

Translating XML-Based Documents

Xml:tm is an XML namespace-based technology designed to substantially reduce the costs of translating XML documents while at the same time providing the architecture to build consistent authoring systems. xml:tm is an open standard maintained by XML-Intl for the benefit of those involved in the translation of XML documents.

The advent of text in electronic format posed a number of problems for translators. These problems were:

1. How to manage the differing encoding standards and their corresponding font support and availability
2. How to present the text to translators without having to purchase additional copies of the original creation program
3. How to translate the text while preserving the formatting
4. How to build translation memories for these documents to reduce the cost of translation and improve consistency

The problem was exacerbated by the veritable "Tower of Babel" of differing authoring and composition environments, from Interleaf to PageMaker. The typical approach was to write filters that would "lift" the text to be translated from its proprietary embedded environment and to present it to translators in a uniform but equally proprietary translation environment. After translation the text would then be merged with the original document, replacing the source language text.

ISO 8879:1986 SGML

A serious attempt to tackle the plethora of competing formats and their embedded nature was made in 1986 with the advent of ISO 8879 Standard Generalized Markup Language (SGML). The aim of ISO 8879 was to separate the content of documents from their form. SGML arose at a time of great and rapid change in the IT industry. The architects attempted to make the standard as flexible and open to change as possible. This laudable aim unfortunately produced something that was very difficult and expensive to implement. In addition, SGML tackled only the aspect of content. Form was tackled by ISO/IEC 10179:1996 Document Style Semantics and Specification Language (DSSSL), but this proved equally difficult to implement.

HTML

The efforts of the ISO 8879 committee were not in vain. SGML allowed for the creation of HTML, which enabled the World Wide Web to catapult the Internet from a vehicle used by academics and computer scientists to what we know today. HTML was initially based on strict adherence to the SGML standard, but soon diverged as the limitations of ISO 8879 became apparent.

XML

By 1996 the W3C began to look for a solution that was better than HTML. What was required was something that would allow for the semantic exchange of information. It needed to be able to propel the Internet from displaying only static pages to a core semantic Web, allowing for the exchange of data. The efforts of the W3C resulted in XML 1.0. This addressed many

of the architectural limitations of SGML, allowing for easier manipulation and parsing of the semantics. In addition to many very good features, the architects of XML introduced a powerful new concept called "namespace." XML namespace allows for the mapping of more than one representation of meaning onto a given document. This feature is now used extensively in supporting standards such as XSL, XSLT, XML Schema, and FOP.

The Future

The success of XML has been phenomenal, although much of it has yet to become visible to end users. XML has spawned much feverish activity in the developer community and has created some excellent open source tools and libraries such as those provided by the Apache Foundation and SourceForge. Even strongly proprietary companies have had to accept the importance of XML. Much excellent work is also being conducted by standards organizations, such as OASIS and the W3C, on XML-based standards like XLIFF for the translation of documents. XML is driving the future of the World Wide Web. It's providing the foundation for important future Web standards such as XML Web services, electronic data exchange, and so on.

Our premise is that the case for XML is so compelling that all leading vendors of word processing and composition systems will have to support it in the near future. In terms of translation, the arguments are even more convincing. It can be up to five times more expensive to translate and correct the layout of documents written in proprietary systems than in XML. Sun Microsystems and the OpenOffice organization already supply an excellent XML-based alternative to proprietary systems, which can also read proprietary systems such as Word and convert them to XML. Microsoft has also announced support for XML in the next version of Office.

With this view of the future in mind we've concentrated our efforts on how best to exploit the very rich syntax and capabilities of XML.

xml:tm

xml:tm radically changes the approach to the translation of XML-based documents. It is an open standard created and maintained by XML-Intl, for the benefit of those involved in the translation of XML documents.

At the core of xml:tm is the concept of "text memory." Text memory is made up of two components:

