XML Translation Workflow

Jean Sénellart, Christian Boitet, Laurent Romary

SYSTRAN New Generation:
The XML Translation Workflow

Jean Senellart
SYSTRAN S.A.
1, rue du Cimetière
95230 Soisy-sous-Montmorency
France
senellart@systran.fr

Christian Boitet
GETA, CLIPS, IMAG, BP 53
385 rue de la Bibliothèque
38041 Grenoble cedex 9
France
Christian.Boitet@imag.fr

Laurent Romary
LORIA
Campus Scientifique, BP 239
54506 Vandoeuvre-les-Nancy
France
Laurent.Romary@loria.fr

Abstract

Customization of Machine Translation (MT) is a prerequisite for corporations to adopt the technology. It is therefore important but nonetheless challenging. Ongoing implementation proves that XML is an excellent exchange device between MT modules that efficiently enables interaction between the user and the processes to reach highly granulated structure-based customization. Accomplished through an innovative approach called the SYSTRAN Translation Stylesheet, this method is coherent with the current evolution of the "authoring process". As a natural progression, the next stage in the customization process is the integration of MT in a multilingual tool kit designed for the "authoring process".

1 Introduction

Easy and deep customization of Machine Translation for a customer's specific needs has always been a challenge for MT vendors. As the corporate environment becomes increasingly multilingual, the importance of this issue grows. The typical corporate MT customer produces technical support documentation in one language and may need to provide multilingual support for an increasing number of non-English speaking customers. Human translation is not an option for such needs: the number of documents is huge, the database is frequently updated, and not all documents need to be translated into each target language. Finally, the prohibitive costs and the inability to quickly turnaround voluminous amounts of context-specific monolingual content into parallel multilingual corpora make human translation unsuitable for this kind of application. Alternatively, machine translation provides on-demand translations at relatively low costs. But to improve the usability level, the translation system needs to be customized. Classically, a customization process has two main components: terminology customization and engine customization. The first component is developed and maintained by the user (SYSTRAN Intuitive Coding, 2003), while the second (engine customization) traditionally relies on SYSTRAN's expertise and is therefore more costly. Note that in a typical production-level customization process, MT is applied at the very end of the text production workflow. This means that MT applies on publication format and for this reason does not benefit from potential XML structure that is commonly used in today's leading Content Management systems. On the contrary, working on this publication format leads to complex "format filtering" issues.

This article illustrates that the ongoing advancement of SYSTRAN New Generation (NG) engines for XML provides tools tailored for use concurrent with the progressive evolvement of content. This opens a new perspective for fine-grained MT customization, as this development overrides the traditional problematic aspects of the "engine customization" and "format filtering".

We describe two major applications of XML in the SYSTRAN New Generation Systems.

2 MT XML Workflow

In this section, we depict the meaning of an "XML Workflow" for complex translation engines, and

further explain how this workflow improves translation quality by providing users (through the translation authoring tool or integration with a generic authoring tool) with a very simple way of interacting with internal low-level linguistic modules.

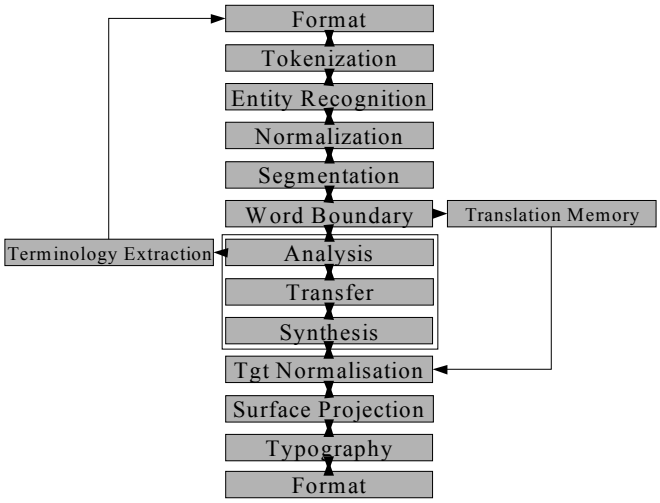


Figure 1a - General SYSTRAN translation workflow

2.1.1 General MT Needs and Module Interaction

At a basic level, the MT process is perceived as a sequence of "independent" modules as represented in Figure 1a, in which:

- The workflow input is formatted text (html, doc, ... file).
- The workflow output is a formatted file with equivalent structure representing the translation of the input text.
- The process parameters are general translation options and resources (choice of user dictionary, linguistic switches, etc.).

The linguistic core of a Machine Translation "analysis-transfer-generation" process is defined in Figure 1b.

