Publishing with XProc

Transforming documents through progressive enhancement

Nic Gibson
Corbas Consulting / LexisNexis

The problem

- Authors write using Microsoft Word (and they *like* it)
- We want rich, semantic structure
- Authors are more important than we are
  - we *cannot* impose structured authoring tools

A solution

- Convert Microsoft Word content to structured, semantic XML
- Build an environment which encourages code reuse
- Use a pipeline engine

Word & WordML

```
<w:p w:rsidR="001C33A0" w:rsidRDefault="0017200C">
  <w:pPr>
    <w:pStyle w:val="Heading1"/>
  </w:pPr>
  <w:r>
    <w:t>Important Title</w:t>
  </w:r>
</w:p>
<w:p w:rsidR="001D4F3B" w:rsidRDefault="0017200C">
  <w:r><w:t>Normal paragraph</w:t></w:r>
</w:p>
<w:p w:rsidR="0017200C"
  w:rsidRDefault="0017200C" w:rsidP="0017200C">
  <w:pPr>
    <w:pStyle w:val="ListParagraph"/>
      <w:numPr>
        <w:ilvl w:val="0"/>
        <w:numId w:val="1"/>
      </w:numPr>
  </w:pPr>
  <w:r><w:t>Bulleted paragraph</w:t></w:r>
</w:p>
```

Word & WordML

Important Title

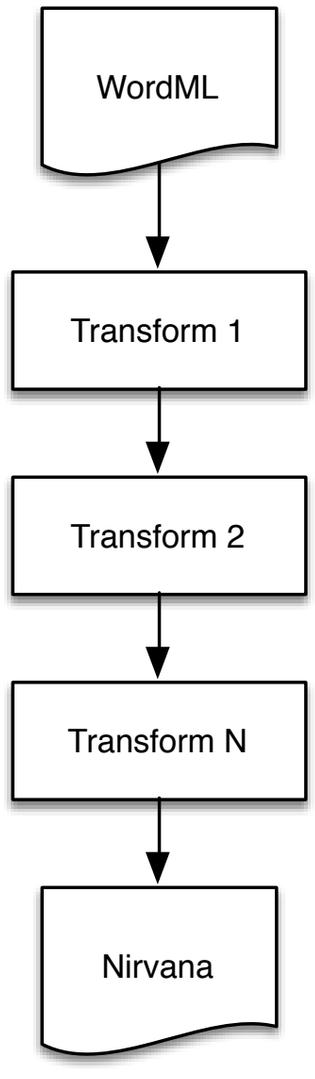
Normal paragraph

- Bulleted paragraph

```
<title>Important Title</para>
<para>Normal paragraph</para>
<itemizedlist>
 <list-item>
   <para>Bulleted paragraph</para>
 </list-item>
</itemizedlist>
```

Progressive enhancement

If a transformation is broken into simple steps focussing on a single part of the conversion, the conversion as a whole will be simpler.



- neutral format
- specialise elements
- group blocks
- add sections

Progressive enhancements…

```
<p cword:style="Heading1">Important Title</p>
<p>Normal paragraph</para>
<li cword:style="ListParagraph>
Bulleted paragraph
</li>
```

```
 <section>
   <h1>Important Title</h1>
   <p>Normal paragraph</para>
   <ul><li>Bulleted paragraph</li></ul>
 </section>
```

```
<h1>Important Title</h1>
<p>Normal paragraph</para>
<ul><li>Bulleted paragraph</li></ul>
```

```
<h1>Important Title</h1>
<p>Normal paragraph</para>
<li cword:style="ListParagraph">
Bulleted paragraph
</li>
```

Environment

- There are requirements
  - pipeline XML in process
  - simplicity of use
  - configurability
  - avoid repetition
  - **manifest files**
  - **generate XSLT from configuration**
  - **pipe the XML**

XProc

- XProc gives us the environment
- XProc is *hard to get started with*
- We need to do that hard part *once*

xproc starts here!

