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

Measuring the Complexity of Multi-language Software Applications

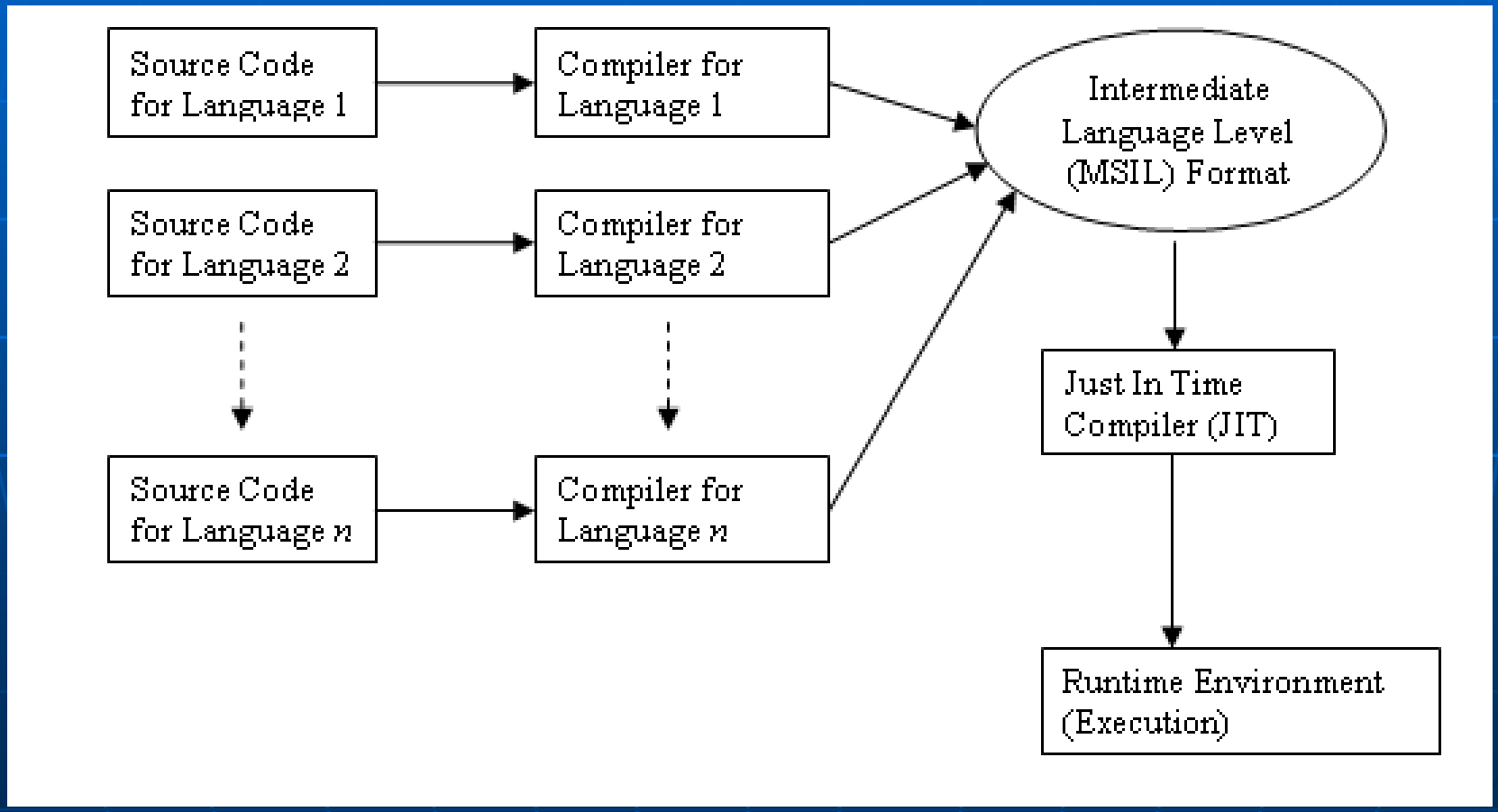
Center of Applied Software Engineering Research (CeASER), Butler University

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 multi-language 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.

