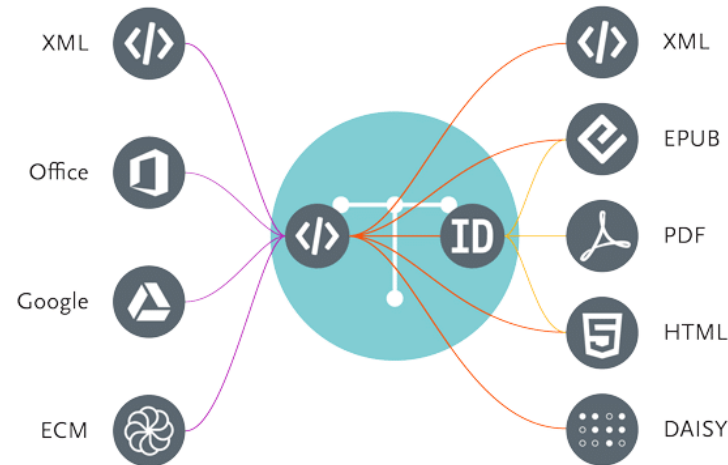


Getting Useful XML out of Microsoft Excel.

Gayanthika Udeshani
and the XSLT team.

Purpose



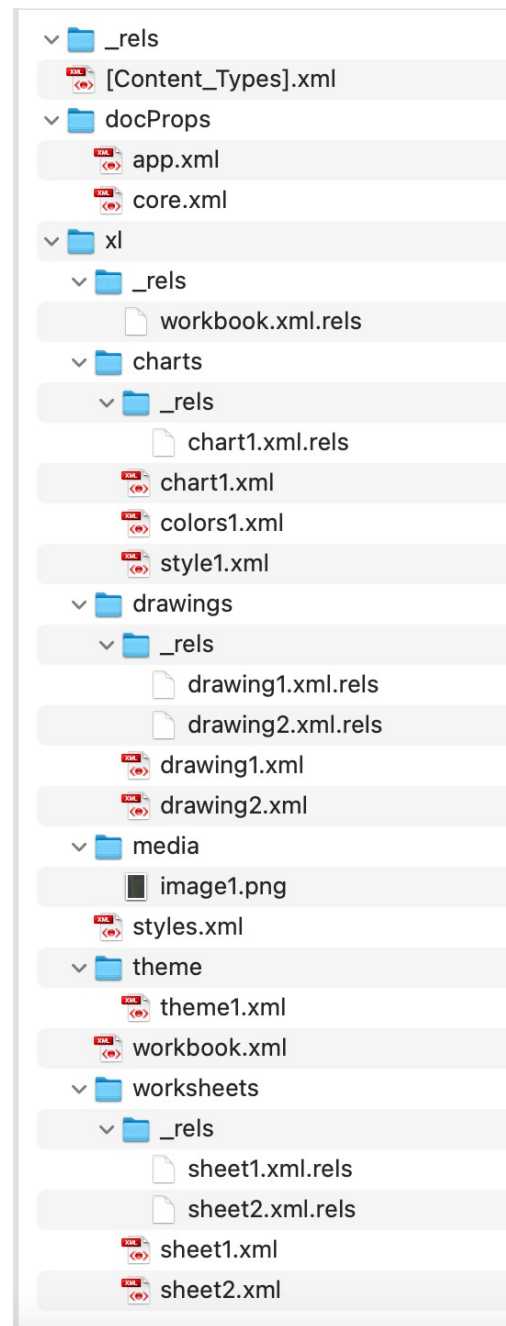
A solution that **transforms** a **Microsoft Excel Open XML Spreadsheet (XLSX)** file into a shallow-structured XML file used at Typefi, called **Content XML (CXML)**, using XSLT and XProc.

