CMS-2 Source Monitor and Analyzer

Design Document – Team Ducks

Author: Matthew Gimbut

Team Members

Matthew Gimbut

Curtis Baillie

Johan Burke

Chris Curreri

Tom Harker

Abbi Palen

Revision 1

March 31, 2017

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to explain the design of the CMS-2 Source Monitor and Analyzer tools. The CMS-2 Source Monitor and Analyzer were created to replace dated tools created in Fortran which were meant to create reports based on an analysis of CMS-2 files. This document should allow a new developer to continue the project in the future with ease.

1.2 Scope

This design document details both the Source Monitor and Source Analyzer tools created for ASRC Federal Mission Solutions. It provides a broad overview to all included systems, as well as an in-depth look at how the systems work.

2. Design Overview

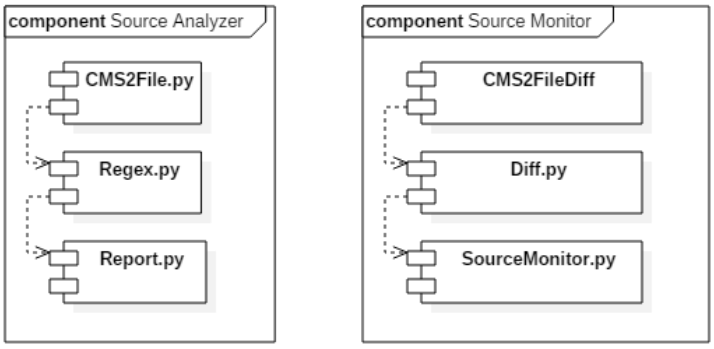
2.1 Technologies Used

The CMS-2 Source Analyzer was built using Python version 3.0.1. While there are no external dependencies for the program, it does make use of several Python packages including re, os, sys, difflib, and git.

2.2 Component Overview

The CMS-2 Source Monitor and Analyzer program is broken down into two major components which each run independently. These components are the Source Analyzer and the Source Monitor. The Source Analyzer is responsible for dissecting a set of CMS-2 files and counting items of interest. The Source Monitor is used to compare different versions of the same file. Each component contains three subsections which work together to produce the output. These components are shown in Figure 1.

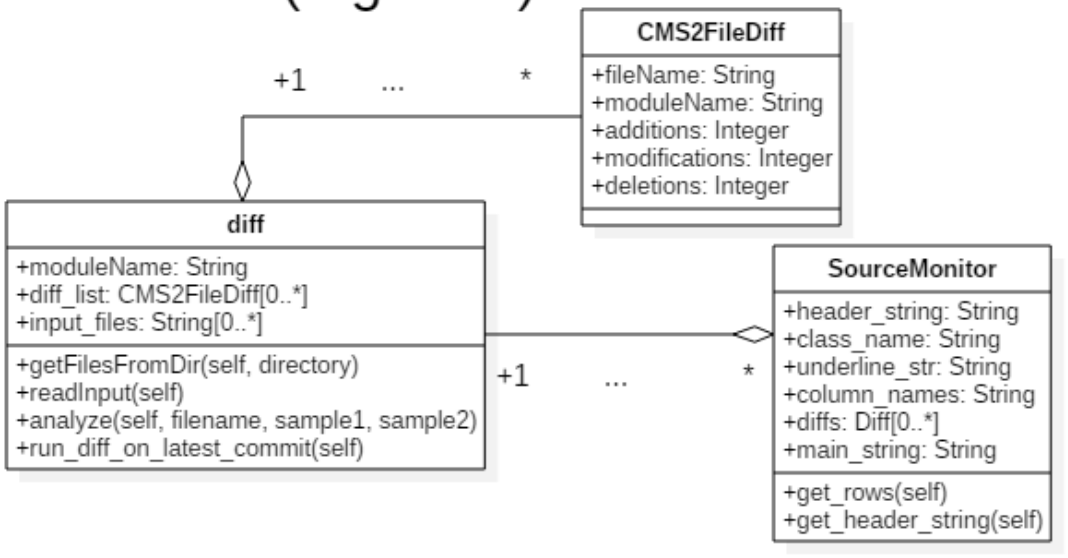
Project Components (Figure 1)



