

Translating XML Documents with xml:tm
by Andrzej Zydron
January 07, 2004

A Russian translation of this article is available at xmlhack.ru

Introduction

Sooner or later someone will want to have your XML document translated into another language. In fact XML documents are much easier to translate than other electronic documents because they separate out form from content, and they conform to a rigorous standard and defined syntax. There are various approaches to improving the translation process.

Machine Translation

Language technology has had a mixed history over the past 40 years. The early promises of cheap automated translation soon lead to dissolution and effectively a marginal role for this technology in providing "gisting" information for certain foreign language texts. There have been significant advances in language technology since then, and we all benefit from these on a day to day basis when we use spelling and grammar checkers and complex search engines. Nevertheless we are still a long way away from usable machine translation based on free format text, although there has been some success for very tightly controlled text in very narrow domains.

Translation memory

In order to reduce translation costs in an environment where documentation can change frequently to reflect improvements and innovation in a product lifecycle the best answer to date has been the use of translation memory. In comparison with machine translation this is a relatively primitive approach to language technology but can bring considerable benefits.

Translation memory works by aligning previously translated text in a target language with the source language. This is accomplished either by the use of a manual tool or automatically by using a controlled environment for the translation process. Alignment is usually done at a sentence level. This affords the best level of usable granularity. The aligned source and target text is held in a repository. The next time the document is updated the repository is searched in order to locate any text that has not changed. Where such a sentence is identified the source language text can be replaced with the target language text. This low tech method has nevertheless provided benefits in terms of translation consistency and reduced costs.

The main weakness of this approach is the fact that how a piece of text is translated in a given target language can depend on its context. When text is pulled in from a translation memory repository it does not possess any of the context within which it existed in the original document. Because there is no contextual information regarding the target language text, a translator is still required to proof read the matched text and adapt it if required. The proof reading process, although less expensive than straight forward translation, still consumes time and money.

Translating XML Documents

The approach to translating XML documents to date has been to extract the translatable text and attributes into an external, typically proprietary format where translation memory matches are performed on the data. On completion of the translation process the newly translated sentences are written to traditional non-standard translation memory repositories. XML in these sorts of environments is treated merely as yet another encoding format.

Special mention must be made here of some important XML based standards concerning translation technology:

1. the OASIS XLIFF (XML Localisation Interchange File Format - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xliff) specification which provides an XML framework for interchanging translatable text from any native format. More about XLIFF later.
2. Lisa (the Localisation Industry Standards Association) provides many XML based initiatives under the auspices of its OSCAR working committee (Open Standards for Container/Content Allowing Re-use - <http://www.lisa.org/oscar>) which include amongst others TMX (<http://www.lisa.org/tmx/tmx.htm>). TMX allows for the interchange of translation memories using XML.

All of these excellent standards address the interchange of information using XML rather than the actual translation of XML documents.

xml:tm

xml:tm is a new approach to the problem of translation for XML documents. It is a XML namespace based syntax that uses the power of XML to embed additional information within the XML document itself.

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 -- 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 which can be used to simplify and improve the consistency of authoring.

The following diagram shows the how the tm namespace maps onto an existing XML document:

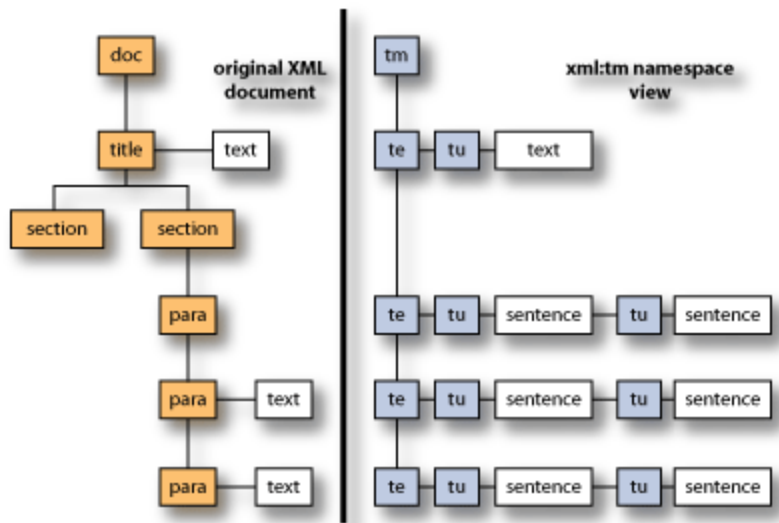


Figure 1: xml:tm mapping diagram

In this diagram "te" stands for "text element" (an XML element that contains text) and "tu" stands for "text unit" (a single sentence or stand alone piece of text).

