OPERATIONAL CONCEPT DOCUMENT

On

CODE ANALYZER

CIS 681 : Software Modeling & Analysis

Instructor

Dr. Jim Fawcett

Submitted By

Karankumar Patel

(SUID : 874028780)

Table of Content

1. INTRODUCTION

1.1 Executive Summary

The sole purpose of the Code analyzer is analysis of source code file and return output to user. Every Developers wants that they have this kind of tools which easily find and list the types defined, members, Size & Complexity of functions, Relationship between all types defined. Developers use this tool to analyze exciting code to understand the requirement, so they don't have to read code aimlessly and Software Testers used to find the bugs in the software when code is changed or deleted by other. The Important feature of this software is that user can pass the argument with different file pattern and option which makes tool generic.

Code analyzer take the Directory/Folder path, file Pattern, and option as user input. After that it parse the command line argument and gives it to File Manager which Collect file from the Directory or Sub Directory with a file matching pattern based on option give as user input. Now Analyzer comes in picture which control all the flow related to parsing, Analyzer pass the file stream to parser and parser uses the service of the Semi Expression & Tokenizer to break down the file stream. Parser apply Different Rules and Calculate the Size & Complexity of each member function in each file. Further it also fins Relationship between various types defined like Class, Interface, Struct and Enum, and store this result in data structure called Repository. At last Display Module check the option and display output in console or store whole output in XML File.

1.2 Specification

Code Analyzer is a tool which used to help Software Developers to analyze the source code. Code Analyzer takes File path, File Pattern and optional command as inputs from command line.There are basically three types of options we can use following way:

- **/S** – It Searched all the Subdirectory of given path to finding all files which matched with file pattern.
- **/R** – It display the Relationship between types defined in the file set.
- **/X** – It causes the output also be written in XML file.

With the given inputs code analyzer first fetch the whole file set from directory, then parser use this file set to find out following various output with respect to Action & rules:

- List out all the types defined with in file set and each member Function names in file set.

- Function size in terms of lines of code function contains.
- Complexity of each Function in each file. Function complexity is the number of scopes the function implements, including braceless scopes.
- Summary for the each file which contains number of member Function, number of types that source file have.
- Relationship betweenall types defined like Class, Struct, Interface, and Enum in all the file set, e.g., Inheritance, Composition, Aggregation, and Using.
  - Inheritance: Derived class is a specialization of the Base class. Derived class inherits all the member of base class except Constructor & Destructor.
  - Composition: Instances of value types could be composed by another class or struct.
  - Aggregation: Instances of reference types may be aggregated by another class or struct.
  - Using: Any type may be used by any class or struct. An instance of the used type is passed as an argument of a method of the using class or struct.

Code Analyzer shall accept the Command Line Arguments following way:

- The Code Analyzer shall accept file specifications using specific file patterns and path from the Command Line Interface. It will help the code analyzer search for a various file pattern like *.cs, *.cpp, *.h and in specified directory.

  Input: Command line: ../Karan/SMA *.cs

  Output on Terminal: shall fetch all the file with extension .cs from SMA Directory.

- The Code Analyzer shall accept option, e.g., /S, /R, /X from Command Line Interface. It will help Code Analyzer to perform specific function as per Option.

  Input: Command line: ../Karan/SMA *.cs /R

  Output on Terminal: shall display Relationship between types defined in all file set.

2. USERS & USES

2.1 Users (Actors)

Code Analyzer tool is one that extract size and complexity of the functions and Relationship between various types defined like Class, Struct, Enum and Interface. This tool is used by different users as per the requirements.

2.1.1 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 theydon't have to read aimlessly code to understand the functionality of the system. By using this tools developers can collect list of function with its name, size and complexity, Relationship of types defined. Sometimes they understand the existing software by reading all this information given code analyzer.

2.1.2 System Analyst

A System analyst is a person whoanalyses 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. Code Analyzer clearly proves to be one of the main tools a System Analyst would use because Code Analyzer gives size and complexity of each function easily.