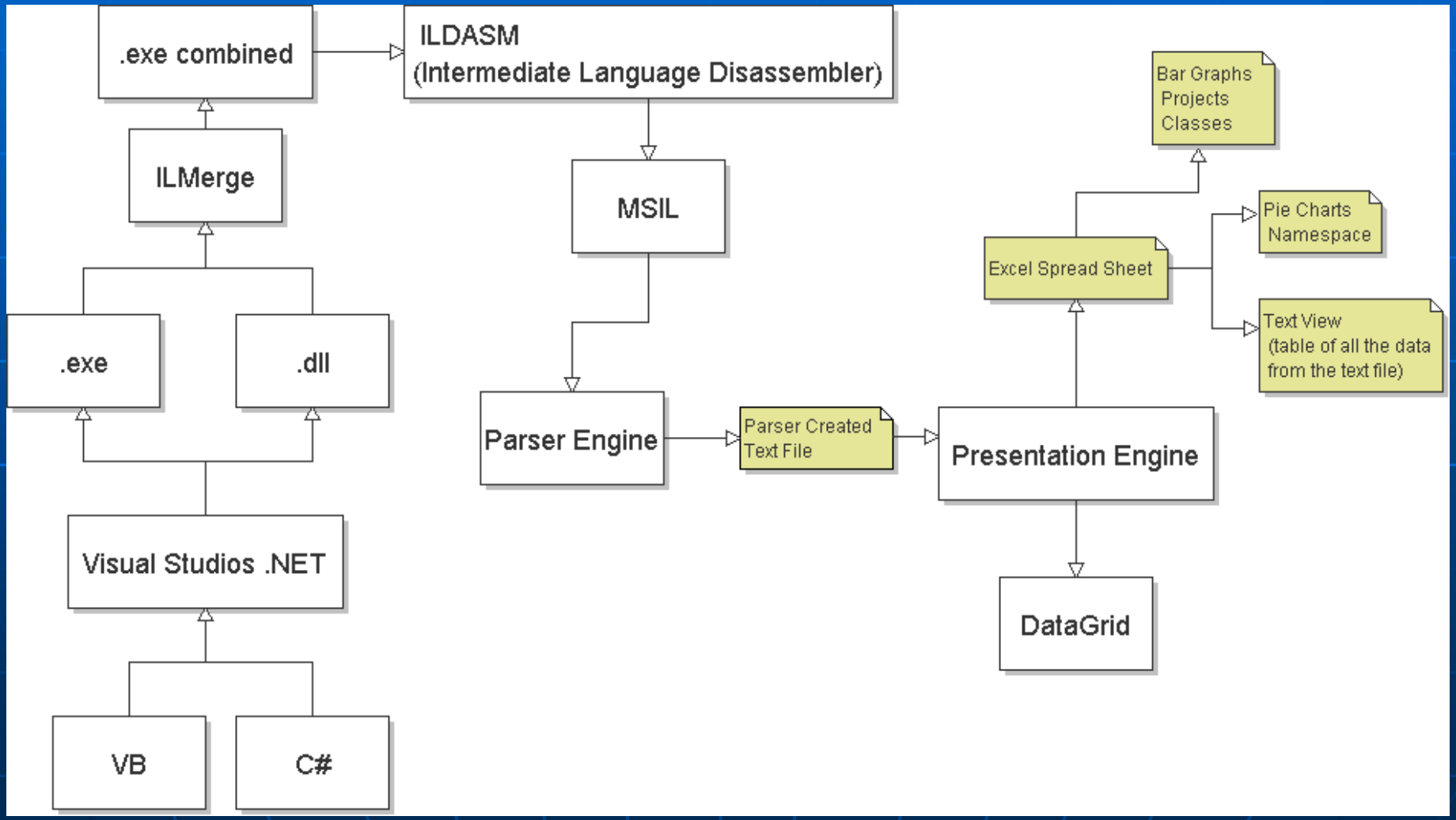
Selected Software Metrics

- **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.

