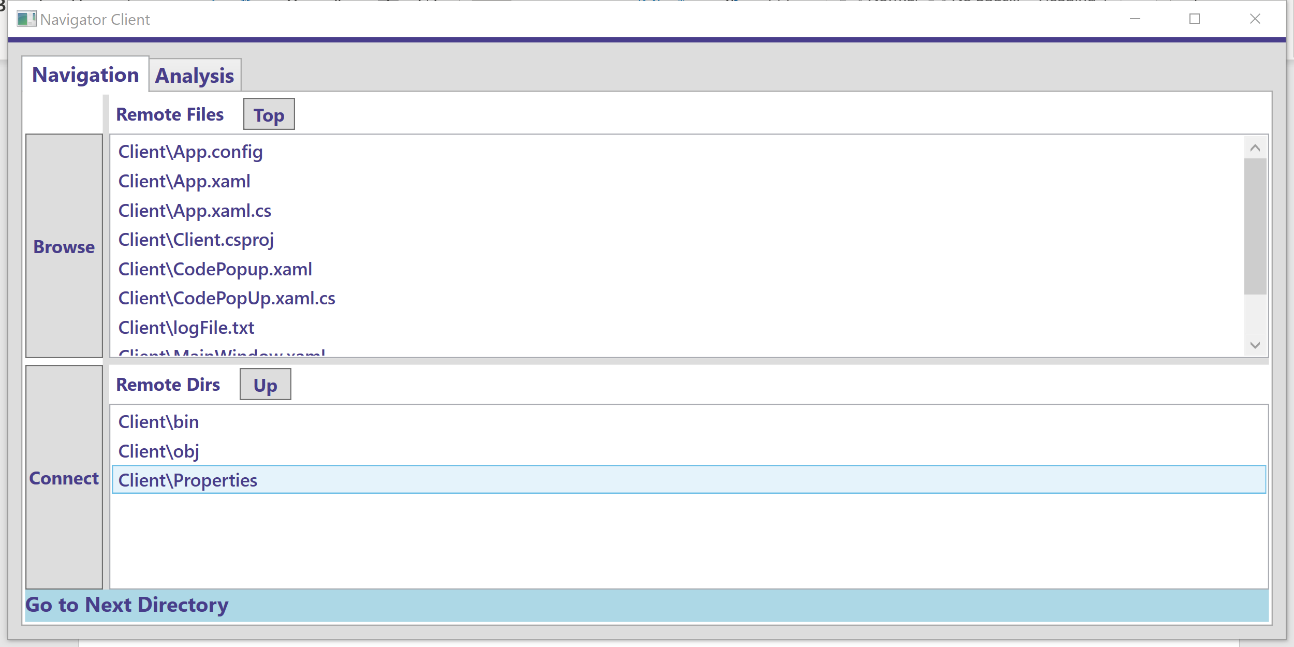


Figure 8: Go to next folder

By double clicking the directories, the client passing the request to server for sending the next folders and the files information, the server get the request and reply the new information to client, so the client get the information and refresh the list boxes. That makes we get into the next directory.