This modular organization is a requirement for maintaining very large coverage of NLP systems (NG system, 2001). It also allows the user to easily combine some of the modules in order to build a new specific NLP application, or customize an existing engine with specific modules.

In a large-scale MT system, such as the SYSTRAN systems, all interaction between these modules are handled through:

- Backtracking – if one selection made at a given stage leads to a dead-end, the process is re-initialized in the context of the first choice and a second section is then explored.
- Delayed selection – some decisions normally made at a given stage are delayed by introducing an explicit "undefined" state.
- Parallel computation – several possibilities are explored at the same time. The duration of the parallel computation is generally short for efficiency reasons. Typically, the disambiguation stage requires such processing.
- "Fix" rules – certain rules look for an expected incorrect result from an earlier process stage and fix this result based on additional information present at the current stage.

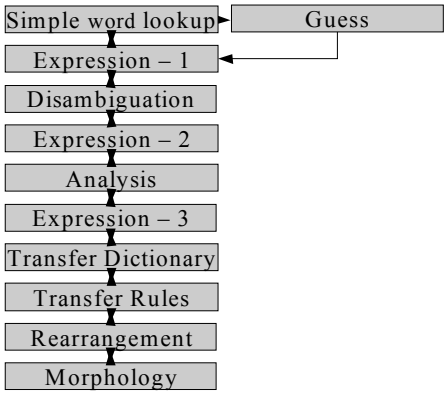


Figure 1b - Schematic "linguistics" workflow

These interaction mechanisms do not represent real "communication channels" between the modules because they cannot be used to control the translation process. On the other hand, they are complex to monitor.

Consequently, none of these solutions is satisfactory for reaching fine-grained customization. Undeniably, to reach this goal, the user should be able to drive the translation process (implicitly or explicitly), not only through additional resources, but also by direct interaction with the different choices made by the modules. We show how this is simplified using a generic XML workflow.

Input Text	A **$100** wrist-watch.
	`<html><hr>A <b>$100</b> wrist-watch.</html>`

first xml representation	`<?xml version="1.0"?>`
	`<document original_format="html"><tag>&<lt;hr></tag><par id="1">A <ts face="bold">$100</ts> wrist-watch.</par>`
	`</document>`

tokenized text after entity recognition	`[...]`
	`<par id="p1" xml:lang="en">`
	`<token type="word" capit="first" id="t1">A</token>`
	`<ts face="bold"><entity type="monval" id="t2">$100</entity></ts>`
	`<token type="word" norm="wrist-watch" id="t3">wristwatch</token>`
	`<token type="punct" id="t4">.</token>`
	`</par>`
	`[...]`

segmented, translated text	`[...]`
	`<par id="1">`
	`<tu_group id="s1">`
	`<tu xml:lang="en">`
	`<token type="word" id="t1" capit="first">A</token>`
	`<ts face="bold"><entity type="monval" id="t2">$100</entity></ts>`
	`<token type="word" norm="wrist-watch" id="t3">wristwatch</token>`
	`<token type="punct" id="t4">.</token>`
	`</tu>`
	`<transtu xml:lang="fr">`
	`<token id="t1" synt="pos=det">Une</token>`
	`<token type="word" type="noun" id="t3">montre-bracelet</token>`
	`<token type="det" id="t2-0">de</token>`
	`<ts face="bold"><entity type="monval" id="t2">$100</entity></ts>`
	`<token type="punct" id="t4">.</token>`
	`</transtu>`
	`</tu_group>`
	`</par>`
	`[...]`

output text	`<html><hr>Une montre-bracelet de <b>$100</b>.</html>`
	Une montre-bracelet de **$100.**

Table 1 - Some intermediary translation steps within the translation of a single html sentence.

2.1 XML - Communication Model Between Modules and with an External Layer

The fundamental idea of the XML Workflow is simple:

- The input document is converted into an XML format. This conversion preserves (hides) the original formatting in `<tag>` nodes, and normalizes font properties into `<ts>` nodes[1].
- All modules in the process maintain this XML structure and compose its evolution.
- This occurs with the following main restriction. In every module, each XML structure output is aligned with the XML structure input. This alignment is done at the paragraph level, sentence and token[2] levels. This is performed by several means: IDREF, "norm" attributes preserving the state before the current transformation (for example, in tokenization or entity recognition) and duplication of structures.

As a result, the input text can be regenerated with additional tagging introduced in a given step of the process, at any time.

An illustration of this workflow is provided in the table above. Note that only some steps of the process are represented.