The following is an example of part of an xml:tm document. The xml:tm elements are highlighted in red to show how xml:tm maps onto an existing XML document.:

```

<?xml version="1.0" encoding="UTF-8" ?>
<office:document-content
xmlns:text="http://openoffice.org/2000/text"
xmlns:tm="urn:xmllint1-tm-tags" xmlns:xlink="http://www.w3.org/1999/xlink">
<tm:tm>
...
<text:p text:style-name="Text body">
<tm:te id="e1" tuval="2">
<tm:tu id="u1.1"> Xml:tm is a revolutionary technology for dealing with the
problems of translation memory for XML documents by using XML techniques to
embed memory directly into the XML documents themselves. </tm:tu>
<tm:tu id="u1.2"> It makes extensive use of XML namespace. </tm:tu>
</tm:te>
</text:p>
<text:p text:style-name="Text body">
<tm:te id="e2">
<tm:tu id="u2.1"> The "tm" stands for "text memory". </tm:tu>
<tm:tu id="u2.2"> There are two aspects to text memory: </tm:tu>
</tm:te>
</text:p>
<text:ordered-list text:continue-numbering="false" text:style-name="L1">
<text:list-item>
<text:p text:style-name="P3">
<tm:te id="e3">
<tm:tu id="u3.1"> Author memory</tm:tu>
</tm:te>
</text:p>
</text:list-item>
<text:list-item>
<text:p text:style-name="P3">

```

```

<tm:te id="e4">
<tm:tu id="u4.1"> Translation memory</tm:tu>
</tm:te>
</text:p>
</text:list-item>

```

And the composed document:

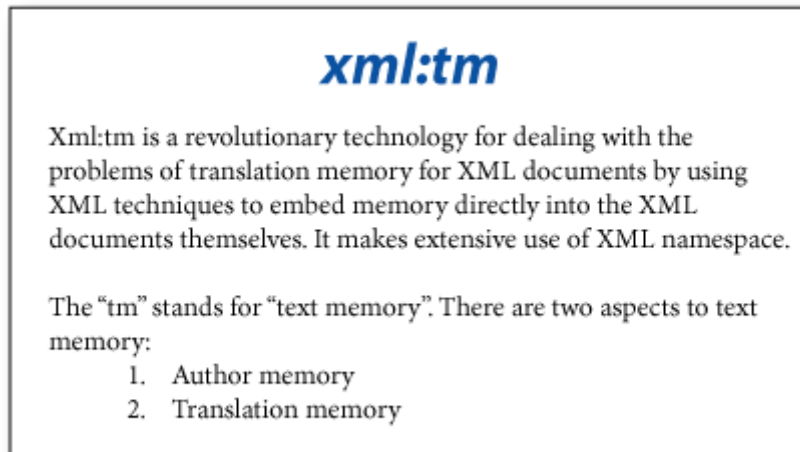


Figure 2: Composed Document

Translation Memory

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

XLIFF

XLIFF is an OASIS standard: [XML Localization Interchange File Format](#). XLIFF is another XML format that is optimized for translation. Using XLIFF you can protect the original document 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.

The following is an example of an XLIFF document based on the previous example:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xliiff PUBLIC "-//XML-INTL XLIFF-XML 1.0//EN" "file:xliiff.dtd">
<xliiff version="1.0">
<file datatype="xml" source-language="en-USA" target-language="es-ESP">
<header>
<count-group name="Totals">
<count count-type="TextUnits" unit="transUnits">40</count>
<count count-type="TotalWordCount" unit="words">416</count>
</count-group>
</header>
<body>

```