The main research areas

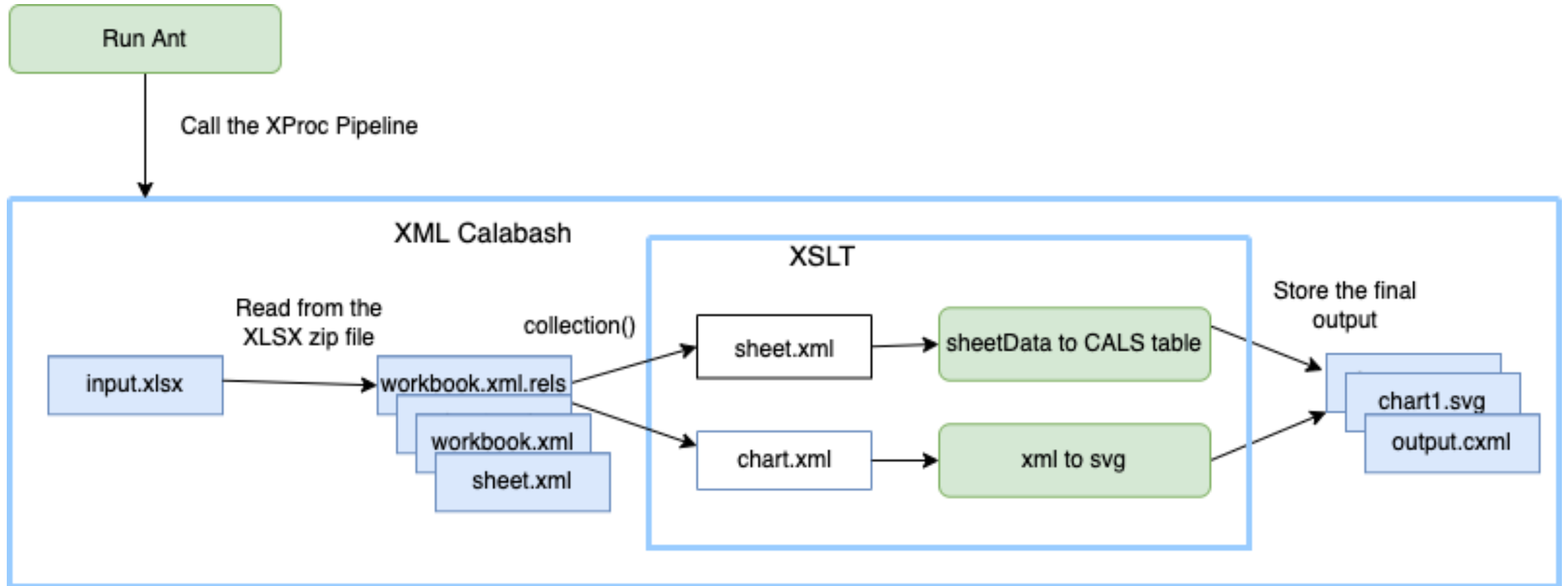
1. The XProc pipeline to read the Excel file content.
2. Map the XLSX table information to the CALS table elements and attributes.
3. Convert the chart.xml into an SVG file.



File structure



High-level architecture



How it works?

- Input parameters.
- Cell range and sheet name:

```
<dimension ref="A1:I836"/>
```

```
<xsl:variable name="cell.range" select="$worksheet/e:dimension/@ref" as="xs:string?"/>
```

```
<sheetView tabSelected="1" showRuler="0" workbookViewId="0">
```

- XSLT 3.0 features, for example: xsl:iterate to calculate the total column width.
- Reusable XSLT functions.

XProc pipeline version 1.0



```
<pxp:unzip>
  <p:with-option name="href" select="$input.file.path"/>
  <p:with-option name="file" select="$filename"/>
  <p:with-option name="charset" select="UTF-8"/>
  <p:with-option name="content-type" select="xml"/>
</pxp:unzip>

<p:store cx:decode="true">
  <p:with-option name="href" select="concat($output-dir, '/', encode-for-uri($filename))"/>
</p:store>
```

collection() could not detect the extracted content as a valid XML file, instead returning an xs:base64Binary object, while the expected output was a document-node().

This XProc pipeline follows these steps:

- Get the input parameters from the ANT script (input file path and the XSLT stylesheet).
- Unzip the XLSX file.
- Iterate over the ZIP file and add the file path as an attribute.
 - This will be used from the XSLT side to read the file content.
- Call the XSLT program and pass the unzip file information as a parameter.
- Write the output files to the disk.

```
<xsl:variable name="worksheet" select="collection()[*/base-uri() eq 'xl/worksheets/sheet1.xml']" as="document-node()"/>
```

Input file

The diagram shows an Excel spreadsheet with the following features highlighted:

- Bold:** Points to cell A2 containing the number 2.
- Italic:** Points to cell B1 containing the letter y.
- Underline:** Points to cell C1 containing the letter z.
- Background color:** Points to cell C3 containing the number 20, which has a yellow background.
- Foreground color:** Points to cell A3 containing the number 4, which is red.
- Border styles:** Points to cell A4 containing the number 6, which has a black border.
- Number formats:** Points to cell C5 containing the number 72.00, which is formatted as a number with two decimal places.
- Merged cells:** Points to cell B7 containing the text *all 3 merged*, which is merged across columns A, B, and C.