Technically, the introduction of this XML workflow is based on the SYSTRAN NG Engine Architecture (NG, Senellart 2001). For current "classical SYSTRAN engines", this new XML Workflow fully replaces the existing workflow

1Preserving character property in the translation process is critical for preserving the identical font style in the text output and for allowing translation memory parsing (TM entries in SYSTRAN systems are full enriched sentences)

2 A "token" becomes a "word" when linguistics really starts.

modules which are not "purely linguistic", and complements the current structure for "purely linguistic" modules. The most important point is that the internal structure (Analysis Area) on which linguistic routines apply is preserved, but is synchronized with this XML structure during any point in the process.

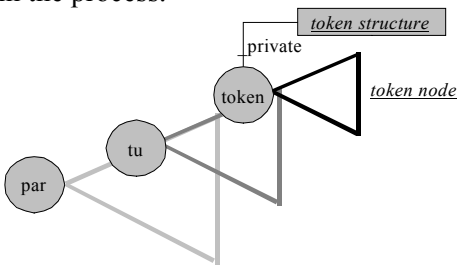


Figure 2 - XML tree and associated memory representation

The actual XML structure updates are performed by a "lazy algorithm" for efficiency reasons. The internal structure handled by the modules is not the XML structure itself, but an optimized memory structure. For example, a "token" node is internally represented by a "token" structure. If the frame for storing the token is still the XML tree, the up-to-date "token" representation is the memory structured stored as a pointer in a private field of the corresponding "token" node. The token node is synchronized when a dump of the XML structure is required, or when an XML operation, such as an XPath evaluation, is performed.

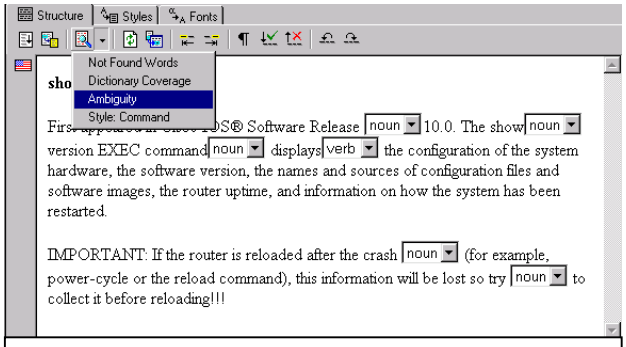


Figure 3 - Conversion of "mark_choice" system tags into html controls

The communication "model" is then applied using a "`mark_choice`", and/or "`post_choice`" markup in the structure, as detailed in the following application "User-Process interaction".

2.2 User-Process Interaction

Based on the previous workflow, defining the interaction between user and process is straightforward:
Let us consider noun-verb disambiguation. Figure 4 displays the feedback and the interaction process:

- The first process is run and stops after executing the disambiguation routine.
- Selected system choices for this routine are integrated in the XML structure via `<mark_choice>` tags.

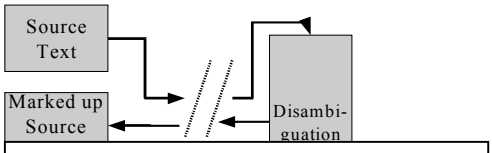


Figure 4 - "markup/interaction/post" process

```
<mark_choice
process_path="//ling/disambiguation[@type='noun
-verb']"          values="verb          noun"
confidence="0.5">verb</post_choice>
```

- The source text is regenerated and tags are converted into user readable tags such as textual marks, or, as in figure 3, into user-friendly html scroll-down control).
- The user reviews and modifies the system choices.
- When translation is eventually run, the user choices are integrated into the XML structure as `<post_choice>` tags.

```
<post_choice
process_path="//ling/disambiguation[@type='noun-
verb'] ">noun</post_choice>
```

- The disambiguation routine replaces the system choices with user choices when provided.

In SYSTRAN's system, the process and markup type identification is performed using an XPath expression in the "process tree". Each module is identified as a "node" in the process tree. Using this classification allows us to define a "communication" scheme with any new modules, as well as the possibility express precise restrictions, such as the following:

```
<!-- markup all noun-verb or noun-adj system
choice in disambiguation process -->
<select
process_path="//ling/disambiguation[@type='noun-
verb' or @type='noun-adj']"/>
```

```
<!-- markup all linguistic choices with a low confidence -->
<select process_path="//ling/*[@confidence &lt; 0.2]"/>
```

2.2 SYSTRAN Translation Stylesheet

In this part, we portray a second application for XML in SYSTRAN NG Engines. This approach is independent of the XML workflow described above, but demonstrates that the combination of both allows the system to reach an effective **customization frame** based on the deep interaction between the translation system and highly granulated structure-based customization.