1. Author memory
2. Translation memory

Author memory

XML namespace is used to map a text memory view onto a document. This process is called segmentation. The text memory view works at the sentence level of granularity - the text unit. Each individual xml:tm text unit is allocated a unique identifier. This unique identifier is immutable for the life of the document. As a document goes through its life cycle, the unique identifiers are maintained and new ones are allocated as required. This aspect of text memory is called author memory. It can be used to build author memory systems that can be used to simplify and improve the consistency of authoring.

```

<office:body>
  <text:sequence-decls>
    <text:sequence-decl text:display-outline-level="0" text:name="Illustration" />
    <text:sequence-decl text:display-outline-level="0" text:name="Table" />
    <text:sequence-decl text:display-outline-level="0" text:name="Text" />
    <text:sequence-decl text:display-outline-level="0" text:name="Drawing" />
  </text:sequence-decls>
  <text:h text:level="2" text:style-name="Heading 2">
    <tm:te id="e1">
      <tm:tu id="u1.1">Xml:tm</tm:tu>
    </tm:te>
  </text:h>
  <text:p text:style-name="Text body">
    <tm:te id="e2">
      <tm:tu id="u2.1">Xml:tm is a revolutionary technique for dealing with the
problems of translation memory for XML documents by using XML techniques and
embedding memory directly into the XML documents themselves.</tm:tu>
      <tm:tu id="u2.2">It makes extensive use of XML namespace.</tm:tu>
    </tm:te>
  </text:p>
  <text:p text:style-name="Text body">
    <tm:te id="e3">
      <tm:tu id="u3.1">The "tm" stands for "text memory".</tm:tu>
      <tm:tu id="u3.2">There are two aspects to text memory:</tm:tu>
    </tm:te>
  </text:p>

```

Listing 1

Listing 1 is an example of part of an xml:tm document. The xml:tm elements are in bold type to show how xml:tm maps onto an existing XML document.

Translation memory

When an xml:tm namespace document is ready for translation, the namespace specifies the text that is to be translated. The tm namespace can be used to create an XLIFF document for translation.

XLIFF

XLIFF (XML Localization Interchange File Format) is an OASIS standard. XLIFF is another XML format that's optimized for translation. Using XLIFF you can protect the original document syntax from accidental corruption during the translation process. In addition, you can supply other relevant information to the translator such as translation memory and preferred terminology.

```

<xliff version="1.0">
  <file datatype="xml" original="k:/Test/Extract/document2-en.diff" source-
language="en-USA" target-language="es-ESP" tool="xml-intl XLIFF Extract"
xml:space="default">
  <header>
    <skl>
      <internal-file crc="428cc65a">
kGu3LUFqvXtU2xfTdVdzStJnNmKE2Cu0KVCbgrTVvbS9Db2GHyi8fzjf9oJk/guUi+y+057oPFK4EFMOI
v/68u/0rf2l4MjHANfYxwU69WAZXFMspLWbv/k4msPz+gNnr+22cTBSuLas34ZclEzJwrykmwNGflbwEw
gQuJ/7WACwlnf0zh86f/AlrGjEIAYgAA</internal-file>
      </skl>
      <count-group name="Totals">
        <count count-type="TextUnits" unit="transUnits">40</count>
        <count count-type="TotalWordCount" unit="words">416</count>
        <count count-type="AlphaNumericWordCount" unit="words">1</count>
        <count count-type="MeasurementWordCount" unit="words">4</count>
        <count count-type="NumericOnlyWordCount" unit="words">3</count>
        <count count-type="PunctuationOnlyWordCount" unit="words">0</count>
        <count count-type="StandAlonePunctWordCount" unit="words">8</count>
        <count count-type="UpdateMatchedWordCount" unit="words">332</count>
        <count count-type="RealWordCount" unit="words">68</count>
        <count count-type="LeveragedWordCount" unit="words">13</count>
      </count-group>
      <prop-group name="ref-maps">
        <prop prop-type="gmap">{i1=t37}</prop>
        <prop prop-type="xmap">{x1=t31}</prop>
      </prop-group>
    </header>
    <body>
      <trans-unit id="t1" translate="no" ts="text:h" reformat="yes"
xml:space="default" datatype="normal">
        <source>Xml:tm</source>
        <target ts="matched">Xml:tm</target>
      </trans-unit>
      <trans-unit id="t2" translate="no" ts="text:p" reformat="yes"
xml:space="default" datatype="normal">
        <source>Xml:tm is a revolutionary technique for dealing with the problems of
translation memory for XML documents by using XML techniques and embedding
memory directly into the XML documents themselves.</source>
        <target ts="matched">Xml:tm es un técnica revolucionaria que trata los
problemas de memoria de traducción en documentos XML usando técnicas XML e
incluyendo la memoria en el documento mismo.</target>
      </trans-unit>
      <trans-unit id="t3" translate="no" ts="text:p" reformat="yes"
xml:space="default" datatype="normal">
        <source>It makes extensive use of XML namespace.</source>
        <target ts="matched">Esta técnica hace extensor uso de XML namespace.</target>
      </trans-unit>
      <trans-unit id="t4" translate="no" ts="text:p" reformat="yes"
xml:space="default" datatype="normal">
        <source>The "tm" stands for "text memory".</source>
        <target ts="matched">"tm" significa "memoria de texto".</target>
      </trans-unit>
      <trans-unit id="t5" translate="no" ts="text:p" reformat="yes"
xml:space="default" datatype="normal">
        <source>There are two aspects to text memory:</source>
        <target ts="matched">Hay dos aspectos de memoria de texto:</target>
      </trans-unit>
    </body>
  </file>
</xliff>