	A	B	C
1	x	<i>y</i>	<u>z</u>
2	2	3	<u>6</u>
3	4	5	20
4	6	7	42
5	8	9	72.00
6	10	12	120
7	<i>all 3 merged</i>		
8	<i>1st two merged 1 @ end</i>		
9	1 @ start	last two merged	

More features:

- Hidden rows/columns
- Date time formatting
- Conditional formatting
- Orientation

Style overrides and shared strings

```
<c r="E1" s="9" t="s">  
  <v>254</v>  
</c>
```

sharedString.xml

```
<si> /sst/si[255]  
  <t>21 May - 27 May 2022</t>  
</si>
```

```
<xf numFmtId="0" fontId="2" fillId="0" borderId="0" xfld="0" applyFont="1" applyAlignment="1">  
  <alignment horizontal="right"/>  
</xf>
```

/styleSheet/cellXfs/xf[10]

style.xml

```
<font>  
  <sz val="12"/>  
  <color rgb="FFFF0000"/>  
  <name val="Calibri"/>  
  <family val="2"/>  
  <scheme val="minor"/>  
</font>
```

/styleSheet/fonts/font[3]

Number format

	<u>general</u>	<u>decimal (2 digits)</u>	<u>currency</u>	<u>24 hour</u>	<u>mm/dd/yy</u>
number format	3.141592654	3.14	\$3.14	16:15:00	01-03-00

```
<row r="8">
  <c r="B8" s="43">
    <v>3.1415926535900001</v>
  </c>
  <c r="C8" s="7">
    <v>3.1415926535900001</v>
  </c>
  <c r="D8" s="8">
    <v>3.1415926535900001</v>
  </c>
  <c r="E8" s="12">
    <v>0.67708333333333337</v>
  </c>
  <c r="F8" s="10">
    <v>3.1415926535900001</v>
  </c>
</row>
```

```
<xf numFmtId="0" fontId="0" fillId="0" borderId="0" xfid="0" applyNumberFormat="1"/> /styleSheet/cellXfs/xf[44]
<xf numFmtId="2" fontId="0" fillId="0" borderId="0" xfid="0" applyNumberFormat="1"/> /styleSheet/cellXfs/xf[8]
<xf numFmtId="164" fontId="0" fillId="0" borderId="0" xfid="0" applyNumberFormat="1"/> /styleSheet/cellXfs/xf[9]
<xf numFmtId="166" fontId="0" fillId="0" borderId="0" xfid="0" applyNumberFormat="1"/> /styleSheet/cellXfs/xf[13]
<xf numFmtId="165" fontId="0" fillId="0" borderId="0" xfid="0" applyNumberFormat="1"/> /styleSheet/cellXfs/xf[11]
```

```
<numFmts count="6">
  <numFmt numFmtId="8" formatCode="",$"#;##0.00_);[Red]\(",$"#;##0.00\)" />
  <numFmt numFmtId="164" formatCode="",$"#;##0.00" />
  <numFmt numFmtId="165" formatCode="mm/dd/yy;@" />
  <numFmt numFmtId="166" formatCode="h:mm:ss;@" />
  <numFmt numFmtId="167" formatCode="[$-409]h:mm:ss\ AM/PM;@" />
</numFmts>
```

```
<xsl:map>
  <xsl:map-entry key="1" select="0" />
  <xsl:map-entry key="2" select="0.00" />
  <xsl:map-entry key="3" select="#,##0" />
  <xsl:map-entry key="4" select="#,##0.00" />
  <xsl:map-entry key="9" select="0%" />
  <xsl:map-entry key="10" select="0.00%" />
  <xsl:map-entry key="11" select="0.00E+00" />
  <xsl:map-entry key="12" select="## ?/?" />
  <xsl:map-entry key="13" select="## ??/??" />
  <xsl:map-entry key="14" select="mm-dd-yy" />
```