2.1.3 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 understand whatthe software is about and information about all the class, function, interface etc. Sometimes they have to ensure that there is not any function which is more complex and large, if no then Software Tester would use code analyzer to find information about project.

2.1.4 Manager

A mangers not necessary be a Software Project Managers, have to deal with a lot of information, they have to keep track of all the activities going under them. We can take a product manager's scenario and can see how a Code Analyzer would help them. If they wants to send the details of a product/software to a customer or a quotation of a product to a consignee, they do not have to look at each and every files/packages searching for the details to find how large project it is and the complexity of the project, they can just give path of project folder as argument to Code Analyzer and then Code Analyzer gives all the information like numbers of files/package, Complexity and size of each function and relationship between all the types defined.

2.2 Uses of Code Analyzer

2.2.1To Browse Relationship between types defined

The most important use of code analyzer is to find Relationship between types defined, it will help them to understand what the program is about. Sometimes there is not proper documentation done by previous programmer in this situation it helps a lot.

2.2.2Code 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 code analyzer extract relationship between classes so by identifying missing class relationship we can find which part of code has been changed.

2.2.3 To enhance Software product

Code analyzer give summary of relationship between types defined so it helps a developers to enhance exciting software easily and efficiently. Further, Developer can know property of function like size and complexity so they can add new feature at appropriate position in code which do not affect the overall system

2.2.4Identifying the risk associated with program

Code analyzer constructs two metrics for each member function: its size in lines of code and complexity defined as the number of elements in its scope tree. So anyone can evaluate risk associated with program based on complexity and size of functions. If program have very large and complex function then risk associated with it is high.

2.2.5 Maintenance of Software

Whenever the feature of application is changed a plenty of time effort required to maintain the application. Code analyzer is very useful for software maintenance.

2.2.6 Collect Information of inheritance tree

One can easily find the depth of inheritance tree for class which is useful to design errorless software architecture.

