DEPENDENCY ANALYZER

OPERATIONAL CONCEPT DOCUMENT (OCD)

CIS 681: SOFTWARE MODELING & ANALYSIS

PROJECT #3

SUNILKUAMR VITTHALBHAI LAKKAD (533201045)

Date: 10-19-2014

Instructor- Jim Fawcett Ph.D.

Table of Content

1. EXECUTIVE SUMMARY

The Dependency Analyzer is network based tool which used to find the dependencies between types defined and between packages. The major feature of this application is that user can connect and select any project for what they want the dependency result. Users also can see output on the Graphical user interface, it helps them to understand the dependency Analysis result. In this system one extra server is used which master server which have the master type table contains the type table from all the server.

This tools is used by various users for various uses like browse dependency information of types and packages, to build test plan, to learn code, to judge the code etc. The intended users of the system list below:

- Software Developers
- System Analyst
- Software Tester
- Software Architects
- Team Leader

This document discussed overall architecture of the system and next sections of this document discuss about the packages and activities of Dependency Analyzer Client, Server and Master Server. This document also discuss Graphical User Interface in views section which describe how GUI look like and which option users have to see output.

At the last, the various critical issue of the Dependency Analyzer system are discuss so while Implementation of the system is going on developers will consider this issues, so that implemented system can successfully meets the customers requirement. The intended critical issues listed below:

- Should clients need to know server directory structures?
- Managing file contention
- File Transfer with chunking
- Complete and Up-to-date Type Tables
- Performance
- Display - Mapping data into information
- What will happen if the connection between client and server is terminated?
- How security will be maintained within the system without any client authentication?
- Master server crashed or down

We will implement client, server and master server communication modes using Windows Communication Foundation (WCF) and Graphical User Interface using Windows Presentation Foundation (WPF).

2. INTRODUCTION

Dependency Analyzer system provides the dependencies between all the types defined and packages in the complete file sets which may resides on the remote machine. It also create a XML file which captures the type and package dependencies found. Since, Dependency Analyzer is network based application, it can be divided majorly into three parts: Dependency Analyzer Client, Dependency Analyzer Server and Dependency Analyzer Master Server. Dependency Analyzer client runs at the client end and provides the graphical user interface to the users to perform various functions like, to select server for which they want to find dependencies, to select project/files from particular server, to display which kind of dependencies information they want.

Similarly, Dependency Analyzer server handles all the request from the clients and performs all the task based on user requests. Server will get two kind of requests from the clients:

1. To get Projects list which resides on the server, so users can choose project for that they want dependencies
2. To analyze particular Project which is chosen by users.

Server send responses to the clients based on its requests. The main task of the server when it get start up is to do type analysis of the projects which resides on it local system and create the type table for it. Then after it will connect with the master server and send this type table for further analysis. There is one Agent at all the server which keep track of change occurs on projects/files, if it found any change then it will notify to server and server do type analysis once again and repeat above same step.

Dependency Analyzer Master Server process the request of all the server, which accept the requests from the all server and merge and store type tables of the server Master Type Table. Whenever Master Server get request for copy of Master Type Table from server it will response server requests by replying Master Type Table as a XML file.

In the next sub sections client, server and master server architecture is discussed in detail.

2.1 Overall system Architecture

System architecture for Dependency Analyzer describes the top level architecture by describing higher level activities of a Dependency Analyzer. After analyzing the requirements of the system, tasks can be divided into following high level parts:

- Connection between Clients & Servers and Servers & Master Server
- Selection of project from particular server
- Building type table when server start up
- Merging type tables of the servers at master server

- Find dependencies between all types defined in the complete file set
- Find dependencies between all packages in the complete file set
- Display dependencies information on Graphical User Interface at client

Since, the Dependency Analyzer application is a client-server based model, there will be multiple concurrent users of the whole system. So, here critical part of the architecture is that how system handles the concurrent requests from multiple clients and how the responses of these requests are generated. A generic view of system model between clients and server is shown in the below diagram.

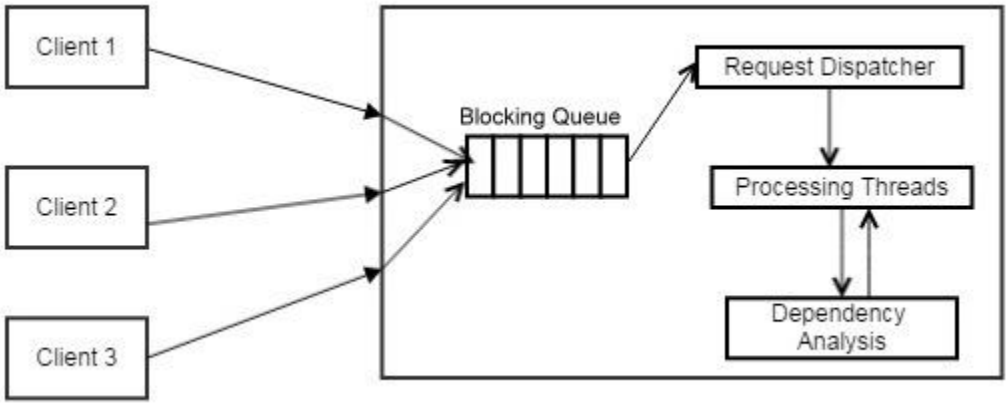

Fig. 2.1: generic view of client and server architecture

As shown in the figure 1.1, there are three important components used to respond to client requests.

- Blocking Queue
- Request Dispatcher
- Processing Threads

Clients will be connected to the server using single communication channel and in order to communicate, both server and client will use the same unique communication channel to send the request and response. Whenever a client will execute any request, it will be queued in the blocking Queue. Blocking Queue will store the requests from all the clients. After a request is stored in the queue, request dispatcher will take the first request from the queue and allocate a separate thread for the processing of request, and then it will take the next request and so on.

In order to process the client requests threads will do dependency analysis.

Similarly, Servers will be connected with Master Server using single communication channel, and Master Server contains the same components which server contains to process the request of the servers.

2.2 Organizing Principles

All Server Contains Thousands of the Folders and files, so it is critical how projects are organized at low level. Because most of time users wants the dependency analysis of particular project rather than a scattered files. So when clients request for set of projects name from server then server must give the set of projects efficient way to clients so users can chose which projects they have analyze. So it is very important that server can find list of projects directory from its root directory efficiently rather than searching each and every Directory and subfolders abruptly. At a Server, projects are organized in such way that server can easily find that this is a project directory and it don't have to search subfolder.

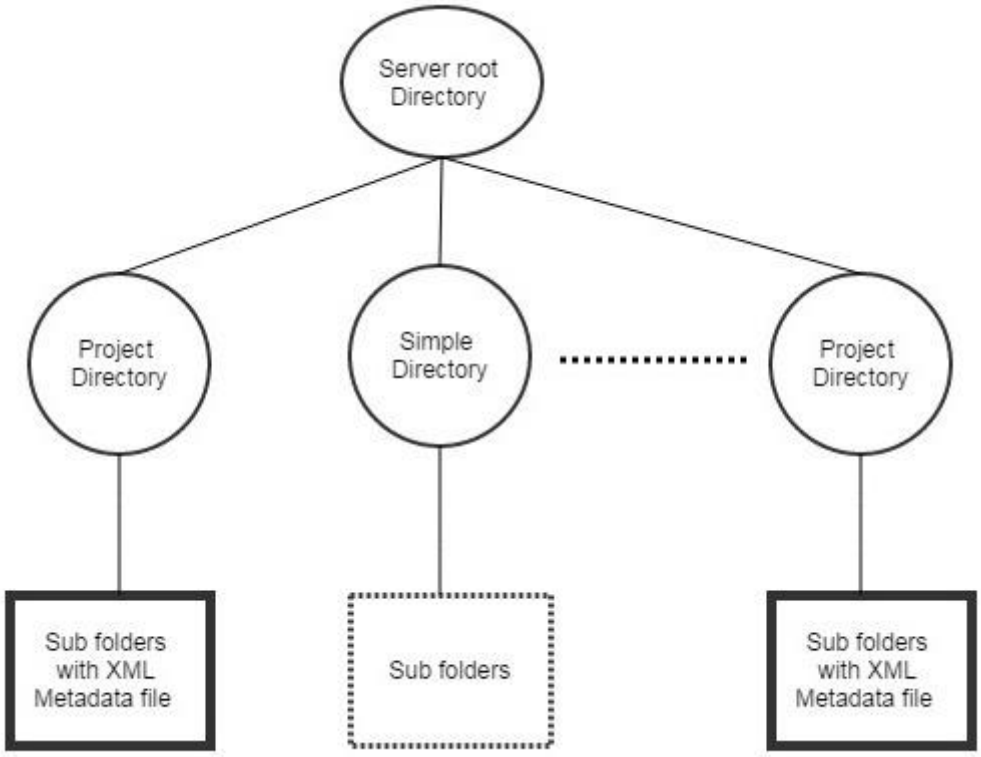

Fig. 2.2.1: Organized Project structure at server

As a shown in figure 2.2.1, Project Directory should contains the XML Metadata file which contains the name of project directory and its path so while scanning process of Project directory server encounter with this XML Metadata file, it will easily identify that this is project directory. So by organizing Project Directory with XML Metadata file we can reduce the scanning time for to collect projects directories.

3. USERS & USES

This section describe various users that are interact with system and the uses of this Dependency Analyzer system.

3.1 Users (Actors)

Dependency Analyzer tool is one that extract dependencies between all types defined and between packages from complete file set which may reside on remote machine. This tool is used by different users as per the requirements.

1. Software Developer

Software developers are those who help organizations achieve goals with software-based solutions. They may update existing software or develop new software to address a specific need or solve a particular problem. Many companies rely on them to contribute to business growth of their company. In the world of software development, writing code for software is not the only thing Developers need to do. In many cases Developers need to modify the code already existing, which may be written by other programmers or themselves long time ago. In order to modify the existing program and make use of it, the first and difficult, step is to understand it. Now, so they don't have to read aimlessly code to understand the flow of project and which package is dependents on other package. By using this tools developers can collect information of dependencies between types defined and packages easily. Sometimes they understand the existing software by reading all this information given by Dependency Analyzer.

2. System Analyst

A System analyst is a person who analyses organizational system by studying the current patterns in the business. The main task of them is to make meaningful system which is efficient, less complex and good structured. Dependency Analyzer clearly proves to be one of the main tools a System Analyst would use because Dependency Analyzer dependencies between types defined and packages. By using this information System Analyst check the code quality and code complexity.

3. Software Tester

Software Tester play an important role in the development of any Software or System. In Development Cycle Once a project reaches a certain stage of development, Software Tester have to ensure it works exactly as it should. At the stage of testing Software Tester have to

build a plan for testing, because initially tester do not which package dependent on other packages. For efficient testing tester should have to test first that packages which are not dependent on other packages. By using Dependency Analyzer Tester can easily extract dependencies between packages and build a good structured plan for testing.