ports ‘carry’ documents

```
<p:declare-step xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:c="http://www.w3.org/ns/xproc-step" version="1.0" name="run-xslt">

    <p:input port="source" primary=“true"/>
    <p:input port="parameters" kind="parameter" primary="true"/>
    <p:output port="result" primary="true"/>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href="word-to-xhtml5-elements.xsl"/>
        </p:input>
    </p:xslt>

    <p:xslt>
        <p:input port="stylesheet">
            <p:document href=“wrap-blocks.xsl”/>
        </p:input>
    </p:xslt>

</p:declare-step>
```

Manifest files

```
<manifest
  xmlns="http://www.corbas.co.uk/ns/transforms/data" xml:base="../
xslt/">
  <item href="word-to-xhtml5-elements.xsl"/>
  <item href="wrap-blocks.xsl"/>
  <item href="“merge_sups.xsl"/>
  <item href="merge_spans.xsl"/>
  <item href="rewrite-para-numbers.xsl"/>
  <item href="group-paras.xsl"/>
  <item href="insert-sections.xsl"/>
  <item href="cleanup.xsl"/>
</manifest>
```

Running that in XProc

```
<p:declare-step xmlns:p="http://www.w3.org/ns/xproc"
  xmlns:ccproc="http://www.corbas.co.uk/ns/xproc/steps" name="transformer"
  xmlns:c="http://www.w3.org/ns/xproc-step" version="1.0">

  <p:input port="manifest"/>
  <p:input port="document"/>

  <p:output port=“result">
    <p:pipe port="result" step=“transform-doc"/>
  </p:output>

  <p:import href="load-sequence-from-file.xpl"/>
  <p:import href="threaded-xslt.xpl"/>

  <ccproc:normalise-manifest name="load-manifest">
    <p:input port="source"><p:pipe port="manifest" step=“transformer"/>
      </p:input>
  </ccproc:normalise-manifest>

  <ccproc:threaded-xslt name="transform-doc">
    <p:input port="source"><p:pipe port="document" step="transformer"/></p:input>
  </ccproc:threaded-xslt>

</p:declare-step>
```

Loading them…

```
<p:declare-step type="ccproc:load-sequence-from-file" name="load-sequence-from-file"
    xmlns:p="http://www.w3.org/ns/xproc" xmlns:data="http://www.corbas.co.uk/ns/transforms/data"
    xmlns:ccproc="http://www.corbas.co.uk/ns/xproc/steps"
    xmlns:cx="http://xmlcalabash.com/ns/extensions"
   version="1.0">

    <p:input port="source" primary="true"/>

    <p:output port="result" primary="true" sequence="true"><p:pipe port="result" step="load-iterator"/></p:output>

    <p:for-each name="load-iterator">

        <p:output port="result" primary="true"/>

        <p:iteration-source select="/data:manifest/*"><p:pipe port="result" step="load-manifest"/></p:iteration-source>

        <p:output port="result"><p:pipe port="result" step="load-doc"/></p:output>

        <p:variable name="href" select="p:resolve-uri(/data:item/@href, p:base-uri(/data:item))"/>

        <p:load name="load-doc">
            <p:with-option name="href" select="$href"/>
        </p:load>

    </p:for-each>

</p:declare-step>
```

Evaluating them…

```
<p:declare-step name="threaded-xslt" type="ccproc:threaded-xslt" exclude-inline-
prefixes="#all"
    xmlns:c="http://www.w3.org/ns/xproc-step" version="1.0"
    xmlns:p="http://www.w3.org/ns/xproc"
    xmlns:ccproc="http://www.corbas.co.uk/ns/xproc/steps">

    <p:input port="source" sequence="false" primary="true"/>
    <p:input port="stylesheets" sequence="true"/>
    <p:input port="parameters" kind="parameter" primary="true"/>
    <p:output port="result" primary="true" />

    <p:option name="verbose" select="'true'"/>

    <p:split-sequence name="split-stylesheets"
      initial-only="true" test="position()=1">
        <p:input port="source">
            <p:pipe port="stylesheets" step="threaded-xslt-impl"/>
        </p:input>
    </p:split-sequence>

<p:count name="count-remaining-transformations" limit="1">
    <p:input port="source">
      <p:pipe port="not-matched" step="split-stylesheets"/>
    </p:input>
</p:count>

<p:xslt name="run-single-xslt">
    <p:input port="stylesheet"><p:pipe port="matched" step=“split-stylesheets"/></p:input>
    <p:input port="source"><p:pipe port="source" step="threaded-xslt-impl"/></p:input>
    <p:input port=“parameters"><p:pipe port="parameters" step=“threaded-xslt-impl"/>
     </p:input>
</p:xslt>
```

split off first stylesheet

how many stylesheets?