```

Listing 2

Listing 2 is an example of an XLIFF document based on the previous example. The magenta colored text signifies where the translated text will replace the source language text.

When the translation has been completed, the target language text can be merged with the original document to create a new target language version of that document. The net result is a perfectly aligned source and target language document.

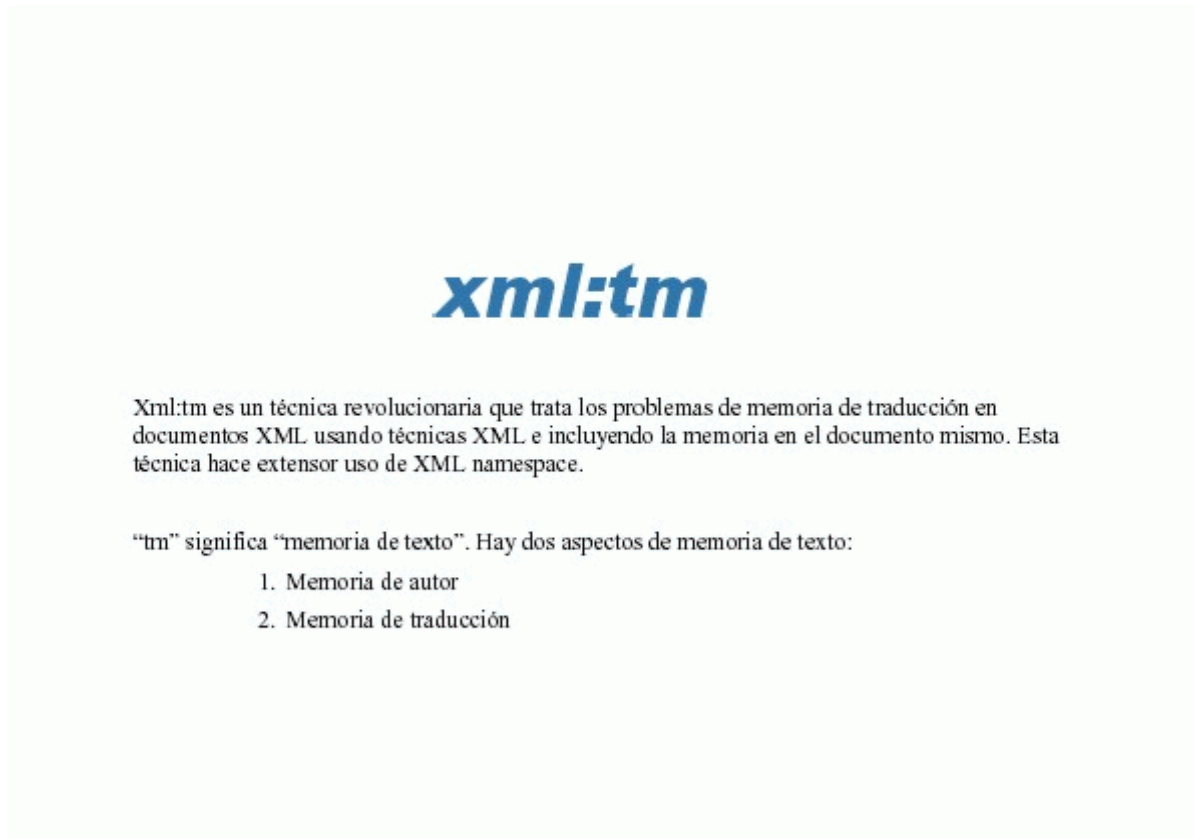


Figure 2

```

<office:body>
  <text:sequence-decls>
    <text:sequence-decl text:display-outline-level="0" text:name="I1"
  <text:sequence-decl text:display-outline-level="0" text:name="Ta"
  <text:sequence-decl text:display-outline-level="0" text:name="Te"
  <text:sequence-decl text:display-outline-level="0" text:name="Dr"
  </text:sequence-decls>
  <text:h text:level="2" text:style-name="Heading 2">
    <tm:te id="e1">
      <tm:tu id="u1.1">Xml:tm</tm:tu>
    </tm:te>
  </text:h>
  <text:p text:style-name="Text body">
    <tm:te id="e2" taval="2">
      <tm:tu id="u2.1">Xml:tm es un técnica revolucionaria que trata l
de memoria de traducción en documentos XML usando técnicas XML e i
memoria en el documento mismo.</tm:tu>
      <tm:tu id="u2.2">Esta técnica hace extensor uso de XML namespace
    </tm:te>
    </text:p>
  <text:p text:style-name="Text body">
    <tm:te id="e3">
      <tm:tu id="u3.1">"tm" significa "memoria de texto".</tm:tu>
      <tm:tu id="u3.2">Hay dos aspectos de memoria de texto:</tm:tu>
    </tm:te>
  </text:p>

```

Listing 3

Listing 3 is an example of a translated xml:tm document. Figure 2 is an example of the text for the source.

The source and target text is linked at the sentence level by the unique xml:tm identifiers. When the document is revised, new identifiers are allocated to modified or new text units. When extracting text for translation of the updated source document, the text units that have not changed can be automatically replaced with the target language text. The resultant XLIFF file will look like Listing 4.

Different Types of Matching

The matching described in the previous section is called "perfect" matching. xml:tm offers unique translation memory matching possibilities to reduce the quantity of text for translation and provide the human translator with suggested alternative translations.

The following types of matching are available:

1. **Perfect matching:** Author memory provides exact details of any changes to a document. Where text units have not been changed for a previously translated document we can say that we have a "perfect match." The concept of perfect matching is an important one. With traditional translation memory systems a translator still has to proof each match, as there is no