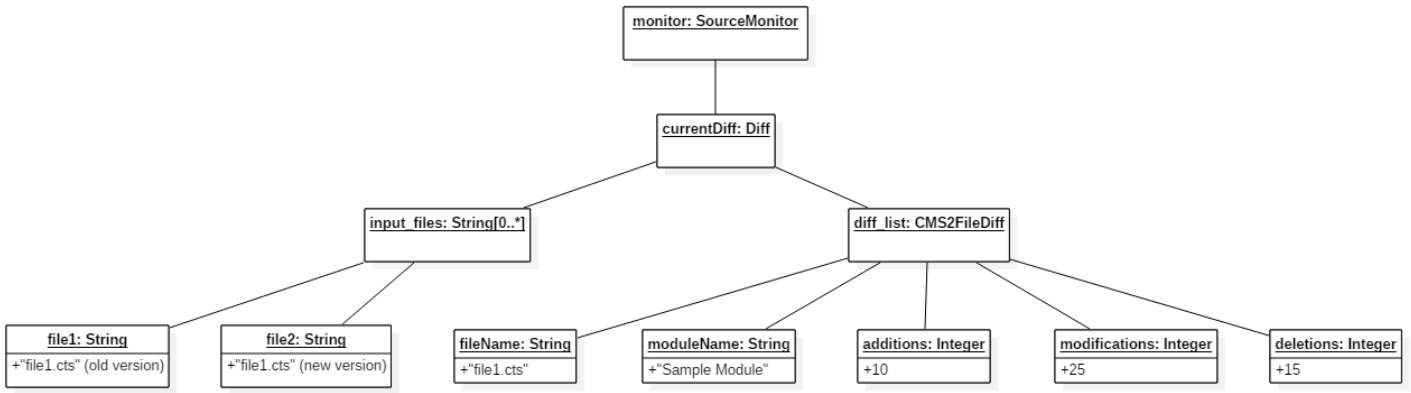
2.3 Class Overview

Figures 2 shows a class diagram of all classes in the Source Monitor component. Figure 2.1 shows how a typical instantiation of these classes would function. Figure 3 shows a class diagram of all classes in the Source Analyzer component.

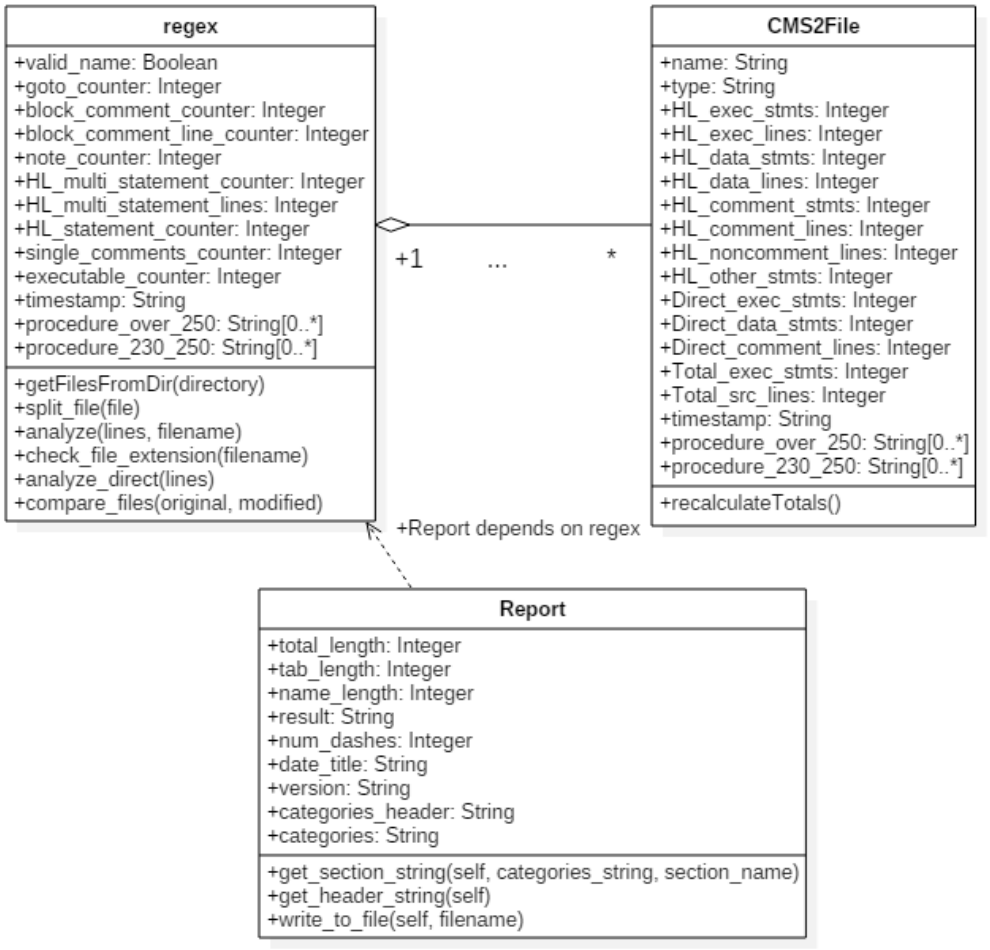
Diff Class Diagram (Figure 2)



