5-6-2020

Undergraduate Code Metrics Analytics for Development Benchmarks

Kevin Geissler

Undergraduate Code Metrics Analytics for Development Benchmarks

By

Kevin Geissler

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

May 6, 2020

Honors Capstone Director: R. Kevin Preston

Computer Science Lecturer

Kevin Geissler 5/6/2020

_____

Student Date

_____

Director Date

_____

Department Chair Date

_____

Honors College Dean Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College
Frank Franz Hall
+1 (256) 824-6450 (voice)
+1 (256) 824-7339 (fax)
honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted as a bound part of the thesis.

In presenting this thesis in partial fulfillment of the requirements for Honors Diploma or Certificate from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department, Director of the Program, or the Dean of the Honors College. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

Kevin Geissler

_____
Student Name (printed)

_____
Student Signature

5/6/2020

_____
Date

Abstract

The C/C++ Code Analysis tool in Visual Studio compares C or C++ source code to predefined rules in a ruleset and can show warning or error messages when rules are violated. This can be useful for catching mistakes that do not stop the program from compiling or warning the user about code that violates guidelines. For this project I created a C++ program in Visual Studio 2019 for use on a Windows operating system that iterates through all the files in a directory, uses command prompt to run analysis on each file, and converts the output listed in the XML log file into a list of defect codes. By running this program on student code provided by a professor, I was able to do basic analytics on the output. The project could detect duplicate code or identify poor programming practices that need to be addressed in the course lectures. The project did not find useful distinctions between the student code for varied assignments.

Development Process

The process of writing code for this project began with researching the analysis tool. Analysis provided by the analytics tool is summarized in Table 1. After learning to use the analysis tool in Visual Studio, the next step was learning the commands to use the analysis tool in the developer command prompt, and then the default command prompt. These commands could be used to make a C++ program analyze other programs with the system() function. I set up a while loop to analyze every file in a folder. The commands would save log files for every analysis. Then additional functions

were used to extract from each of these logs a list of every defect code that the analysis produced. The program also counts these codes to calculate the average number of warnings produced by files in the given folder. The program was finished with the addition of an option to check for duplicate lists among the lists of defect codes. An effort was made to get the program to function with the files in their original forms, but this goal was abandoned due to time constraints and continuing errors caused by inconsistent precompiled headers. These issues were resolved by manually removing #include statements for precompiled headers from each of the files. Files with .txt extensions were renamed with .cpp extensions and spaces in file names were removed. The program went through several revisions. With the default ruleset many of the code samples did not produce any errors, so the C++ Core Guidelines were added to the rules being checked. This caused every program to produce warnings and is the setting that was used for the metrics in this report.

	Description	Example
Filepath	The location of the file with the defect.	`C:\Users\kevin\Desktop\StudentCode\CS1212Fall2018Program2\`
Filename	The name of the file with the defect.	`1018220410-Program_2.cpp`
Line	The location of the defect in the file.	`18`
Colum	The location of the defect on the line.	`8`
Defectcode	A number that identifies the rule that was violated to cause the defect.	`26494`
Description	Text provided to summarize why a defect was detected in the analyzed colde,	`Variable 'weight' is uninitialized. Always initialize an object (type.5).`

Table 1: Summary of Analysis Tool Output

Comparison of Code for CS 121 Projects

The student assignments that were analyzed included four different programming assignments of increasing difficulty and represented a total of 158 different submissions. This project's use of the C/C++ Code Analysis tool was ineffective at identifying different complexity between programming assignments. The additional warnings caused by the increased length of code for more complicated projects may have been offset by the reduced warnings caused by better coding practices of more experienced students. When the default ruleset was used, programs produced few or zero warnings with no clear trend. Figure 1 shows the warnings produced when the C++ Core Guidelines were included with the ruleset. Many of the warnings were caused by included files rather than the student's code.

