Survey Research Operations
Survey Research Center
Institute for Social Research

Leveraging the Capabilities of the Blaise Editor Add-In Tool to Improve the Readability of Datamodel Source Code

Chris Oz

24 April 2012

London, United Kingdom

Overview

- Read Blaise Source
- Regular Expression Programming
- XML Transformations With XSLT
- Apply Transformed Code Back to Blaise
- Before and After
- Lessons Learned & Conclusion

Reading Blaise Source

- No exposed methods in Blaise to access the editor window
  - Implementing Windows (.NET) files to simulate a copy action (Ctrl + C)
- Alternative solutions
  - Using C++ programming to access the running instance of Blaise to hook into the editor (complicated, more development time)

Reading Blaise Source

- **Access Windows clipboard contents to store in local variable for modification**

```
private string  sTextToClipboard()
{
    SendKeys.SendWait("^{HOME}");
    SendKeys.SendWait("^+{END}");
    SendKeys.SendWait("^c");
    object oInfo = Clipboard.GetData("Text");
    return oInfo.ToString();
}
```

class System.Windows.Forms.Clipboard
Provides methods to place data on and retrieve data from the system Clipboard. This class cannot be inherited.

Regular Expression Programming

- Need to isolate specific pieces of Blaise source to transform into a standardized format
- Regular Expressions, also called Regex, are a powerful tool for identifying patterns in a line of text

Regular Expression Programming

```
DATAMODEL TA2011          "Transition to Adulthood Study 2011"

ATTRIBUTES = DK, RF

PRIMARY
    SampleID



 SECONDARY
 Complete

    USES     State        'State'

USES
    Country 'Country'
USES

    Degree   'Degree'
USES
    StateProg       'StateProg'

LOCALS
    I            : INTEGER
PROCEDURE CamRec
    PARAMETERS
```

Regular Expression	Captured text string
(DATAMODEL)\s*(\w*)\s*\"(.*)\"	DATAMODEL TA2011 "Transition to Adulthood Study 2011"
(ATTRIBUTES)\s*=\s*(\w*,*\s\w*)\s\s(\s)*	ATTRIBUTES = DK, RF

Regular Expression Programming

- Using Regex to format Blaise source as XML document

```
private string sFormatWithRegEx(string sInput)
{
    string sTemp = sInput;

    sTemp = Regex.Replace(sTemp, "(DATAMODEL)\\s*(\\w*)\\s*\\\"(.*)\\\"", "<DataModel ID=\"$2\" Desc=\"$3\" />", RegexOptions.IgnoreCase);
    sTemp = Regex.Replace(sTemp, "(ATTRIBUTES)\\s*=\\s*(\\w*,*\\s\\w*)\\s\\s(\\s)*", "<Attributes Attribute=\"$2\" />", RegexOptions.IgnoreCase);
```

- Text blocks identified by Regex formatted into XML

```
<root>
  <DataModel ID="TA2011" Desc="Transition to Adulthood Study 2011" />
  <Attributes Attribute="DK, RF" />
  <Primary Key="SampleID" />
  <Secondary Key="Complete" />
  <Uses Name="State" Value="'State'" />
  <Uses Name="Country" Value="'Country'" />
  <Uses Name="Degree" Value="'Degree'" />
  <Uses Name="StateProg" Value="'StateProg'" />
```

Regular Expression Programming

- **Expanded XML sample using Regex formatting**

```
<root>
  <DataModel ID="TA2011" Desc="Transition to Adulthood Study 2011" />
  <Attributes Attribute="DK, RF" />
  <Primary Key="SampleID" />
  <Secondary Key="Complete" />
  <Uses Name="State" Value="'State'" />
  <Uses Name="Country" Value="'Country'" />
  <Uses Name="Degree" Value="'Degree'" />
  <Uses Name="StateProg" Value="'StateProg'" />
  <Include FileName="TA2011_type.inc" />
  <Include FileName="StateCountry.prc" />
  <Include FileName="make_date_string.prc" />
  <Include FileName="TA2011_Preload.inc" />
  <Include FileName="TA2011_SecA.inc" />
  <Include FileName="TA2011_SecB.inc" />
  <Include FileName="TA2011_SecC.inc" />
  <Include FileName="TA2011_SecD.inc" />
  <Include FileName="TA2011_SecE.inc" />
  <Include FileName="TA2011_SecF.inc" />
  <Include FileName="TA2011_SecW.inc" />
  <Include FileName="TA2011_SecG.inc" />
  <Include FileName="TA2011_SecH.inc" />
  <Include FileName="TA2011_SecK.inc" />
  <Include FileName="TA2011_SecL.inc" />
  <Include FileName="TA2011_SecM.inc" />
  <Include FileName="TA2011_SecN.inc" />
</root>
```