4. Software Architects

Software architects who work on multiple projects to improve source code management and software documentation management. Software Architects use Dependency Analyzer to study coupling and cohesion of each package with help of package dependency structure. This can also be used to make changes to a package based on how dense a package dependency is. If the package is found to have many dependent packages, which all packages would be effected by making a change can be found clearly.

5. Team Leader

Team Leader reviews the performance of their team members by evaluating their projects through Dependency Analyzer. Dependency Analyzer gives team leader detailed information about the type, relationships and dependencies of particular project, by using it he may get to know that which project taken less resources and less complex to understand and implements it for next development.

3.2 Uses of Code Analyzer

This section discuss about various uses of the system.

1. Building a test plans

In software development tester have to ensure work is as it should. For efficient testing tester have to build a test plan, because initially tester do not know dependency structure of packages. Tester should have to test the packages first which are not dependent on other packages.

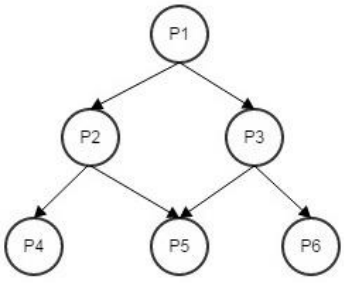

Figure 3.2.1: Example for building test plan

If tester first start testing with Package P1, and when tester reached at testing of P2 and found any bugs then tester changes some useful change in P2, then again tester have to test P1 because it depends on P2. Tester can avoid this problem by building a test plan which start with the packages which are not dependent on other packages.

2. Maintaining a Software

Whenever the feature of application is changed a plenty of time effort required to maintain the application. If Developers changes code of any packages, they could not know which other packages affects and sometime it's very critical for developer to find affected packages. By using Dependency Analyzer developers easily find package dependency and do software maintenance efficiently.

3. Learning a code by using type relationship/dependency

Sometimes software developers have to make a new system using existing system, for this purpose they have to understand which types used in this package are from other packages, which packages dependent on other packages which may reside on remote machine and how system actually works and communicate with other packages. Software developers can use Dependency Analyzer to learn code because it find out all the dependencies between types defined and package of complete file set which may resides on remote machine.

4. To judge code quality by using type relationship/dependency

Any module contains good quality of code if anybody change the code at some extent but which may not highly affects the overall system or affects the minimum numbers of package. Because sometimes it happens when any modules of system resides on the other server which might crashed due to any reason, which do not crashed overall system. In this kind of situation system must give output or process the request at some extent. We can find the code quality of any module by using dependencies between types defined which is easily extract by Dependency Analyzer.

5. Code Changed or Deleted During Software Development

Many time it happens that someone changed and deleted some useful code by mistake then it is very difficult to find which part of code has been changed. But Dependency Analyzer extract dependencies between types defined and packages so by identifying missing dependencies we can find which part of code has been changed.

6. To Browse Relationship/Dependency between types defined

The most important use of Dependency analyzer is to find Relationship/Dependency between types defined, it will help anybody to understand what the program is about.

Sometimes there is not proper documentation done by previous programmer in this situation it helps a lot.

7. To enhance Software product

Dependency Analyzer give summary of dependencies between types defined and packages so it helps a developers to enhance exciting software easily and efficiently. Further, Developer can know which types and/or packages are dependent on which other types and/or package, so they can add new feature at appropriate position in code which do not affect the overall system.

4. Views

Dependency Analyzer Client provides the Graphical User Interface by which users interact with the system. GUI will be implemented by Windows Presentation Foundation (WPF). GUI divided into various following panes so users can easily understand the flow of system.

- Main Window
  1. Server List
  2. Display
  3. Message Log

- Pop up Window
  1. Projects List

4.1 Server List

Initially users don't have knowledge about active server they are available to them. So whenever Graphical User Interface popped up to users, it fetch the list of active server from the XML file. GUI contain Server List pane which display the active servers on window. Server List pane shown in following figure.

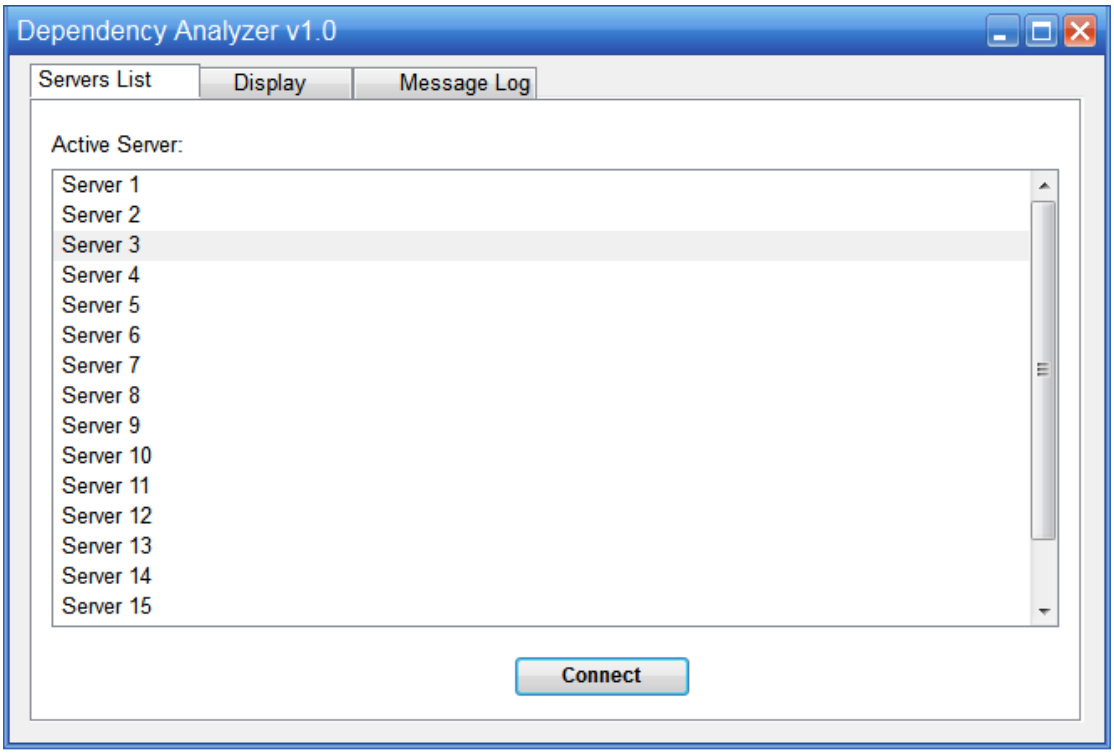

Figure 4.1.1 Server List Pane

As per shown in above figure users can see the active server on Server List pane. Now users select one server for its projects they want dependency analysis. After selecting server user try to connect with that server.

4.2 Projects List

After connecting with server user get a popped up Projects List pane in pop up window, which contains the list of projects. Projects list is send by server so users can easily know that what project they have to select for Dependency Analysis. Project List pane also contains the name of connected server so user can find that they didn't select wrong server. Now After selecting project name for which users want to dependency analysis, users click button on Do Dependency Analysis. GUI pass this action to client, so client send another request for analysis to server and client waits for analysis result from the server.

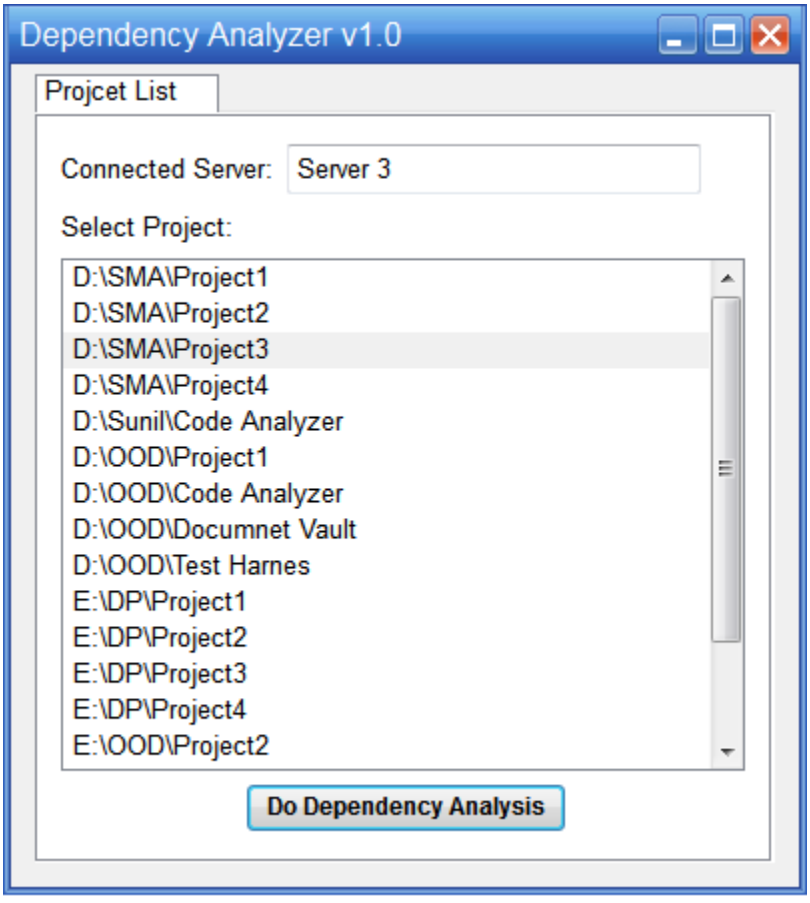

Figure 4.2.1 Projects List Pane

4.3 Message Log

Message Log pane is very useful for users to understand what will happen and when, once client connects to server or users select project for analysis. Message Log contains all the information regarding what and when certain things happens. When user try to connect with the any server, they don't know whether client successfully connected with the server or not, until pop up window with projects List pane is showed up. So when user don't get any responses then they can see message log to ensure that client connected with server or not, because if client didn't successfully connected with server then there is message entry in message log that client unable to connect with server.