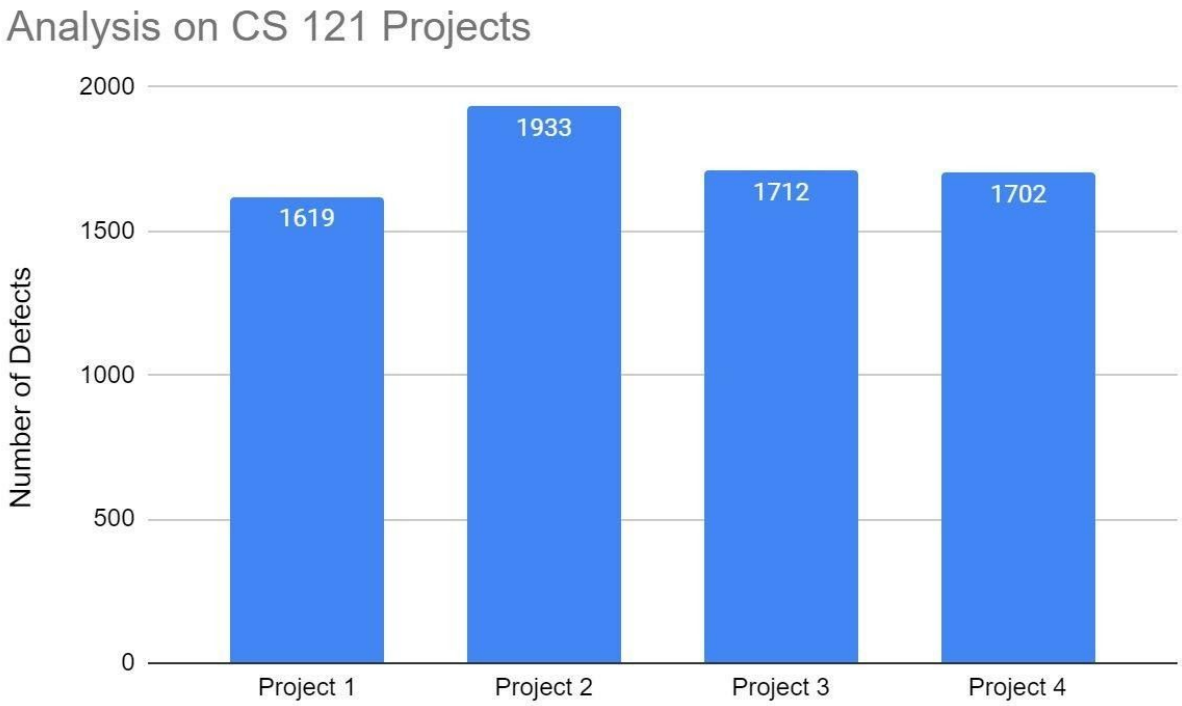


Figure 1: Graph of Results of Analysis

Detection of Duplicate Code

This program can be used to detect duplicate code. Identical files will produce identical sets of defect codes. An option is available to compare the defect codes of generated files and report any files that have identical sequences of defect codes. The function did not produce any false positive results. This method of comparison has advantages over simply comparing all of the characters in each file because it can detect duplicates even when additional blank space or comments are added. While the program was able to successfully identify complete copying, further research into the

efficacy of comparing partial matches of defect codes may be warranted if a professor wants to use the C/C++ Code Analysis tool to detect plagiarism.

Conclusion

This project was able to analyze an arbitrary number of files and use the analysis to detect code duplication. It did not find any meaningful relationships between the warnings produced by code analysis and the difficulty of the projects the code was written for. While It is powerful, the C/C++ Code Analysis tool is meant for detecting defects in code. Attempting to use it for other purposes, as I have done, may not be ideal. I recommend that future research into producing Code Metrics should begin with a broad search for a better way to produce comparable metrics.