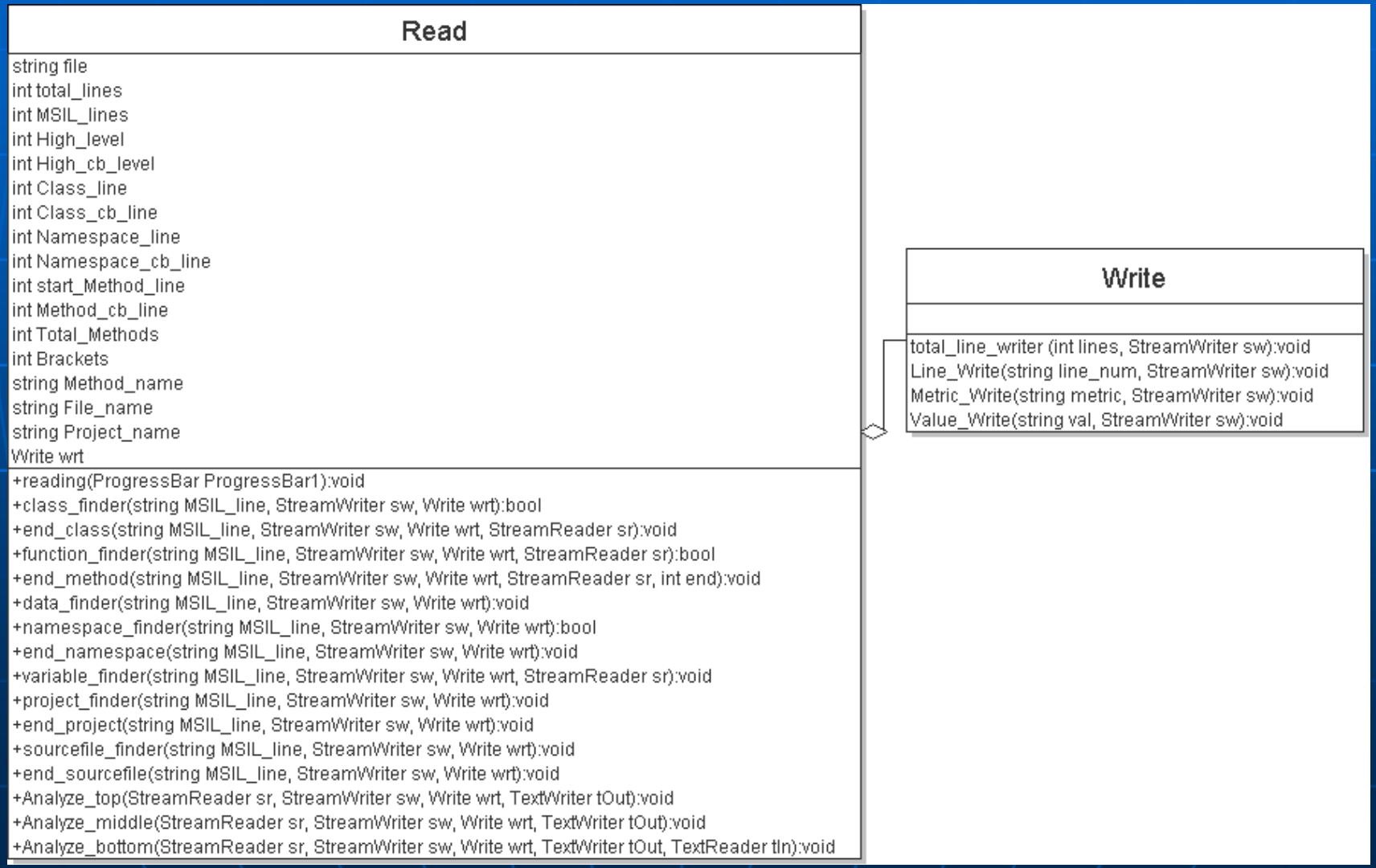
Selected Software Metrics Contd.

