Modification and Rendering in Context of a Comprehensive Standards Based L10n Architecture

David Filip
ADAPT Centre at Trinity College Dublin
david.filip@adaptcentre.ie

Ján Husarčík
Moravia, an RWS company
jhusarcik@moravia.com

Abstract
The main goal of this paper and presentation is to explain – based on some practical examples – some basic concepts behind interoperable exchange and rendering of translation bitext (XLIFF). It explains how rendering of data and metadata provided by the underlying exchange format is critical for making the information available to language specialists performing the translation, editing, and review tasks in an effective way or indeed at all. The rendering concepts are covered in the context of a broader interoperability architecture based on the concepts of application programming interface (API), microservices, service oriented architecture (SOA), enterprise service bus (ESB), messaging, workflow, and service patterns, and canonical format. Solutions that are being developed and recommended within the GALA TAPICC initiative are investigated and explained.

1 Introduction
The translation and localization industry is necessarily fragmented. As usual, fragmentation is an effect of immaturity, but how come that the industry is immature now for some 30+ years? There are some inherent reasons for the industry to remain immature, some of those reasons are the overall societal megatrends of the 21st century in the context of information technology (IT) and in general. First, there is the technology adoption growth in the most linguistically diverse areas of the World. Second, there’s the information and content explosion, driven chiefly by the democratization of the Internet and the Worldwide Web, manifest in the rise in user generated content. Finally, there are two industry specific reasons, there is a very low barrier to entry and there is always a need for hyperlocal expertise, so that the effort can never be fully centralized, and the supply chain remains necessarily complex.

Despite all the above challenges, there are players in the industry who benefit from fighting the chaos and from building well thought through, interoperable, and mature solutions for industrial translation and localization. These efforts make use of established best practices from other industries and of industry specific standardization.

This paper and presentation explore different approaches of XLIFF "Modifiers", i.e. roundtrip Agents performing Modifications, to the changes of the XLIFF Documents, comparing their pros and cons and providing recommendations on selecting the most suitable approaches. We also describe how to render the information available in XLIFF documents in translation and editing tools to provide value to language specialists (i.e. translators and reviewers) to allow the use of XLIFF data and metadata in an optimal way. Most of the discussed concepts are accompanied by examples of recommended and discouraged practices.

---

¹ The current version of XLIFF is XLIFF 2.1 (Filip et al., 2018) which is backwards compatible with XLIFF 2.0 (Comerford, Filip et al., 2014) that it supersedes. XLIFF 2 is not backwards compatible with XLIFF 1.2 (Savourel et al., 2008) that is no longer maintained. ISO 21720:2017 - XLIFF (XML Localization interchange file format) (Comerford, Filip et al., 2017) is identical with XLIFF 2.0.
2 Interoperability in L10n Architectures

This section is an enterprise integration patterns\(^2\) crash course that is necessary for a translator or other type of non-technical industry stakeholder to understand – from the helicopter point of view – what hides behind his or her ability to receive a translation project, translate the content in scope, return it, and get paid because what they provided (and received) possibly through a series of middle men is still useful for the end user for which the localization customer ordered it in the first place.

Most of the current webpages are managed using Content Management Systems (CMS). There are thousands of these and only a handful of them were designed with some internationalization consideration, so you have systems that have to run separate instances for different languages, systems that don’t support Unicode encodings, systems with crazy proprietary formats that never considered that they might need to be extracted for translation. Many potential translation customers think, oh, we just give the translator the password to the system and let the m translate directly in the system. This is of course not sustainable and doesn’t scale. Professional translators need their professional tools and whenever there is more than one person involved in producing the final translated text, it is critical that the translator can access the job in the form of a bitext, i.e. a segmented and aligned artefact holding the content at the same time both in the source and the target language; interlinearly (traditional way) or tabularly (current best practice).

Of course, there are use cases during a multilingual production when only source or only target need be displayed. But the alignment must not be broken at any intermediate step lest the subsequent localization and translation transformations are made impossible or extremely costly by introducing the need of realignment.