Source Monitor Object Diagram (Figure 2.1)



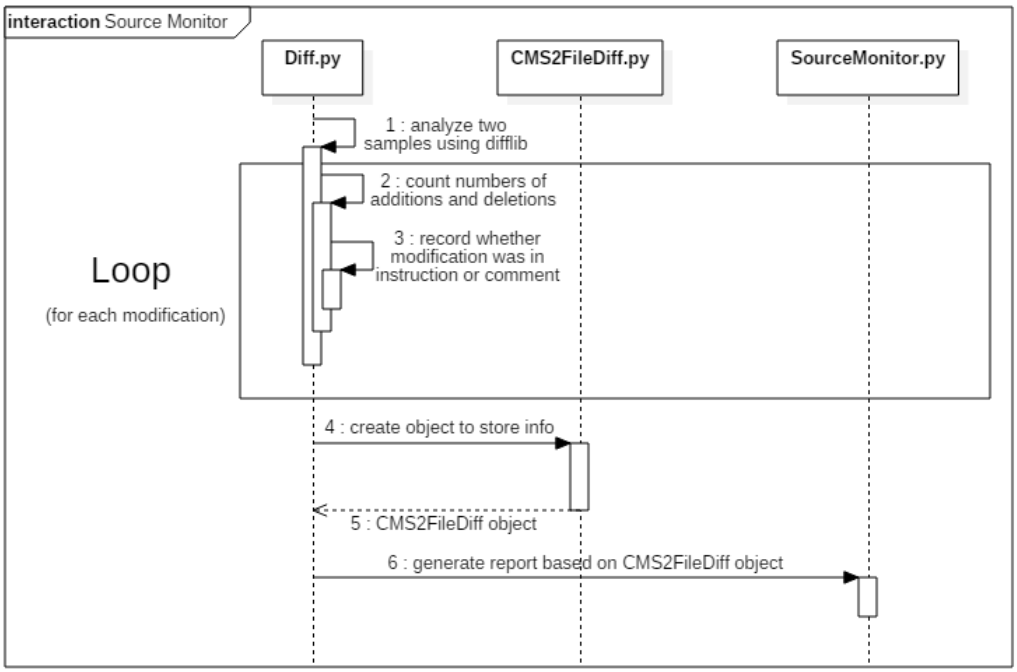
Regex Class Diagram (Figure 3)