3. ACTIVITY DIAGRAM

3.1 Diagram

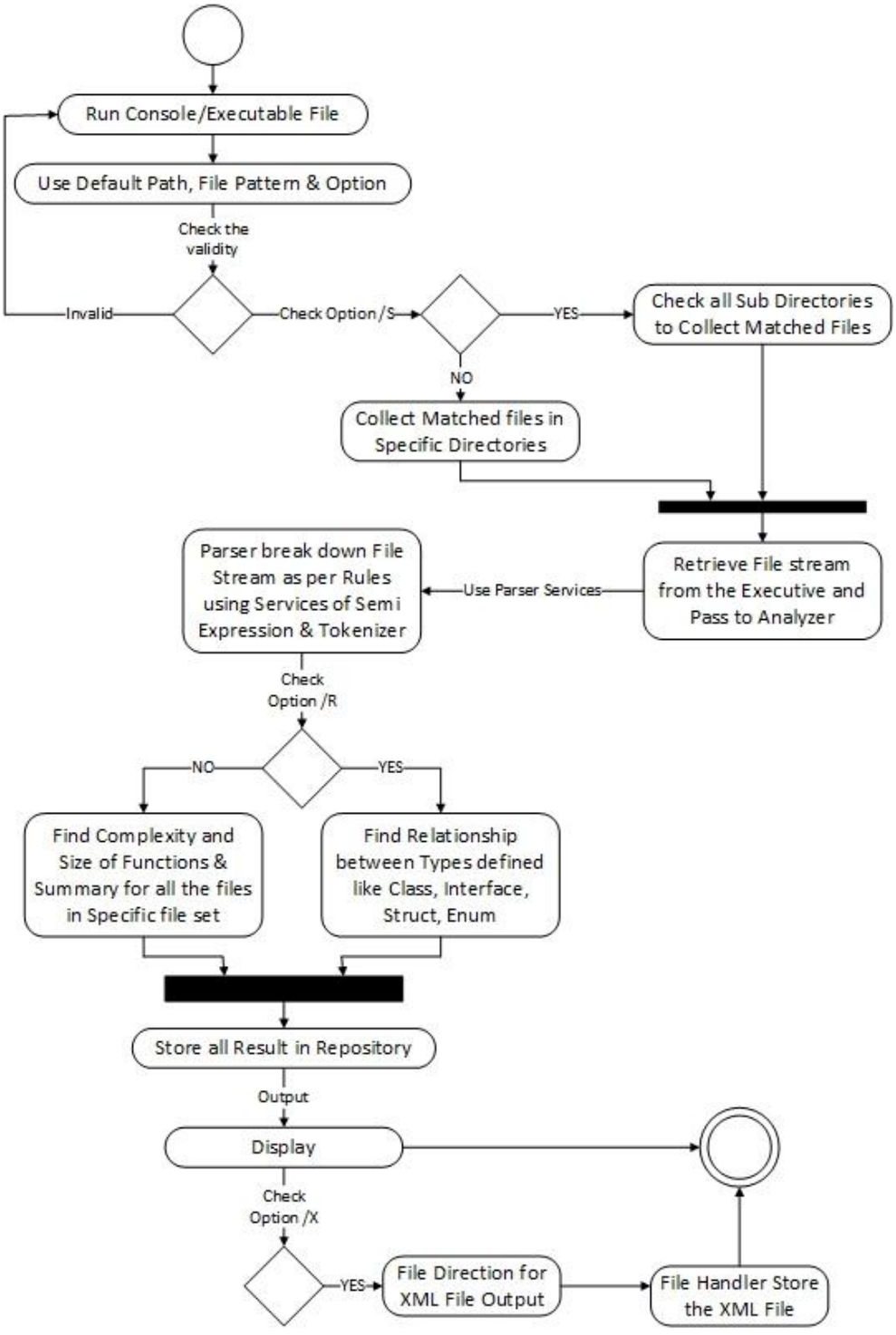

Figure 1: Activity Diagram for Code Analyzer

The above Activity diagram show High level Description of Code analyzer's overall activity.

3.2 Activity Description

3.3.1 Run console/Executable File

There is two option to execute the functionality of the code analyzer. When anyone run console or Executable File code analyzer gives output based on the command line argument.

3.3.2 Directory Path, file Pattern and Option Matching

User gives the Directory/Folder Path, file pattern and option as input to code analyzer. So the first task of it to parse the arguments and check the validity of the arguments. If there is invalid arguments like invalid path, file pattern or option then code analyzer gives appropriate error message.

3.3.3 Collecting File Stream from Directory or Sub Directories

Code analyzer check input arguments if there is **/S** option then it rooted all the sub directories and fetch file from each sub directories with pattern matching. If there is not **/S** option then it only collect file from specific directory with Pattern matching.

3.3.4 Analyzer Retrieves File Stream from Executive

Analyzer Retrieves File stream from Executive to decompose all the file so code analyzer could calculate Size and Complexity of Functions and Relationship between types defined. Analyzer control all the flow for parsing activity.

3.3.5 Parser Detect all the Types defined and members in each file

As soon as the file stream is acquired from the Analyzer, Parser do decomposition with the service from Semi Expression and Tokenizer. Then, after Tokenizer and Semi Expression finished their job and return the decomposed file stream which includes only key tokens for analysis to parser. Parser uses the Rules to detect all the Types Defined and members in the each file.

3.3.6 Find Complexity & Size of each Function in each file

The first Responsibility of code analyzer is to find Complexity and size of each Function and store results in to repository. Here Size is in terms of line of count of function and Function complexity is the number of scopes the function implements, including braceless scopes.

3.3.7 Find Relationship between Types Defined

The Second Responsibility of code analyzer is to find Relationship between all the Types Defined found during decomposition. For this operation code analyzer take one file at time and using Action & Rules package find basic relationship between all the Types defined, and store this result to in repository to further uses.

3.3.8 Store all The Result in Repository

All the information found during parsing like types defined, members, size & Complexity of function, relationship stored in a data structure which is called as a Repository. Data stored in repository is used by Display, File Redirection.

3.3.9 Display Output

Display activity display output of code analyzer based on user input. It is progress of code analyzer which display all the result like all the types defined, members, size & Complexity of the functions, summary of the each file in specified file stream, Relationship between types defined. Further it checks the **/X** option in user input to store all the output in XML File.