The need to extract isn’t new, i.e. it didn’t first appear with the advent of CMS, albeit tool makers and service providers compete since time immemorial in their capability to handle zillions of native formats directly. The need to extract has been exacerbated by the rise of CMS; some formats such as rtf or binary software resources are declining but overall proliferation of CMS means more formats, albeit arguably better structured and sometimes better internationalized, so that the extraction can be easier, albeit with wild and unnecessary variations. The truth is, in any efficient industrial setting, there always is extraction. The tool maker or service provider may hide the extraction from the customer, and the customer is usually happy because they don’t want to know. The moment when they start to want to know is usually when they start to wonder.. Wait a minute, what’s this engineering you are charging me, I just gave you my files and you translated them, right? So, where’s engineering in it?

2.1 Enterprise Integration

The paradigms driving a modern-day Enterprise IT architecture are cloud and microservices. An API itself doesn’t guarantee an effective integration. APIs allow programmatic connections to be made, however it doesn’t say between what the programmatic connection should be. Every tool vendor, either on the CMS front or on the translation tool front offers an API on some terms.

An enterprise has many IT capabilities and so the concepts of service oriented architecture (SOA) makes sense. Capabilities are offered as services in the wider enterprise architecture. But this is pretty vague, so the related notions of microservice and service layer or service bus developed. Although you can offer and integrate many kinds of services, the ideal state to which an enterprise architecture aspires is to provide services that are as small as possible to

\(^2\) It all started with this seminal paper by Gregor Hohpe (Hohpe, 2002). See also (Hohpe and Woolf, 2015; Hohpe, 2016).
as many users as possible. So you can build very small and very specialized tools that each perform just one microservice and thus are easy to maintain, to manage their life cycle from requirements gathering, through inception, adoption, and productive use to rescission.

If you consider an enterprise with a large number of microservices, it becomes clear that you need a bus or layer that routes the service requests and probably also an API management tool as well as a service catalogue. The notions of a workflow and workflow token (the bitext or XLIFF Document in the localization industry) comes handy. In a generalized enterprise architecture, the messaging and service bus patterns play an important role. It is not realistic that a single canonical data format covers a whole enterprise architecture. The interoperability of service layer, messaging architecture, or an Enterprise Service Bus (ESB) relies on introducing an envelope format that wraps native formats. In this sense the messaging or ESB architecture are called native format agnostic, they ship and route whatever the various areas need routed and shipped. Service integrations and brokers in specific IT areas then benefit from exchanging data in a standardized or canonical format. The canonical format for exchanging translation data is necessarily a bitext format. Existence of a well designed canonical data format promotes the growth of the ecosystem and drives down integration cost. The only open standard bitext format is XLIFF, the only maintained open standard bitext format is XLIFF 2.

2.2 TAPICC

The Translation API Cases and Classes (TAPICC)\(^3\) initiative is a collaborative, community-driven, open-source project to advance API standards for multilingual content delivery. The overall purpose of this initiative is to provide a metadata and API framework on which users can base their integration, automation, and interoperability efforts. All industry stakeholders are encouraged to participate. The standard TAPICC relies on for bitext interchange is XLIFF 2.

TAPICC reuses some very important notions from the area of enterprise integration patterns, the most important one is the canonical message format. Canonical message format is what makes a messaging or ESB pattern effective in situations where many to many systems need integration. TAPICC Track 1 describes how to route an XLIFF Document through a supply chain, it defines a standard API model. It tells you what metadata and in what serialization you need to send along with your XLIFF payload in order to successfully exchange a translation project. TAPICC Track 1 describes only asynchronous exchange of project level payload. TAPICC Track 2 – just recently started – uses the XLIFF 2 data model but not in its traditional XML serialization. It uses the JLIFF format. The fact that both JLIFF and XLIFF are based on the same object model makes the two tracks semantically interoperable.

2.3 XLIFF

3 Manipulating XLIFF

In this section we discuss manipulation (Modification\(^4\)) of XLIFF Documents during the localization roundtrip, with focus on changes made by human language specialists within all kinds of editing (translation, editing, review etc.) environments such as Computer Aided Translation (CAT) tools or Translation Management Systems (TMS) and their associated workbenches or online editing environments.

---

\(^3\) See the TAPICC Charter (TAPICC Steering Committee, 2017).