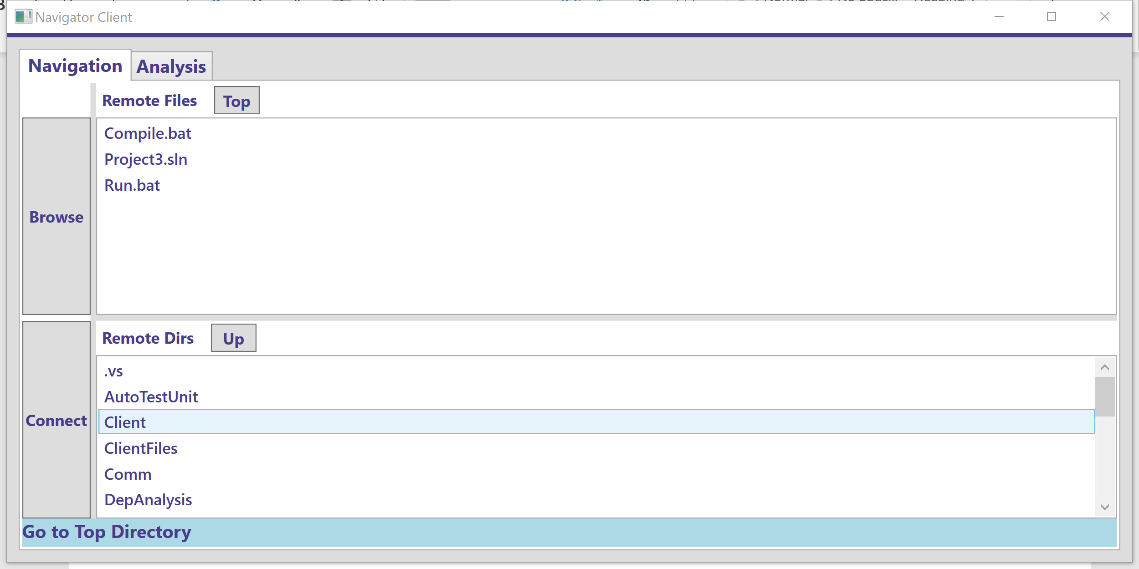


Figure 9: Back to the root folder

By pressing the Top button, the client sends the message to client for requesting the information of directories and files under the root address. The server replies the client and send back the message which contains the directories and files under the root address. And the client receives the reply, then refresh the list boxes and show that to users. The mechanism of Up button is same like Top button, however the different folders and files information.

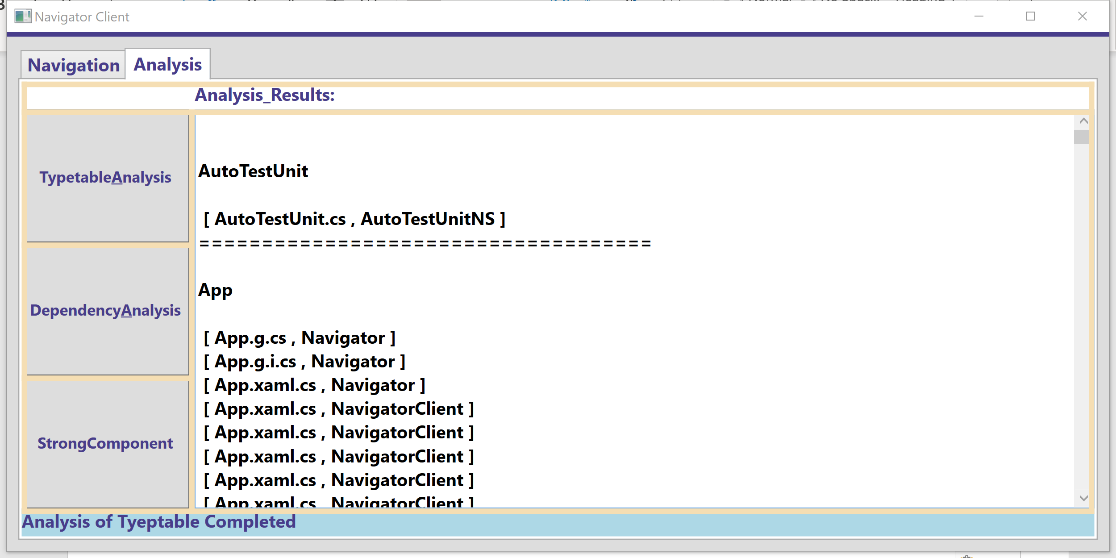


Figure 10: Analysis

This is another tab item in the main window. In this tab item, we have three buttons, two text boxes and one label. These three buttons is used for three types of analysis, typetable analysis, dependency analysis and strong component analysis. And the big text box is used to show the results. The picture above shows the results which are sent from server after typetable analysis. When pressing the button, the client sends the message to server requesting for information of typetable analysis of all files under the address. The server receives the message and recalls the typetable function and replies the message. The client receives the message and extract the arguments. The text box refreshes the information and then post them in the text box.

# **8. Reference**

<https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/Lectures/StudyGuideOCD.htm>

<https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/lectures/Project1-F2018.htm>

<https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/lectures/Project2-F2018.htm>

<https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/lectures/Project3-F2018.htm>

<https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/lectures/Project4-F2018.htm>

<https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/Project1HelpF2018/>

<https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/Project2HelpF2018/>

<https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/Project3HelpF2018/>

<https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/Project4HelpF2018/>

<https://ecs.syr.edu/faculty/fawcett/handouts/Webpages/BlogParser.htm>