CSE-681 Software Modeling and Analysis
Instructor- Dr. Jim Fawcett

Project #4

Remote Package Dependency Analysis: Operational Concept Document

Rucha Guntoorkar
SUID 453497450

Contents

1. Executive Summary

The Remote Code Analyzer will extract the lexical content from source files, analyzing its code syntax, perform the dependency analysis using the typetable obtained from type analysis and finding strong components. Client gives the input files by browsing the subdirectories.

The user or client browses the subdirectories and select the files. On clicking the button, request message is sent to the server through message passing queue. Server and client are connected through comm. whenever the sender sends the request it is posted to sender queue. It is then popped from queue and sent to the receiver. The message is added to receiver's receive queue, which then picked by the receiver. The client can request for get files, get directories, dependency analysis and strong component. Whenever the client request for dependency analysis, type analysis is performed first on the semiExpression return by the lexer. While doing type analysis, semiExpressions are checked against the rules in the parser. Typetable holds the result of type analysis.
The files are again passed for the dependency analysis. This time semiExpression are checked against the typetable. The result is stored in the dependency table. Graph is constructed using the dependency table and strong component analysis is done using Tarjan's algorithm.

This project will provide the following capabilities:

- Client can browse through the subdirectories and files from server side
- Client can request for strong component and dependency analysis, server files and directories by passing messages
- Server can get the client request and can send reply message containing the result of the request
- Server can perform dependency analysis, type analysis, parsing and finding strong components, lexical analysis, returning list of files and subdirectories requested by the server

The following are the users and uses of this project:

- Project Manager: to check the quality of software designed by developer
- Developers/students: for checking the dependencies between the packages, getting the strong components and analyzing each and every package from the code
- TA: to check whether all the requirements are met by the students and if the project is doable in the future

The critical issues for this project are given below and solution for the issues are discussed later in the document:

- The application should not be crashed because of the port. It should be flexible enough to select another available port
- Performance of the system can be degraded for large number of input files and client request
- If the messages are read by unauthorized person then it is a security concern
- The application should be able to handle the huge output size

2. Introduction

The software or any big system consists of large number of small partitions or packages. To successfully implement the software it is necessary to test each and every part of software carefully before implementing. When it comes to modifying or implementing new feature in the software, every dependent part of the project should be analyzed and tested carefully. In this case code analysis tool plays the vital role. This project will implement the Lexical Scanner, finding package dependency, parsing and finding the strong component which will help in code analysis.

2.1 Objective and Key Idea

In this project client will send the request for dependency analysis or strong component. The server after getting the request processed the requests using the input files and sends the reply message along with output of the request. Application will now emphasize on C# code but later it can be modified and make available for other languages as well.

2.2 Organizing Principles

This project will be implemented using C# 7.0 and .NET framework, WPF and WCF in Visual Studio 2017. This project includes following main components: remote server, client, FileMgr, MessagePassingComm, Environmnet. Client will be created using WPF and MessagePassingComm will be implemented using WCF.

3. Application Activities

Application activities give the activities for server side activities and MessagePassingComm along with the diagram.

3.1 Server Activities

- Server gets the request message from client which contains the command.
- Command can be to get strong components and dependency analysis for the input files from the client
- server reads the files from the directory

- Type analysis is performed on the given sets of the files after lexical analysis and type table is populated
- Dependency analysis is performed on the given sets of files and dependency table is returned
- If the message request is for dependency analysis then the reply message sends the output of dependency analysis
- Graph is constructed using dependency table and it is passed to strong components analysis
- Strong components are identified using Trajan's Algorithm
- For the strong component request the result is returned along with the reply message.