\(^4\) See (Filip et al., 2018) http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#definitions
3.1 XLIFF Extraction and Merging Best Practice (XLIFF EMBP)

One of TAPICC deliverables, the XLIFF EMBP\(^5\) describes the most efficient way of Extracting translatable content along with metadata from various native file formats based on the real-world experience of the editors and other TAPICC volunteers. This deliverable passed its public review and will be donated by GALA to the OASIS XLIFF TC for publication as a TC Note.

Extraction processes have several dependencies that affect their outcomes and thus XLIFF features that will be effectively available to the translators. Albeit both human and machine translators are affected, this paper and presentation concentrates on effective and actionable display of human consumable data and metadata during the localization roundtrip.

3.1.1 Native Format

We call native format, the file or data format used for authoring of the source content that becomes subject to Extraction for localization purposes. We recognize native formats of varied expressivity. For instance, MarkDown and DocBook are very nearly on the opposite ends of the expressivity scale — the former being a lightweight, easy to write shorthand language, carrying the least necessary amount of information for conversion into a valid HTML document, while the latter is a feature-rich mark up language (based on XML) which can provide virtually any sort of metadata.

Another example is a Photoshop document, where one can extract not only the source text from the vector layers but also the additional resources (e.g. the raster layer) for rendering of in-context preview.

3.1.2 Content Type

The content type, such as user assistance, knowledge base article, marketing material, or user interface is another important factor that affects the design of Extraction processes. Specific content types can entail additional limitations, for instance size and length restriction limits for various types of user interfaces.

Making the translator aware of the maximum length of a string in the target language can prevent problems from occurring in the first place, thus preventing costly rework or bringing significant testing savings via automation.

3.1.3 Task Type

Adjusting CAT tool’s graphical user interface (GUI) to fit the needs of the task at hand helps the human language specialist to focus on the goal without being overwhelmed by redundant information while keeping all and just the necessary tools at hand.

Translators have different GUI requirements compared to proof-readers, editors, post-editors, spot-check reviewers (quality assessors), or even monolingual subject matter experts. Apart from the general task type, human specialists will have varied personal preferences also related to a specific phase of their task performance, also depending on issue types they are facing etc.

3.1.4 Enrichment

After Extraction, but typically before the manual transformation tasks (translation, edit, proof et al.) begin, mature translation systems would perform a variety of largely automated, process and metadata driven enrichment tasks that should add value for the human specialist working down stream.

\(^5\) (Filip and Husarčík, 2018).
The originally Extracted text can be re-segmented based on suitable rules\(^6\), followed by pre-translation from a translation memory, Machine Translation (MT) or other sources. Terminology identified, based on markup from the native format or by entity extraction, can be processed and referenced from the inline content.

### 3.1.5 Process

The localization process along with the tools being used affect the frequency and the granularity of the handoffs. Continuous localization aims for low volume and high frequency handoffs, which can limit the amount of information available to the translator (e.g. only a single UI resource from the whole dialog can be handed off at times), compared to a weekly handoff of newly written user assistance articles along with supporting multimedia.

### 3.2 XLIFF Rendering

XLIFF rendering means the transformation of data and metadata within an XLIFF Document into an interactive visual representation within a specific GUI. Rendering engines are key component parts of other human centric software such as Web browsers or CAT tools. User device characteristics, such as dimensions and display ratio of the available screen estate should be taken into account for the optimal User eXperience (UX). Typically, wide-screen devices are suitable for working with tabular layouts with multiple columns.

Tool specific features not directly related to the XLIFF standard, such as typing prediction, target string versioning, commenting, chat, instant messaging and other social features are out of the scope of this paper and presentation.

Some of the discussed features are well-known from old-school CAT Tools. Since the advent of cloud based TMS platforms with a Web browser GUI, the translator UX seems to be limited to the most basic functionality. Advanced translation centric rendering features are often neglected in favour of responsiveness and good looks.