3.3.10 File Redirection for XML File Output

File Redirection activity required if there is **/X** option in user input to store whole result in XML file, then after file Handler store this XML file.

4. PARTITIONS

4.1 Package Diagram

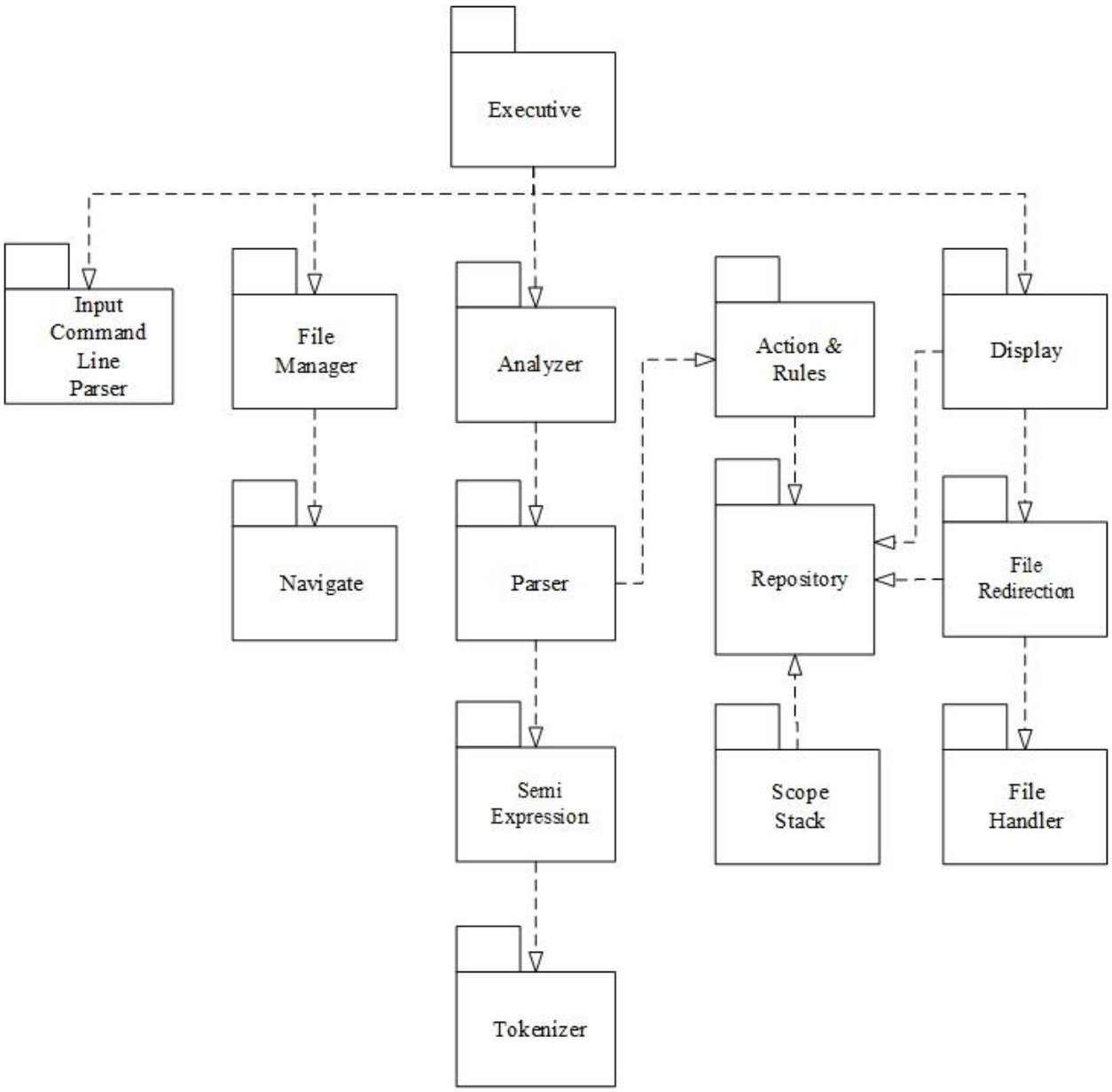

Figure 2: Package Diagram for Code analyzer

4.2 Package Description

Package Description describe the functionality of each and every packages of code analyzer.