Appendix A: Project Code

```
     #include <iostream>
     #include <fstream>
     #include <string>
     #include <vector>
     using namespace std;
     int main()
     {
      if (false) //true will check for duplication, false will
analyze files
      {
       vector<string> Codes;
       vector<string> Names;
       //this is the location of the defect codes used to check
for duplication
       string temp = "cd
C:/Users/kevin/Desktop/StudentCode/Output3/ && dir /b>
file_names.list";
       char tab[1024];
       strcpy_s(tab, temp.c_str());
       system(tab);
       string fileName;
       ifstream
MyReadFile("C:/Users/kevin/Desktop/StudentCode/Output3/file_names
.list");
       while (getline(MyReadFile, fileName)) {
        string code = "";
        ifstream
MyReadFile2("C:/Users/kevin/Desktop/StudentCode/Output3/" +
fileName);
        string lineText;
         while (getline(MyReadFile2, lineText)) {
          code = code + lineText;
         }
       Codes.push_back(code);
       Names.push_back(fileName);
       MyReadFile2.close();
       }
       MyReadFile.close();
       for (int i = 0; i < Codes.size(); i++)
       {
        for (int j = 0; j < Codes.size(); j++)
        {
```

```
         if (i != j && Codes.at(i) == Codes.at(j))
         {
          cout << Names.at(i) << " and " << Names.at(j) << " are
the same\n";
         }
        }
       }
      }
      else
      {
      int totalErrors = 0;
      int totalFiles = 0;
      int totalFails = 0;
      //floderName is the folder that contains the .cpp file that
will be analyzed
      // string floderName =
"C:/Users/kevin/Desktop/StudentCode/CS1212Fall2018Program1";
       //string floderName =
"C:/Users/kevin/Desktop/StudentCode/CS1212Fall2018Program2";
      string floderName =
"C:/Users/kevin/Desktop/StudentCode/CS1212Fall2018Program3";
      //string floderName =
"C:/Users/kevin/Desktop/StudentCode/CS1212Fall2018Program4";

      string temp = "cd " + floderName + " && dir /b>
file_names.list";
      char tab[1024];
      strcpy_s(tab, temp.c_str());
      system(tab);
      // Create a text string, which is used to output the text
file
      string fileName;
      // Read from the text file
      ifstream MyReadFile(floderName + "/" + "file_names.list");
      // Use a while loop together with the getline() function to
read the file line by line
      while (getline(MyReadFile, fileName)) {
       if (fileName.substr(fileName.length() - 4) == ".cpp") {
        string temp = "cd /Program Files (x86)/Microsoft Visual
Studio/2019/Community/VC/Auxiliary/Build && vcvarsall x86 && cl
/EHsc /Zs /analyze:logC:/Users/kevin/Desktop/StudentCode/Rules/"
+ fileName + " /analyze:plugin EspXEngine.dll /analyze:quiet
/analyze:WX- /analyze:only
/analyze:rulesetC:/Users/kevin/Desktop/StudentCode/Rules/RuleSet1
.ruleset " + floderName + "/" + fileName;
```

```
        char tab[1024];
        strcpy_s(tab, temp.c_str());
        system(tab);
        // Create a text string, which is used to output the text
file
        string fileName2;
        // Read from the text file
        ifstream
MyReadFile2("C:/Users/kevin/Desktop/StudentCode/Rules/" +
fileName);
        if (MyReadFile2.peek() ==
std::ifstream::traits_type::eof()) {
         totalFails++;
        }
        else
        {
         totalFiles++;
         string lineText;
         //This is the forder where the defect codes are stored
         ofstream
errorList("C:/Users/kevin/Desktop/StudentCode/Output3/" +
fileName);
         while (getline(MyReadFile2, lineText)) {
          int start = lineText.find("<DEFECTCODE>") + 12;
          if (start != string::npos) {
           int stop = lineText.find("</DEFECTCODE>", start);
           if (stop != string::npos) {
            string code = lineText.substr(start, stop - start);
            errorList << code << "\n";
            totalErrors++;
           }
          }
         }
         errorList.close();
         MyReadFile2.close();
        }
       }
      }
      MyReadFile.close();
      cout << "\n\nAverage Errors\n" << totalErrors / totalFiles
<< "\n\nFailed Files\n" << totalFails << "\n\n\n\n";
     }
      return 0;
     }
```