A “Rendering Module”\(^7\) is one of the features proposed to be included in the next XLIFF version, XLIFF Version 2.2, which is currently in the inception phase. No version of the XLIFF standard so far has ever provided any rendering guidance. Rendering of the localization data and metadata seems to be one of the last vestiges of the undocumented tribal knowledge in localization. Rendering guidance would be beneficial not only for incumbent tool makers and translators, who’d benefit from better and more relevant data and metadata rendering. It would also lower the technology barrier for displaying and modifying XLIFF. The technology would become readily available to browser engine makers. It would become relatively easy to develop the necessary styling artefacts and contribute them to the major open source browser engine projects (Chromium Blink or Gecko).

Sample files for this presentation can be found in GALA TAPICC GitHub repository\(^8\). Texts used in the samples are:

---

\(^6\) Extraction and segmentation are two different processes — the former is a natural language agnostic engineering task that requires only the knowledge of the native format, while the latter depends on understanding the source natural language at least to the extent necessary to identify segment (often sentence) boundaries.

\(^7\) Current modules in XLIFF are basically specialized data models that are separated from the XLIFF Core by being in another XML namespace. It is not yet quite clear if the Rendering Module would require any new elements or attributes. More likely, this “module” will become a new vertical feature similarly to the advanced validation capabilities that were added as part of XLIFF Version 2.1.

\(^8\) [https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples](https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples). These examples are part of the previously cited TAPICC deliverable (Filip and Husarčík, 2018).

3.2.1 Rendering Modes

A handful of bitext rendering methods are common: tabular or interlinear for Modification, source or target only for context and preview purposes. Occasionally, the user might be presented with other types of visualization. The option to seamlessly switch between the four within one tool, or to customize the existing views is extremely rare.

3.2.1.1 Tabular Modes

Currently, the tabular layout is the most common one, supported by virtually all CAT tools on the market, with the source column on the left and the target column on the right. This follows the left-to-right reading direction common in the western culture. Additional details, such as segment number, state, and lock tend to be included. On the other hand, the possibility to customize the table columns, i.e. change the order of the columns, and add or hide them is rare.

Ideally, depending on the device, complexity of the text, and the role of the user, there should be at least two ways to render the table: the elementary, two column, layout with source and target only; and a complex one, providing detail about file structure, segment order, state, and translatability.

Maximum length of a single line with readable text (source/target/references/notes, etc.) should be limited on extremely wide displays to allow for comfortable reading experience.

![Simple Tabular Layout](image)

Figure 1 Simple Tabular Layout.

---

9 (Coleridge, 2018).
10 ("Internationalization and localization," 2018)
11 ("Internacionalizácia a lokalizácia," 2015)
12 (Franz, 2014).
3.2.1.2 Monolingual and Interlinear Modes

In addition to the above, source-only and target-only preview, and interlinear layouts should be available for in context editing.

![Complex tabular layout with file structure, segment order handles and metadata](image)

*Figure 2 Complex tabular layout with file structure, segment order handles and metadata.*

![Source only rendering](image)

*Figure 3 Source only rendering.*

![Target only rendering](image)

*Figure 4 Target only rendering. Engine could revert to source where target not available, displaying a warning, potentially using colour coding.*
Figure 5 Interlinear layout with source and target text alternating.

Note that in the above examples, the rendering is based on metadata provided by XLIFF’s Format Style Module\(^\text{13}\), not on a tool’s proprietary processing of the native format. This way Extractors can provide enough information for the rendering engine to transform the XLIFF Document using HTML markup regardless of the CAT tool.

### 3.2.2 File Structure Representation

For a complex document or User Interface (UI) localization, there is value in representing their structure to the user, rather than just displaying a flat table of source and target strings. There are multiple dimensions of the structure: Fragment Identification\(^\text{14}\); attributes name\(^\text{15}\), type\(^\text{16}\); and the Format Style module attributes.

The XLIFF fragment identification (fragid) mechanism let’s one address XLIFF Documents’ structural and inline elements using absolute and relative references. While the XLIFF specific fragid mechanism is different from the native XML fragid mechanism\(^\text{17}\) and the concepts could need some getting used to, once understood, it provides more value than a simple unit or segment number.

Attribute name has been designed to store identifiers of the original resources from the native format, providing additional context especially for UI localization.

Attributes type on structural elements can be used for custom values describing the native format.