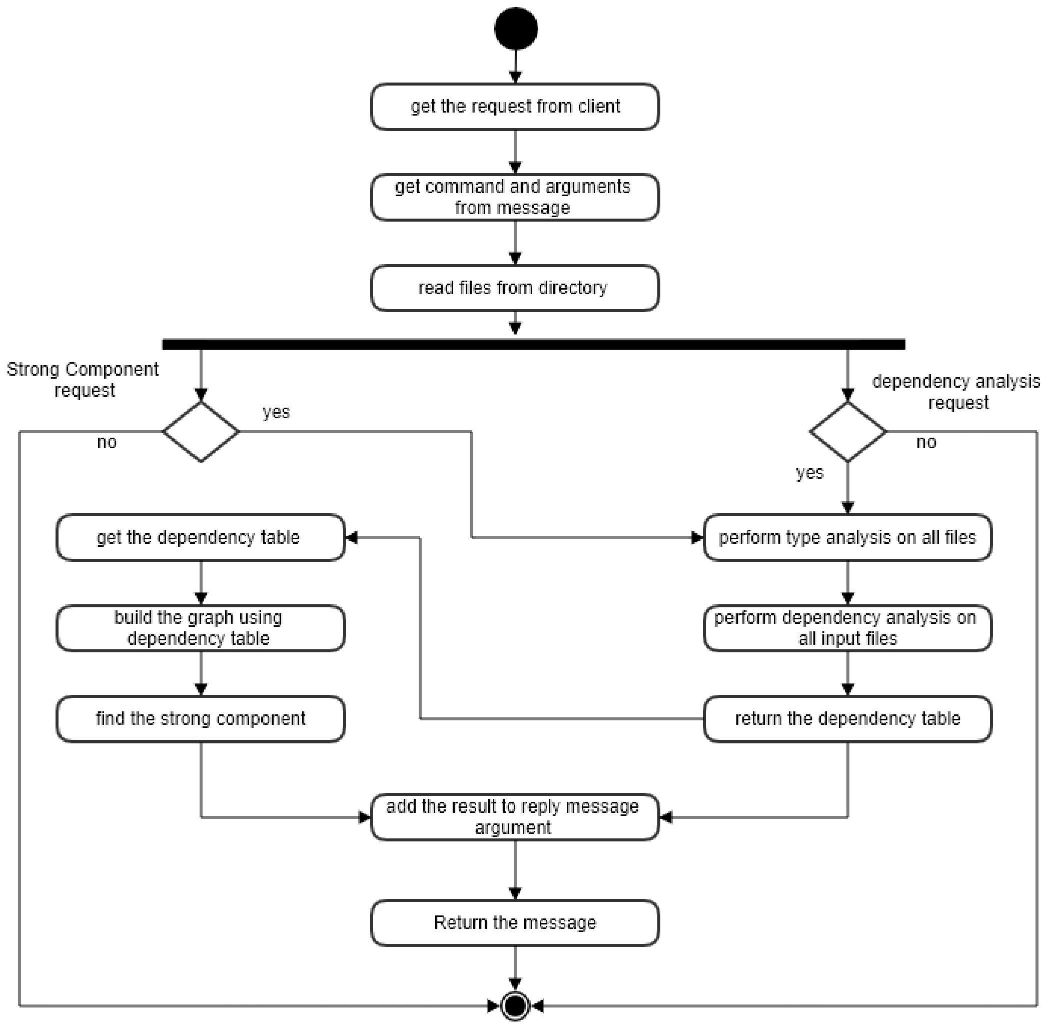

get the request from client
get command and arguments from message
read files from directory
Strong Component request
yes
no
dependency analysis request
yes
no
get the dependency table
build the graph using dependency table
find the strong component
perform type analysis on all files
perform dependency analysis on all input files
return the dependency table
add the result to reply message argument
Return the message

Figure 3.1: Server Activities

3.2 MessagePassingComm Activities

- When the sender posts the message it is enqueue in the sender queue
- That message is extracted from the queue and sent through the channel
- After establishing the connection it is received from receiver end it is added in the receiver queue
- Once getMessage is called the message is extracted from the queue