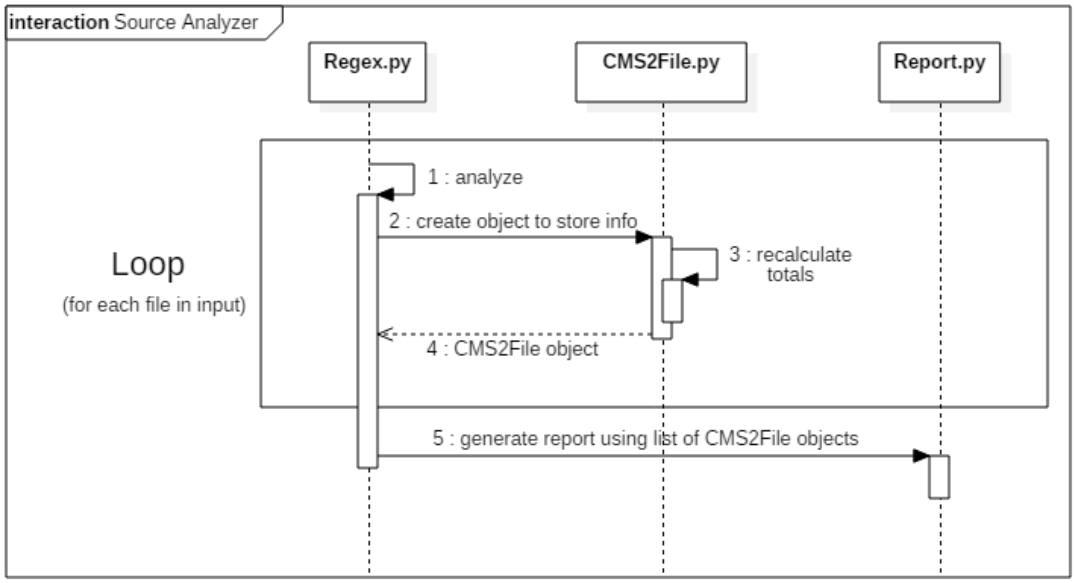
3. Functionality Overview

3.1 Sequence Diagrams

Figures 4 and 5 are a sequence diagrams to describe the flow of data in the SM and SAN components respectively.



Source Monitor Sequence Diagram (Figure 4)