The Format Style Module attributes can be used to annotate the XLIFF nodes using a subset of HTML 5 markup.

All of the above can be represented in the tabular view in the form of a hierarchical tree layout.

\(^{13}\) (Filip et al., 2018) [http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#fs-mod](http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#fs-mod).

\(^{14}\) (Filip et al., 2018) [http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#fragid](http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#fragid).

\(^{15}\) (Filip et al., 2018) [http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#name](http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#name).

\(^{16}\) (Filip et al., 2018) [http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#type](http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#type).

\(^{17}\) XLIFF fragid has business process driven scopes of uniqueness, hence it cannot reuse common XML fragid mechanisms. Documenting the fragid mechanism was one of critical prerequisites of registering XLIFF (.xlf) as a media type on the IANA Standards tree [https://www.iana.org/assignments/media-types/application/xliff+xml](https://www.iana.org/assignments/media-types/application/xliff+xml).
Naturally, it should be possible to reorder or hide the available columns based on the user’s preferences and needs, e.g. for a particular task type or work stage. UI localization analogy.
Note, that in order to maximize utilization of the available screen estate, XLIFF units containing only a single segment are rendered as omitting the segment level. The layout would have to adapt to display the segment level if a unit’s content were split.

3.2.3 Segment Representation

As shown in Figure 7 or Figure 8, a single <segment> or <ignorable> inside an XLIFF unit can be rendered as omitting the transient segment level. Since a unit can contain multiple <segment> and <ignorable> nodes, a tree structure is more suitable (see Figure 2).

This helps to preserve the integrity of units, as the segmentation can only be changed within one unit. Scope of bulk operations is also easier by allowing the user to seamlessly work with groups of segments.

3.2.3.1 Segmentation Modification

As discussed in 3.1.4, the Extractor does not need to perform segmentation¹⁸, because segmenting requires knowledge of the natural language to a certain extent. As such, the Extractor can produce units with only one segment corresponding for example to a whole paragraph or text.

![Figure 9 Unit with a single segment corresponding to the whole stanza](http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#segmentationModification)

Such units can be manually segmented, which is a common feature of CAT tools, albeit not all of them respect the XLIFF unit boundary. The more valuable it would be for the translator to see the unit boundaries in the tree view, preventing merging failures due to re-segmenting further in the process.

![Figure 10 Result of manual segment splitting](http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#annotations)

Another frequent issue is joining of previously split segments — while it’s desirable to preserve original segmentation, it’s not always possible within the tool’s UI. XLIFF can facilitate this by using annotations¹⁹.

---

¹⁸ (Filip et al., 2018) [http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#segmentationModification](http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#segmentationModification)

¹⁹ (Filip et al., 2018) [http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#annotations](http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html#annotations)
3.2.3.2 Segment Order Modification

It’s often desirable to change the order of the translated segments within a unit, for instance in case the strings need to be alphabetically ordered. XLIFF provides the order attribute to store such changes. CAT tool’s editor GUI can facilitate such segment reordering in two ways: add a handle next to the target string for the user to drag and drop segments into the expected order; or by providing a segment order index in form of a dropdown, the user can select from. The CAT tool has to observe the Constraints and Processing Requirements of the XLIFF specification to preserve integrity of the document.

3.2.4 String Translatability

XLIFF specification provides several ways to control translatability of the extracted text that are fit for different purposes:

- the translate attribute on structural elements;
translate annotations, and the <ignorable> node.

Locked structural elements and ignorable, if ever presented to the user in the CAT tool GUI, are usually represented with a simple lock icon and cannot be Modified during the translation. Thus, these are also usually excluded from the quality control.

What the tools seem to struggle with is the translate annotation. While some correctly protect the substring; others hide it completely from the user, preventing them from accessing the context; or fail to protect the string from modification; even more so in case of annotations spanning across segments.

```
<unit id="u2">
  <segment>
    <source>Wikipedia's article about <mrk id="1" translate="no" type="term">i18n</mrk> and <mrk id="2" translate="no" type="term">l10n</mrk> first paragraph in Slovak reads:
    <sm id="3" translate="no"/>V informatike je Internacionalizácia a lokalizácia ... kultúry.<source>
  </segment>
  <segment>
    <source>Internacionalizácia ... písma.</source>
  </segment>
  <segment>
    <source>Lokalizácia ... jednotky.<em startRef="3"/> [shortened]</source>
  </segment>
</unit>
```

**Code Snippet 1 XLIFF fragment with translate annotation**

![Figure 14 Spanning translate annotation correctly represented in the editor's UI](image)

The annotation should be preserved even as the segmentation changes.

