

Butler University Digital Commons @ Butler University

Scholarship and Professional Work - LAS

College of Liberal Arts & Sciences

2006

A metrics Tool for Multi-language .NET Software Applications

Panos K. Linos Butler University, linos@butler.edu

G. McGullogh Butler University

E. Maier Butler University

Follow this and additional works at: http://digitalcommons.butler.edu/facsch_papers Part of the <u>Software Engineering Commons</u>

Recommended Citation

Proceedings of the 18th Undergraduate Research Conference, Butler University, Indianapolis. (P. Linos with G. McGullogh and E. Maier) "A metrics Tool for Multi-language .NET Software Applications" April 21, 2006

This Presentation is brought to you for free and open access by the College of Liberal Arts & Sciences at Digital Commons @ Butler University. It has been accepted for inclusion in Scholarship and Professional Work - LAS by an authorized administrator of Digital Commons @ Butler University. For more information, please contact fgaede@butler.edu.

Measuring the Complexity of Multi-language Software Applications



Center of Applied Software Engineering Research (CeASER), Butler University Ezekiel Maier, Greg McCullough and Dr. Panos Linos

Microsoft Visual Studio .Net Software Development Environment

- Microsoft's visual programming environment for creating Web Services based on XML
- The product suite provides forms for building a user interface, features for integrating existing application data, and for debugging.
 - Comes with the .Net Framework
 - Includes several programming languages (e.g. C#, VB, etc.)

Microsoft Intermediate Language Framework (MSIL)

Our Hypothesis

The measurement of software metrics at the MSIL (Microsoft Intermediate Language) level can be as effective as measuring such metrics at the level of each individual language (e.g. VB, Java, C# etc.).

Significance

If we show that measuring metrics at the MSIL is possible (and it is effective) then we will avoid the need for building syntax parsers for each separate programming language used in the .NET environment (which is a costly and difficult task).

Our Research Plan

- Identify a set of software metrics for multilanguage applications.
- Develop a parsing tool for MSIL code.
- Develop a metrics visualization tool.
- Launch an empirical case study to analyze and evaluate our results.

Additional Tools Used

ILMerge – Intermediate Language Merger

- A utility that merges multiple .NET assemblies into a single assembly.
- Developed by Michael Barnett of Microsoft Research.
- Downloadable from the Internet
- Needed to condense all source into one file for scanning.
- ILDasm Intermediate Language Disassembler
 - A tool that allows the user to view MSIL code
 - Packaged with the .Net framework
 - Needed to create a MSIL text file for scanning.

Selected Software Metrics

Source Lines of Code (SLOC)

• This metric is defined as the count of program lines of code excluding comment or blank lines.

Class Total Source Size (CTSS)

 It refers to the total number of lines of code inside the body of each class.

Class Actual Source Size (CASS)

• It refers to the lines of code (excluding lines that contain comments or blanks) inside the body of each class.

Weighted Methods Per Class (WMC)

- It refers to the number of methods defined in a class.
- Weighted Data Per Class (WDC)
 - It refers to the number of instance variables defined in each class.

Selected Software Metrics Contd.

Number of Children (NOC)

It refers to the number of immediate sub-classes of a class.

Depth of Inheritance Tree (DIT)

 It refers to the maximum inheritance path from a given class to the root class.

Member Variables over Methods Ratio (MVR)

It refers to use of the class.

Weighted Properties Per Class (WPC)

- It refers to the type and number of property procedures of a class.
- Weighted Constructors Per Class (WCC)
- It refers to the type and number of constructors of a class.

Our Process

UML class diagram for the parser

Read		
string file		
int total_lines		
int MSIL_lines		
int High_level		
int High_cb_level		
int Class_line		
int Class_cb_line		
int Namespace_line		
int Namespace_cb_line		106-11-
int start_Method_line		vvrite
int Method_cb_line		
int Total_Methods		total line writer (int lines. Streem&kriter cyalyoid
int Brackets		Line Write/etring line, pum StreamWriter sw).void
string Method_name		Motrie Write/etring motrie Stream/Ariter sw).void
string File_name		Weinc_vviite(string metric, StreamWriter sw).void
string Project_name	⊳.	
Write wrt		
+reading(ProgressBar ProgressBar1):void		
+class_finder(string MSIL_line, StreamWriter sw, Write wrt):bool		
+end_class(string MSIL_line, StreamWriter sw, Write wrt, StreamReader sr):void		
+function_finder(string MSIL_line, StreamWriter sw, Write wrt, StreamReader sr):bool		
+end_method(string MSIL_line, StreamWriter sw, Write wrt, StreamReader sr, int end):void		
+data_finder(string MSIL_line, StreamWriter sw, Write wrt):void		
+namespace_finder(string MSIL_line, StreamWriter sw, Write wrt):bool		
+end_namespace(string MSIL_line, StreamWriter sw, Write wrt):void		
+variable_finder(string MSIL_line, StreamWriter sw, Write wrt, StreamReader sr):void		
+project_finder(string MSIL_line, StreamWriter sw, Write wrt):void		
+end_project(string MSIL_line, StreamWriter sw, Write wrt):void		
+sourcefile_finder(string MSIL_line, StreamWriter sw, Write wrt):void		
+end_sourcefile(string MSIL_line, StreamWriter sw, Write wrt):void		
+Analyze_top(StreamReader sr, StreamWriter sw, Write wrt, TextWriter tOut):void		
+Analyze_middle(StreamReader sr, StreamWriter sw, Write wrt, TextWriter tOut):void		
+Analyze_bottom(StreamReader sr, StreamWriter sw, Write wrt, TextWriter tOut, TextReader tIn):void		

MSIL Keywords

- .assembly name
- //Source File
- .namespace
- I/ end of namespace
- .class
- I/ end of class name
- .field
- .method
- .entrypoint
- .maxstack
- // Code size
- .local
- .ctor
- I/ end of method name

Parser Code Snippet

//Finds information about each class
public static bool class_finder(string MSIL_line, StreamWriter sw)

if ((MSIL_line.Trim()).StartsWith(".class"))

//FOUND CLASS NAME

int first_char_pos = MSIL_line.LastIndexOf(" ") + 1; string class_line = MSIL_lines.ToString(); string metric = "class"; string val = MSIL_line.Substring (first_char_pos,(MSIL_line.Length - first_char_pos)); Write.Line_Write(class_line, sw); Write.Metric_Write(metric, sw); Write.Value_Write(val, sw); return true;

return false;

}

{

{

Example of Parser Output

- class Example_Code
- {
- public static void Main()
- int x = 7;

- **3**
- class
- Example_Code
- **4**
- function(1)
- *Main[29]
- **5**
- variable[0]
- x(int32)

UML Class Diagram for the GUI

Example: TheBank Windows Application

- Describe how many classes it has etc.
- Show an example of the MSIL code
- Show an example of the produced output
- Show some charts from Greg's part here to show all the metrics collected

Conclusions

Describe what we have accomplished so far

Live demo