Appendix B: Slide Show

Using Code Analysis with Visual Studio 2019

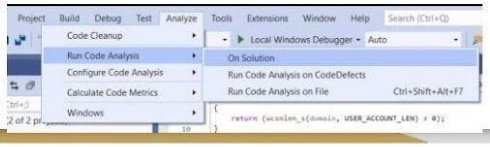
Presentation by Kevin Geissler

Introduction

The code Analysis tool compares C or C++ source code to predefined rules and can show warning or error messages when rules are violated. This can be useful for catching mistakes that do not stop the program from compilings or warning the user about code that violates guidelines.

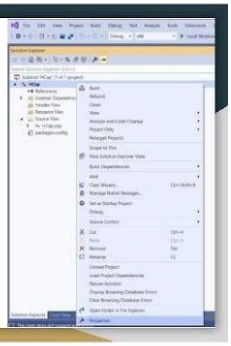
How to use Code Analysis

Select “Analyze”, then “Run Code Analysis”, and then the choose to analyze the project, solution, or file.



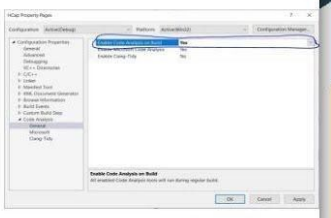
How to Enable Code Analysis on Build

Right Click on your project’s name in the Solution Explorer. Then left click on “Properties”.



How to Enable Code Analysis on Build (cont)

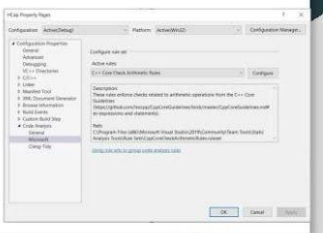
Under “Code Analysis” Change “Enable Code Analysis On Build” to “Yes”.