XML Transforming With XSLT

- XML needs to be transformed back into programming language recognized by Blaise
- XSLT chosen for its advantages in XML transformation
- Allows each organization to create their own formatting rules/guidelines

XML Transforming With XSLT

- **Example of XSLT Programming**

```
<root>
  <DataModel ID="TA2011" Desc="Transition to Adulthood Study 2011" />
  <Attributes Attribute="DK, RF" />
  <Primary Key="SampleID" />
  <Secondary Key="Complete" />
  <Uses Name="State" Value="'State'" />
  <Uses Name="Country" Value="'Country'" />
  <Uses Name="Degree" Value="'Degree'" />
  <Uses Name="StateProg" Value="'StateProg'" />
  <Include FileName="TA2011_type.inc" />
  <Include FileName="StateCountry.prc" />
  <Include FileName="make_date_string.prc" />
  <Include FileName="TA2011_Preload.inc" />
  <Include FileName="TA2011_SecA.inc" />
  <Include FileName="TA2011_SecB.inc" />
  <Include FileName="TA2011_SecC.inc" />
  <Include FileName="TA2011_SecD.inc" />
  <Include FileName="TA2011_SecE.inc" />
  <Include FileName="TA2011_SecF.inc" />
  <Include FileName="TA2011_SecW.inc" />
  <Include FileName="TA2011_SecG.inc" />
  <Include FileName="TA2011_SecH.inc" />
  <Include FileName="TA2011_SecK.inc" />
  <Include FileName="TA2011_SecL.inc" />
  <Include FileName="TA2011_SecM.inc" />
  <Include FileName="TA2011_SecN.inc" />
</root>
```

```
<xsl:template match="root">
  <xsl:apply-templates select="@* | node()" />
</xsl:template>

<xsl:template match="DataModel">
  <xsl:value-of select="'\r\n'" />
  <xsl:value-of select="'DATAMODEL '"/>
  <xsl:value-of select="@ID" />
  <xsl:value-of select="' '"/>
  <xsl:value-of select="concat('&amp;quot;', @Desc, '&amp;quot;')"/>
  <xsl:value-of select="'\r\n'" />
</xsl:template>

<xsl:template match="Attributes">
  <xsl:value-of select="'\r\n'" />
  <xsl:value-of select="'ATTRIBUTES = '" />
  <xsl:value-of select="@Attribute"/>
  <xsl:value-of select="'\r\n'" />
</xsl:template>

<xsl:template match="Primary">
  <xsl:value-of select="'\r\n'" />
  <xsl:value-of select="'PRIMARY '"/>
  <xsl:value-of select="@Key"/>
  <xsl:value-of select="'\r\n'" />
</xsl:template>
```

Apply Transformed Code to Blaise

- Using Microsoft's .NET libraries to simulate the keystrokes for paste (Ctrl + V)
  - Same approach as when retrieving text from Blaise in step 1

Before and After

Before

```
DATAMODEL TA2011          "Transition to Adulthood Study 2011"

ATTRIBUTES = DK, RF

PRIMARY
    SampleID



 SECONDARY
 Complete

    USES     State        'State'

USES
    Country 'Country'
USES

    Degree   'Degree'
USES StateProg 'StateProg'

LOCALS
    I             : INTEGER
```

After

```
DATAMODEL TA2011 "Transition to Adulthood Study 2011"

ATTRIBUTES = DK, RF

PRIMARY SampleID

SECONDARY Complete

USES
    State     'State'
USES
    Country     'Country'
USES
    Degree     'Degree'
USES
    StateProg     'StateProg'
```

Lessons Learned & Conclusion

- Currently no Blaise libraries allowing programmers access to tabs (open files) inside of Blaise Control Centre
  - Necessitated keypress simulation to retrieve text from Blaise

Lessons Learned & Conclusion

- Simulating keyboard input prone to user intervention and subsequent application error
  - Future goal: lock keyboard input while keystrokes are being simulated

Lessons Learned & Conclusion

- Text reformatting not 100% accurate due to nature of text editor inside Blaise
  - Tabs are relative to text on the screen, not row/column coordinates
  - Solution: Use string length calculations inside XSLT to add/remove spaces from tab spacing as needed

Conclusion

- Questions
  - Contact Info: ozc@isr.umich.edu