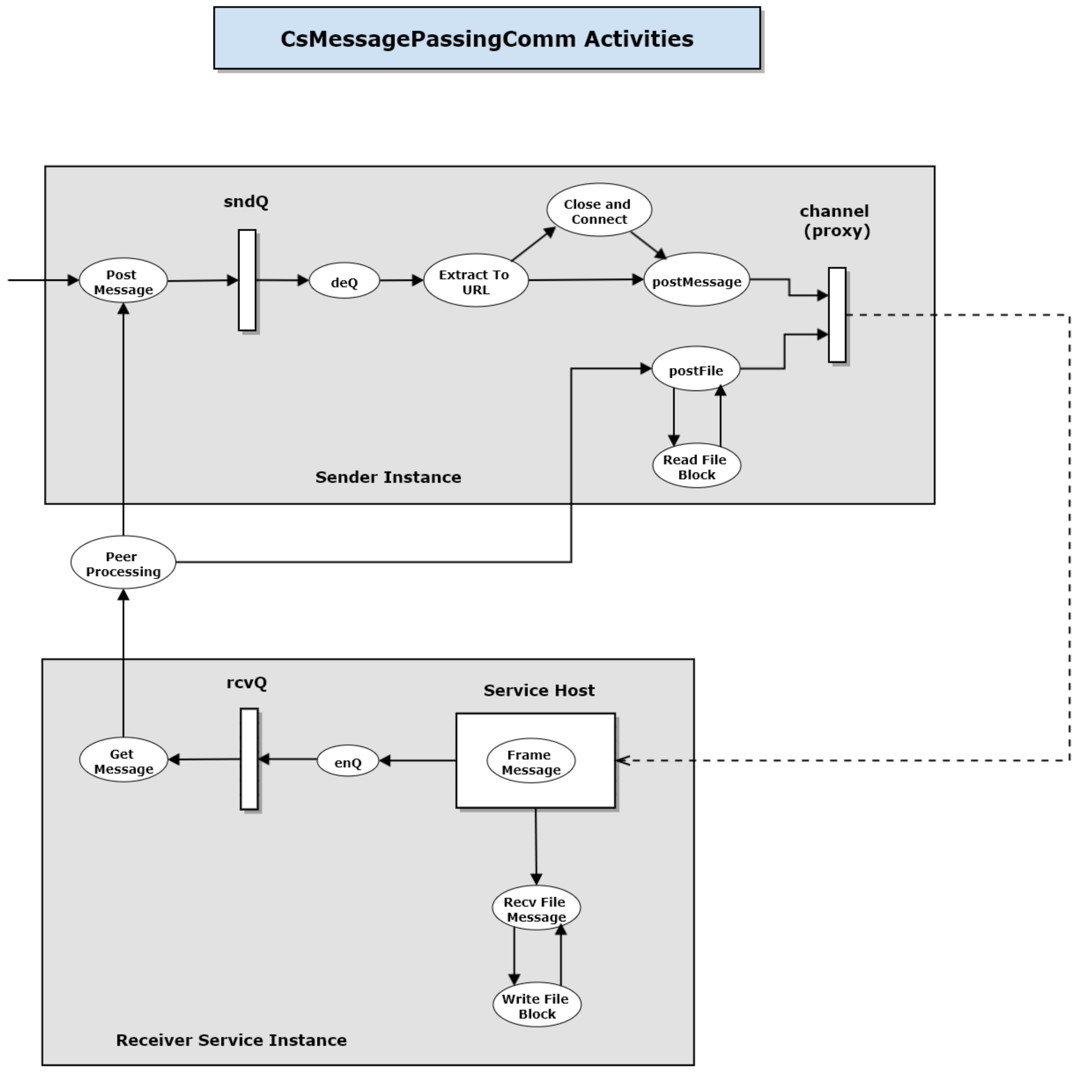


Figure 3.2: MessagePassingComm Activites

4. Partition

4.1 Packages

4.1. Application partition

The diagram gives the high level partition of application. It contains following

- **Remote Server:**
  It has all the functionalities of code analyzer. When the server gets the request it processes the command in the request and returns the output to the client.
- **FileMgr**
  It contains the file operations like get files and get the parent directories and returns the files and directory
- **Client:**
