CSE – 681 SOFTWARE MODELING AND ANALYSIS

Instructor – Dr. Jim Fawcett

PROJECT #4

REMOTE PACKAGE DEPENDENCY ANALYSIS

AMRUTA JOSHI
SUID – 744991145

NOVEMBER 5, 2018

Table of Contents

1. EXECUTIVE SUMMARY

This document represents the Operational Concept document of Remote Package Dependency analysis. In large industries where there are millions of lines of code being processed and executed, it is important for the compilers to work efficiently. The compilation process is divided into many smaller programs to minimize the memory capacity challenges faced by the computer systems. Whenever the source code is being entered in the computer system, first task the compiler does is that it performs the proper analysis of the source code.

In order to work on big systems we need to partition the code into smaller parts and then test each of the parts before inserting them into baseline. Soo as the new small parts are entered in the baseline we again need to fix the latency errors and run the test sequences for the entire baseline. Soo thus, code analysis does this job. Code Analysis is done for effective management of the processes as well as for testing of the applications so that the proper functioning of code is done large scale level. A toolset is required to inspect the various aspects of code analysis. This document explains the entire architecture and working of the Code Analysis toolset which includes:

- Extracting of the lexical content from source code files
- Analysing the syntax from its lexical content
- Based on the syntax, building appropriate type table with the help of which dependency can be identified between different files.
- Additionally, we can also find the strong components between various different files based on the dependencies between the files.
- Using the client server architecture to find the dependencies between the remotely located files.

This project majorly encompasses the whole software development process lifecycle such as:

- Meeting the requirements of the client
- Designing the software
- Coding
- Analysing the source code
- Change Management
- Testing
- Integration

The intended users of this project will be developers, quality assurance staff, testing team, Maintenance team, virtual assistant. Developers will use this toolset for analysing the code written by the other developers, on which they have to work. This work can be done remotely. The Quality assurance staff can perform the analyses of the code and can run on the remote repository from the clients on their desktops. Testing team can test the code on their remote machines so that it consumes less efforts. Maintenance team can perform the analysis of the code and can use the tools to fix the errors and make changes in the code by working independently from their own remote machines. Virtual assistant can also use the tools of the code analysis to analysis the code functionality.

The later section will give the brief and detailed explanation about the uses of this project, the users, critical issues related to the project. Also, the further parts will explain the project with the help of activity, class and package diagram.

Some of the critical issues associated with the project are given below. These issues needs to be identified and the solutions to resolve this issues are explained. We need to address the following types of issues :

- In context to project 2, handling of the special cases while processing the Tokenizer.
- How to peek for more than 2 characters as .NET streams does not supports peeking for 2 characters.
- In context to project 4, some of the issues:
  1. Traffic Congestion
  2. How will the client understand where to send the messages ?
  3. Robustness
  4. Extensibility
  5. Design

2. INTRODUCTION

In project 2, we had worked on the small part of this project named Lexical Scanner using State Based Tokenizer. In that project we had created various methods to open the file, collect the tokens from the file using some rules in Tokenizer package, passing the tokens to SemiExpression package and collecting the semiExpressions along with their special cases.

Based on project 2, in project 3 we created more packages namely TypeTable, TypeAnalysis, DependencyAnalysis, StrongComponent, Graph. We had created the TypeTable to store the types information. The Dependency Analysis package will be used to check the dependencies between the different files. Strong Component package will be used to find the strong components between the files. This will be dependent on dependency analysis package.

After working on these small parts, the main aim of the project was to integrate all the small parts to build a big subsystem. In Project4, we had integrated all the packages which we created in project 2 and project 3. Along with that, we have also created some new packages namely Client package, Server package and Comm package. The architecture and working of this project is explained further in detail.

