Enabling Language Translation with XML Tools and Standards

Bryan Schnabel, XML Information Architect, Tektronix, Inc. with Gail Toft-Vizzini, ed.

Maintaining consistency between a source document and its translated counterparts can be complex and troublesome. Innumerable challenges can arise with character sets, version control, text in graphics, tables, expansion of text, updates, and so on. Using XML for translation can help overcome some of these challenges.

In this article, I explain how XML tools and standards can help remedy tricky issues related to translation.

The Traditional Approaches

Because the challenge of translating documentation is not new, there are a number of traditional approaches. With the traditional approaches, the content to be translated is embedded in a desktop publishing, word processing, or graphic illustration software product. One of two things typically happens. Either the translator must translate the content in the software product, or custom programming must be done to perform a kind of automated text extraction and post-translation recomposition.

Three traditional approaches to translation are

- a supplier uses a potentially complex desktop publishing or graphics tool for layout or uses a custom text extraction and recomposition tool
- an in-house localization department uses a potentially complex desktop publishing or graphics tool for layout or uses a custom text extraction and recomposition tool
- a hapless document writer uses a cut-and-paste method to piece together a translated document

Typically, each of these approaches has to grapple with content and format being intertwined.

XML is a Natural Migration

Many companies have implemented XML to make their documentation process more efficient. Their motives usually include

- efficient layout
- single sourcing
- automation
- data exchange
- e-commerce
- customer requirements
- government requirements

"XML can be a key to solving difficulties associated with translation...."

XML can be a key to solving difficulties associated with translation because it effectively separates content from format and enables the translator to focus on the translation, not the software package the content is embedded in.

XML offers operating system independence (Mac, Linux, Unix, Windows). Usually, authors write in their word processing or desktop publishing software of choice. If the translator needs to translate the content on a different operating system, a potential risk of data corruption arises when the characters are moved from one operating system to the next. XML is operating system neutral, thereby mitigating this risk.

A similar risk exists when the author and translator use different software packages. The author might use his favorite word processing or desktop publishing software. Or a graphics illustrator might use her favorite graphics software. The files might then be saved or filtered into a different package for the translator.

Bryan Schnabel
XML Information Architect
Tektronix, Inc.

bryan.s.schnabel@exgate.tek.com

Bryan Schnabel is the XML Information Architect for Tektronix, Inc. As a world leader in test, measurement, and monitoring, Tektronix is one of the principal players in enabling the coming together of computers and communications. Bryan is a seasoned XML practitioner. He embraces XML as a portable, scalable platform and vendor-independent means to best utilize and protect a company's valuable data. Upon completing his Bachelor of Science and Master's degree at Central Michigan University, Bryan began information architecting for the automotive industry. He began solving problems with standards-based technology early on, first with SGML, then with XML. His accomplishments include serving on the J2008 Automotive Standard Committee, being the Detroit Director of the Midwest SGML Forum, establishing and directing the OregonXML Forum, and serving on the OASIS XLIFF Technical Committee.

XML is vendor independent, thereby mitigating this risk as well.

XML has universal character support. It is based in Unicode, which becomes very useful once translation is required in non-Latin characters.

XML has standards-based facilities for manipulating content, like XSL, XSLT, and XPath, and standards-based facilities for enforcing structure, like XML schema and DTDs.

When XML is used for translation it can

- reduce the number of steps in the translation process
- help translators zero-in on the work they have to do
- make it easier to update translations when the source changes

All these benefits can result in cost and time-to-market savings for the translation process.

And an XML solution, because it is standards-based, is a portable, scaleable solution.

Applying XML to Images

XML provides an excellent way of describing and modeling data. That model and description can be processed by a computer to produce desirable effects. Consider the following simple XML as a way to model or mark up an image.

```
<?xml version="1.0" encod-
ing="utf-8"?>

<ImageFile>

<Circle id="c1" size="2in"
brdr="off" fill="red" loc="ctr" />

<TextBox>

<PositionRelation

CenterTo-id="c1" OffsetXdim="0"
OffsetYdim="0" />

<text id="t1">This is the picture
on my ski</text>

</TextBox>

</ImageFile>
```

You see that metadata provides detail about the geometry of the elements that could be used to create an image. You also see that the string of text is marked up as content in the `<text>` element.