Date time formats

```
<xsl:value-of select="format-date(tps:get-date-from-the-serial-no(.), '[M01]/[D01]/[Y01]')"/>
<xsl:value-of select="tps:get-time-from-the-value(xs:double(.), $num-formats[1])"/>
```

```
<xsl:function name="tps:get-date-from-the-serial-no" as="xs:date">
  <xsl:param name="serial-no" as="xs:decimal"/>

  <xsl:variable name="base-date" select="xs:date('1899-12-31')" as="xs:date"/>

  <xsl:sequence select="$base-date + floor($serial-no) * xs:dayTimeDuration('P1D')"/>
</xsl:function>

<xsl:function name="tps:get-time-from-the-value" as="xs:string">
  <xsl:param name="value" as="xs:decimal"/>
  <xsl:param name="format" as="xs:string"/>

  <xsl:variable name="time" select="24*$value" as="xs:double"/>

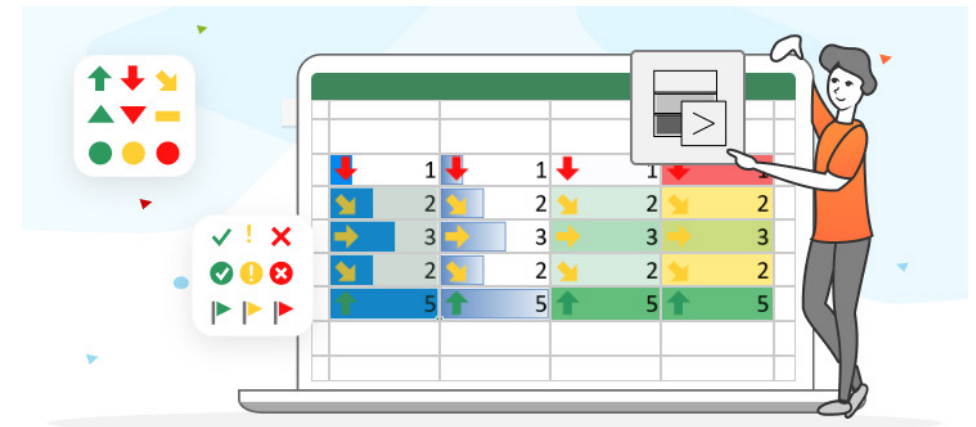
  <xsl:sequence select="floor($time) || ':' || round(($time mod 1)*60) || (if($format eq 'h:mm:ss') then '00' else '')"/>
</xsl:function>
```

```
3.1415926535900001 -> base-date + 3 days -> 1900-1-3
24 * 0.14159 -> 3.3982224 -> 3:24 AM
(0.3982224*60 )
```