The next section describes the overall architecture of this project. We will discuss about the communication mechanism between the client and server machines. The client system has the GUI that displays various functionalities for sending and receiving messages along with the results and also displaying the result. It provides the facilities for connecting a channel to the remote server and then sending the request to the server. The server system will consist of all the operations to be performed whenever it gets requests from the client. Basically, in short terms it is the package which will be residing on a remote machine that exposes on HTTP endpoint for Communication channel connections.

All the communication between the machines is done through the comm package which implements the asynchronous message passing communication using Windows Communication Foundation (WCF). The WCF provides a set of communication functionalities such as wrapping sockets and windows IPC. The Comm package is explained in detail in further context.

2.1 ORGANIZING PRINCIPLE

The organizing principles of this project will be useful will finding the dependencies between the files on client and server machines. It will be also helpful will providing solutions to the critical issues which will be explained further. There are many new packages included in this project namely Comm , client, server.

2.2 KEY ARCHITECTURAL IDEAS

The key idea to build the project is to work on the remote and local machines using client and server. The project mainly is the integration of all the three projects. In this project we have to establish the connection between client and server. This is done by Comm package. The comm package consist of the address and port number of the sender and receiver. The client sends the request to the server using a channel. This is done by postMessage method and getMessage method. This two methods are explained in detail in further sections. The Server contains all the files. The client requests to perform certain operations on these files. So the main aim of client server architecture is that the client sends the request to the server, the server should send the acknowledgement back to the client. All this process of sending and receiving of messages is explained further in detail. We have implemented these using WCF and WPF concept.

3. USERS AND USES

This section describes in detail the users and uses of this project. It is most important that all the users can interact with the system to get the necessary information without any complex procedures. The system should be easy to use so that all the users can use it and can obtain the information in the simple and effective manner.

USERS:

1.1 QUALITY ASSURANCE STAFF

- The quality assurance staff can analyse the quality of the code by running the code in their remote repositories from clients on their own desktops. Thus, it is very easy for the staff members to analyse the quality of the code without spending a lot of time.

1.2 DEVELOPER

- A developer team will analyses the code. They are responsible for the development of the project. Suppose, there are more than 1 developer working on the same project then it sometimes becomes difficult to analyse the code. Thus, developers can use the code analysis tools to analyse the code through remote repositories.

1.3 TESTING TEAM

- The testing team will test the code to check various functionalities. The tools of code analysis can help to check the dependencies between the files. Thus, the testing team can check dependencies of the remotely located files from clients.

1.4 Pattern Recognition

- It is used in pattern recognition and input verification systems.
- This can be done on remote machines.

4. PARTITIONS

In this section, we will discuss about different packages which we are using while building this project. We will also discuss about different functions these packages perform and also their interactions with other packages. The package diagram