```

<trans-unit id="t1">
<source> Xml:tm</source>
<target ts="matched"> Xml:tm </target>
</trans-unit>
<trans-unit id="t2">
<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> 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. </target>
</trans-unit>
<trans-unit id="t3">
<source> It makes extensive use of XML namespace. </source>
<target> It makes extensive use of XML namespace. </target>
</trans-unit>
<trans-unit id="t4">
<source> The "tm" stands for "text memory". </source>
<target> The "tm" stands for "text memory". </target>
</trans-unit>
<trans-unit id="t5">
<source> There are two aspects to text memory: </source>
<target> There are two aspects to text memory: </target>
</trans-unit>
<trans-unit id="t6">
<source> Author memory </source>
<target> Author memory </target>
</trans-unit>
<trans-unit id="t7">
<source> Translation memory </source>
<target> Translation memory </target>
</trans-unit>
...

```

The magenta colored text signifies where the translated text will replace the source language text as shown below:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xliiff PUBLIC "-//XML-INTL XLIFF-XML 1.0//EN" "xliiff.dtd">
<xliiff version="1.0">
<file datatype="xml" source-language="en-USA" target-language="es-ESP">
<header>
<count-group name="Totals">
<count count-type="TextUnits" unit="transUnits">40</count>
<count count-type="TotalWordCount" unit="words">416</count>
</count-group>
</header>
<body>
<trans-unit id="t1">
<source> Xml:tm</source>
<target> Xml:tm </target>
</trans-unit>
<trans-unit id="t2">
<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> 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">
<source> It makes extensive use of XML namespace. </source>
<target>E sta técnica hace extensor uso de XML namespace. </target>
</trans-unit>
<trans-unit id="t4">
<source> The "tm" stands for "text memory". </source>
<target> "tm" significa "memoria de texto". </target>
</trans-unit>
<trans-unit id="t5">
<source> There are two aspects to text memory: </source>
<target> Hay dos aspectos de memoria de texto: </target>
</trans-unit>
<trans-unit id="t6">
<source> Author memory </source>
<target> Memoria de autor </target>
</trans-unit>
<trans-unit id="t7">
<source> Translation memory </source>
<target> Memoria de traducción </target>
</trans-unit>
...

```

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.

The following is an example of a translated xml:tm document:

```

<?xml version="1.0" encoding="UTF-8" ?>
<office:document-content
xmlns:text="http://openoffice.org/2000/text"
xmlns:tm="urn:xmllint1-tm-tags" xmlns:xlink="http://www.w3.org/1999/xlink">
<tm:tm>
...
<text:p text:style-name="Text body">
<tm:te id="e1" taval="2">
<tm:tu id="u1.1"> 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. </tm:tu>
<tm:tu id="u1.2">E sta técnica hace extensor uso de XML namespace. </tm:tu>
</tm:te>
</text:p>
<text:p text:style-name="Text body">
<tm:te id="e2">
<tm:tu id="u2.1"> "tm" significa "memoria de texto". </tm:tu>
<tm:tu id="u2.2"> Hay dos aspectos de memoria de texto: </tm:tu>
</tm:te>
</text:p>
<text:ordered-list text:continue-numbering="false" text:style-name="L1">
<text:list-item>
<text:p text:style-name="P3">
<tm:te id="e3">
<tm:tu id="u3.1"> Memoria de autor </tm:tu>
</tm:te>
</text:p>
</text:list-item>
<text:list-item>
<text:p text:style-name="P3">
<tm:te id="e4">

```

```

<tm:tu id="u4.1"> Memoria de traducción</tm:tu>
</tm:te>
</text:p>
</text:list-item>

```

This is an example of the composed translated text:

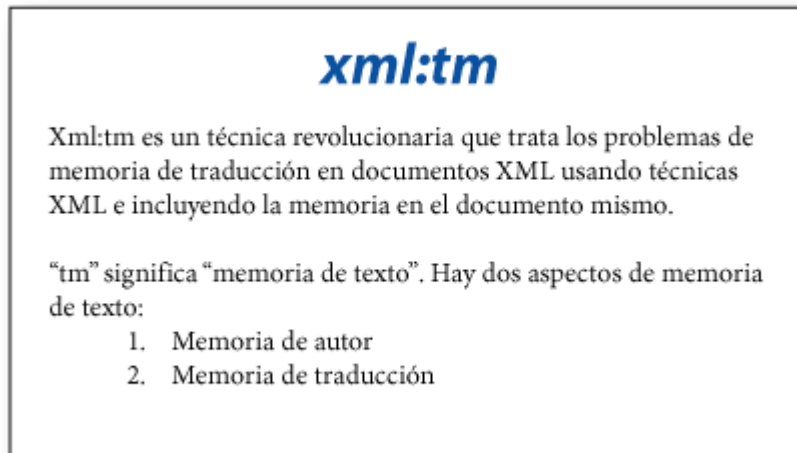


Figure 3: Composed Translated Document

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 resulting XLIFF file will look like this:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE xliiff PUBLIC "-//XML-INTL XLIFF-XML 1.0//EN" "xliiff.dtd">
<xliiff version="1.0">
<file datatype="xml" source-language="en-USA" target-language="es-ESP">
<header>
<count-group name="Totals">
<count count-type="TextUnits" unit="transUnits">40</count>
<count count-type="TotalWordCount" unit="words">416</count>
</count-group>
</header>
<body>
<trans-unit id="t1">
<source> Xml:tm</source>
<target ts="matched"> Xml:tm </target>
</trans-unit>
<trans-unit id="t2">
<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">
<source> It makes extensive use of XML namespace. </source>

```

```

<target ts="matched">E sta técnica hace extensor uso de XML namespace.
</target>
</trans-unit>
<trans-unit id="t4">
<source> The "tm" stands for "text memory". </source>
<target ts="matched"> "tm" significa "memoria de texto". </target>
</trans-unit>
<trans-unit id="t5">
<source> There are two aspects to text memory: </source>
<target ts="matched"> Hay dos aspectos de memoria de texto: </target>
</trans-unit>
<trans-unit id="t6">
<source> Author memory </source>
<target ts="matched"> Memoria de autor </target>
</trans-unit>
<trans-unit id="t7">
<source> Translation memory </source>
<target ts="matched"> Memoria de traducción </target>
</trans-unit>
...

```

Perfect Matching

The matching described in the previous section is called “perfect” matching. Because xml:tm memories are embedded within an XML document they have all the contextual information that is required to precisely identify text units that have not changed from the previous revision of the document. Unlike leveraged matches, perfect matches do not require translator intervention, thus reducing translation costs.

The following diagram shows how Perfect Matching is achieved:

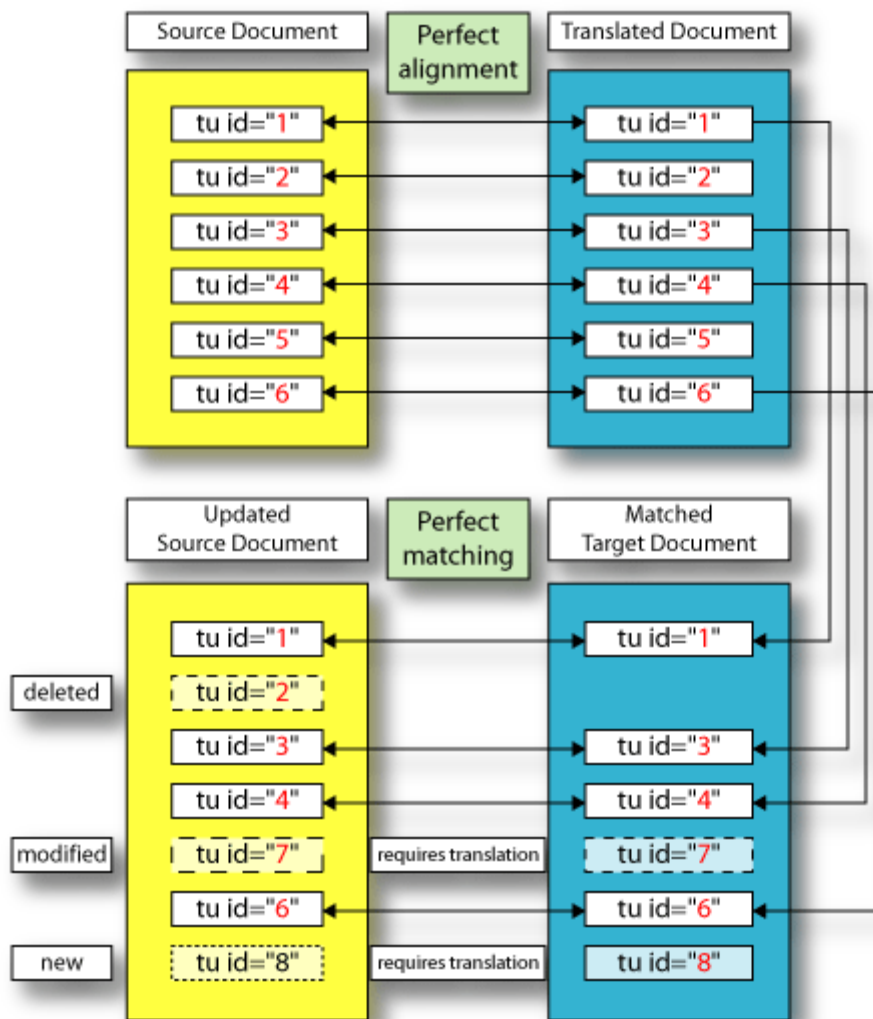


Figure 4: Perfect Matching Mechanism Diagram

Matching with xml:tm

xml:tm provides much more focused types of matching than traditional translation memory systems. 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 proof, thereby saving on the cost of translation.

2. 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.

3. 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.

4. In document fuzzy matching

During the maintenance of author memory a note can be made of text units that have only changed 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. Fuzzy matching

The text units contained in the leveraged memory database can also be used to provide fuzzy matches of similar previously translated text. In practice fuzzy matching is of little use to translators except for instances where the text units are fairly long and the differences between the original and current sentence are very small.

6. 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.

The following is an example of non-translatable text in xml:tm:

```
...
<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 </tm:tu>
</tm:te>
</text:p>
<text:p text:style-name="P13">
<tm:te id="e45">
<tm:tu id="u45.1" 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>
</text:p>
...

```

And an example of the composed text:

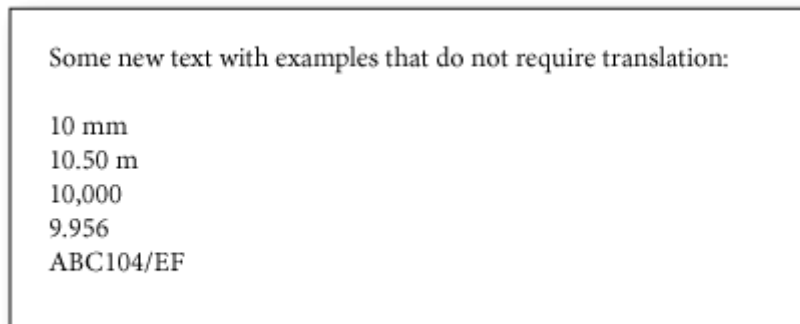


Figure 5: Composed Non-Translatable Text

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, rather than the supplier. This is an important distinction and allows for a tighter control of costs.

XLIFF and Online Translation

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 Web. 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 work flow problems can be totally avoided. Using XML technology it is now possible to both reduce and control the cost of translation as well as reduce the time it takes for translation and improve the reliability.

Traditional translation scenario:

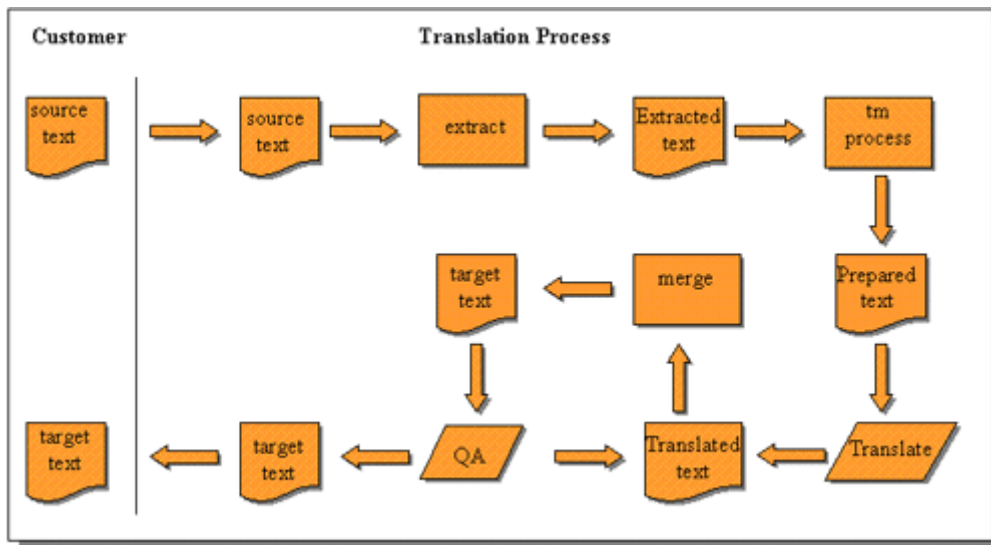


Figure 6: Traditional Translation Scenario

In the xml:tm translation scenario all processing takes place within the customer's environment:

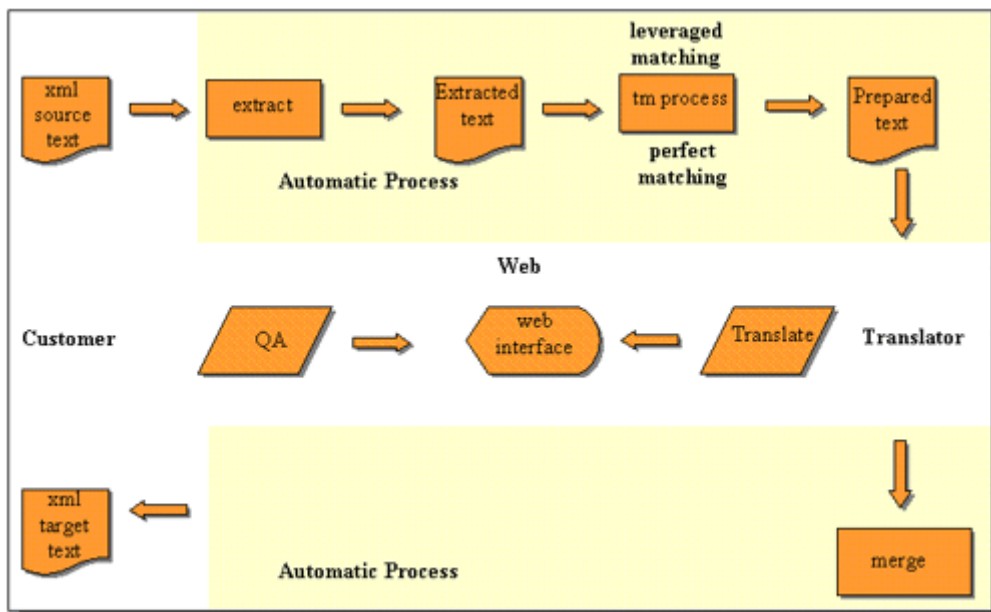


Figure 7: xml:tm Translation Scenario

An example of a web based translator environment can be seen at the following URL: <http://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:

- The ability to build consistent authoring systems.
- Automatic production of authoring statistics.
- Automatic alignment of source and target text.
- Aligned texts can be used to populate leveraged matching tm database tables.
- Perfect translation matching for unchanged text units.
- In-document leveraged and modified text unit matching.
- Automatic production of word count statistics.
- Automatic generation of perfect, leveraged, previous modified or fuzzy matching.
- Automatic generation of XLIFF files.
- Protection of the original document structure.
- The ability to provide on line access for translators.
- Can be used transparently for relay translation.

Summary

xml:tm is a namespace based technology created and maintained by Xml-Intl based on XML and XLIFF for the benefit of the XML community. Full details of the xml:tm definitions (XML Data Type Definition and XML Schema) are available from the Xml-Intl web site (<http://www.xml-intl.com>). Xml-Intl also supplies an implementation of xml:tm using Java and Oracle, which includes linguistically aware database leveraged and fuzzy matching.

There are future plans to incorporate a grammatical namespace in addition to the text memory namespace so that grammatical information can be embedded into XML documents and exchanged between applications.

xml:tm is best suited to 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.

The xml:tm approach reduces translation costs in the following ways:

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