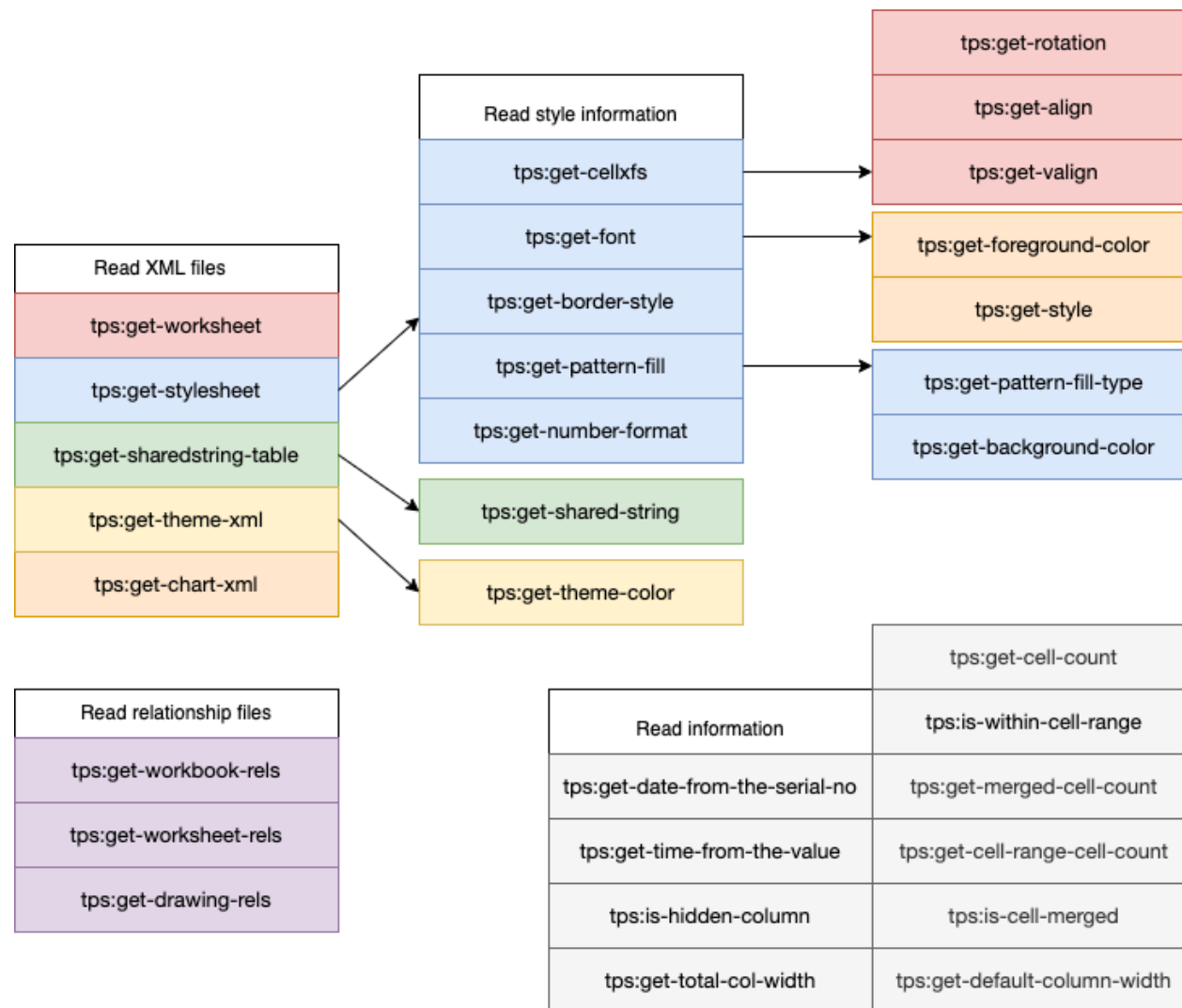
Custom number formats

("\$" * #,##0.00_);_(" \$" * \(#,##0.00\);_(" \$" * "-"??_);_(@_)

```
<xsl:when test="$section-count eq 4 and (. > 0)">
  <xsl:value-of select="format-number(., $num-formats[1])"/>
</xsl:when>
<xsl:when test="$section-count eq 4 and (. &lt; 0)">
  <xsl:value-of select="format-number(., $num-formats[2])"/>
</xsl:when>
<xsl:when test="$section-count eq 4 and (. = 0)">
  <xsl:value-of select="format-number(., $num-formats[3])"/>
</xsl:when>
<xsl:when test="$section-count eq 2 and (. &gt;= 0)">
  <xsl:value-of select="format-number(., $num-formats[1])"/>
</xsl:when>
<xsl:when test="$section-count eq 2 and (. &lt; 0)">
  <xsl:value-of select="format-number(., $num-formats[2])"/>
</xsl:when>
```



XLSX table to CALS



XLSX table to CALS

- It is the most widely adopted industry standard of XML representation for tables.

XLSX Element	CALS table element
sheetData	<pre><table> <tgroup cols="number of columns"> <colspec/> <thead> </thead> <tbody> </tbody> <tfoot> </tfoot> </tgroup> </table></pre>
col	<pre><colspec></pre>
row	<pre><row></pre>
c	<pre><entry> @namestart specifies the starting cell number for horizontally merging cells. @nameend specifies the ending cell number for horizontally merging cells. @morerows specifies the number of additional rows for vertically merging cells. @valign specifies the vertical alignment for the cell contents. @align specifies the horizontal alignment for the cell contents.</pre>