  Client is the GUI developed using WPF. It selects the files which is on the server sides and request for strong component and dependency analysis by clicking on the button
- **Environment:**
  Environment contains all the attributes required for establishing connection between client and server
- **IMessagePassingCommService:**
  It provides the interface which contains the operation and data contract required for WCF
- **MessagePassingCommService:**
  It provides the service. It provides the channel to connect client and server and implements message passing queue to send and receive the requests

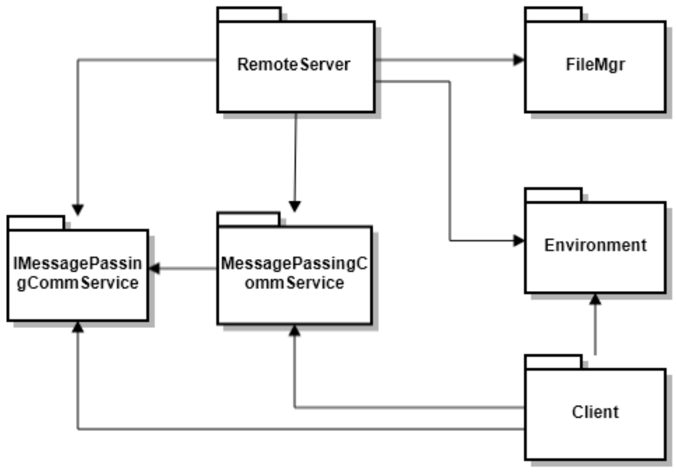
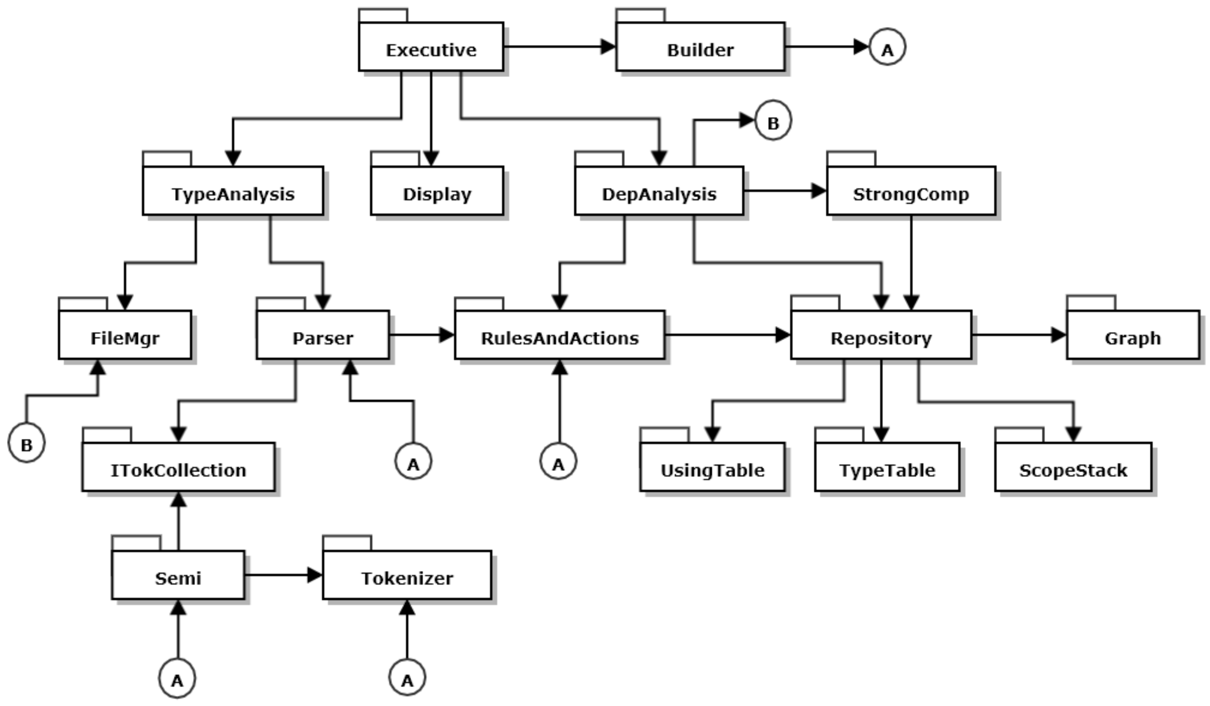