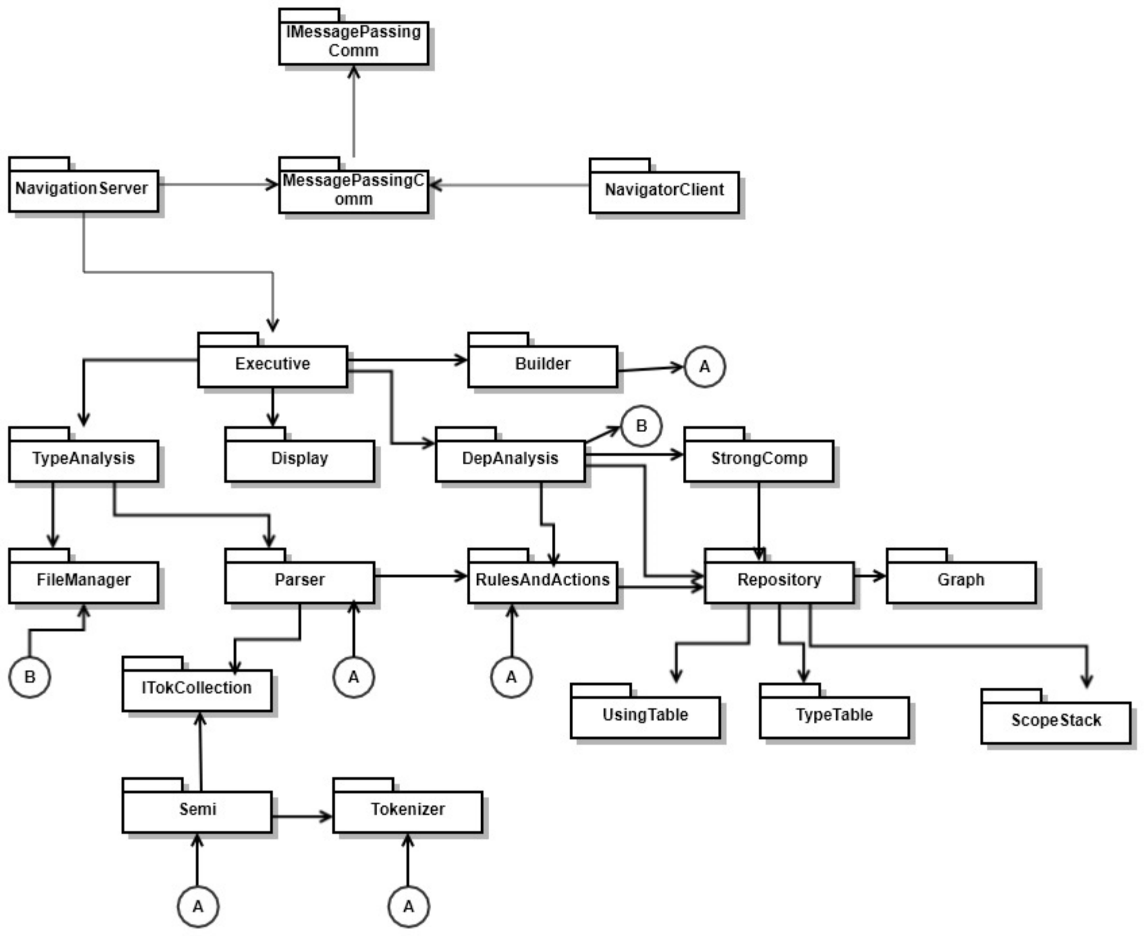


FIGURE 4.1.1- PACKAGE DIAGRAM FOR REMOTE DEPENDENCY ANALYSIS

So, the package diagram consists of the following packages:

1) Executive :
   - The executive package is the main starting point of the project. It performs testing of various different applications.
   - It points towards type analysis , display, dependency analysis methods.
   - Basically, we can directly go to these packages from the executive package.
   - It displays the output of various packages.

2) Type Analysis
   - This package stores the type information from all the files
   - For storing the type analysis from different files, we require a container named typeTable.
   - Thus, the files which are of types such as class, namespace, alias, delegate, enum, struct are stored in the typetable.
   - This package then further points towards the parser and FileMgr.

3) Parser
   - This package contains all the files which needs to be analysed
   - We store all the files which needs to be analysed from the arguments,
   - In this package, two methods are created ProcessCommandLine and ShowCommandLine.
   - ProcessCommandLine will store the list of files which needs to be analysed
   - ShowCommandLine shows the files which is required for the analyzation purpose.
   - Once the required file is opened, proper operations are performed to parse the file and we get the resultant tokens and semiexpressions.

4) Rules and Actions:
   - This package defines the rules and actions which needs to be performed on files.
   - Different actions to be performed are:
     a) PushStack – This action pushes the element with type and name onto stack. The elements are then stored in the repositories. It also stores the start line number and the scope count.
     b) PopStack – This action pops the element from the stack. Thus, the element which is popped out from the stack is removed from the repository.