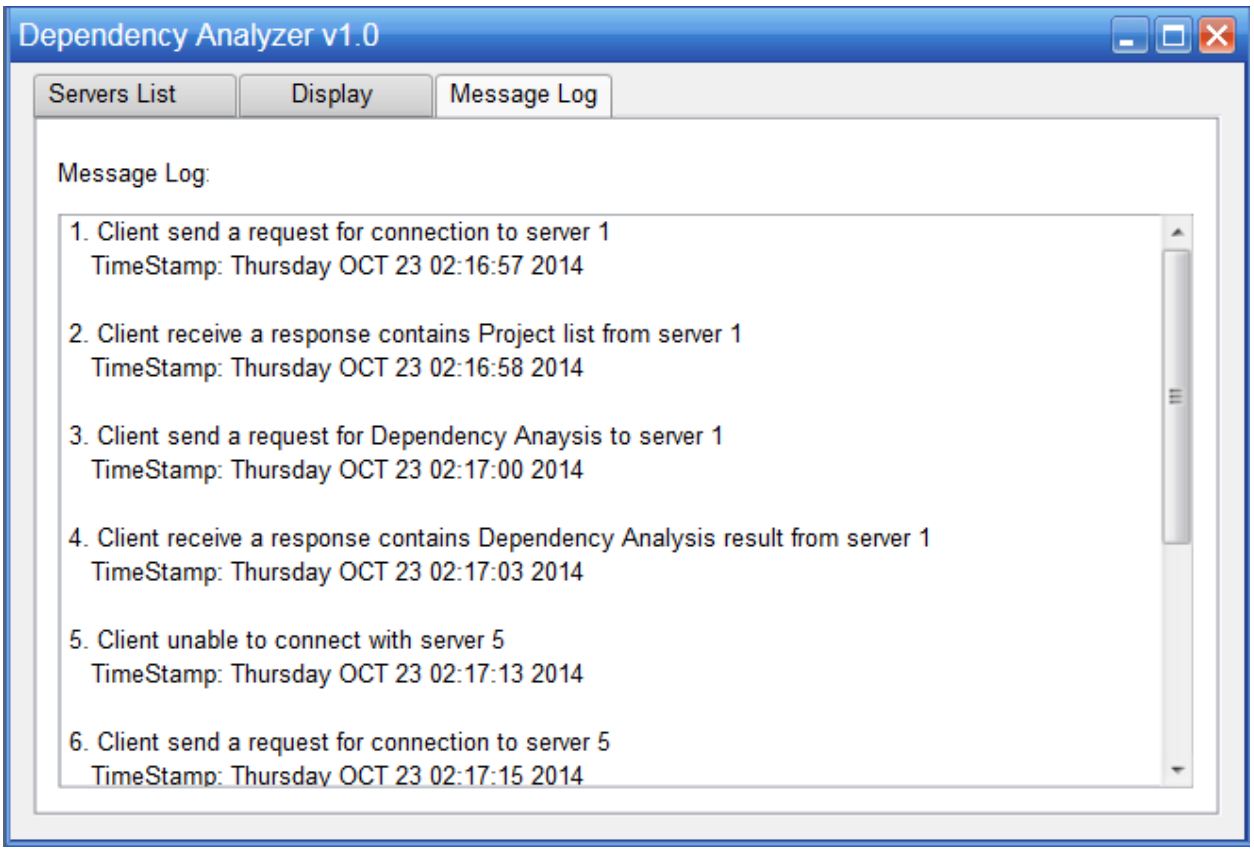

Figure 4.3.1 Message Log Pane

As shown in figure there is message entry for each communication done between client and server. We can see that message entry 5 says that client is unable to connect with server, so user can easily find that why they didn't get project list from the server.

4.4 Display

When users select Project to do dependency analysis, user wait for the result from the server. When user select Display pane in main GUI Window they can find three display option as button.

1. Type Dependency

Whenever user select the Type Dependency option then client display only type dependency result from the dependency analysis result.

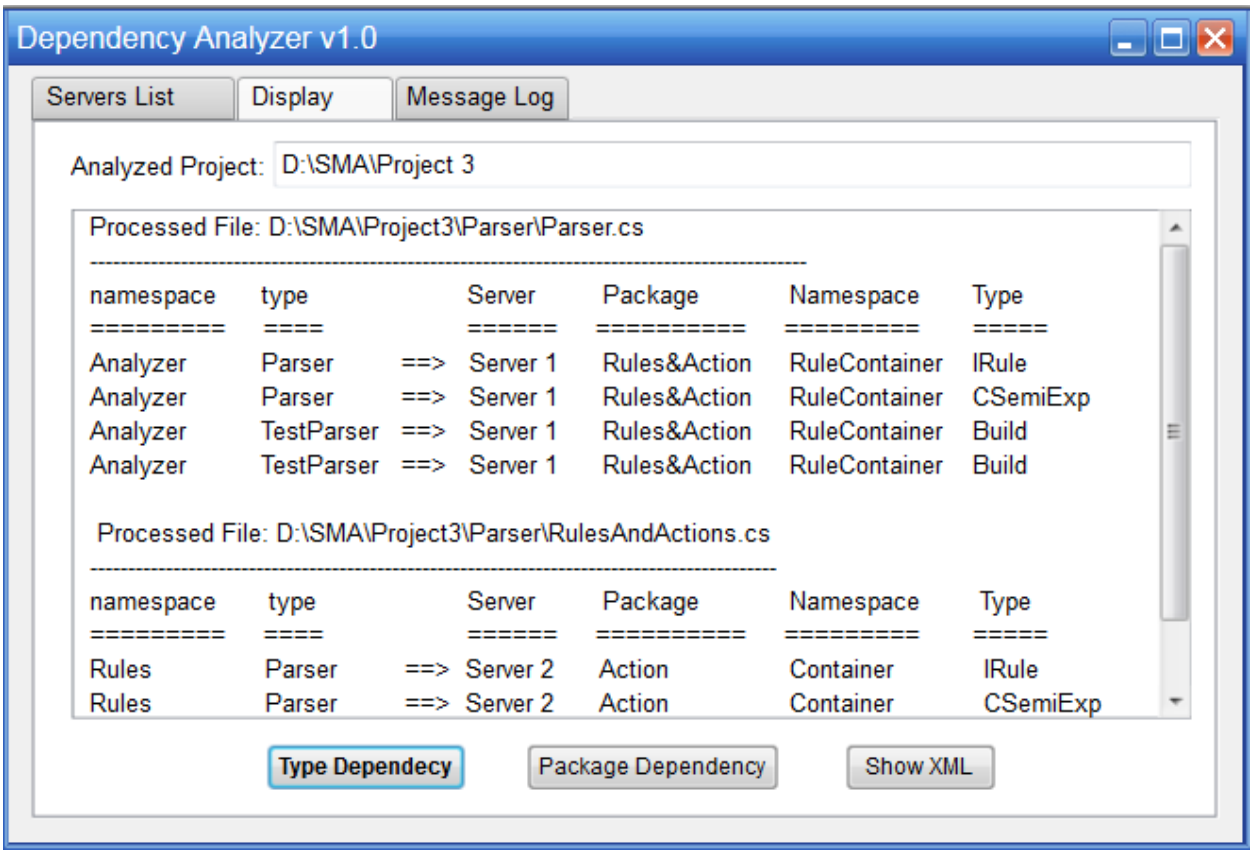

Figure 4.4.1 Display Pane – Type Dependency Option

In Display window users can also get to which project was selected by them to analyze. This option display output on the screen with following property.

- **Namespace:** Which describe the namespace of the particular type which depends on the other type.
- **Type:** The name of type of processed file which depend on the other type.
- **Server:** Which describe the server name/address where package resides on which processed file's type is dependent.
- **Package:** The name of the package on which processed file's type is dependent.

- **Namespace:** which show the namespace of particular type on which processed file's type is dependent.
- **Type:** The name of type on which processed file's type is dependent.

When users select this option client display result for all the file one by one, so user can see result of the all using scroll bar easily.

2. Package Dependency

Whenever user select the Package Dependency option then client display only package dependency result from the dependency analysis result.

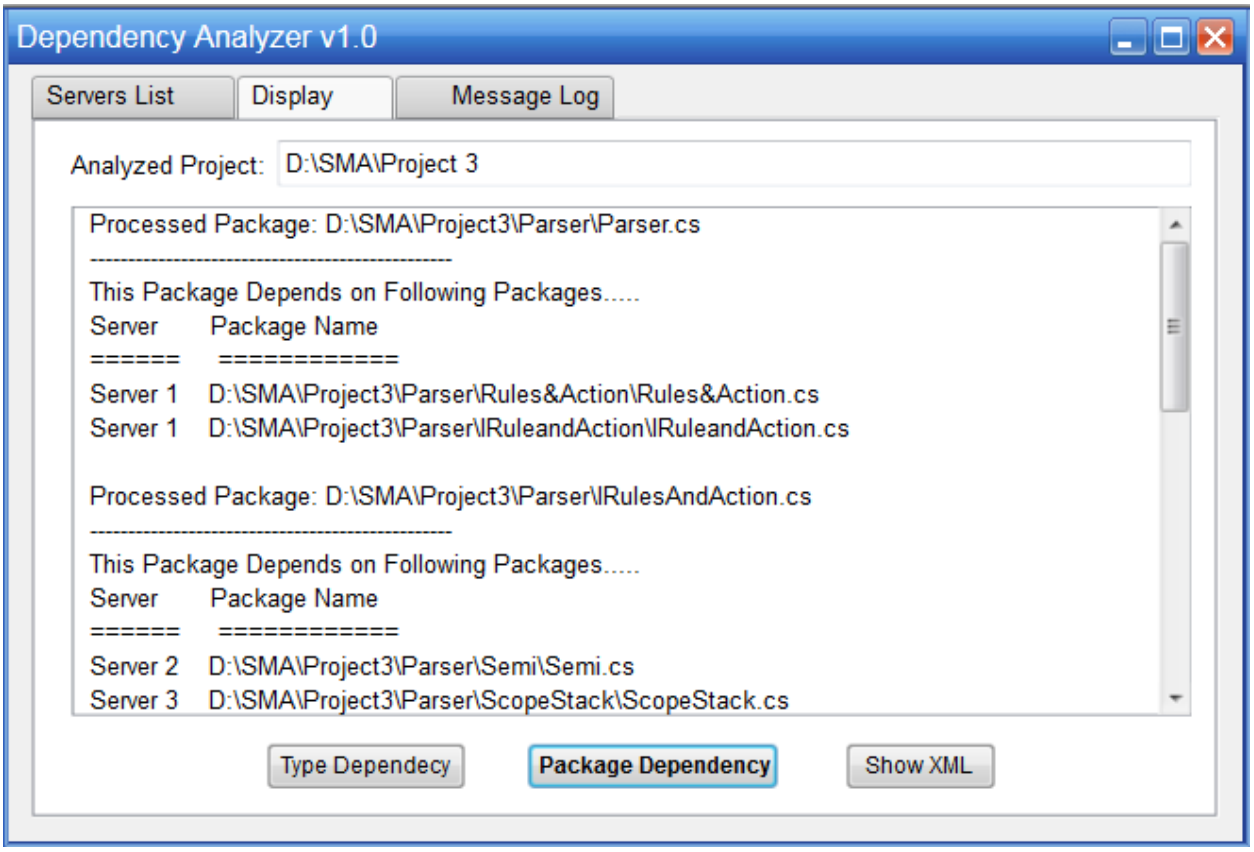

Figure 4.4.1 Display Pane – Package Dependency Option

As shown in above figure when user select this option client display result for the all package. Here client display first package name and then list out all other packages with their server name on which processed package is dependent.