- **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


.exe combined
ILDASM
(Intermediate Language Disassembler)
ILMerge
MSIL
.exe
.dll
Parser Engine
Parser Created
Text File
Presentation Engine
Bar Graphs
Projects
Classes
Pie Charts
Namespace
Excel Spread Sheet
Text View
(table of all the data
from the text file)
Visual Studios .NET
DataGrid
VB
C#

UML class diagram for the parser



MSIL Keywords

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

Parser Code Snippet

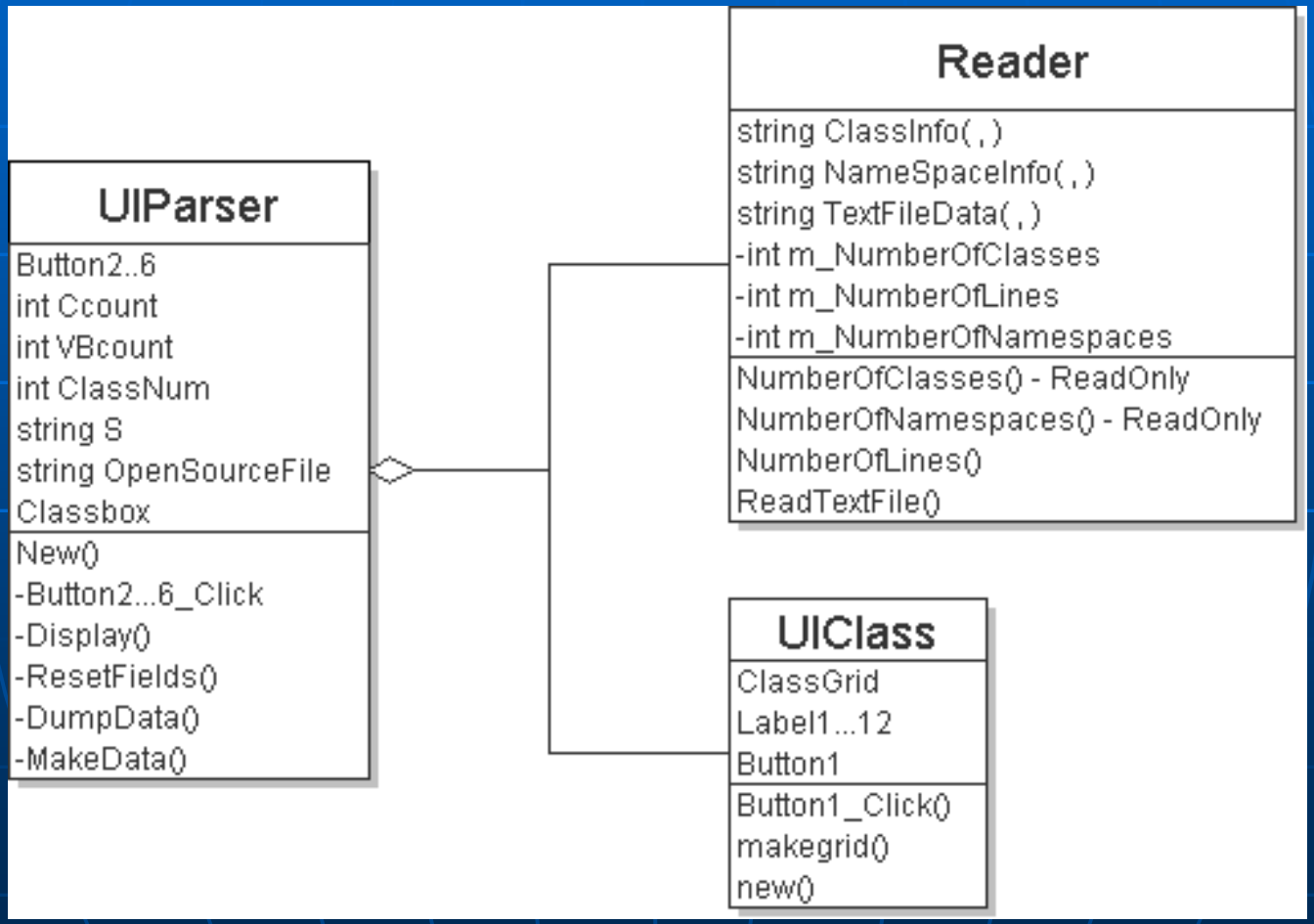
```
//Finds information about each class
public static bool class_finder(string MSIL_line, StreamWriter sw)
{
    if ((MSIL_line.Trim()).StartsWith(".class"))
    {
        //FOUND CLASS NAME
        int first_char_pos  = MSIL_line.LastIndexOf(" ") + 1;
        string class_line = MSIL_lines.ToString();
        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

Future steps

Tool Demo

- Live demo