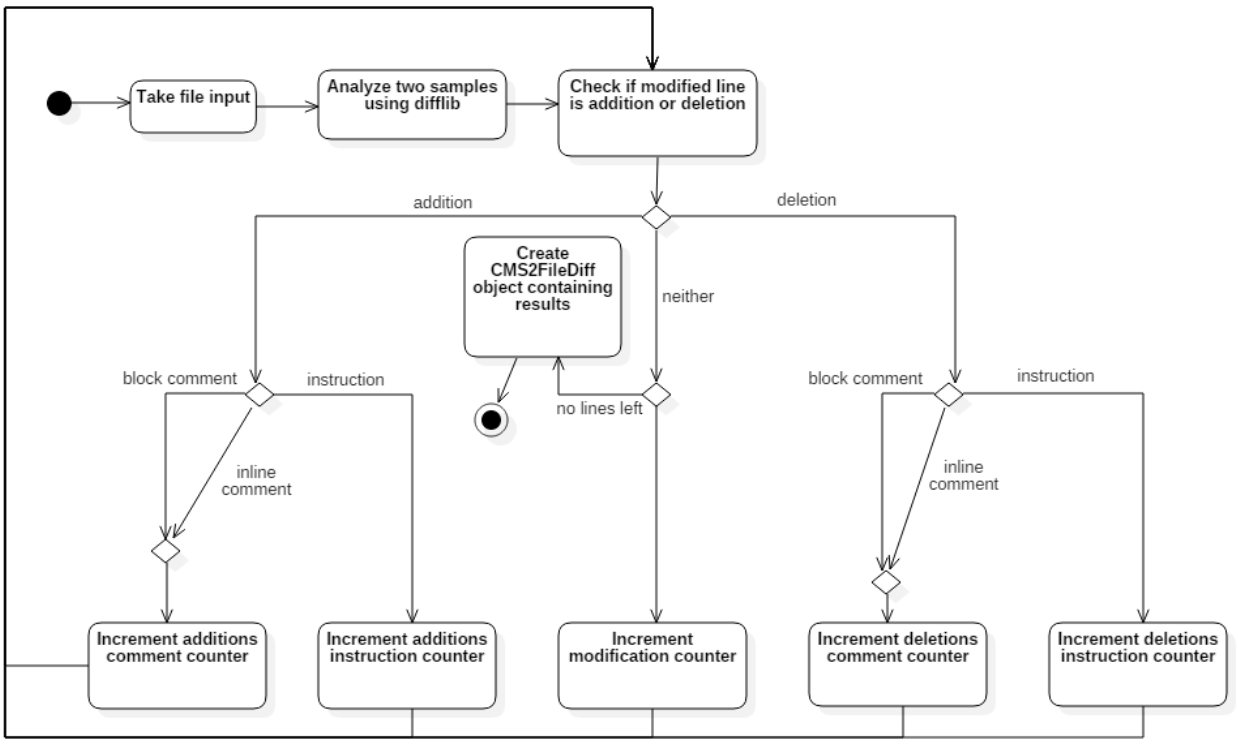


Source Analyzer Sequence Diagram (Figure 5)

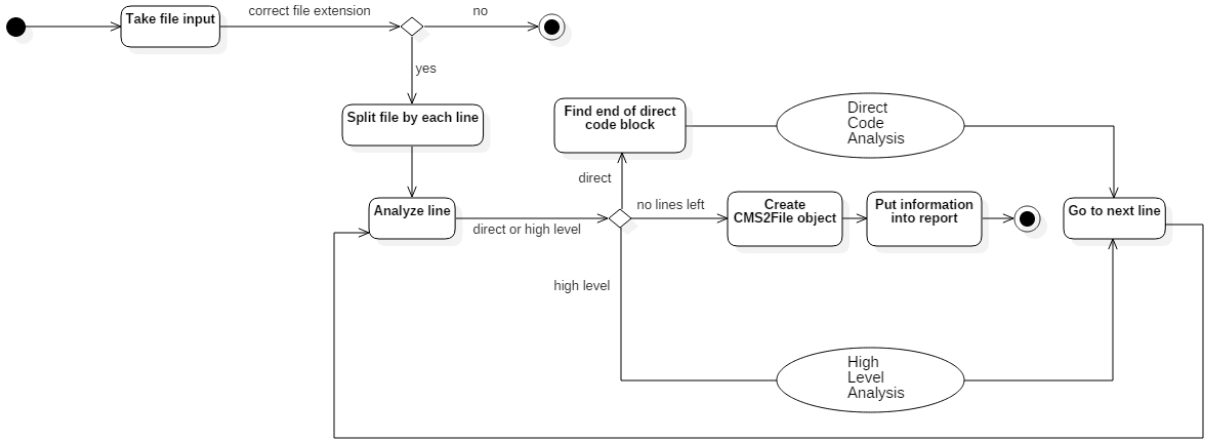
3.2 Activity Diagrams

Figure 6 shows an activity diagram to describe the process of diff.py in the SM component. Figures 7, 7.1, and 7.2 all describe the process of regex.py in the SAN component. Figure 7 provides a broad overview of how the code executes. Figures 7.1 and 7.2 provide a more in-depth look at the execution of direct code analysis and high level code analysis respectively.