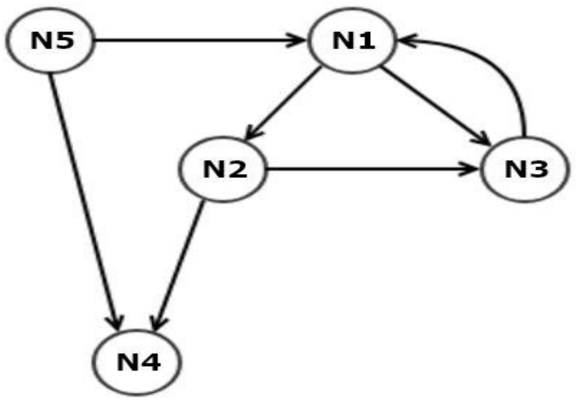
     It records the end line number and the scope count.

     c) PrintFunction – This action prints the semi's present in the repository along with the line number
     d) SaveDeclaration – This action saves the types, namespace from the semiExpressions.

- Different rules defined are:
  a) Detect NameSpace
  b) Detect Class
  c) Detect Function
  d) Detect AnonymousScope
  e) Detect PublicDeclartaion
  f) Detect LeavingScope
  g) Detect enum
  h) Detect delegates
  i) Detect struct
  j) Detect alias

5) Dependency Analysis –
   - Based on the types stored in the typeTable we can find the dependencies between them.
   - This can be done if and only if one file uses the name of any type defined in other file.
   - For example, file A is dependent on file B if and only if it uses the name of any tye defined in file B.

6) Strong Component:
   - A strong component package depends on Dependency Analysis.
   - Whenever a file can be reached from any other file by the set of direct dependency link then we can find the strong component.
   - The concept of strong components comes from the directed graphs.



   -

   - Thus, to implement the strong component we use the graph class. We can implement this by using the Tarjan's Algorithm.

7) Comm:
   - This package implements asynchronous message passing communication.
   - This package consist of sender, receiver and Comm classes.
   - The sender class constructs sender using address and port. It also contains the methods like postMessage, postFile.
   - The receiver class constructs of the receiver instance. It also consist of postMessage, getMessage methods.

8) Client :
   - This package defines the WPF application processing of the client.
   - It provides the navigation into subdirectories in both the local client and remote server.
   - The GUI of client is also maintained in this package.

9) Server:
   - This package consist of the message dispatcher that will handle the processing of all the incoming and outgoing packages.
   - The operations to be performed like TypeTable analysis, dependency analysis is also stored in this package.
   - In this package the To, From, Arguments , Command methods are used based on which the appropriate actions are performed.