way to ascertain the appropriateness of the match. Proofing has to be paid for - typically at 60% of the standard translation cost. With perfect matching there is no need to proofread, thereby saving on the cost of translation.

2. **Leveraged matching:** When an xml:tm document is translated, the translation process provides perfectly aligned source and target language text units. These can be used to create traditional translation memories, but in a consistent and automatic fashion.

3. **In-document leveraged matching:** xml:tm can also be used to find in-document leveraged matches, which will be more appropriate to a given document than normal translation memory leveraged matches.

4. **In-document fuzzy matching:** During the maintenance of author memory a note can be made of text units that have changed only slightly. If a corresponding translation exists for the previous version of the source text unit, then the previous source and target versions can be offered to the translator as a type of close fuzzy match.

5. **Non-translatable text:** In technical documents you can often find a large number of text units that are made up solely of numeric, alphanumeric, punctuation, or measurement items. With xml:tm these can be identified during authoring and flagged as non-translatable, thus reducing the word counts. For numeric and measurement-only text units it is also possible to automatically convert the decimal and thousands designators as required by the target language.

```

<text:ordered-list text:continue-numbering="true" text:style-name="L5">
  <text:list-header>
    <text:p text:style-name="P9">
      <tm:te id="e41">
        <tm:tu id="u41.1">Some new text with examples of text that does not require
translation:</tm:tu>
      </tm:te>
    </text:p>
  </text:list-header>
</text:ordered-list>
<text:p text:style-name="Hanging indent">
  <tm:te id="e42">
    <tm:tu id="u42.1" type="measure">10 mm</tm:tu>
  </tm:te>
</text:p>
<text:p text:style-name="Hanging indent">
  <tm:te id="e43">
    <tm:tu id="u43.1" type="measure">10.50 m</tm:tu>
  </tm:te>
</text:p>
<text:p text:style-name="Hanging indent">
  <tm:te id="e44">
    <tm:tu id="u44.1" type="numeric">10,000</text:span>
  </tm:tu>
</tm:te>
</text:p>
<text:p text:style-name="P13">
  <tm:te id="e45">
    <tm:tu id="u45.1" status="n" type="numeric">9.956</tm:tu>
  </tm:te>
</text:p>
<text:p text:style-name="P13">
  <tm:te id="e46">
    <tm:tu id="u46.1" type="alphanum">ABC104/EF</tm:tu>
  </tm:te>

```

Listing 5

Some new text with examples of text that does not require translation:

10 mm
10.50 m
10,000
9.956
ABC104/EF

Figure 4

Listing 5 is an example of non-translatable text in xml:tm. An example of the composed text is shown in Figure 4

Word Counts

The output from the text extraction process can be used to generate automatic word and match counts by the customer. This puts the customer in control of the word counts.