3. Show XML

When user select this option then client read all the content from the Analysis result's XML file and display on the window. When client get result of dependency analysis it store XML file into local storage but users don't have knowledge how it look like. So if users have to know that how XML file look like it select this option on display pane.

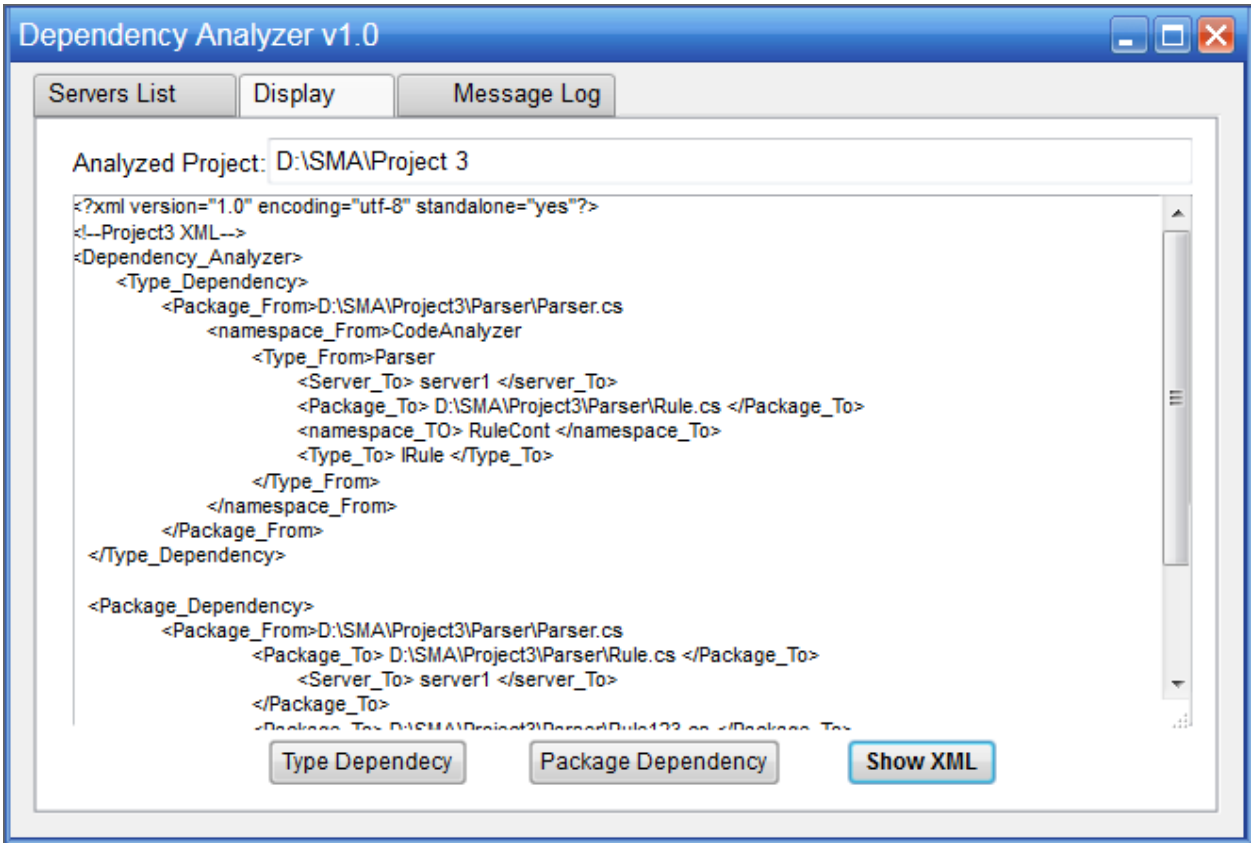

Figure 4.4.1 Display Pane – Show XML Option

5. Partitions

Dependency analyzer is a network based application, which can be partitions into several high level partitions.

5.1 Dependency Analyzer Client Partitions

The tasks of client system can be divide following way:

- To start client module and present Graphical User Interface
- Send a particular requests to server for processing
- Receiver different responses and interpret from server
- To show different analysis on GUI
- To catch up and show any error occurs while client is running
- To save XML file which receiver get from the server as response
- Encoding and Decoding of XML file for further processing of that file

The Client partitions diagram shown below, which is based on task mentioned above.

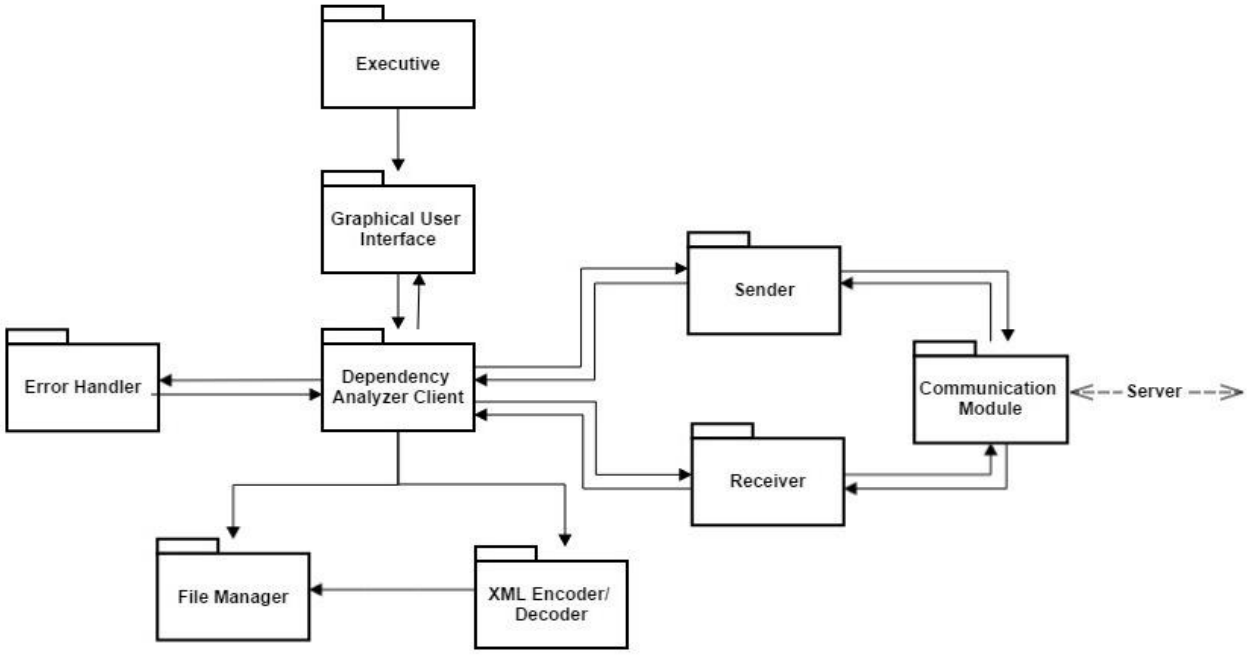

Figure 5.1.1 Package Diagram for Dependency Analyzer Client

The client system partitions with the following module.

1. Executive

Executive is important module of this client system, because it will start the whole system. This module contains a main method which start the Graphical User Interface, by which user can interact with system easily.

2. Graphical User Interface

Graphical User Interface is module by which users will communicate with system. This module takes an input from user and present appropriate output. It will contains a main windows with many UI tabbed panes shown in client view. By Graphical User Interface users will perform following task:

- Users will select and connect to the server for which they want analysis.
- Users can select particular project for analysis.
- Users can chose which dependencies information they want like type dependencies or package dependencies.
- Users can also see how analysis result's XML file look like.
- Users come to know if there is any error while client system is running using Message Log Pane.

Whenever users perform any actions on GUI, it will sends this action to Dependency Analyzer client module which decide further processing.

3. Dependency Analyzer Client

Dependency analyzer client is heart of the client system, it will interact with Graphical User Interface, Error Handler, File Manager, XML Encoder/Decoder, Sender and Receiver.

When users select any server for connection from GUI then it pass connection related message to sender, and when server send response as XML file which contains the list of project it will just called XML Decoder module to decode XML file and store this data into some data structure like List. After that it will pass this data structure to GUI and GUI represent this list of project on interface window. Now User further select project for analysis by GUI then this module send this request to sender, when receiver got response as XML File which contains the dependency information it will first store this XML file via File Manager and call XML Decoder module and further processing will take place as above.

Sometime it error encounter while client is running then it will take an error message from error Handler and send this information to GUI so users come to about this error.

4. Communication Module:

This Module take care of communication between client and server, it perform following task when client system is running.

- It establish unique communication channel between client and server.
- It encode the message whenever it get request from client and send it to server.
- It decode the message whenever it get response from server and send it to Dependency Analyzer client.

Here encoding and decoding a message means, there are some messaging environment which have to follow by both client and server.

5. Error Handler

This module is responsible to handle all the possible exceptions/errors which will be thrown by all other modules. It handles the exception/error and returns proper error message to Dependency Analyzer Client, client further show this error to users by GUI.

6. Sender

Sender Module is used when a client send request to server for the Users to send a file or message on the server. It contains following two components:

- Sending Queue: A queue is maintained by the sender to send the requests to server. Whenever sender get a request from the client it will queued request.

- Sending Thread: Sending thread take a request one by one from the Sending Queue and send this request to server. To send the request messages to server, it will interacts with communication module.

7. Receiver

Receiver Module is used when Server send response to the client which may contains simple processing message or file. It consist following two components:

- Receiving Queue: A queue is maintained by the receiver to receive the requests/response from the server. Whenever receiver get a request/response from the server it will queued request/response.

- Receiving Thread: Receiving thread take a request/response one by one from the Receiving Queue and send this request/response to client. To receive the request/response messages from the server, it will interacts with communication module.

8. File Manager

This module is used to store and retrieve file from the storage as per Dependency Analyzer client's requirement. Whenever client get dependency analysis result as XML file from the server it store this file in local storage using File Manager.

9. XML Encoder/Decoder

This module is used to encode and decode XML file from/or to the storage as per Dependency Analyzer Client's requirement. Whenever client get dependency analysis result as XML file from the server it pass this file to this module, it will perform XML decoding and store information in some data structure like List. After that this List is passed to client and client will passed this to GUI for displaying result.

5.2 Dependency Analyzer Server Partitions

The tasks of server system can be divide following way:

- To start server module and do startup processing like projects scanning, type analysis of the projects
- Receive a request from the client and process it
- Send back a response to the client which may be dependency result or list of projects
- Send a request to Master Server for copy of up to date master type table or to add/update type table of server to master type table if any change occurs in projects/packages at server side
- Receive a response from Master Server which contains copy of master type table
- To find dependencies between types defined and packages using master type table
- Encoding and Decoding of XML file for further processing of that file

The Client partitions diagram shown below, which is based on task mentioned above.