Graphics software designed to process XML might render the previous code example

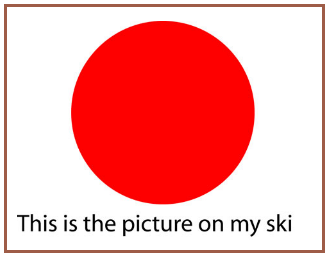


Figure 1. If graphics software could render XML

XML could be used to model data that could be processed by graphics software. It is also reasonable to expect that XML could be used to model data that could be easily processed by software designed for translators.

Consider the following simple example. The previous graphic image XML file could be transformed through XSLT to a format more easily processed by translation tools. The goal of this XML format is to isolate and preserve the hierarchy data *from* the translator and present text in a useful, accessible way *to* the translator.

```
<TransDoc>

<Wrap name="Circle" Watt1="c1"
Watt2="2in" Watt3="off"

Watt4="red" Watt5="ctr">

<Wrap name="TextBox" trans-unit-
ref="t1">

<Meta name="PositionRelation"
at1="c1" at2="0" at3="0" />

</Wrap>

</Wrap>

<trans-unit idref="t1">

<source lang="en">This is the pic-
ture on my ski</source>

<target lang="de">This is the pic-
ture on my ski</target>

</trans-unit>

</TransDoc>
```

The metadata preserves the detail about the geometry of the elements that were used by the graphic image XML. The string of text is marked up as content in the `<trans-unit>` element. In this case, the string is presented in a way that could be made available to a translator. A source string is provided in the `<source>` element ("This is the picture on my ski"), along with an attribute that says the source language is English. The target string is provided in the `<target>` element, along with an attribute that prescribes the target language to be German (de).

The translator would be shielded from the hierarchy data and would edit only the target text string.

```
<TransDoc>
<Wrap name="Circle" Watt1="c1"
Watt2="2in" Watt3="off"
Watt4="red" Watt5="ctr">
<Wrap name="TextBox" trans-unit-
ref="t1">
<Meta name="PositionRelation"
at1="c110" at2="0" at3="0" />
</Wrap>
</Wrap>
<trans-unit idref="t1">
<source lang="en">This is the pic-
ture on my ski</source>
<target lang="de">Dieses ist die
Abbildung auf meinem Ski</target>
</trans-unit>
</TransDoc>
```

An XSL (eXtensible Stylesheet Language) transformation could be applied to get the translated text back into the image format.

```
<?xml version="1.0"
encoding="utf-8"?>
<ImageFile>
<Circle id="c1"
size="2in" brdr="off"
fill="red" loc="ctr" />
<TextBox>
<PositionRelation
CenterTo-id="c1" Off-
setXdim="0" OffsetY-
dim="0" />
<text id="t1">Dieses
ist die Abbildung auf
meinem
Ski</text>
</TextBox>
</ImageFile>
```

Then, the graphics software could render the same image, only this time it would have translated text, as shown in Figure 2.

While the previous example made use of XML and XML tools, it also made use of "home-grown" XML document types called ImageFile and TransDoc.

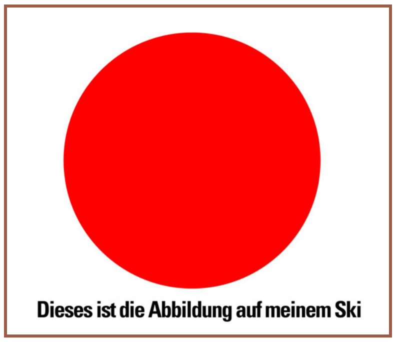


Figure 2. If graphics software could render translated XML

These homegrown document types could be made to work, but it might cause a problem down the road. That is to say, with joint development and customization, Company A and Translation Vendor A might develop a way to use the homegrown ImageFile and TransDoc document types. But what if Company A needs to work with Translation Vendor B? Almost certainly there would need to be more development and more training to get everyone up to speed on the custom document types.

"…industry standards …facilitate efficiency, portability, and interchangeability."

The Value of XML Standard Doctypes

Today, there are a couple of industry-standard document types that provide the means to enable language translation in graphics files using XML tools and standards:

- Scalable Vector Graphics (SVG)
- eXtensible Localisation Interchange File Format (XLIFF)

Some advantages come with using XML:

- content separated from format
- universal character support (UTF)
- no dependency on proprietary formats

And by using industry standards for translation, you gain advantages because the standards facilitate efficiency, portability, and interchangeability.