XLIFF and Online Translation

XLIFF is an OASIS standard for the interchange of translatable text in XML format. xml:tm translatable files can be created in XLIFF format. The XLIFF format can then be used to create dynamic Web pages for translation. A translator can access these pages via a browser and undertake the whole of the translation process over the Internet. This has many potential benefits. The problems of running filters and the delays inherent in sending data out for translation, such as inadvertent corruption of character encoding or document syntax, or simple human workflow problems, can be totally avoided. Using XML technology it's now possible to reduce and control the cost of translation as well as reduce the time it takes for translation and improve reliability.

An example of a Web-based translator environment can be seen at www.xml-intl.com/demo/trans.html.

Benefits of Using xml:tm

The following is a list of the main benefits of using the xml:tm approach to authoring and translation:

1. The ability to build consistent authoring systems
2. Automatic production of authoring statistics
3. Automatic alignment of source and target text
4. Perfect translation matching for unchanged text units
5. In-document leveraged and modified text unit matching
6. Automatic production of word count statistics
7. Automatic generation of perfect, leveraged, previously modified, or fuzzy matching
8. Automatic generation of XLIFF files
9. Protection of the original document structure
10. The ability to provide online access for translators
11. Can be used transparently for relay translation

Figure 5 shows a traditional translation scenario, and Figure 6 shows an xml:tm translation scenario.