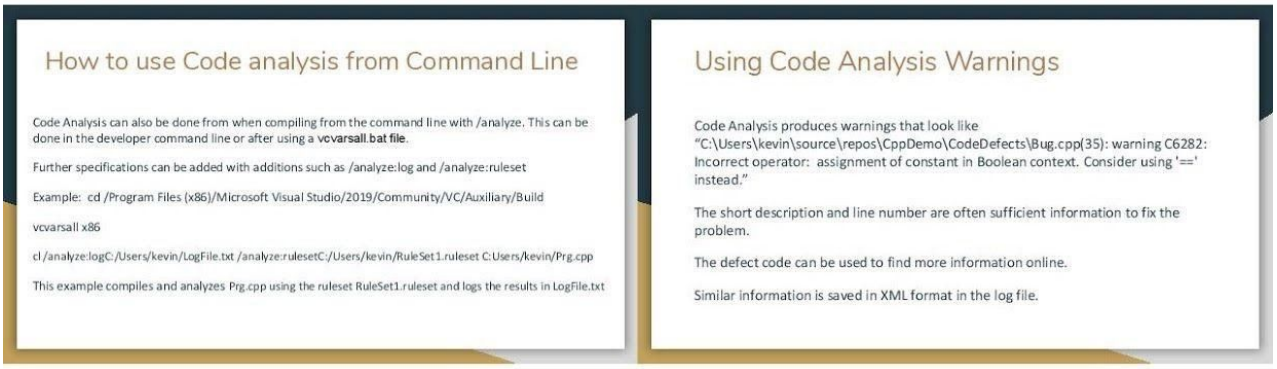


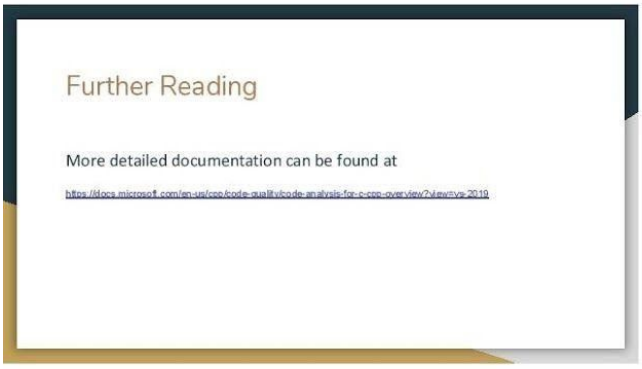
How to Choose a Rule Set

A rule set can be chosen on the “Microsoft” tab.

The rule set determines what code structures will be detected when the code is analysed.




How to use Code analysis from Command Line
Code Analysis can also be done from when compiling from the command line with /analyze. This can be done in the developer command line or after using a vcvarsall.bat file.
Further specifications can be added with additions such as /analyze:log and /analyze:ruleset
Example: cd /Program Files (x86)/Microsoft Visual Studio/2019/Community/VC/Auxiliary/Build
vcvarsall x86
cl /analyze:logC:/Users/kevin/LogFile.txt /analyze:rulesetC:/Users/kevin/RuleSet1.ruleset C:Users/kevin/Prg.cpp
This example compiles and analyzes Prg.cpp using the ruleset RuleSet1.ruleset and logs the results in LogFile.txt
Using Code Analysis Warnings
Code Analysis produces warnings that look like "C:\Users\kevin\source\repos\CppDemo\CodeDefects\Bug.cpp(35): warning C6282: Incorrect operator: assignment of constant in Boolean context. Consider using '==' instead."
The short description and line number are often sufficient information to fix the problem.
The defect code can be used to find more information online.
Similar information is saved in XML format in the log file.


Further Reading
More detailed documentation can be found at
https://docs.microsoft.com/en-us/cpp/code-quality/code-analysis-for-c-cpp-overview?view=vs-2019

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

David Cook <dac0010@uah.edu>

Re: Geissler Honors Capstone

Kevin Preston <rkp0001@uah.edu> Wed, May 6, 2020 at 1:47 PM
To: Kevin Geissler <keg0015@uah.edu>
Cc: "Dr. Ranganath" <ranganat@cs.uah.edu>, William Wilkerson <wilkerw@uah.edu>, David Cook <dac0010@uah.edu>

Kevin,

I approve your submission of the Honors Capstone project.

Please follow the directions in the email from David Cook for final submission.

R. Kevin Preston
University of Alabama in Huntsville
Computer Science Department - Lecturer

On Wed, May 6, 2020 at 12:55 PM Kevin Geissler <keg0015@uah.edu> wrote:

> Mr. Preston,
>
> Thank you for your comments. I have made the changes you recommended, and attached the new version of my project to this email.
>
> Please let me know if this is acceptable, or if I need to make any other changes.
>
> Best,
>
> Kevin Geissler
>
> On Wed, May 6, 2020 at 8:48 AM Kevin Preston <rkp0001@uah.edu> wrote:
>
> > Kevin,
> >
> > See the comments in the attached Word document. Let me know if you have any questions. We may be able to get this submitted by the deadline.
> >
> > R. Kevin Preston
> > University of Alabama in Huntsville
> > Computer Science Department - Lecturer
> >
> > On Wed, May 6, 2020 at 5:57 AM Kevin Geissler <keg0015@uah.edu> wrote:
> >
> > > Mr. Preston,
> > >
> > > I have attached my Honors Capstone project to this email for your approval.
> > >
> > > If you decide to approve my project please message me, your Department Chair, the Honors Dean at wilkerw@uah.edu, and David Cook at dac0010@uah.edu.
> > >
> > > I would appreciate it if you can respond today, because today is the last day that I can submit the capstone without filing a form for an extension.
> > >
> > > Best,
> > >
> > > Kevin Geissler

Geissler_Kevin_SP20 Honors Capstone.pdf
480K