What Exactly are SVG and XLIFF?

Strictly speaking, SVG and XLIFF are XML applications, complete with validating DTDs or XML schemas. Therefore, each can be transformed with XML tools that implement XML standards, like XSLT and XPath.

SVG is a language for describing two-dimensional graphics in XML. SVG allows for three types of graphic objects: vector graphic shapes, images, and text. Graphical objects can be grouped, styled, transformed, and composited into previously rendered objects.

SVG has several strengths worth noting:

- It is a World Wide Web Consortium standard.
- It is an XML document.
- It is a nonproprietary image format.
- SVG can be opened, manipulated, and saved in popular graphics packages like Adobe Illustrator, Corel Draw, and Visio.
- SVG can be processed with XML tools and standards.
- SVG can be viewed in a browser.

XLIFF (XML Localization Interchange File Format) defines a specification for an extensible localization interchange format that will allow any software provider to produce a single interchange format that can be delivered to and understood by any localization service provider. The format is tool-independent, standardized, and supports the entire localization process.

XLIFF has several strengths worth noting:

- It is a standard of OASIS (Organization for the Advancement of Structured Information Standards).
- It is an XML document.
- It is a nonproprietary document format.
- XLIFF can be opened, manipulated, and saved in popular XML packages like ArborText, XML Spy, and NotePad.
- XLIFF is being developed by translators, tool vendors, and documentation experts.
- Tools in a translator's toolbox will be XLIFF compliant.

The XML Round Trip

To use XML tools and standards to enable language translation, a round trip of the content occurs.-

With an admittedly not-so-trivial upfront development, the nuts and bolts are pretty straightforward. Each step below corresponds with the number in the diagram. See Figure 3.

1. The image is created in a graphics software package (example, Adobe Illustrator). The file is saved as SVG.
2. An XSLT script is invoked that transforms the SVG XML instance to an XLIFF XML instance.
3. The XLIFF is delivered to the translation vendor. The translation vendor uses its XLIFF-enabled tools to translate the text (optionally having the capability to view the SVG for strings of text, in context). The translation vendor returns the translated XLIFF instance.
4. An XSLT script is invoked that transforms the translated XLIFF XML instance back to an SVG XML instance.
5. The translated image file is maintained and saved as an SVG XML instance.

Using SVG and XLIFF for Translation: An Example

The following image was created in Adobe Illustrator and saved as SVG. See Figure 4.

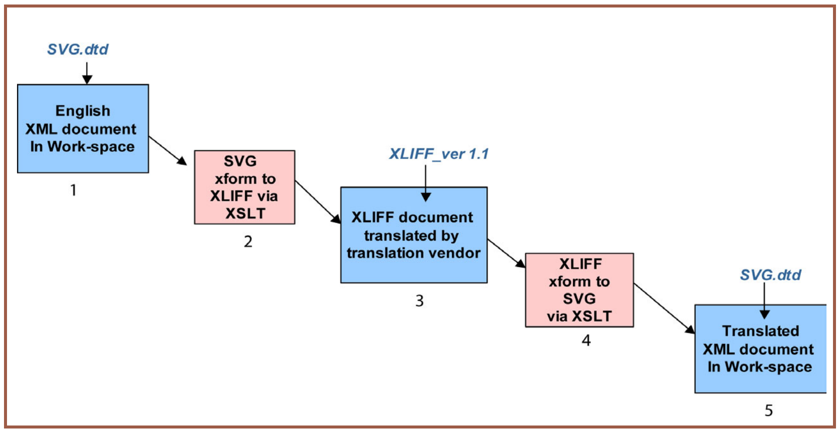

SVG.dtd
English XML document In Work-space
1
SVG xform to XLIFF via XSLT
2
XLIFF_ver 1.1
XLIFF document translated by translation vendor
3
XLIFF xform to SVG via XSLT
4
SVG.dtd
Translated XML document In Work-space
5