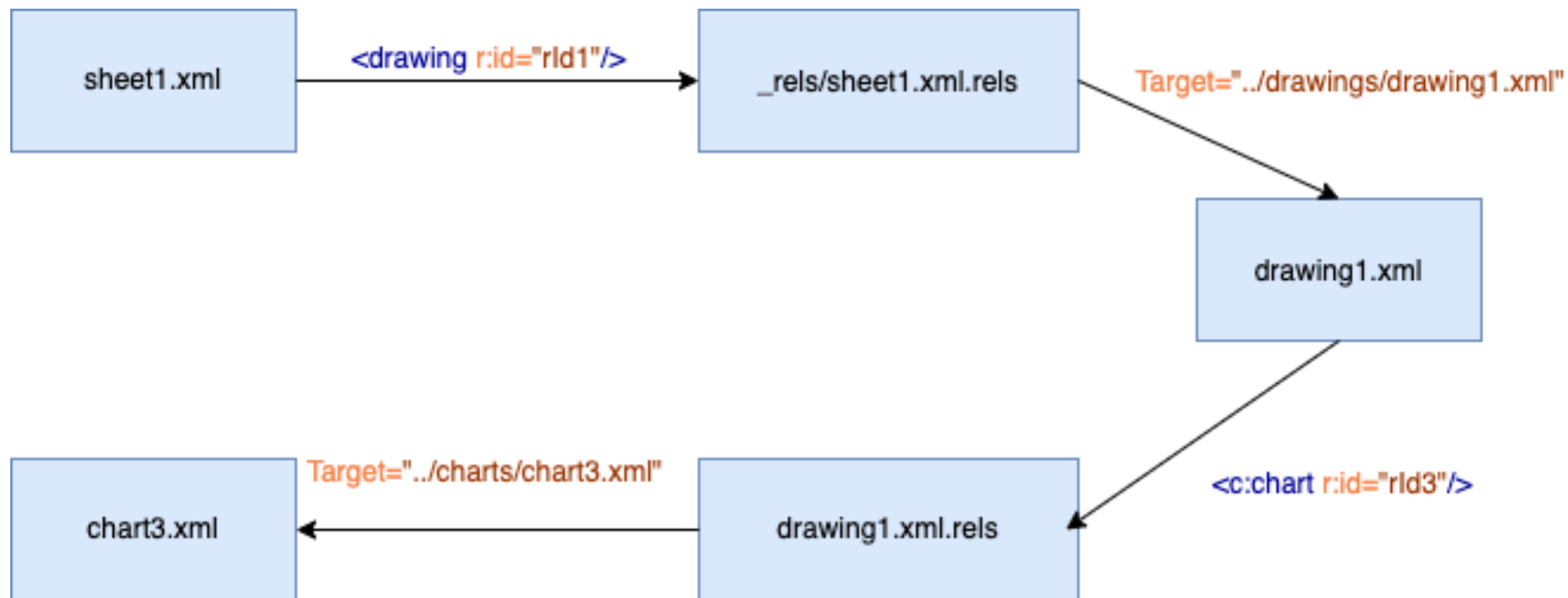
XLSX table to CALS

- Processing instruction which contains style information.

```
<entry namest="2" nameend="3" align="center" valign="bottom"><?style bold?>Heading 1
  <?cell-format background-color:FFFFC7CE; foreground-color:FF9C0006; ?>
</entry>
```

```
<xsl:template match="processing-instruction('cell-format')">
  <tps:c type="cell-format">
    <xsl:value-of select="."/>
  </tps:c>
</xsl:template>
```

XLSX with charts and graphs



Charts to SVG

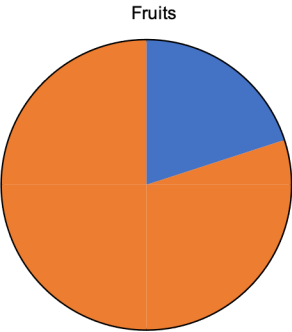
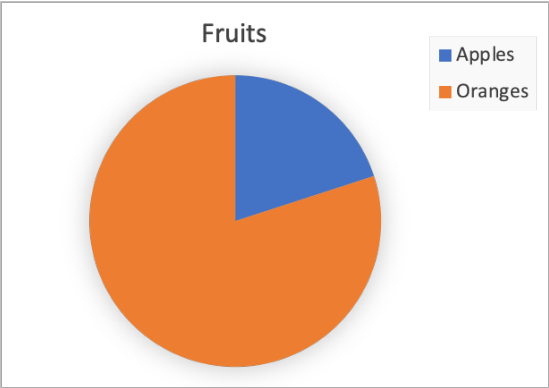
Element	Description
<defs>	Store graphical elements
<path> M x, y A L x,y	Move to Elliptic arc Line to
<rotate>	Rotate
<clippath>	Specify a boundary to be drawn
<circle>	Circles
<use>	Duplicates the elements
<rect>	Rectangles
<text>	Text elements
<line>	Lines

```
<xsl:choose>
  <xsl:when test="$is-line-chart">
    <path d="M {50 + (position() - 1)*100},{chart-height - c:v}
          L {50 + (position())*100}, {chart-height - following-sibling::c:pt/c:v} Z" stroke="{color}"/>
  </xsl:when>
  <xsl:otherwise>
    <rect x="{40 + (position() - 1)*100}" y="{chart-height - c:v}" width="50" height="{c:v}"
          style="fill:{color};stroke:rgb(0,0,0);stroke-width:0"/>
  </xsl:otherwise>
</xsl:choose>
```