The server system partitions with the following module.

1. Executive

Executive is important module of this server system, because it will start the whole system. This module contains a main method which call the Dependency Analyzer Server Module for further useful processing.

2. Dependency Analyzer Server

This module is very important for server system because it will interact with other modules of the server system. With startup processing it will call Projects Scanner module which is

responsible for collecting projects list that are available in local directory, after that it will pass this list to XML Encoder/Decoder module to convert into XML file and store into local storage using XML File Manager. So whenever Server got request for project list from the client it will directly fetch XML file from the local storage using XML File Manager which contains the projects list information and passed to the client.

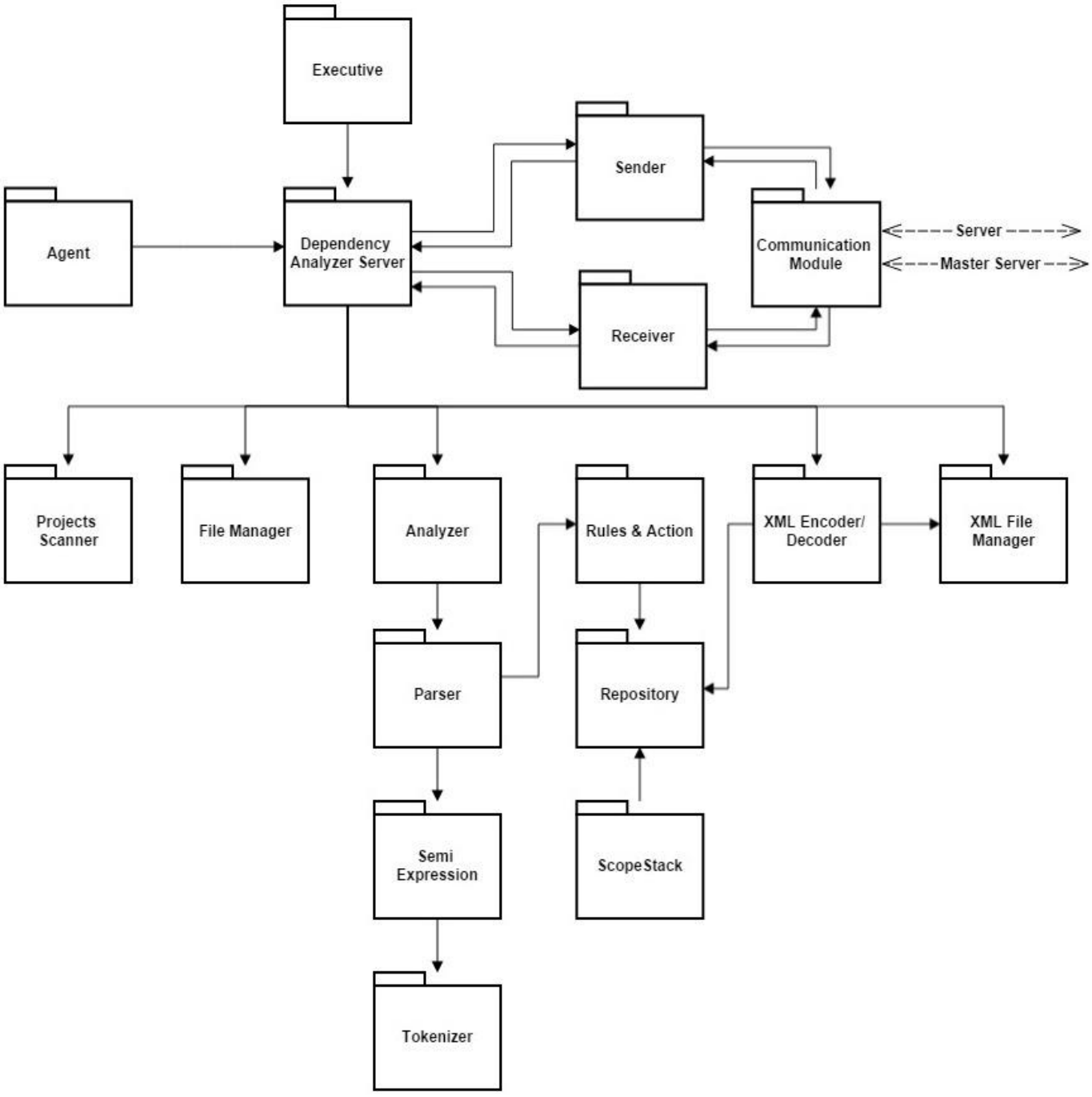

Figure 5.2.1 Package Diagram for Dependency Analyzer Server

Whenever it will called by Executive, it will initially do type analysis of the projects which resides on the local storage and store into local type table at repository. After that it will call XML Encoder/Decoder module to convert that local type table into XML file and store into local storage using XML File Manager. Once type analysis done by server it will connect with master server and send this local type table to master server for merging into master type table.

Now, when server get request for dependency analysis on particular project, it will contact to master server for copy of master type table and it will get copy of master type table as XML file. Once it get XML File it will call XML Encoder/Decoder module to convert this XML file in some useful data structure like list, and store at Repository for dependency analysis. After that Dependency Analyzer Server call file manager to collect list of file of that project, then it will call Analyzer for dependency analysis which used master type table reside at Repository for this dependency analysis. The result of this analysis will store at Repository in another data structure, once dependency analysis get done server will call XML Encoder/Decoder to convert result into XML file, at the end it will transfer XML file as a result of the analysis to the client.

3. Projects Scanner

This module is called by Dependency Analyzer Server whenever it get started. The main motive of this module is to scan all the project which reside on local directory of the server. Initially client don't have knowledge how many projects are available at server and what the path of that projects. This project scanning also useful in type analysis because server don't have to search projects blindly, server can use the list of project which created by this module.

4. File Manager

File Manager receive project directory path from the server, it will rooted every file and collect all the file from subdirectory. At last it will pass file stream to upper package.

5. Analyzer

Analyzer package is root package for all the activity done after file collections. It will collect file stream from Executive and pass to Parser package to find out dependencies between types defined and packages. It control all the activity relating to analysis.

6. Parser

Parser package receives file stream from Analyzer. Parser Package is a container of specific rules to do parsing efficiently. Each rule is a write to detect certain grammatical construction. For example, a non-keyword name before a pair of brackets and before an open braces &

after keyword 'class' means a class name. So, when the Parser Package detects the name of class, it will use the Actions & Rules Package to execute all the operations for a new class, including recording the class name. However, Parser Package uses the service of Semi Expression and Tokenizer Packages to accomplish the complete.

7. Semi Expression

Semi Expression Package provides a class, CSemiExp, which extracts certain token sequences according to some specific rules. After that Semi Expression receive the decomposed stream from Tokenizer, Semi Expression shall return a stream with meaningful characters to Parser.

8. Tokenizer

Tokenizer Package provides a CToker class that supports reading words from a string or file stream, called tokens. Tokens are constructed using specific set of rules that are useful to detect class, namespace etc. Basically, Tokenizer will decompose the file stream by erasing all the spaces, white space and comments. Then, it passes the decomposed file stream as tokens back to Semi Expression.

9. Actions & Rules

Actions & Rules Package contains the set of rule for finding several useful code analysis information. This package also responsible for recording the data its collects. Action & Rules Information fetch and store useful information in Repository Package

10. Repository

Repository Package is used for to store information during analysis. Actions & Rules Package uses Repository Package as the container to store the result of the decomposition. For example, after the Actions & Rules Package gets the various types defined, the program needs to store the information in somewhere for later uses. It is Repository Package which used in this kind of situation.

The other use of Repository to store master type table, because whenever Analyzer doing dependency analysis it required master type table for look up particular type.

11. Scope stack

Scope Stack Package is used as temporary storage for source code analysis activity. Scope stack provides the facilities to track the position in source code by pushing and popping namespace, class, struct and enum via the class ScopeStack.

12. XML Encoder/Decoder

This module is used to encode and decode XML file as per Dependency Analyzer server's requirement. This Modules used by server in following cases:

- Initially when server will startup, Project scanner scan all the project list which is later stored in XML File using this module.
- Initially when server will startup, initial type analysis is taken place and the local type table created at repository. At the end this type table concerted into XML File using this module. After this XML file is passed to master server for merging the master type table
- When server got a copy of master type table as XML file from master server it has to decode this XML file into some useful data structure like List.
- When actual dependency analysis is going on analyzer store the result in to repository at the end. Server convert this result into XML file and send back this XML file to client.

13. XML File Manager

This module is used to store and retrieve file from the storage as per Dependency Analyzer server's requirement. This Modules used by server in following cases:

- Initially when server will startup, Project scanner scan all the project list which is later stored in XML File. This XML file is stored in local storage using File Manager so whenever it get request from client for project list it just fetch XML file from storage and give it to client.
- Initially when server will startup, initial type analysis is taken place and the local type table created at repository. At the end this type table converted into XML File and stored at local storage.
- When actual dependency analysis is going on analyzer store the result in to repository at the end. Server convert this result into XML file and stored at local storage so it will send back this XML file to client.

14. Agent

This module is always rendering local directory to find any change occurs at the projects/packages in local directory of the server. If it found any change then it notify to Dependency Analyzer Server regarding this change, now server perform following task:

1. First it will call project scanner for new project list
2. It will call Analyzer to do new type analysis and store that result into XML file.
3. It will contact with master server and update the master type table reside at master server.

Communication Module, Sender and Receiver Modules same as discussed in Dependency Analyzer Client partitions, but they have to take care of both Client and Master Server.

5.3 Dependency Analyzer Master Server Partitions

The tasks of master server system can be divide following way:

- To start master server system and listing from server request.
- Receive a request from the server and process it
- Send back a response to the server which may be copy of master type table
- To merge type tables from all the server into master type table
- Encoding and Decoding of XML file for further processing of that file

The Master Server partitions diagram shown below, which is based on task mentioned above.