In the XML workflow, the input structure is mapped into an internal XML generic structure. However, this mapping only allows for the handling of shallow document structure. A typical example is PDF translation. For most PDF documents, the information that the PDF filter can retrieve is limited to a sentence list. Numbering may be preserved but without minimal structure information that clarifies two list items as consecutive and belonging to the same enumeration. Bold face font mat be preserved but not the style category that generated it. Even worse, words with internal elision may not be correctly reconstituted; not because of the poor quality of the PDF import filter, but because that information is no longer present in the document, and can only be reconstituted, at best.
This limitation example of PDF documents is not unique. To a lesser extent, we observe the same phenomenon in all classical document formats.
For example, an html page generated from a newspaper database does not contain information pertaining to the database structure, only its presentation[3]. The newspaper website contains different articles presented together and probably stored internally with domain information. But in most cases, this information is no longer part of the html structure; and even when present, is only expressed as comments but not sufficiently structured for automatic processing.
Finally, it is important to note that this situation is more than a question of format. Most word processors are designed to handle named styles (for example, paragraph or character styles), but this option is usually ignored, as manual direct formatting is preferred.

Undoubtedly, XML content management and structured documents represent the future of text authoring. Several pure XML publication formats, like DocBook and specialized technical XML formats like XBRL (Extensible Business Reporting Language), are now recognized asstandards. Even major word processors and other content editing tools today integrate XML as a reference content format (such as OpenOffice, Office11).

Figure 5a-Classical use of Machine Translation, downstream text production using Word Processor Translation

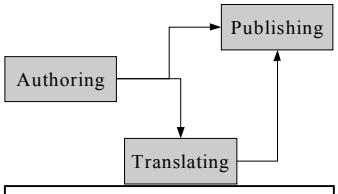


Figure 5b- Use of Publishing and Translation Stylesheets downstream

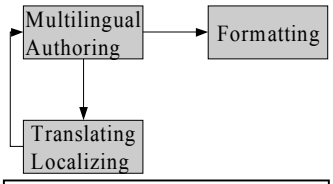


Figure 5c- Integration of XML multilingual Workflow in the Authoring Process

3 Even if HTML language was to some extent designed to represent meta-information through tags like **<quote>**,

```
<systran:output>
    <source xml:lang="en">The <name>Z12</name> <Strong>driver</Strong> is
loaded.</source>
    <target xml:lang="fr">Le <mark type="dict" name="global">
        <Strong>driveur</Strong></mark> <name>Z12</name> est chargé.</target>
    <options>
       <dicts><dictionary name="global"/></dicts>
       <marks><dict_mark type="xml"/></marks>
       <misc><extract_nfw/></misc>
       <ling><imperative type="technical"/></ling>
    </options>
    <nfw_list>Z12</nfw_list>
</systran:output>
```

code sample 1- Translation output - a tree fragment that contains source, target, not found word list, and translation options. Among translation options, "mark-up" of user-dictionary is activated.

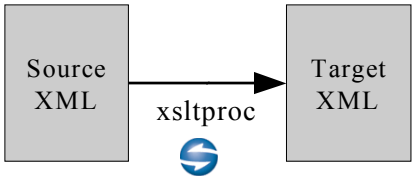
Below we display that the concept of "translation filter" is totally overridden by the "Translation Stylesheet" (TS).

2.2.1 Definition - Translation and Formatting Stylesheets

A stylesheet is a way of associating "style" to formatting marks. In other words, the stylesheet is "*the file that describes how to display a given (XML) input file*".

XSL (Extensible Stylesheet Language) is a powerful language used to express stylesheets. As a complete language based on XML objects, XSL provides the opportunity of transforming a source XML document into a target XML document, which goes far beyond "formatting".

By extending "formatting" stylesheets and comparing publishing in the content production workflow (figure 5), we use this transformation mechanism to produce fine-grained customized multilingual content.



Translation Stylesheet

The Translation Stylesheet is thus "*the file that describes how to translate a given (XML) input file*".

2.2.2 SYSTRAN Translation Stylesheet

SYSTRAN Translation Stylesheet is based on a single extension function "`systran:translate`". The whole translation process is driven by a xslt processor (xsltproc).

Technically the `systran:translate` function has the following "xslt" APIs:

```
node-set        systran:translate(node-set
text,node-set translation_option)
node-set          systran:translate(string
text,node-set translation_option)
```

The input is either a node-set or a string (depends on the calling context in the XSL process).

The functions return a tree fragment that contains the translated node-set, and the translation process feedback, such as a list of NFWs (Not Found Words) or sentence dictionary, etc.

