

技術翻訳とマークアップ言語

出来 信一

技術翻訳の作業を含む出版プロセスの多くでは、作業文書の形式として SGML や XML などのマークアップ言語が利用されている。それにより、翻訳者は本質的に翻訳でない作業からある程度解放され、翻訳作業にいっそう集中できる。

今となってはマークアップを全く利用しないという選択肢は考えられないであろう。しかし、マークアップ言語の利点を十分に引き出すには、まだまだ課題が残されている。技術翻訳でこれまで作業してきた経験に基づいて、マークアップ言語を利用する問題点を以下にまとめてみた。これらの点は、実際には翻訳だけでなく文書の作成や処理に関連したあらゆる分野に当てはまるに違いない。

1. 翻訳者自身の課題

マークアップ採用は翻訳作業の効率向上につながるが、その一方で、翻訳者にとっては、マークアップについてそれなりの理解が求められるようになった。情報処理の背景の乏しい翻訳者にとっては、少し負担かもしれない。しかし、技術翻訳の分野に限って言えば、建設的な影響をもたらすものでもある。すなわち、XML の導入自体が情報処理の大きな流れの 1 つということからすると、マークアップの導入は翻訳者が専門分野に関する理解を増し加えるよう翻訳者を後押しするものとなり得るのである。少なくともワープロソフトの書式設定を覚えることに比べれば、はるかに教育的である。

2. 原文作成での課題

マークアップの導入により翻訳者だけでなく、当然ながら原文執筆者もマークアップ言語を利用することになる。文書を最初にどうタグ付けするかは、後の処理を左右することになるので重要である。

時折、翻訳作業を困難にするような複雑なタグ付けがされていて、苦勞することがある。その一例は条件分岐である。たとえば、

```
See <ph otherprops="html"><xref href="folder/file.ext#id">/<xref>. </ph><ph otherporps="PDF">the manual, "Installation Guide". </ph>
```

ここでは、ターゲットが html なら See <xref href="folder/file.ext#id">、また PDF なら See the manual, "Installation Guide". となるよう、<ph> 要素の otherprops 属性を使って条件分岐がコーディングされている。訳文でも同様に切り替わるようにタグを付ける必要があるが、多くの翻訳ツールでは最小のピリオドを文の区切りと判断して翻訳単位を分けてしまうので、ともすると条件分岐を見落としてしまう可能性があるし、場合によっては翻訳データベース中の望ましくない訳文で自動的に置き換えられてしまう可能性がある。

もっと複雑な例として、原文の 1 か所に存在する条件分岐を訳文では離れた 2 か所に分散して訳さなければならぬことがある。そのような場合、条件分岐タグを増やすか、または文全体を切り替えるよう、一部を重複して訳さなければ

ならない。長文の中に他のタグに混じってこのようなタグが複雑に付いていると、エディタでテキストを眺めても、原文と訳文の対応を確認するのは容易ではない。

翻訳作業の点からすると、是非とも 1 文全体が条件分岐の単位となるように原文を作成してほしいものである。

3. ツールの課題

マークアップ言語が使用されるのであれば、何らかの形で機械的処理が施されることが意図されているであろう。翻訳作業自体に使用されるツールを含めて、マークアップ文書の処理ツールに期待したい点はいくつかある。

● 翻訳単位の区切り方

タグ付けの方法が翻訳作業に微妙に影響する場合がある。たとえば、要素タグの属性値文字列を翻訳しなければならない場合があるが、属性値だけを 1 つの翻訳単位とするか、それともタグ全体を単位とするかによって、翻訳データベースに格納される形態が異なってくる。

また、同一英文に対しても、セクションのタイトル、索引項目、用語集の見出し、箇条書きの 1 項目というように文脈に応じて言い回しを微妙に変化させなければならぬことがある。そのような場合、それを囲むエレメントタグも翻訳単位に含めるなら、将来訳文を再利用する際に訳文選択を誤る可能性が少なくなる。たとえば、

Setting the attribute values

というフレーズを、タイトルでは「属性値を設定する」と訳し、索引

項目では「属性値の設定」と訳すかもしれない。その場合、

```
<title>Setting the attribute values</title>
<idxterm>Setting the attribute values</
idxterm>
```