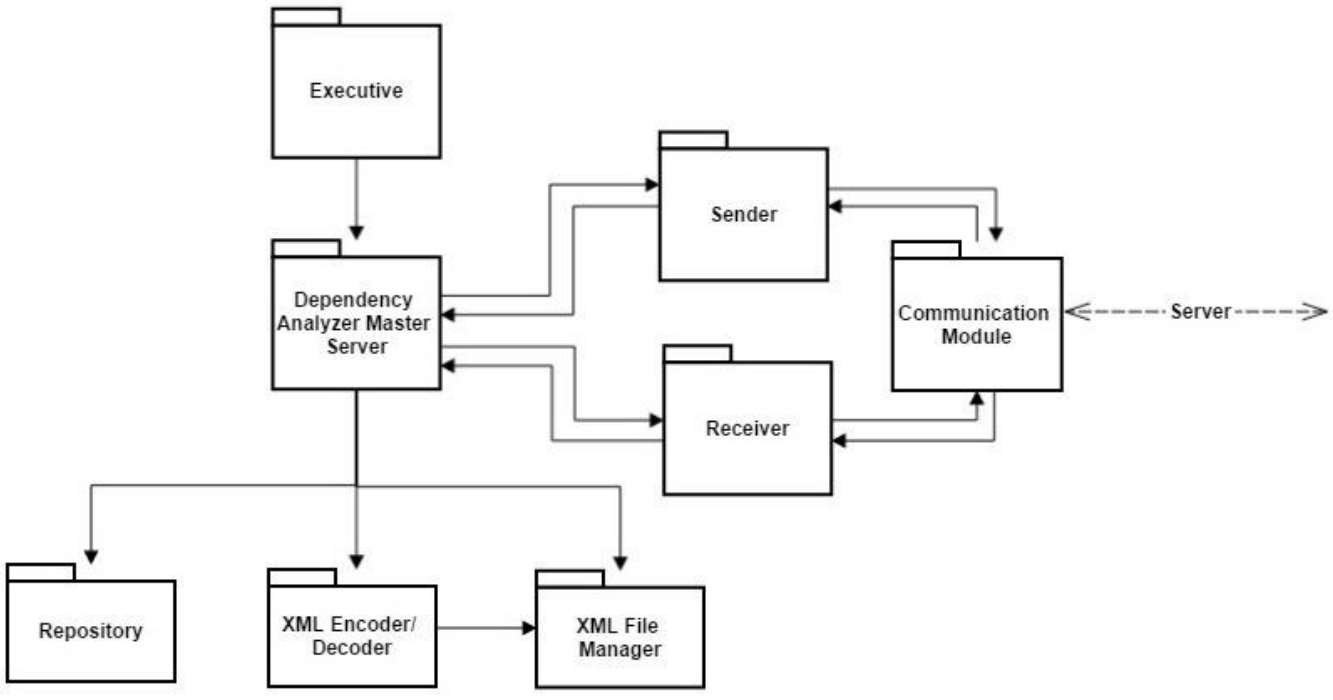

Figure 5.3.1 Package Diagram for Dependency Analyzer Master Server

The master server system partitions with the following module.

1. Executive

Executive is important module of this master server system, because it will start the whole system. This module contains a main method which call the Dependency Analyzer Master Server Module for further useful processing.

2. Dependency Analyzer Master Server

Initially it is listening request from the server to merge server's type table into master table. Whenever it get request for merging it first call repository which contains useful data structure into which it store the type table of server. When this process is done it pass this type table to XML Encoder/Decoder to Encode this table into XML file, this converted file is stored into local storage using XML File Manager.

Now it get request from other server for merging type table it will call XML File Manager and fetch XML file which contains the master type table then it Decode this file into some data structure like list and send it to repository. Now repository have master type table, master server add type table of server into this master type table. Now Repository have full master type table which contains types of all server. Now server called XML Encoder/Decoder to Encode this table into XML file, this converted file is stored into local storage using XML File Manager which overwrite old XML file.

Whenever master server get request for replace data into master type table for particular server then it follow same above procedure but at the time of adding new data in master type table it replace old type data with new type data.

Master server get request for copy of master type table from any server, then it just fetch XML File from the local storage and it will pass this file to server as a response.

3. Repository

Repository used data structure like list to merge type table of all other server into master type table. Repository called by server whenever any changed is required in to master type table like to add new type data or replace type data of particular server.

4. XML File Manager

XML File Manager store and load XML file from the local storage. Here File Manager works with only one file. Because whenever it have to store master type table's XML file it just overwrite it with old XML file so every time whenever it fetch XML file which contains latest information.

6. ACTIVITY DIAGRAM

This section of document discussed about various activity will take place at system, for ease the activity divided into three activity diagrams. Each of one describe various activity will take place at three different sub system like Client, Server and Master Server.

6.1 Dependency Analyzer Client's Activities

As discussed in Partitions, top level tasks of the Dependency Analyzer Client can be divide in following way:

- Start client system
- Get a list of Active server
- Select particular server for that users wanted to do Dependency Analysis
- Get a Projects list from the selected server and chose one project for Dependency analysis
- Receive analysis result as a XML file from the server
- Save XML File at local storage
- Select different option to show various content on GUI

The Activity Diagram for Dependency Analyzer Client shown below which consist activities will take place at client side.

Users start the client system with the help of Executive module, initially client don't have knowledge of how many servers are currently running. Client get this information from the XML file which contains the list of currently running server. Whenever client system start it pop up Graphical User Interface via it users can select particular server to analyze projects resides on server's local directory.

When users select particular server for analysis, client send a request for connection to that server with message that it wants Projects list which resides on server local directory. If client didn't get response in a specified time it will try to connect one more time with the same request message. At the end when client get connected to the server, server send Projects list information in one XML file. Once receiver receive this XML it will pass this file to client and client Decode this XML file using XML Encoder/Decoder module and display on GUI.

Now users can select a project on which users want dependency analysis, when user select a project client send request message for dependency analysis to server. Server will do its job and send dependency analysis result as XML file. Client receive this XML file and store at local storage for further use. Further client also Decode this XML file in some data structure like List to display on GUI.

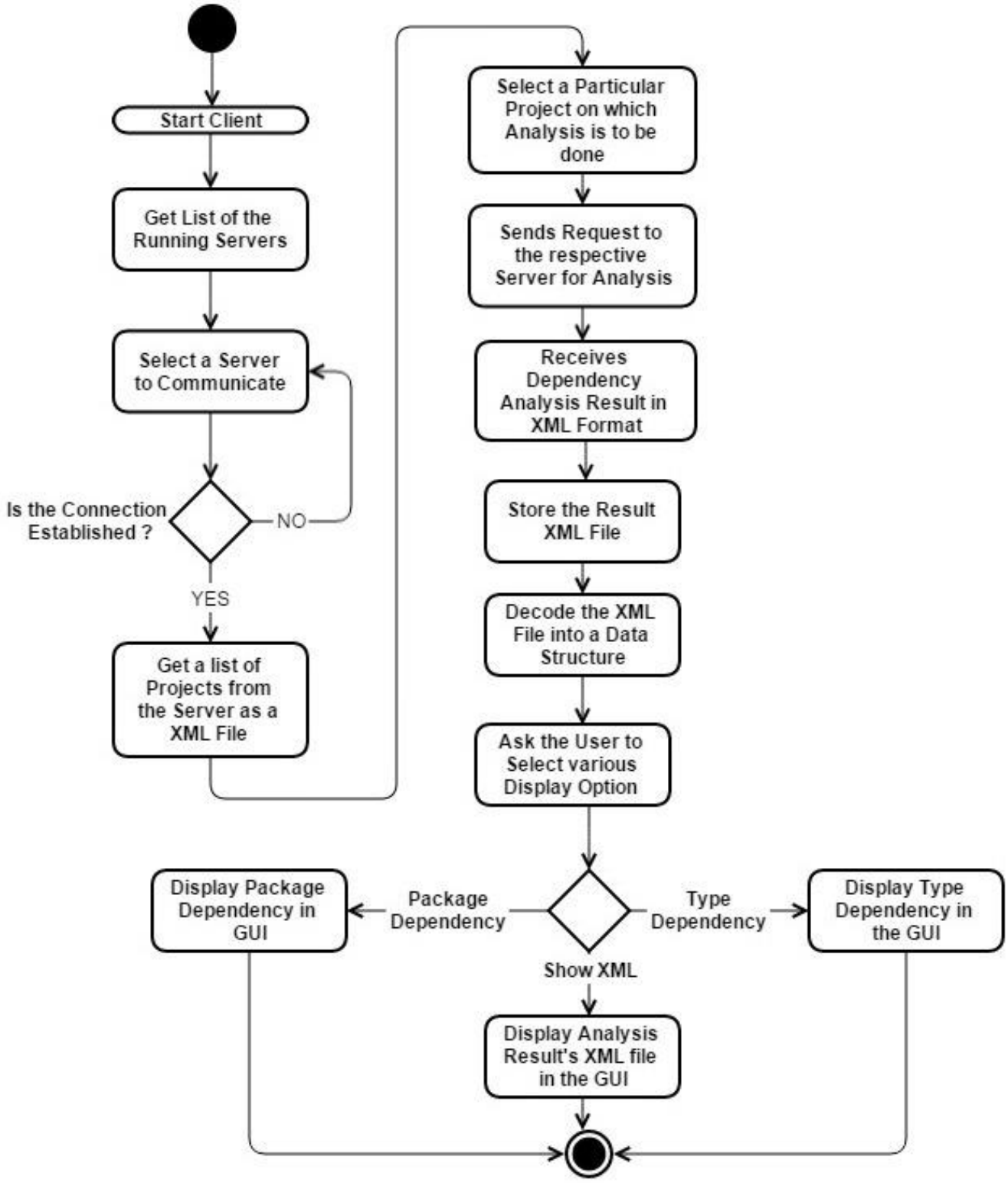

Figure 6.1.1 Activity Diagram for Dependency Analyzer Client

There is option regarding which dependency results user wants see, if user select type defined dependency then client fetch results from the data structure and display on the GUI and if user select package dependency then client do same task and display result on the GUI. There is also one other option Show XML, when user select this option client read the data from analysis result's XML file and show content XML on the GUI.

6.2 Dependency Analyzer Server's Activities

As discussed in Partitions, top level tasks for the Dependency Analyzer Server can be divide in following way:

- Do Type Analysis with startup
- Do Project Scanning with startup
- Send type analysis results to Master Server
- Listing to clients requests, and process different requests with different processing
- To check always if there is any change at Projects/Packages
- Receive a copy of up to date master type table for dependency analysis
- Do dependency analysis and store result in XML file and send back to this file to client

The Activity Diagram for Dependency Analyzer Server shown below which consist activities will take place at server side.

When server get started it has to do initial task. With startup server start projects scanning at the local directory and save the project list on the one XML file. Secondly it will start the type analysis for all the projects which resides at server's local directory and store the result type table into data structure, after at the end of type analysis it just convert the result into XML file.

Now, Server try to contact master server to send its type table to merge it with master type table. After successful connection with master server it will send the type table's XML file to master server.

After completing startup processing, it will listen the client requests. Once server get request from the client it will check weather request is for to get projects list from the server or it is for dependency analysis. If the client's request for projects list then server just fetch XML file from the local storage and send it to client as response.