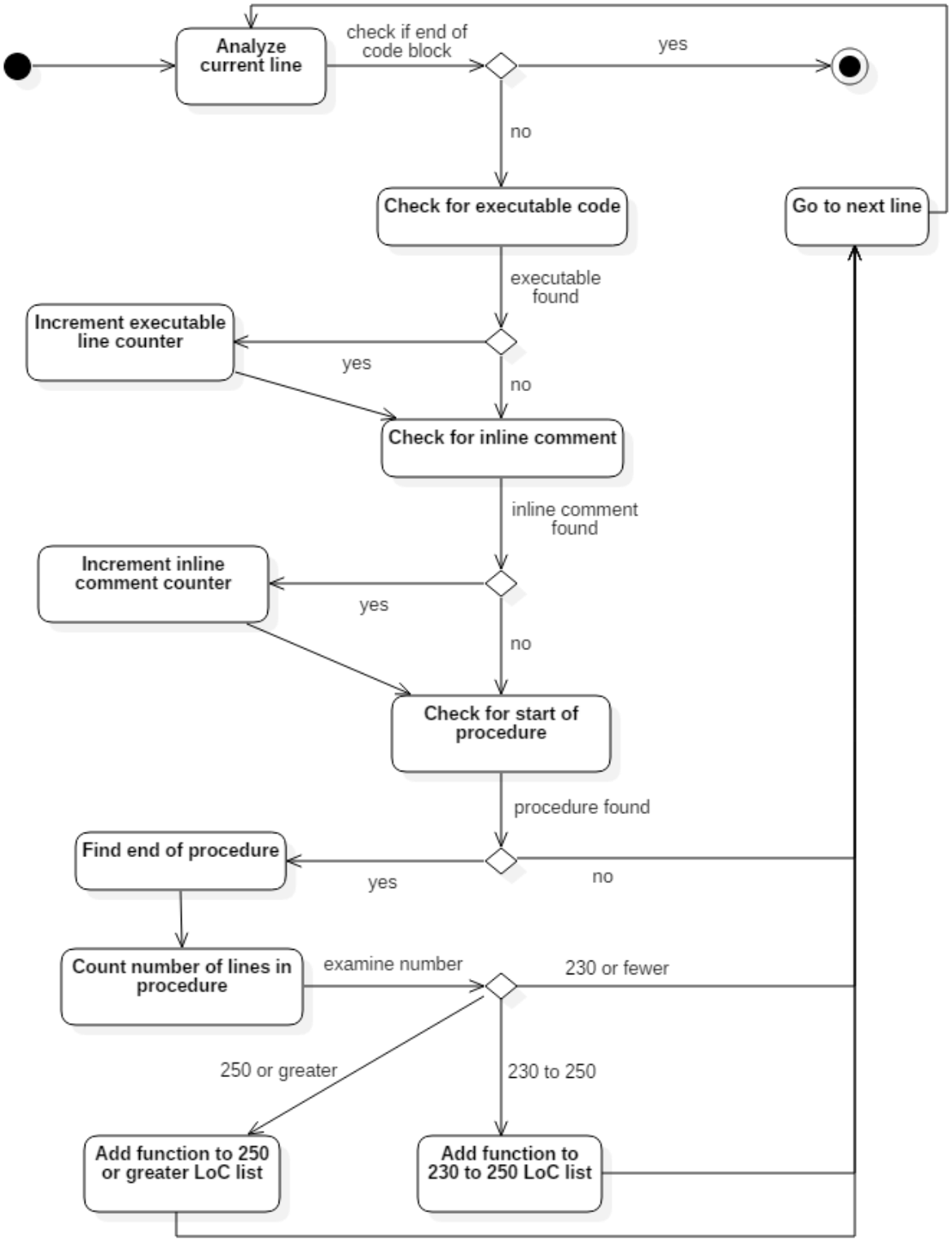
Diff.py Activity Diagram (Figure 6)