のようにそれを囲むエレメントタグまで含めて翻訳単位とすれば、訳文再利用の際にも確実に文脈に忠じた訳を採用することができる。翻訳ツールで、このような点まで考慮して翻訳単位を切ることが望ましい。

● 参照の問題

マークアップにより相互リンクや文字列参照が実現されていることが少なくないが、参照元を見ただけでは参照先の文字列がわからないことが多い。たとえば、文字列 Product A を参照するタグとして、以下のさまざまな形式が考えられる。

```
(1) <ref source="../proA/ProductAIntroduction.xml">Product A</ref> has been released.
(2) <ref source="../proA/ProductAIntroduction.xml" /> has been released.
(3) <ref source="../proA/ProductAIntroduction.xml" str="Product A" /> has been released.
```

(1) の場合、すべての参照箇所ですべての参照箇所を翻訳しなければならないし、日本語名称が変更になった場合はすべての参照箇所に変更を加えなければならない。不便のように思えるが、一方で、翻訳作業では、実際に現れる文字列が何であるかを調べる手間が省けるという利点がある。

(2) の場合、名称変更にも自動処理で対応できるため、マークアップの機能を十分に利用できる。しかし、翻訳者にとっては、このタグが実際にどんな文字列に置き換えられるのかは調査しなければならないという点で不便である。

特に、翻訳作業用に提供されるファイル群の中に参照先(この例では ProductAIntroduction.xml)が含まれているとは限らないので、(1) のように文字列が明示されているほうがありがたい。

(3) は、最も融通の利く方式となり得る。str 属性は単なる覚え書きとして使用し、組版時には参照先の文字列を実際に取り出すようにするならば、(1) と (2) の利点が両立する。この方法ならば、名称変更時に str 属性の修正を仮に忘れても、何を表すタグかの概略は知ることができる。str 属性の代わりに desc 属性による説明だけにしてもよいだろう。

4. マークアップ言語自体に期待すること

● 直感的にわかりやすいマークアップ仕様

xhtml のように手軽にレンダリングができるスキーマもあるが、それが容易でないスキーマもある。たとえば前述の <ref> 要素は、空要素でありながら文字列に置き換えられることを意図したものである。文脈によっては、前後を見るだけで大体推測できるかもしれない。しかし、前後の文のつながり具合によっては、これが何らかの文字列に置き換わることに気付かない可能性もある。

できることならば、タグ自体が文字列としての内容を持つことを直感的に示す形にできないだろうか。たとえば、

```
<&ref source="../proA/ProductAIntroduction.xml" str="Product A" />
```

のように "<&" で開始することによりこれが文字列に置き換わることを明示する方法、あるいは、

```
&ref source="../proA/ProductAIntroduction.xml" str="Product A";
```

のように言えばエンティティ参照に属性を付けたような形が考えられる。こうすれば、何らかの実体、特に文字列に置き換わるものであることがより直感的に示されると思う。

● 属性値の問題

現在の XML 仕様では、属性値を囲む文字として二重引用符 (") とアポストロフィ (') の 2 種類が可能である。さらに、属性値内にタグ終了文字 (>) をそのまま記述することも許されている。これは、柔軟性が高いと言えるかもしれないが、たとえば正規表現で正確にタグを指定しようと思うと、この柔軟性があるためにタグの終わりを判定するのが(可能だが)容易でない。

「属性値は必ず'" で囲み、引用符を含めるには " を使用し、属性値内にはタグ開始文字 (<) だけでなくタグ終了文字 (>) も使用できない」と規定したとしても、文書作成上それほど不便ではないと思うのだが、どうだろうか。こうすれば、文字列検索時にタグは <[^>+> というシンプルな正規表現でマッチするようになる。さらには、多くの XML 関連ツールにおいても処理がよりシンプルになり、パフォーマンス上都合がよいはずである。技術翻訳では文字列検索を実行することが少なくないし、タグを検索に含めたいこともしばしばである。XML の将来のバージョンで検討されることを願うものである。

5. まとめ

マークアップ言語利用は確かに多くのメリットをもたらしたが課題もある。今後の発展に期待したい。