Figure 4.1.1 Application Partition Diagram

4.1.2 Server Partition Diagram

Following is the server package diagram which highlights all the packages in the server side.

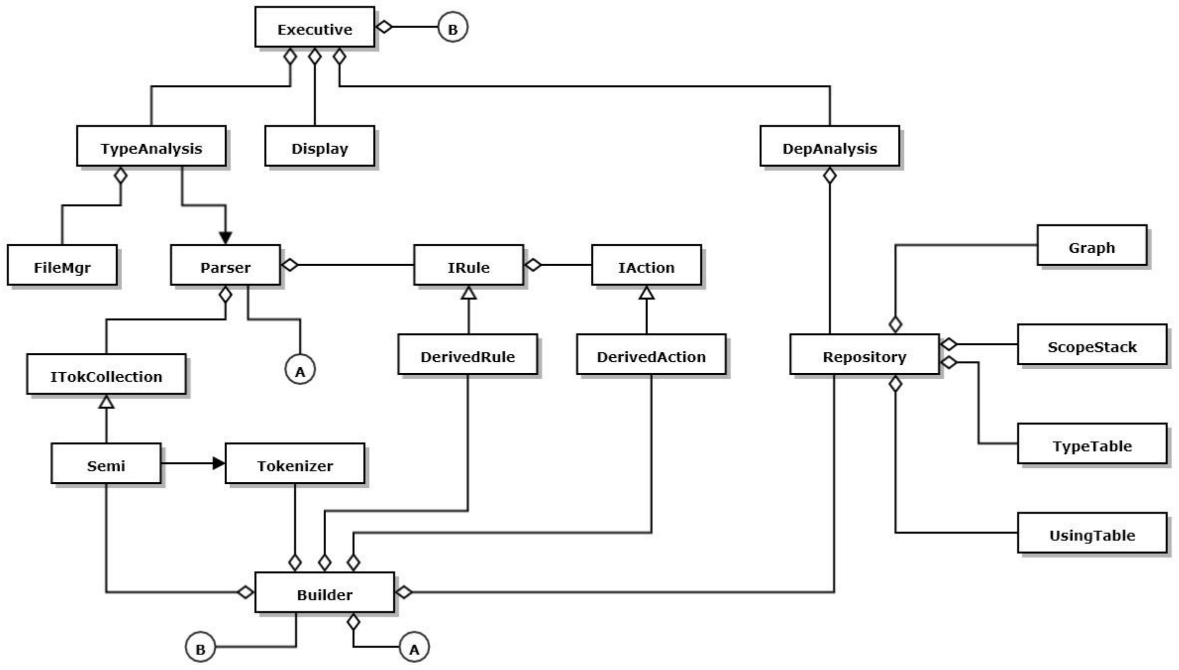


4.1.2 Server Package Diagram

4.2 Class Partition

4.2.1 Server Class Diagram

Following is the server class partition diagram which gives classes, interfaces and relationship among them.



4.2.1 Server Class Diagram

4.2.2. **MessagePassingComm Class partition:**

Following is the MessagePassingComm class partition diagram which gives classes, interfaces and relationship among them.