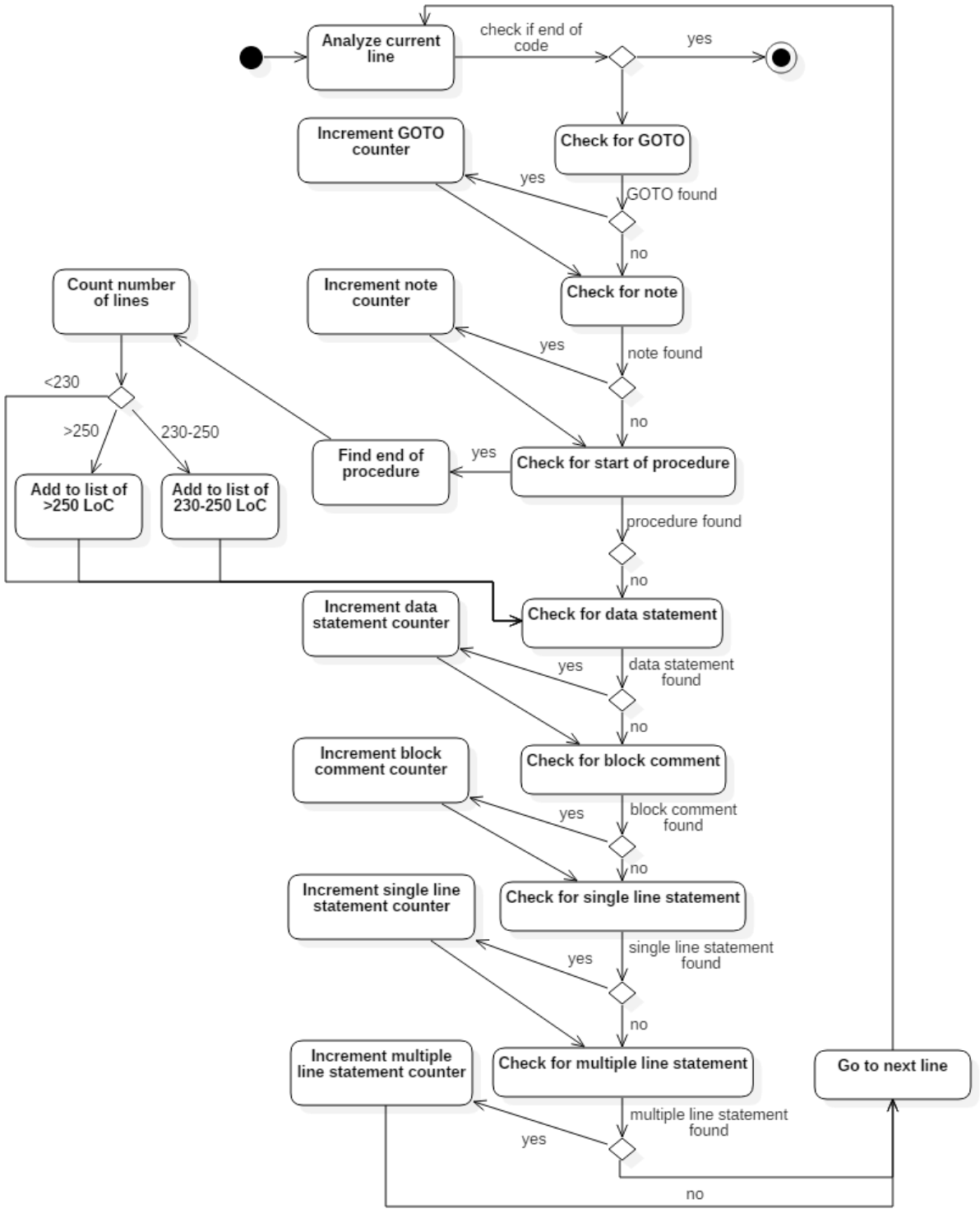
Regex.py Activity Diagram (Figure 7)



Direct Code Analysis (Figure 7.1)



High Level Code Analysis (Figure 7.2)



4. Expanding the System

4.1 Expanding Regex Functionality

To add regex functionality, the new code should be inserted into the regex.py class. Simply add another check to see if a line matches a regular expression which detects whatever pattern the developer is looking for. The information can then be saved in a new field added to the CMS2File.py class.

4.2 Expanding Diff Functionality

To add diff functionality, the new code should be inserted into the diff.py class. Any new information that must be stored should be stored in a new data field in the CMS2FileDiff.py class.

5. Terms and Definitions

Term	Definition
CMS-2	Programming language used by the US Navy
Regex	Regular Expression
SAN	Source Analyzer
SM	Source Monitor

6. Revision History

Version	Date	Updated By	Revision
1.0	March 30, 2017	Matthew Gimbut	Initial Release