The set of translation options is represented by an option tree, and controls the type of information returned by the function. The following example shows a typical output structure.

```
<xsl:template match="CommandName">
  <systran:dnt type="noun">
     <xsl:copy-of select="."/>
  </systran:dnt>
</xsl:template>
```

code sample 2 - XSLT template transforming <CommandName> element into <systran:dnt>. The node content is not translated, but integrated in the sentence analysis as a full noun, preserving the sentence structure.

.

Based on the `systran:translate` function, the translation stylesheet describes which XML area to translate and with which options through "xsl

---

<code>, ...

```
<systran:output>
    <source xml:lang="en">The <name>Z12</name> <Strong>driver</Strong> is
loaded.</source>
    <target xml:lang="fr">Le <mark type="dict" name="global">
        <Strong>driveur</Strong></mark> <name>Z12</name> est chargé.</target>
    <options>
       <dicts><dictionary name="global"/></dicts>
       <marks><dict_mark type="xml"/></marks>
       <misc><extract_nfw/></misc>
       <ling><imperative type="technical"/></ling>
    </options>
    <nfw_list>Z12</nfw_list>
</systran:output>
```

code sample 3 - Translation output. It is a tree fragment containing source, target, not found word list, and translation options. Among translation options, "mark-up" of user-dictionary is activated.

templates". Assume that we have an input XML containing formatted text "Para" nodes that we want to translate and "Note" nodes that we want to remain in the source language. The xsl template in *code sample 4* shows how Para nodes are processed. The main process is based on the following scheme:

- Define the XML area to be translated (can be a tree fragment or a string, if translation of attribute values is needed)
- Preprocess (via another template) the tree fragment: this module typically interprets "token level" tags and converts those tags into internal SYSTRAN tags with appropriate semantics. For example, in *code sample 5*, a template converts `CommandName` tags into `systran:dnt` (Do Not Translate) SYSTRAN tags, and associates a part of speech to this word. Therefore, the analysis of the complete sentence is done with the acknowledgement that a token should not be translated as it behaves like a noun.

3 Conclusion and Perspectives

This article discusses the impact related to the introduction of XML in the New Generation SYSTRAN MT Architecture. This has and continues to open new views for Machine Translation use, as well as the ability to successfully and simply perform MT customization. The XML frame is fully functional; the next stages will focus on its industrialization through a new preprocessing interface, a new customization methodology, and the exploration of new linguistic opportunities. For example, since the XML workflow enables the dynamic definition of "translation options" as a part of the communication channel, a direct application of this new option could allow some modules to generate on-the-fly local dictionaries.

The current evolution of transforming the

```
<xsl:template match="Para">
    <xsl:comment>SYSTRAN translation</xsl:comment>
    <xsl:variable name="preprocess">
        <xsl:apply-templates select="." mode="preprocess"/>
    </xsl:variable>
    <xsl:variable
       name="translation"
       select="systran:translate(exslt:node-set($preprocess)/*,$OPT)"
    />
    <xsl:apply-templates
              select="$translation/target/*" mode="postprocess"/>
```

code sample 4 - xsl template describes the translation of "Para" element. <Para> nodes are localized, preprocessed, translated with the current XML option $OPT, and then regenerated using a postprocess template.

authoring process to structured format initiates better content control and responds to the increasing need for defined exchange standards. Since “exchange” now implies multilinguality, the natural next step is the introduction of multilingual consideration, as illustrated in Figure 5. The ability to rate the machine translatability of a text should also be considered at the same level as the validation of text structure or spell-checking. Beyond the technical issues regarding the interaction of multilingual tools and text production, as presented in this paper, this evolution renews the linguistic definition of translation tools.

3 Acknowledgments

Special thanks to Daniel Veillard, architect and main developer of libxml2 and libxslt ; these are efficient, complete, portable, and free XML libraries that provide a large number of existing standards related to markup languages (XML, XSL, XPath, XInclude, Relax NG, XML Schemas, etc.). See http://www.xmlsoft.org.

4 Bibliographical References

Senellart J., Dienes P., Váradi T. 2001. New Generation SYSTRAN Translation System. In *Proceedings of MT Summit IIX*

Senellart J., Yang J., Rebollo A. 2003. SYSTRAN Intuitive Coding Technology. In *Proceedings of MT Summit IX.*

Boitet, C. 2000. Handling text and corpus in Ariane-G5, a complete environment for multilingual MT. In Proceedings of ACIDCA'2000/

Lieske C., McCormick S., Thurmair G. 2001. The Open Lexicon Interchange format (OLIF) comes of Age. In Proceedings of MT Summit IIX.

Walsh N., Muellner, L. 1999. DocBook: The Definitive Guide. *O'Reilly Books*.