evaluate that stylesheet

Evaluating them…

```
<p:choose name="determine-recursion">
  <p:xpath-context><p:pipe port="result" step="count-remaining-transformations"/></
p:xpath-context>

  <p:when test="number(c:result)>0">
    <p:output port="result" sequence="true"><p:pipe port="result" step="run-single-xslt"/
></p:output>

    <ccproc:threaded-xslt name="continue-recursion">
      <p:input port="stylesheets"><p:pipe port="not-matched" step="split-stylesheets"/></
p:input>

      <p:input port="source"><p:pipe port="result" step="run-single-xslt"/></p:input>
    </ccproc:threaded-xslt>

  </p:when>

  <p:otherwise>
    <p:output port="result">
      <p:pipe port="result" step="terminate-recursion"/>
    </p:output>

    <p:identity name="terminate-recursion">
      <p:input port="source"><p:pipe port="result" step="run-single-xslt"/></p:input>
    </p:identity>

  </p:otherwise>

</p:choose>

</p:declare-step>
```

more stylesheets?

recurse

terminate

One more feature…

```
<manifest
  xmlns="http://www.corbas.co.uk/ns/transforms/data" xml:base="../
xslt/">
  <item href="word-to-xhtml5-elements.xsl"/>
  <item href="wrap-blocks.xsl"/>
  <item href="merge_sups.xsl"/>
  <item href="merge_spans.xsl"/>
  <processed-item stylesheet="build-mapping-stylesheet.xsl">
    <item xml:base="../mapping/" href="word-map.xml"/>
  </processed-item>
</manifest>
```

```
<map xmlns="http://www.corbas.co.uk/ns/transforms/map"
    xmlns:cword="http://www.corbas.co.uk/ns/word"
    source-attribute="cword:style"
    ns="http://www.w3.org/1999/xhtml"
    source-element="p">
    <mapping source-value="Title" target-element="h1" heading-level="1"/>
    <mapping source-value="SubTitle" target-element="h2" heading-level="2"/>
    <mapping source-value="ClauseTitle"  target-element="h3" heading-level="2"/>
</map>
```

What does that do?

```
<xsl:template match="p[@cword:style = 'Title']">
  <h1 xmlns:map="http://www.corbas.co.uk/ns/transforms/map" map:level="1">
    <xsl:apply-templates select="@*[local-name() = 'id']"/>
    <xsl:apply-templates select="@*[not(local-name() = 'id')]"/>
    <xsl:apply-templates select="node()"/>
  </h1>
</xsl:template>

<xsl:template match="p[@cword:style = 'SubTitle']">
  <h2 xmlns:map="http://www.corbas.co.uk/ns/transforms/map" map:level="2">
    <xsl:apply-templates select="@*[local-name() = 'id']"/>
    <xsl:apply-templates select="@*[not(local-name() = 'id')]"/>
    <xsl:apply-templates select="node()"/>
  </h2>
</xsl:template>

<xsl:template match="p[@cword:style = 'ClauseTitle']">
  <h3 xmlns:map="http://www.corbas.co.uk/ns/transforms/map" map:level="2">
    <xsl:apply-templates select="@*[local-name() = 'id']"/>
    <xsl:apply-templates select="@*[not(local-name() = 'id')]"/>
    <xsl:apply-templates select="node()"/>
  </h3>
</xsl:template>
```

Things we learned…

- Meta programming rocks
- XProc wraps the complexity
- XProc can’t do threading of content
- Works on other things apart from WordML

https://github.com/Corbas/xproc-tools

https://github.com/Corbas/mapping-tools