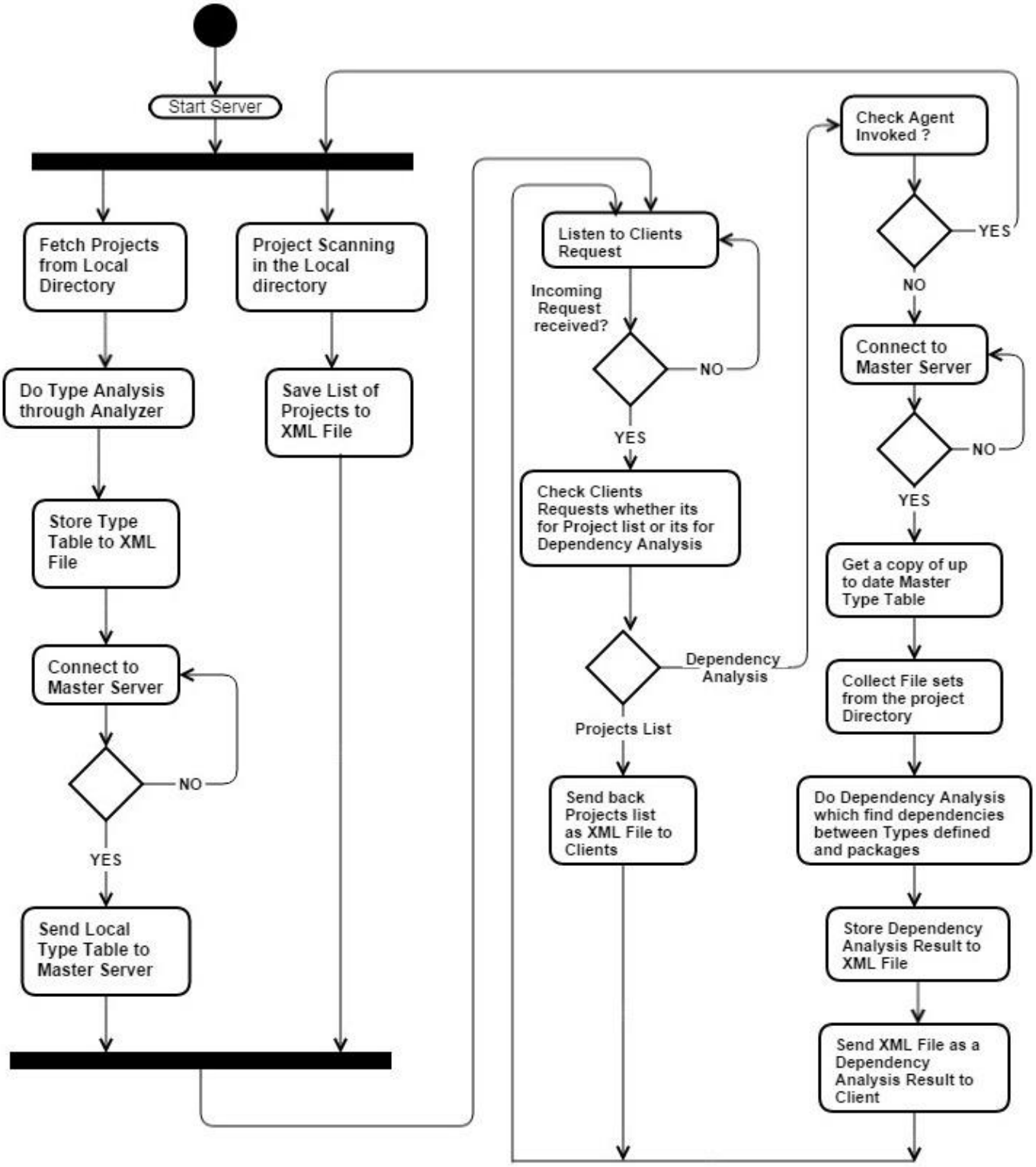

Figure 6.2.1 Activity Diagram for Dependency Analyzer Server

If the client's request for dependency analysis then server checked that in between this mean time Agent Invoked or not, if agent have been invoked then server will start startup processing again as it discussed. If agent haven't been invoked then server will contact master server for up to date master type table, once it get the master type table from the master server it will store this master type table at repository in new data structure to use for look up when dependency analysis will start by analyzer.

Server process the client request message once again and parse the project name and its path on which users want the analysis. Once server get the project path it will call File manager, which collects the files set from the project directory and give it to server.

Now actual dependency analysis will begin, server call the Analyzer and pass it to files sets, Analyzer perform type dependency and package dependency. While performing this dependencies analyzer use up to date master type table resides at Repository for error prone analysis result. When Analyzer done with dependencies analysis it just store the result on the XML File using XML Encoder/Decoder module and store at local storage using XML File Manager. At the end, server fetch this file from the local storage and send it to client as response.

6.3 Dependency Analyzer Master Server's Activities

As discussed in Partitions, top level tasks for the Dependency Analyzer Master Server can be divide in following way:

- Listening the request from servers
- Process the different requests of the servers
- Add or Update the type tables of the server into master type table
- Send a copy of master type table to server

The Activity Diagram for Dependency Analyzer Master Server shown below which consist activities will take place at master server side.

When master server get startup, it will listen the servers requests. If it got the request from the server it will check whether the request for add/update type tables into master type table or it for to get copy of master type table. If request is for to get copy of master type table then master server fetch the XML file which contains the up to date master type table and send it to server.

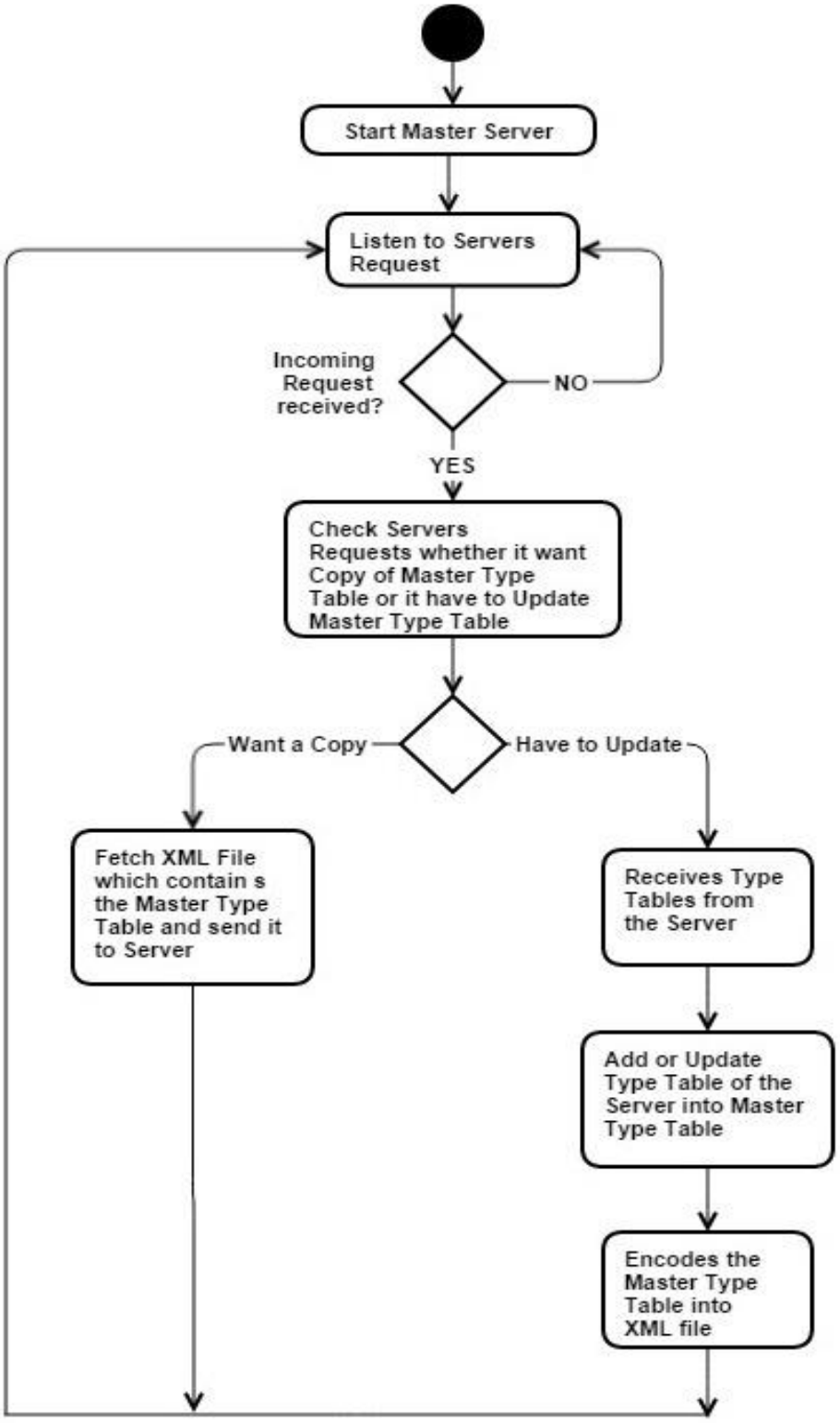

Figure 6.3.1 Activity Diagram for Dependency Analyzer Master Server

If the request for add/update the type table into master table then master server do following activity:

- If the master server get request for first time from the server to add type table to master type table then it will call XML file manager and fetch the XML file which contains the master type table. Then master server decode this XML file and send it to Repository, at Repository the type table of the server will add into master type table. Then server will do reverse processing it will Encode master type table to XML file and store at local storage.

- If the master server get request for update master type table because at the server side Agent found changes on the Projects/Packages, then master server do all the above processing but at Repository it will replace old type table of the server with new type table into master type table.

7. Critical Issues

While designing the architecture of a system, there can be several critical issues which need to be addressed. In this section, some of the critical issues related to the implementation of Dependency Analyzer are as follows.

7.1 should clients need to know server directory structures?

In this Dependency Analyzer tool users are interested to analyze particular projects through clients which may resides on remote machine like server. But the issue is that how users know what is path for particular project on server is, without path information users can not specify particular project directory for analysis.

Solution:

First consider scenario at server side when it start up.

When server is start up one thread will start the scanning for project directory, thread will try to find solution file with ***.sln*** extension and if it encounter this file in any directory it will consider this directory as project directory and it will create one XML file which collects the Project directory information, after that it will store full path relative to that server in the one XML file. At this moment thread also create one XML Metadata file inside this project directory which contains full path of this directory & information that state this directory is project directory. So at the end of scanning process XML file contains the full path of all project directory which resides on the server.

Now, when users want to analyze particular project on particular server, it will just connect to the server and send request for list of projects which resides on the server. After that client will display this list on GUI and users can select project for which they want dependency information.

7.2 Managing file contention

Competition for resources (File) leads to contention. The term is used especially in networks to describe the situation where two or more nodes attempt to transmit a message across the same wire at the same time. That is, two or more nodes may try to send messages across the network simultaneously. The contention protocol defines what happens when this occurs.

In this System at server side we are using multiple threads which take request from Blocking Queue and processes that request concurrently. At a moment when different multiple clients want a dependency analysis for particular project, so when multiple threads processing their request each thread want to access files at same time which may lead file contention.