Charts to SVG

Excel

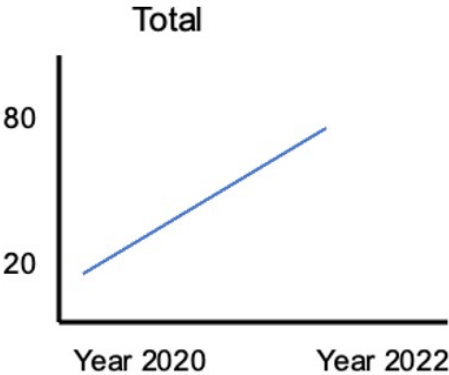
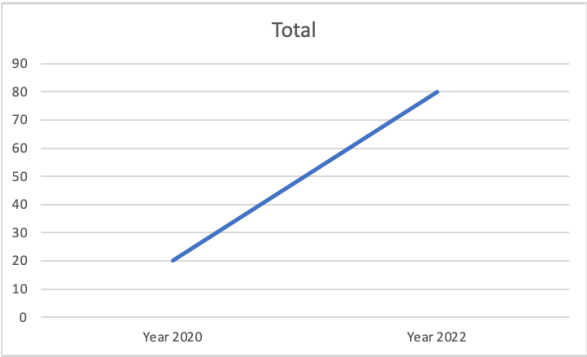
SVG



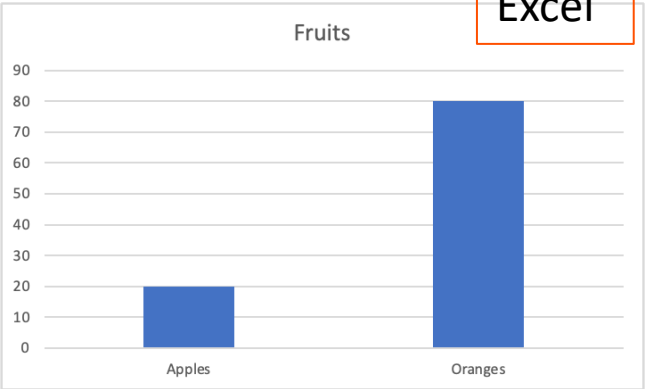
■ Apples
■ Oranges

Excel

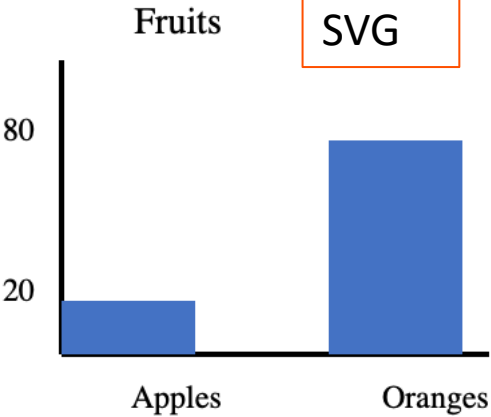
SVG



Excel



SVG



Discussion

- The solution uses Xproc, XSLT, Xpath, and ANT scripts and provides a CXML file as the final output.
- CALS tables are derived from sheet.xml and other related files.
- The chart.xml → SVG file—but using a java library would be more efficient for complex graphs.
- It considers the common input XLSX file structure in corresponding order, but an arbitrary file structure will be considered in future versions.
- `doc('jar:file:/Users/gayanthika/Documents/proservice/Research-work/xslx2cxml/data/in/pie.xlsx!/xl/workbook.xml')` is available with XSLT 3.0, but we chose X\Proc.

Why XSLT?

- Limitations of Apache POI—for huge files using the default POI classes you will likely need a very large amount of memory.
- The previous approach, with the Apache POI, read the XLSX file using the streaming fashion, but again with some limitations on what information you can read out of the file.
- The existing application supports Word documents that are processed using XSLT features; therefore, our goal is to use a similar approach in the Excel converters.
- The Excel file structure is complex, but we still prefer XSLT.
- Future work : Calculation chain, formulas, pivot tables etc.