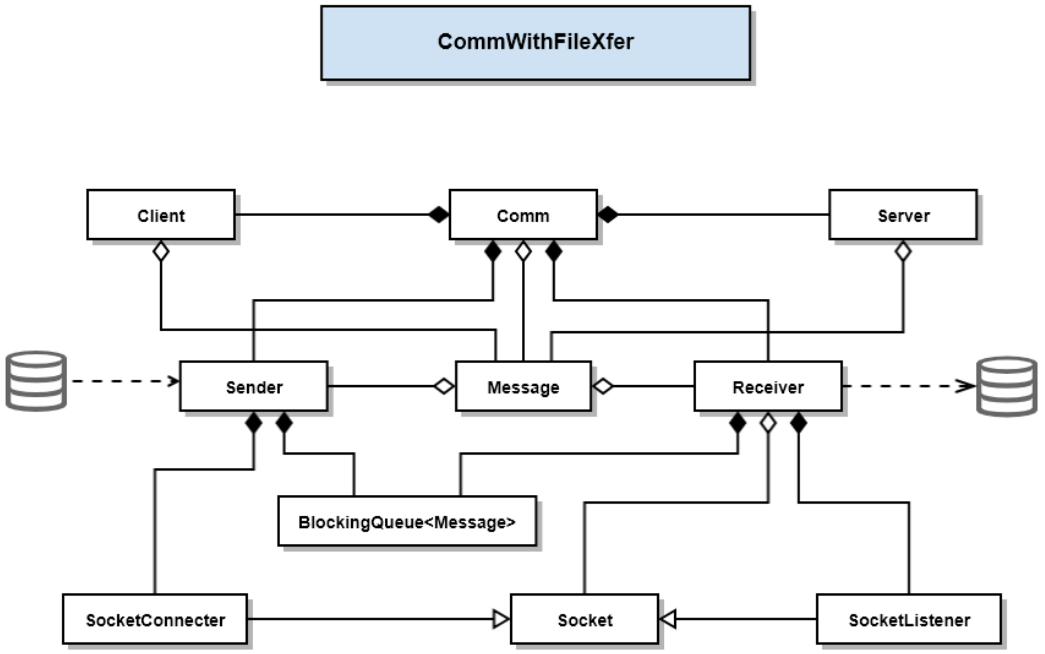


Figure 4.2.1 MessagePassingComm class diagram

5. Uses

- Developer:
  The developer can use this application to analyze their application to find the dependencies and the strong components. It will be useful to analyze the design efficiency of their project and to detect the flaws.
- Managers:
  Before handling the application to the client the manager can analyze the software using this tool for checking the quality of software.
- Student:
  By analyzing the application with this tool a student can understand the design efficiency of his project and can improve his project by checking the strong components and dependency analysis.
- Teaching Assistant:
  By using the parser functionality of the tool, a TA can check whether the student has fulfilled all the requirements.

6. Critical Issues

- While passing messages from client to server or server to client there are chances of connection failure as the port could be used by another process. In the case application will crash until the port becomes available
  Proposed Solution: There should be provision of port change when one port is busy on client and server side
- Performance can be the critical issue especially when client will send large number of huge files. Server will take long time to process the request especially for dependency analysis. Client will have to wait for longer time. Type analysis, Dependency analysis, lexical analysis can be implemented on different server
  Proposed solution: Multithreading can be used and each file can be processed simultaneously.
- Accessing the request and reply messages by unauthorized user can be the security concern. In the big companies there are more chances of such thing.
  Proposed Solution: Only the authorized user should be able to send the request message and receive the reply message requests.
- When the client will give the large number of files as input, the output size will be huge for those files. Displaying such big result on GUI can be an issue.
  Proposed output: The output can be stored in the file for convenience.

7. References

- https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/Project1HelpF2018/
- https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/Lectures/StudyGuideOCD.htm
- https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/midterm/MTF18/
- https://ecs.syr.edu/faculty/fawcett/handouts/CSE681/code/Project4HelpF2018/