4.2.1 Executive

Executive package is heart of this package which maintain and use the functionality given by other package and control the whole flow of program. An Executive package also known as an executable that contains an exe file which used to run application. It arranges the tasks in order which to be carried out.

4.2.2 Input command Line Parser

Input command Line Parser is the first package by which user interact with this Code Analyzer tool. This package provide functionality to manipulate command line argument. The main task of this package is to take path, file patterns and different option from the user and provide an error message if there is erroneous data in the command line argument. The reason behind to take all these arguments from users to let them work with specific file, Directory and option. At last if all the arguments are right then it sends these data to File Manager.

4.2.3 File Manager

File Manager receive directory path from Input command Line Parser, and its try to find out valid path throughout to the target directory. If it found invalid path then File Manager informs about this invalid path to Input command line parser. If it's valid path then File manager pass this path to Navigate package.

4.2.4 Navigate

First, Navigate package check the option passed in command line and if it found **/S** option then it will rooted every file with the same pattern passed in argument and collect all the file from subdirectory. If it does not found **/S** option then it just collect file with matching pattern from directory only. At last it will pass file stream to upper package.

4.2.5 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 Function size, Complexity, Relationship between types defined. It control all the activity relating to parsing.

4.2.6 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 means a function name. So, when the Parser Package detects the name of function, it will use the Actions & Rules Package to execute all the operations for a new function, including recording the function name, beginning to count the function size, and beginning to calculate the complexity of that function. However, Parser Package uses the service of Semi Expression and Tokenizer Packages to accomplish the complete.

4.2.7 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.

4.2.8 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 calculate size of the function, Complexity. 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.

4.2.9 Actions& Rules

Actions & Rules Package contains the set of rule for finding several useful code analysis information. This package also responsible for recording and calculating the data its collects, i.e. line of count for each example, complexity of each function with braceless scope, Relationship between types defined like Interface, Struct, Enum, class. Action & Rules Information fetch and store useful information in Repository Package, i.e. if code analyzer has to find the relationship between types defined then in first pass it store information in Repository which it fetch during second pass.

4.2.10 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 function names, sizes and 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.

4.2.11 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, function and control via the class ScopeStack.

4.2.12 Display

Display Package is used to display output to potential user. Display package fetch useful data stored in Repository Package like types defined, members, size and complexity of the functions and relationship to display the output of users arguments.

4.2.13 File Redirection

File Redirection check command line argument if users gave a **/X** option as argument then File redirection Package store the whole output in XML file. Further, File Redirection manager interacts with Executive, upon user request to re direct the output of the Code Analyzer is redirected to a file on file Handler.