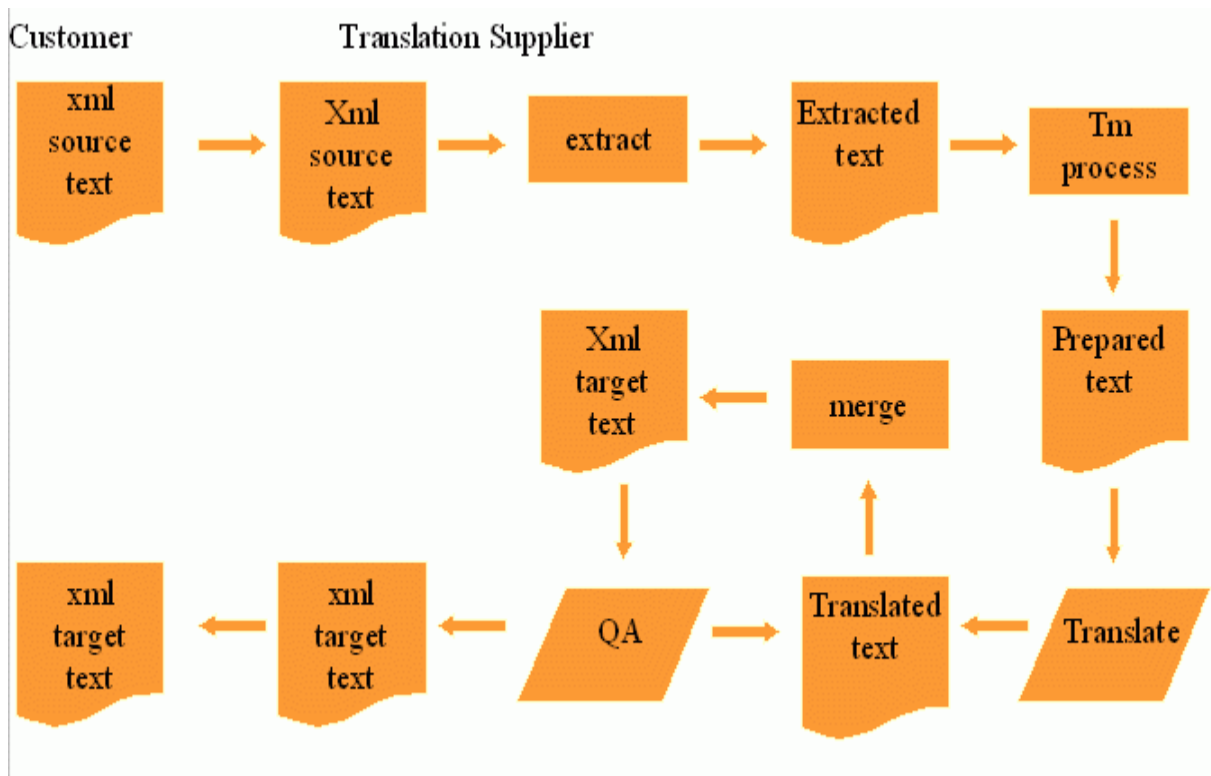


Figure 5

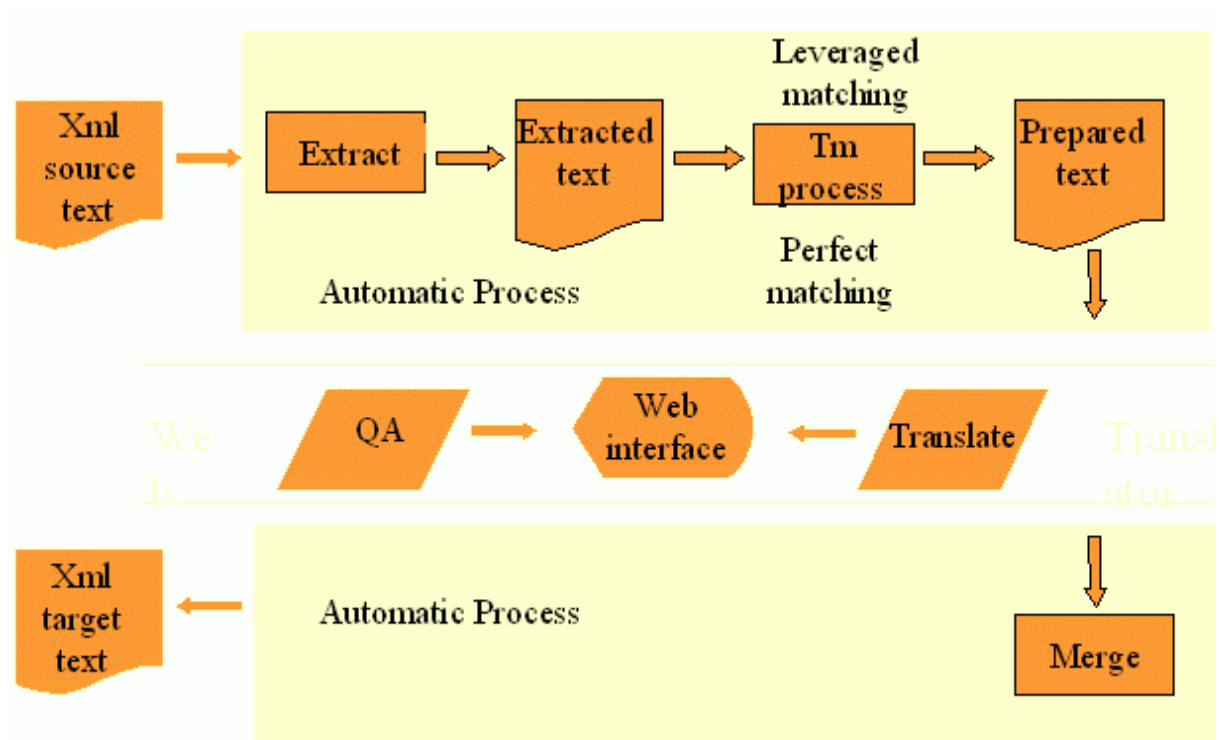


Figure 6

Summary

xml:tm is an open standard created and maintained by XML-Intl based on XML and XLIFF.

Full details of the xml:tm definitions (XML Data Type Definition and XML Schema) are available from the XML-Intl Web site (www.xml-intl.com). XML-Intl also supplies an implementation of xml:tm using Java and Oracle.

xml:tm is best suited for enterprise-level implementation for corporations with a large annual translation requirement and a content management system. During the implementation process xml:tm is integrated with the customer's content management system. xml:tm reduces translation costs in the following ways:

1. Translation memory is held by the customer within the documents.
2. Perfect matching reduces translation costs by eliminating the need for translators to proof these matches.
3. Translation memory matching is much more focused than is the case with traditional TM systems, providing better results.
4. It allows for relay translation memory processing via an intermediate language.
5. All TM, extractions, and merge processing is automatic; there is no need for manual intervention.
6. Translation can take place directly via the customer's Web site.
7. All word counts are controlled by the customer.
8. The original XML documents are protected from accidental damage.
9. The system is totally integrated into the XML framework, making maximum use of the capabilities of XML to address authoring and translation.

References

- The Apache Foundation: <http://xml.apache.org>
- SourceForge: www.sourceforge.net
- OASIS: www.oasis-open.org
- W3C: www.w3c.org
- Sun Microsystems: www.sun.com
- OpenOffice: www.openoffice.org