Solution: we can provide Locking operation for this problem. Whenever any thread want access the file for reading system will check the file is locked by another thread or not, if not then thread access this file and locked down this file. So now any other thread wants to access this file but it is locked by another thread so this thread will go to sleep for certain amount of time.

7.3 File Transfer with chunking

In Windows Communication Foundation if we to send large messages then we will face problem limited amount of memory which used to buffer those messages.

Solution: One possible solution is to partitions whole large message into smaller messages called chunks, and send these chunks to the receiver one chunk at time, and recollect this chunks and construct original message on the receiver side. The system itself could do this kind of chunking or it could be custom channel for it. For this chunking process we assign sequence number to all chunks and with the last chunks we will number of chunks send so far. So at the receiver side, receiver collect all the chunks and find number of chunks passed by sender with the help of last chunks to ensure all the chunks have been collect or not. Now based on Sequence numbers receiver reconstruct original message.

7.4 Complete and Up-to-date Type Tables

To analyze particular project which is chosen by user, server required type tables which contains type information from all the server which were stared. But there is an issue if any projects or packages get changed by any developer, now for analysis all server wants fresh type tables which contains fresh and up to date information from all the servers.

Solution: There are possibly three solution,

1. We can use Master Server for merging type information from all the server, this approach is used in this document.

2. Initially when servers get startup it will do type analysis and store type information in local type table. Now when client connect with different server, client will pass message to all the server which contains the address of the each server. So this point all the server pass its type table to other servers and each server will get type tables from other servers. All server collects type tables and merge into new up to date master type table which contains type information from all the server.

When any changes occurs at projects or package, server will do type analysis for this change and create local type table. After that it will send message to all the server which contains new type table, and all the server just replace this type information into master type table for that particular server.

In this kind of model message overhead will increase because any server get changed in its projects then it has to notify other server to update master type table.

3. Initially client has XML File which contains addresses of the all server. When client program get started client send request to each server to collect type table from all the server and then client merge all the type table into master type table. After then when server connect with particular for project analysis it will also send master type table as part of message when it request to server for dependency analysis. So whenever clients send an analysis request to any other server it will also pass master type table as request message.

Now when any changes occurs at projects or package at a server, server directly send a new types table to client and client just replace this type information into master type table for that particular server.

7.5 Performance

A project may have thousands of files for analysis. Those thousands of files may have multiple types, relationships and dependencies among them. Initially servers supposed to scan all the files and read the all type from the thousands of the file it will take significant time. It will impact the start-up time of the server and overall performance will be degraded.

Secondly, when actual dependencies finding activity is going by Analyzer, it have to look up in master type table for particular type but master type table contains the thousands of type. So if Analyzer look each and every entry in master table then it will take lots of time to finding particular one type, we can imagine that to find type dependency analyzer have to always look up into master type table which take more time and performance will be degraded.

Solution: When get started the first time, it will scan all the projects resides on the local directory with multiple threads to prepare project list. Using of multiple thread will make a server's startup processing fast. To build type table used of multiple thread is great way.

Now for Dependency Analyzer, we have to break up master type table in some data structure like hash table. Hash table contains entry per other server's data, means if master type table contains the data of 15 server then there is 15 entry in hash table which all contains type table of the 15 different type table of the server. So now Analyzer have to look up in master table for particular type then it will create 15 threads, these all threads will look up in hash table with different index. This reduce significant amount of time to look up master type table.

7.6 Display - Mapping data into information

Client get a dependency analysis result as XML file from the server, which contains tons of data. But if we display this data as it comes from the server and users cannot get idea about result. But as we discussed that client decode this XML file some data structure, using this data structure client can print output easily shown in views section.

But output contains thousands of package so if user have to watch out any particular package then they have to scroll down output and search that package randomly.

Solution: we can provide one extra pane called Search Package shown in below figure.

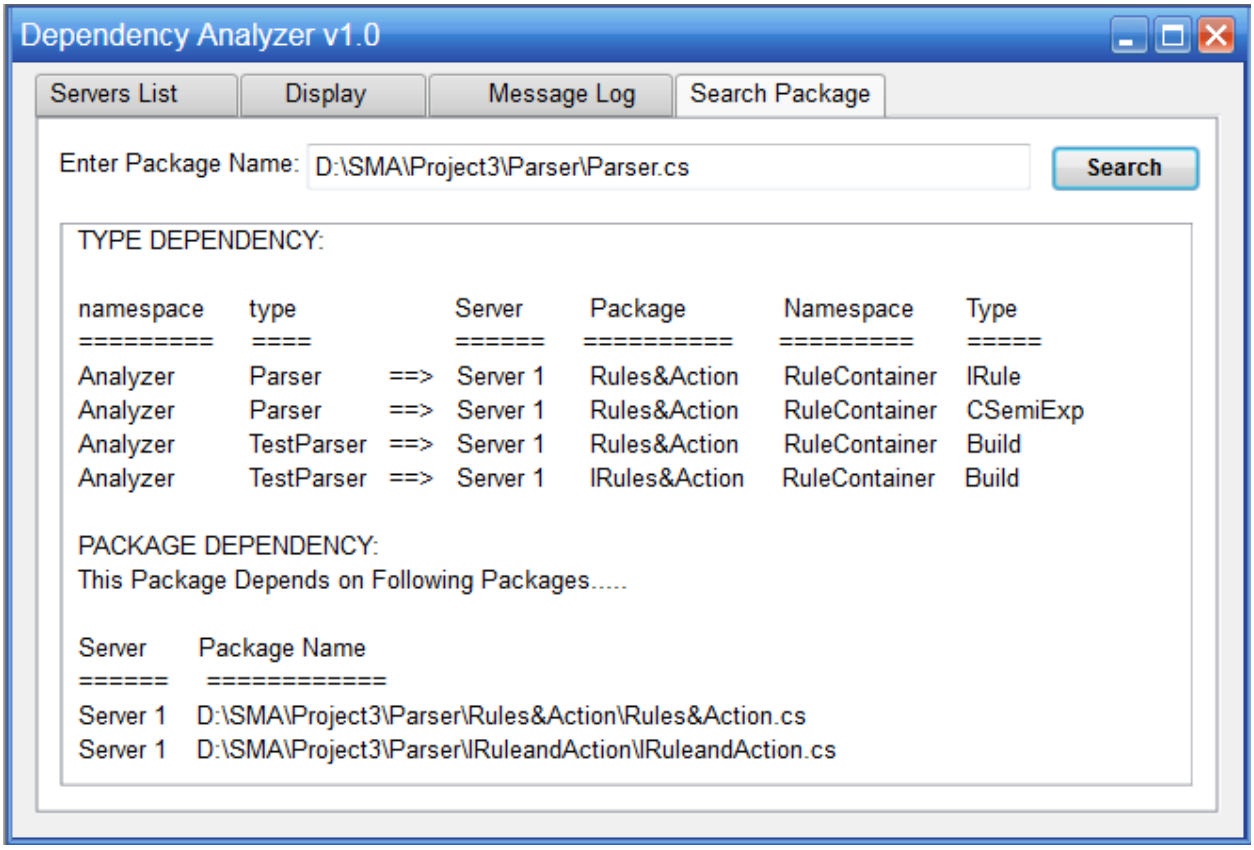

Figure: 7.6.1 Search Package's result

When users want dependency analysis result for particular package then it just enter the package name with it path in text field and press Search button. Client get this action and search the both dependency results in analysis result and display that result of that particular package on the interface window.

7.7 What will happen if the connection between client and server is terminated?

Since Dependency Analyzer is a network based application, it is possible that communication between client & server or server & master server is terminated in between a transaction. Here transaction means file transferring or normal message passing.

Solution: There are two scenario in this problem:

1. The connection between client and server get terminated while client is idle, there is not any request is in pending state for the client, client again make attempt to connect with the server until client and server connected successfully. We can implement thread which try to connect server on unsuccessful connection after sometime.

2. The connection between client and server get terminated while a transaction is in process. Let's take example, client send a request for Project analysis to the server, connection get terminated while server is sending the result to client. So client do not get response from the server. To solve this problem we will implement one thread which ping to client when server is not able to send response back to client. This thread will try to connect to a client for a certain time limit and after that this connection will considered as dead. In between this time if thread successfully connected with clients then server will send result of analysis to the client.

7.8 How security will be maintained within the system without any client authentication?

Dependency analyzer is network based application, so sometime it happens unauthorized client may send request to server for analysis. So how server can identify that client is legitimate and it have to accept the request from the client.

Solution: There could be two possible solutions so server can identify legitimacy of client.

1. **Access control list:** Dependency analyzer can be secure with the help of windows access control list. It define the way to add trustee users in to a network so any outsider will not access this system.

2. **Server side admin interface:** We can provide separate security admin interface module at the server side which have list it's contain list of allowed client's IP addresses. So whenever server get request from the client it first call the security module to check the legitimacy of the client and if client is legitimate then server will process client's request.

7.9 Master server crashed or down

Initially all server have copy of master type table from the master server, when any projects or packages get changed server have to send updated type table to master server so all server get up to date master type table. But sometimes it happens when Master server crashed or down then above type table change will reflected to master type table because server cannot

connect to the master server. So all the servers don't have up to date copy of master type table at this point, users will not get reliable dependency output.

Solution: We can provide Backup server which always contains the up to date copy of master type table. So when any server could not connect to the master server because it is crashed or down then server will directly connect to the backup server. After that server will send request to Backup server to update copy of master type table at backup server or to get up to date copy of master type table.

When Master server will recover it will get up to date copy of master type table from the backup server, and now onwards any server will connect with master server.

8. CONCLUSION

The Dependency Analyzer is a tool which is used process medium and large sized of projects. It is network based application by which user select the server and project for that they want dependency analysis. The system contains three different module like client, server and master server for ease and error prone analysis. Dependency Analyzer also provide Graphical User Interface by which users can interact with system easily and they also select various option for result. Here we discussed how various Users (Actors) uses this tools for various purposes. Users can use this Code Analyzer to understand, enhance, test and reused exiting System. The Dependency Analyzer divided into various package so we can correlate the functionality of each modules.

We have discuss various important critical issues which may rises while implementation of Dependency Analyzer. The critical issues reasonably solved during implementation otherwise it causes serious design flaw which might affect the execution of Dependency Analyzer.

7. REFERENCE

[A] Parser, Tokenizer, Semi Expression Prototype, Handout provided by Jim Fawcett
[B] OCD Study Guide, provided by Jim Fawcett
[C] Project Helper Fall 2014, provide by Jim Fawcett
[D] Best Project 3, OCD of Prateek Sharma, provide by Jim Fawcett
http://ecs.syr.edu/faculty/fawcett/handouts/CSE681/BestProject3s/
[E] Gliffy.com, Online Diagram Software, http://www.gliffy.com/