4.3 Context Diagram:

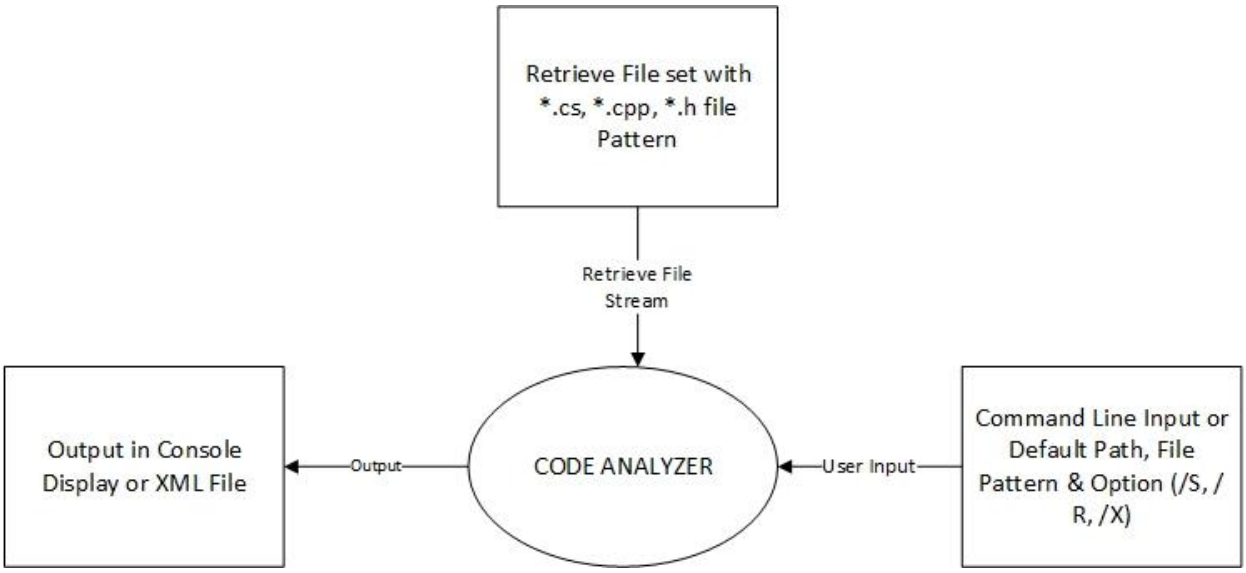

Figure 3: Level-0 Context Diagram

Figure 3 shows how the system (Code analyzer) and the external entities that interact with the system. Firstly Code Analyzer takes the input from the user or use Default Path, File Pattern & Option set by Developer. As per the Input Code analyzer Retrieve File set from the Specified path with file pattern and give appropriate output on console or save output in XML file.