Figure 3. XML Roundtrip

Here's a snip of the SVG XML code:

```
<switch i:knockout="Off"
i:objectNS="&ns_flows;" i:object-
Type="pointText">

<foreignObject requiredExten-
sions="&ns_flows;" x="0" y="0"
width="1" height="1" over-
flow="visible"><flowDef
xmlns="&ns_flows;">

<region><path
d="M127.89,175.478"/></
region><flow  xmlns="&ns_flows;"
font-family="'Helvetica-Con-
densed'" font-size="9" text-
align="center" text-align-
last="center">

<p><span>Processor board assem-
bly</span></p>

</flow>

</flowDef>

<x:targetRef
xlink:href="#XMLID_1_" /></forei-
gnObject>

<text id="XMLID_1_" trans-
form="matrix(1 0 0 1 80.6636
175.4775)"><tspan x="0" y="0"
font-family="'Helvetica-Con-
densed'" font-size="9">

Processor board assembly</
tspan></text>

</switch>
```

The second step is to transform the SVG to XLIFF. Figure 5 shows the transformed XML, as it looks in Arbortext Epic, with simple screen styling.

The strings of text are presented to the translator. In its untranslated state, the source and target elements contain matching English text.

The third step is for the translation vendor to translate the target strings, as shown in Figure 6. The XML tool could be adapted to display translated strings in a different style or color for the convenience of the translator.

The fourth step is to transform the translated XLIFF XML instance to an SVG XML instance.

Here's a snip of the SVG XML code, with the text stings translated to French:

```
<switch i:knockout="Off"
i:objectNS="&ns_flows;" i:object-
Type="pointText">

<foreignObject requiredExten-
sions="&ns_flows;" x="0" y="0"
width="1" height="1" over-
```

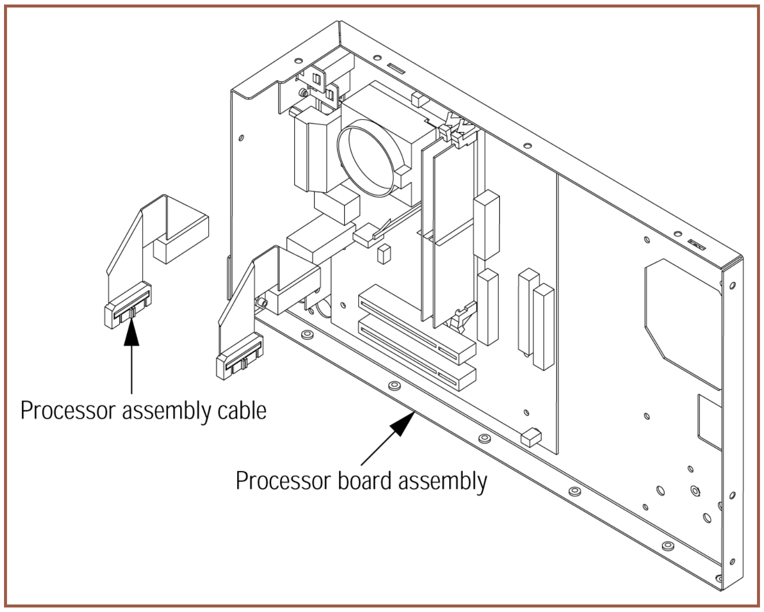
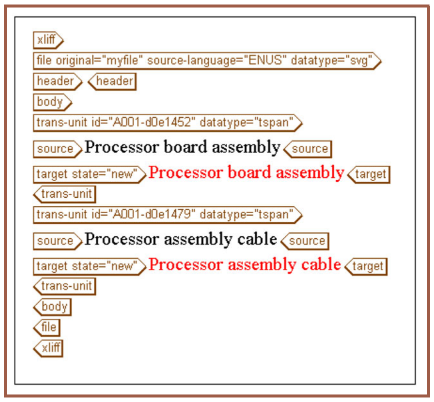
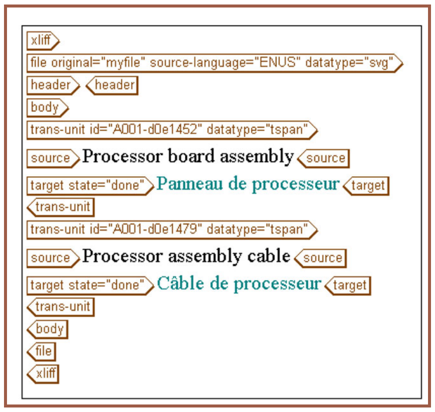