4.1 Class Diagram:

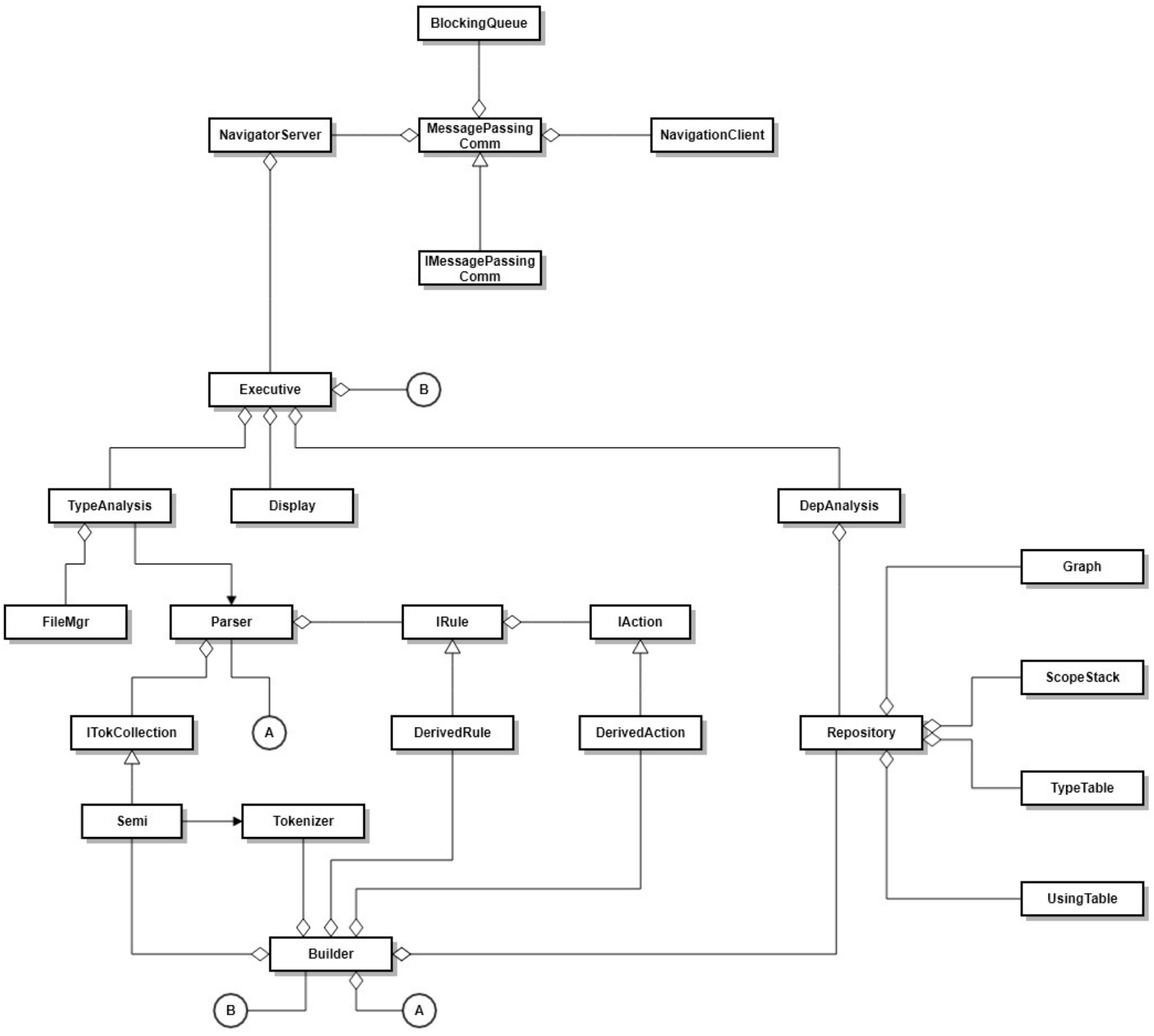
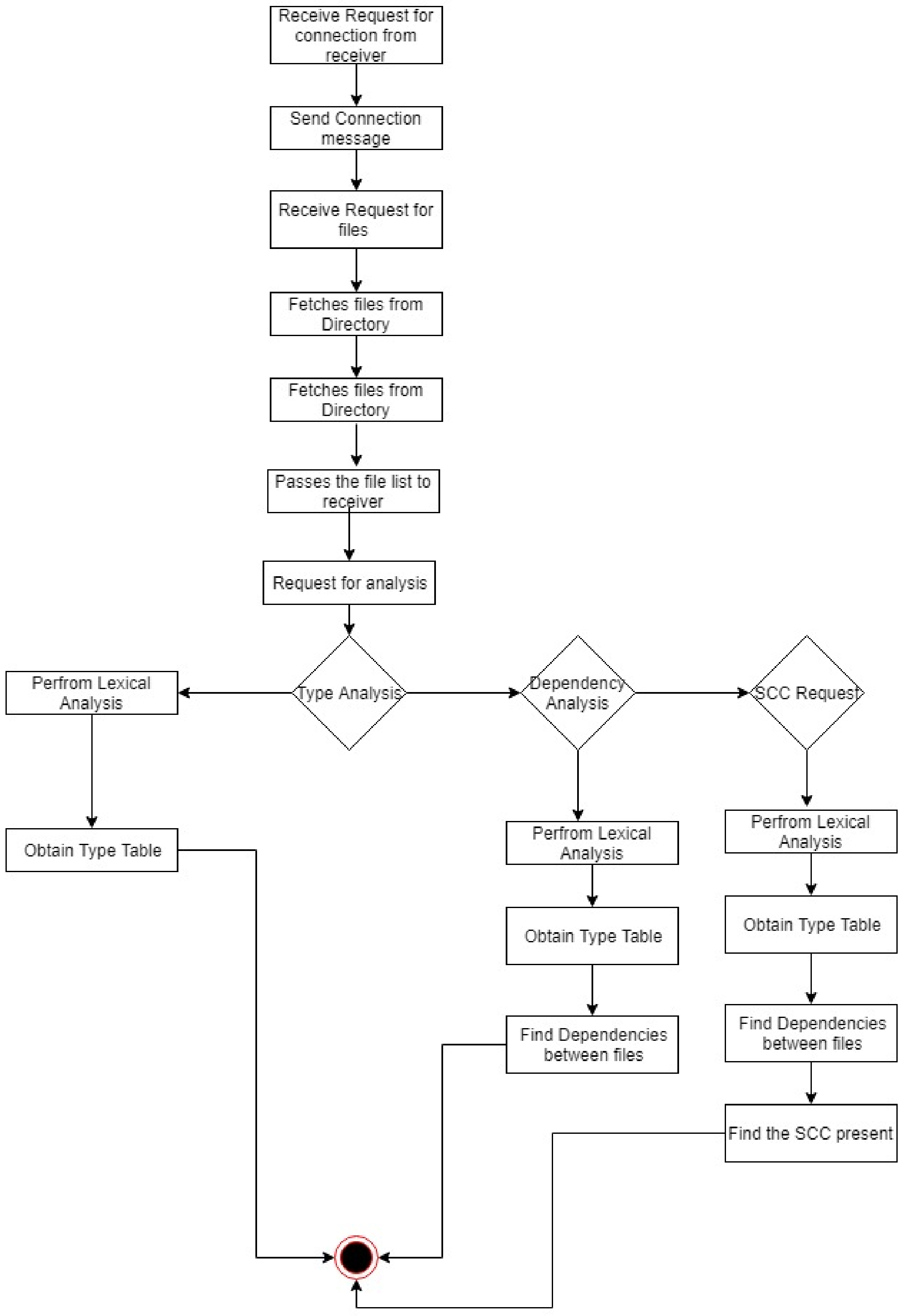