<https://poi.apache.org/components/spreadsheet/limitations.html>



References

1. Microsoft Corporation. February 15, 2022. Office Implementation Information for ISO/IEC 29500 Standards Support. <https://interoperability.blob.core.windows.net/files/MS-OI29500/%5bMS-OI29500%5d.pdf>
2. Spreadsheet styles. <http://officeopenxml.com/SSstyles.php>
3. Microsoft Office, API Reference. <https://docs.microsoft.com/en-us/dotnet/api/documentformat.openxml.spreadsheet.sheetformatproperties?view=openxml-2.8.1>
4. "Retrieving data from Excel XML using XSLT". <https://stackoverflow.com/questions/33422652/retrieving-data-from-excel-xml-using-xslt>
5. "How to turn an XML file into SVG using XSL?" <https://stackoverflow.com/questions/8056671/how-to-turn-an-xml-file-into-svg-using-xsl>
6. XSLT to generate SVG tutorial, 2016. http://edutechwiki.unige.ch/en/XSLT_to_generate_SVG_tutorial
7. Florent Georges, 2008, Simple SVG chart generation with XSLT. <http://fgeorges.blogspot.com/2008/04/simple-svg-chart-generation-with-xslt.html>
8. Max Froumentin, Vincent Hardy, W3C, Using XSLT and SVG together: a survey of case studies. <https://www.w3.org/People/maxf/papers/2002-07-SVGOpen/svgopen.html>
9. SVG attribute reference. <https://developer.mozilla.org/en-US/docs/Web/SVG/Attribute>
10. A pure XSLT/SVG pie chart. <http://www.sleepingdog.org.uk/svg/chart/pie/>
11. SvgCharts4XSL. <https://franklinefrancis.github.io/SvgCharts4Xsl/>
12. Adrian McMenamin, 2011, Using XSLT to manipulate an SVG file. <https://cartesianproduct.wordpress.com/2011/07/16/using-xslt-to-manipulate-an-svg-file/>
13. SVG: Scalable Vector Graphics. <https://developer.mozilla.org/en-US/docs/Web/SVG>
14. Ben Hauser, Content XML 3.0 and earlier. <https://help.typefi.com/hc/en-us/articles/228240428-Content-XML-3-0-and-earlier>
15. Laura Powers, 2021, "Local style overrides." https://help.typefi.com/hc/en-us/articles/360001491415#h_3be5e014-f255-49b8-8454-6c7681861b1a
16. Erik Bruchez, "Unzipping an Excel 2007 .xlsx file and extracting data in XPL/XSLT." <https://gist.github.com/ebruchez/1245692>
17. Dave Bruns, "Excel customer number formats." <https://exceljet.net/custom-number-formats>
18. Appendix A, Full XML schema. https://docs.microsoft.com/en-us/openspecs/office_standards/ms-xlsx/6624db33-496c-47f7-a562-a54cb01b133f
19. Apache POI, HSSF and XSSF Limitations. <https://poi.apache.org/components/spreadsheet/limitations.html>
20. Harvey Bingham, OASIS Technical Resolution TR 9503:1995, Exchange Table Model Document Type Definition. <https://www.oasis-open.org/specs/a503.htm>
21. Ari Nordström, Balisage: The Markup Conference 2020, Balisage Paper: Pipelined XSLT Transformations. <https://www.balisage.net/Proceedings/vol25/print/Nordstrom01/BalisageVol25-Nordstrom01.html>
22. David Maus, 2018, XProc Step by Step: Implementing a DOCX to TEI step. <https://dmaus.name/blog/2018.14/index.html>
23. Community Group Report 1 July 2022, XProc 3.0: An XML Pipeline Language. <https://spec.xproc.org/master/head/xproc/>
24. Norman Walsh, Achim Berndzen, Balisage: The Markup Conference 2019, Balisage Paper: XProc 3.0. <https://www.balisage.net/Proceedings/vol23/print/Walsh02/BalisageVol23-Walsh02.html>
25. Erik Siegel, 2020, XProc 3.0 - Strategies for merging documents. <https://www.xml.com/articles/2020/11/16/xproc-30-strategies-merging-documents/#sect-pass-as-collection>
26. How do I convert Excel Serial Date Numbers in an XML file to mm/dd/yyyy for SQL Server in an SSIS Package? <https://stackoverflow.com/questions/64018372/how-do-i-convert-excel-serial-date-numbers-in-an-xml-file-to-mm-dd-yyyy-for-sql>
27. How to change date format in Excel and create custom formatting. <https://www.ablebits.com/office-addins-blog/2015/03/11/change-date-format-excel/#excel-date-format>

Thank you!

gudeshani@typefi.com



Gayanthika Udeshani
and the XSLT Team, Typefi