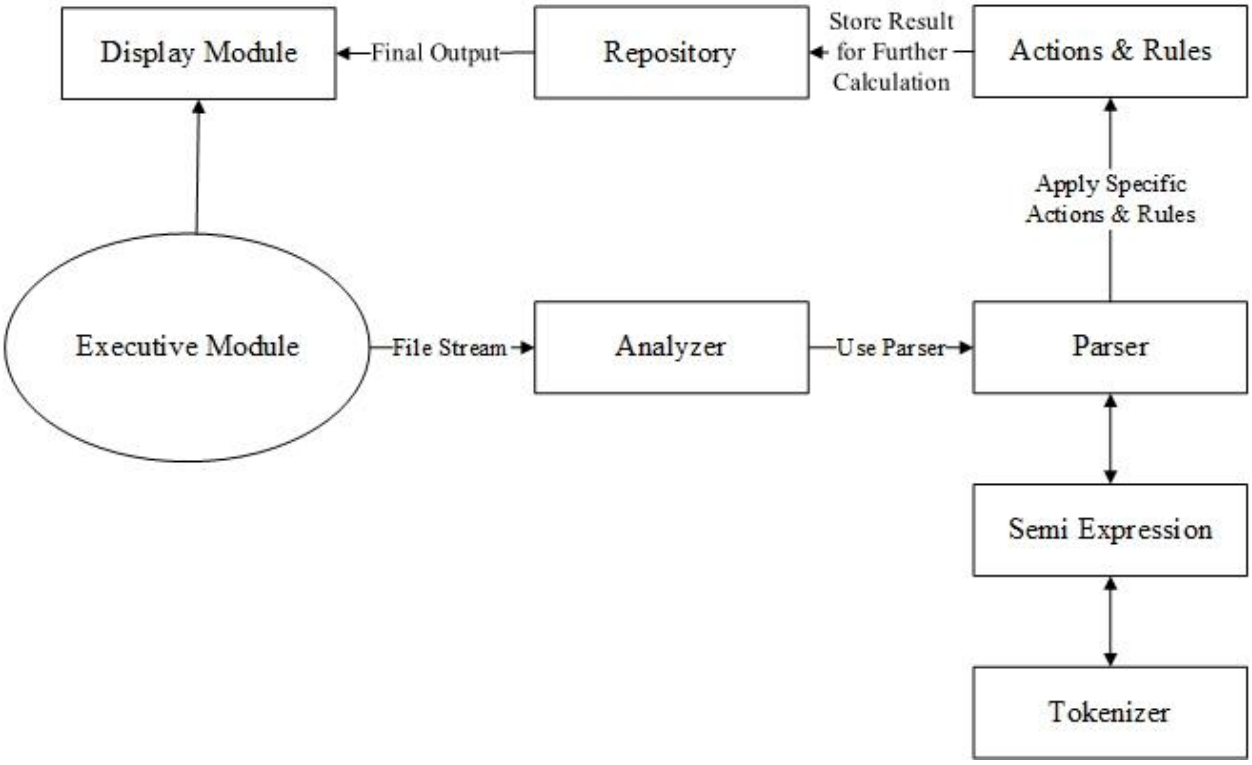

Figure 4: Level-1 Context Diagram

Figure 4 shows how the detailed modules communicate with each other in the Code analyzer Module appearing at Figure 3.

Executive Module pass the file stream to Analyzer which send this file stream to parser which uses the services of Semi Expression and Tokenizer to decompose the File stream. Then, Actions & Rules apply on the decomposed file stream to analyze the stream to generate results such as function size and complexity. After that, the results will be store to Repository for further uses. At last, the final result wills be sent to Display Module for displaying or save to XML File.

5. CRITICAL ISSUES

5.1 Directory/Folder and File Change during Analysis Processing

This tool is most probably used to analyze multi-user project. Sometimes it may happens that other user may change the following property of the Directory and file:

- Existing files or Directories might be removed and/or renamed
- New Files or Directories might be added

This operation affect the output of the code analyzer.

Solution: To apply lock mechanism for disabling write access may not be a feasible option because code analyzer release lock at the end of processing. So to use code analyzer efficiently there is mechanism which ignore the certain change in files and Directories which execution of code analyzer.

5.2 XML File Output size Issue

When output has to store in XML file then there is one problem is that XML file does not use local disk efficiently. So when tools have to parse large number of files then it required more space further user may execute this tool many time to keep track and understand the output.

Solution: When user provide /X option in argument; at end of processing while File Redirection have to store output in XML file, it check that there is XML file created previously if so then it pop up overwriting option for overwrite single XML file. When tool has to process giant project then this solution save large memory space.

5.3 Performance for large number of file set

This is very common issue may face by Code analyzer. When this tool have to process large number of file set, to compute various output it may takes longer time than it required usual operation.

Solution: To do this task efficiently developers have to build thread model to do parallel execution for each different file. Later, output from each thread will be merged to do further analysis. It will save lots of processing time.

5.4 User Provide Invalid/Different File Pattern and/or Option

Command line input gives freedom to enter Different File Pattern and Option as Input for user. Sometimes users unintentionally enter Invalid File Pattern and/or Option which might affect the execution of the tool or Crashed the tool.

Solution: While Parsing of Input command arguments code analyzer should have to check the validity of arguments. Every time File pattern and Option should be checked by code analyzer to process each data efficiently. *.cs File pattern and /S, /R, /X option is valid for this tool so every time other pattern and/or option should be ignored by code analyzer.

5.5 Display large amount of output in console

When Code analyzer have to execute large number of file, output shown on console causes problem for users. Users do not understand the output on console easily. Further, sometimes users could not track useful information from the output.

Solution: Graphical user interface is feasible solution to show output of all processing done on file. Developer can provide separate option to show types defined, member functions, Size & Complexity of the functions, Relationship between various types defined. So users can easily browse what they want from the output.

6. CONCLUSION

The Code Analyzer is a tool which is used process medium and large sized of projects. It takes directory path, file patterns and option as command line input. Users can pass *.cs file pattern and /S, /X, /R as option to argument to process project as per their requirement. Code Analyzer gives summary of all file sets, Size & Complexity of the functions, Relationship between various types defined as output. Code Analyzer also provide option to store whole output in XML file. Here we discussed various how Users (Actors) uses this tools for various purposes. Users can use this Code Analyzer to understand, enhance, test and reused exiting System. The Code Analyzer divided in various package so we can correlate the functionality of each modules.

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