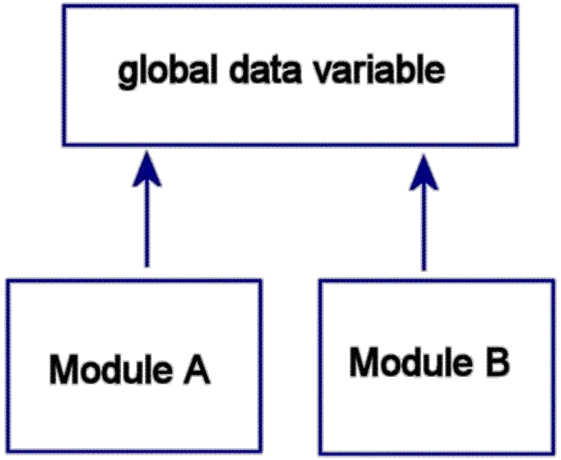
scrML Tool Implementation on a C# Environment

by Eduardo Jaime

Next Steps

Develop a tool using srcML.NET that:

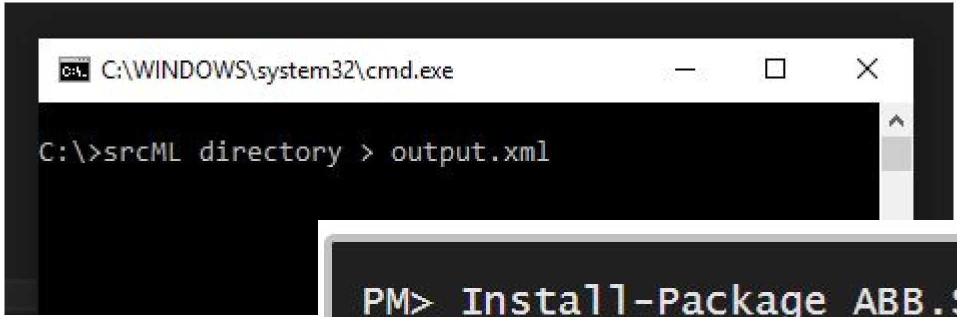
- Converts the selected code base to srcML format.
- Queries the selected srcML File.
- Detects Common Coupling.



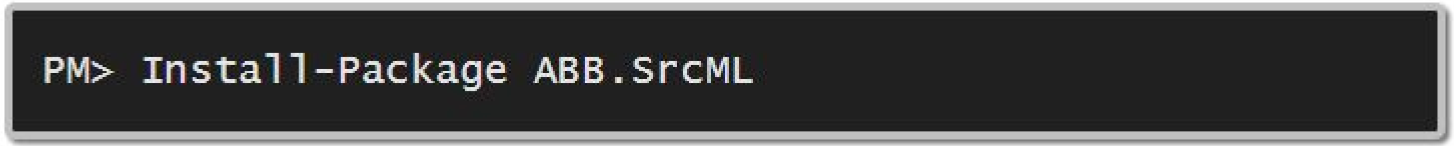
Approach Update

Combined command line tool to convert the code base to XML and a custom made tool made in C# to query the file using XPathDocument.

- Easy set up.
- XPathDocument offers a straightforward implementation.


C:\WINDOWS\system32\cmd.exe

```
C:\>srcML directory > output.xml
```



```
PM> Install-Package ABB.SrcML
```

srcMLFile vs XPathDocument

- srcML.Net offers a class named **srcMLFile** but the native C# **XPathDocument** class was chosen instead.
- srcMLFile can load a document in srcML Format and offers some functions to query the document using LinQ. It can also be used to modify the codebase.
- XPathDocument is a C# class that loads a document in XML format and returns nodes when queried against an XPath Expression.
- This approach was chosen because its simple implementation.

XML File Structure

```
<unit xmlns="http://www.srcML.org/srcML/src" revision="0.9.5">
    <unit language=”c#” filename=”SoruceFile1.cs”>
        (Comments, variables and methods)
    </unit>
    <unit language=”c#” filename=”SoruceFile2.cs”>
        (Comments, variables and methods)
    </unit>
</unit>
```

UNIT Structure

```
<unit xmlns:cpp="http://www.srcML.org/srcML/cpp" revision="0.9.5"
        language="C#"
        filename="ConstructorHandling.cs"
        hash="336e2f0959f373b22eab5cdef109f7a2d72c196a">
    <decl_stmt>(Declaration statement)</decl_stmt>
    <function>(Method definition)</function>
</unit>
```

Variable Declaration Structure

XML node that represents the line: **int myInt;**

```
<decl_stmt>
    <decl>
        <type> <name>int</name> </type>
        <name>myInt</name>
    </decl>;
</decl_stmt>
```

Function Structure

XML node that represents the line: **static void doWork() { more code... }**

<**function**>

<**specifier**>static</specifier>

<**type**> <name>void</name> </type>

<**name**>doWork</name>

<parameter_list>()</parameter_list>

<**block**> { (Comments, variables declarations and method calls) } </block>

</function>

XPath Expressions for Method Granularity

Objective	**XPath Expression**
Get all the files in the code base.	//src:unit/src:unit
Get filename for each unit.	//src:unit/@filename
Get all the fields.	//src:class/src:block/src:decl_stmt/src:decl/src:name
Get all the methods that use those fields.	//src:function[descendant::src:name='FIELDNAME']/src:name

Results

The tool CouplingDetector was developed to offer the following functionality:

1. Convert a code folder into a srcML Format file by implementing srcML.Net from ABB.
2. Analyse a srcML File to detect common **coupling between the methods of a class**.
3. Query a selected document against an XPath Expression and shows the results.

Analysis Algorithm (Method granularity)

The following steps were taken to detect Common Coupling

1. Select all the Unit Nodes.
2. Iterate through the results.
3. For each Unit select all the declared Fields.
4. Iterate through the results.
5. For each Field, select the methods that use them.

The following XPATH query was used to detect the methods that used the Fields.

```
//src:function[descendant::src:name='FIELDNAME']/src:name
```

Conclusion

You can actually use any programming language that you like. Java, C#, VB. NET, or whatever language with XPath support to design your code analysis tools.

LinQ can also be used, but XPath Expressions were more than enough for this project.

srcML focuses on code analysis, a huge difference between this and JAVA ASDT.

Parse, add stop words, more set up VS just running a command and being ready to query.

Thank you!