FIGURE 4.4.1- CLASS DIAGRAM FOR REMOTE PACKAGE DEPENDEANALYSIS

5. APPLICATION ACTIVITIES

The activity diagram describes the flow of activities which will occur in the project that we will build. Following are the steps for the entire system:

Activity Diagram from
Server's View



6. CRITICAL ISSUES

- 6.1 Performance

The performance is determined by the efficiency of the compilation and compiler portability. It should be fast so that compilation time is less, and it can be used by other applications effectively.

Solution: -

Performance issue can be solved if there is proper interaction among the packages. The client should send the proper request to the server. The server should send the acknowledgment back to the client. Thus, To, FROM, ARGUMENTS, COMMAND should be properly written.

- 6.2 Traffic Congestion

Traffic congestion is one of the issue in client server architecture. It may be possible that the server many be busy handling some other request and the client goes on sending new requests to the server. This will lead to traffic congestion.

Solution:

The queue should be maintained for handling the request at the client side. It should be designed in such way that the server should not get flooded with the requests. All the requests should be maintained in the queue.

- 6.3 Complete Dependency on defined set of rules

The dependency depends on some specific rules. To show the strong component and graph first the files should be dependent.

Solution:

Define the specific rules from which the dependencies between the files can be easily known.

7 REFERENCES

1) http://ecs.syr.edu/faculty/fawcett/handouts/CSE681/
2) https://en.wikipedia.org/wiki/Lexical_analysis
3) http://ecs.syr.edu/faculty/fawcett/handouts/CSE681/presentations/notation.pdf
4) Tokenizer Code given by Prof. Jim Fawcett.