Figure 4. Sample of Line Art saved as SVG



```
xliff
file original="myfile" source-language="ENUS" datatype="svg"
header header
body
trans-unit id="A001-d0e1452" datatype="tspan"
source Processor board assembly source
target state="new" Processor board assembly target
trans-unit
trans-unit id="A001-d0e1479" datatype="tspan"
source Processor assembly cable source
target state="new" Processor assembly cable target
trans-unit
body
file
xliff
```

Figure 5. Untranslated XLIF



```
xliff
file original="myfile" source-language="ENUS" datatype="svg"
header header
body
trans-unit id="A001-d0e1452" datatype="tspan"
source Processor board assembly source
target state="done" Panneau de processeur target
trans-unit
trans-unit id="A001-d0e1479" datatype="tspan"
source Processor assembly cable source
target state="done" Câble de processeur target
trans-unit
body
file
xliff
```

Figure 6. Translated XLIF

```
flow="visible"><flowDef
xmlns="&ns_flows;">

<region><path
d="M127.89,175.478"/></
region><flow  xmlns="&ns_flows;"
font-family="'Helvetica-Con-
densed'" font-size="9" text-
align="center" text-align-
last="center">

<p><span>Panneau de processeur</
span></p>

</flow>

</flowDef>

<x:targetRef
xlink:href="#XMLID_1_" /></forei-
gnObject>

<text id="XMLID_1_" trans-
form="matrix(1 0 0 1 80.6636
175.4775)"><tspan x="0" y="0"
font-family="'Helvetica-Con-
densed'" font-size="9">

Panneau de processeur</tspan></
text>

</switch>
```

The SVG could then be viewed in the graphics software or in the browser. See Figure 7.

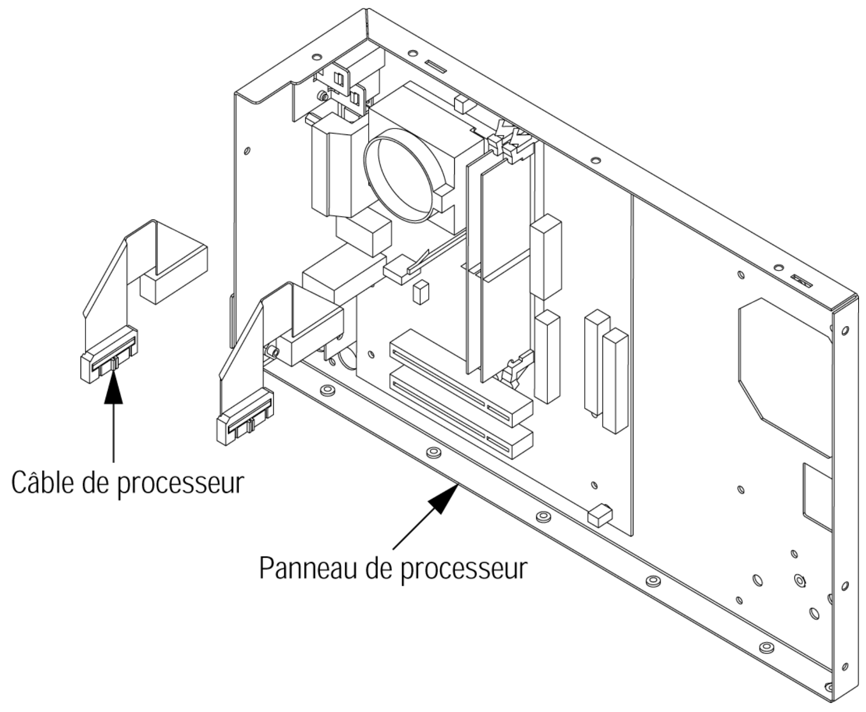
Conclusion

Translating documentation will always be a tricky process. Documentation groups continue to improve processes, tools, and workflows. Content-management systems, be they a straightforward use of high-end software packages or optimized workflow practices, are continuously being refined to meet the translation challenge. Translation operations are maturing and improving, developing practices and tools, such as translation memory, to take on the unique dimensions that language translation brings to the table.

The choice of XML makes tools and standards more successful. Using standards that reflect the experience and expertise from a number of connected industries is a terrific start, but casting the standard in XML brings additional efficiency. The input of the group brings robustness and strength to the standard, while the use of XML provides portability, scalability, universality, and stability to the process of managing documentation.

Reprinted with permission

Best Practices newsletter, Volume 5, Number 4, August 2003.


Câble de processeur
Panneau de processeur