### 3.2.5 Inline Elements

#### 3.2.5.1 Spans

XLIFF lets the Extractor represent various inline codes from the native format, the most familiar being formatting, for instance *italic* or **bold**. Formatting can span across segments. Well-formed codes can be represented using `<pc>` and `<sc>`/`<ec>` interchangeably (with Constraints). CAT tools, however, do not offer a consistent user experience for these two

---

20 (Filip et al., 2018) [http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#translateAnnotation](http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#translateAnnotation).
alternatives. While the pairing of <pc> tends to be preserved both visually and in the background, for <sc/>/<ec/> it’s not always the case.

Using the first paragraph as the sample, if the first segment used <pc> and the second one <sc/>/<ec>, they would usually use carets and squares respectively, with index numbers.

![Common depiction of pc and sc/ec in CAT tools](image)

Had the formatting started in the first segment and ended in the second, the rendering would be even less desirable.

![Formatting spanning across segments.](image)

Note, how the role of the code is usually understood within a segment, this is seldom the situation for codes spanning more than one segment.

![Expected rendering](image)

![Expected rendering, more complex example.](image)

Interpreting the role of the inline element is a related topic. XLIFF provides the type and subType attributes. The XLIFF-defined subtypes xlf:lb, xlf:pb, xlf:b, xlf:i, and xlf:u are associated with the type fmt, while the subtype xlf:var is associated with the type ui. These can be easily interpreted and should be fully supported by any CAT tool.

### 3.2.5.2 Metadata representation

XLIFF inline codes can contain various metadata, resulting in lengthy plain-text representation.

```xml
<ph dataRef="1" disp="[User Name]" equiv="{id}" id="1" subType="xlf:var" type="ui" canCopy="no" canDelete="no" />
```

A string, such as this one can obscure translator’s perception of the segment text, even more so if there are multiple inline codes present. For this reason, CAT tools usually offer at least
two display modes for inline elements: full text and Id only. The issue with displaying just the Id is that it can (and often will) hide important context information or possible restrictions.

While the info could be still available as a tooltip, it requires the translator to move their hand from the keyboard to the mouse and put the cursor over the tag of interest. Such an action takes valuable time and is less than ergonomic.

For this reason, it’s beneficial if the tool can also provide a partial text mode, which displays only the necessary info. It can provide the value of the disp attribute that has been specifically designed to provide the display equivalent of a tag.

![Figure 19 Rendering partial tag text.](image)

Additional info, for instance that the element cannot be removed in target text, can be visualized by modifying the shape of the tag icon. It’s preferable to colour coding to accommodate accessibility needs of translators with colour vision deficiency.

### 3.2.5.3 Hotkeys

User Interface (UI) translation is also specific due to the presence of the hotkeys. These are usually encoded using the `<ph/>` inline element. Relying on CAT tool’s default behaviour for rendering of `<ph/>` can lead to issues with translator’s User eXperience (UX) and readability of the text.

![Figure 20 Default behaviour for ph](image)

Ideally, the tool should mimic the traditional behaviour of UI for hotkeys — underline. This approach makes the editing interface less cluttered and easier to understand.

![Figure 21 Interpreting ph as a hotkey placement](image)

### 3.2.6 Metadata

The XLIFF Standard allows the *Enrichers* and *Modifiers* to embed additional data and metadata into the document during the localization roundtrip. This information can be stored in both the core and various modules.

One of the modules is the ITS Module, which defines a subset of data categories available in the W3C ITS 2.0 Recommendation: Allowed Characters, Domain, Locale Filter, Localization Quality Issue, Localization Quality Rating, Provenance, Text Analysis. All of these categories could be populated within a single XLIFF Document. Had the tool presented the information to the user within a single view, the amount of information would be overwhelming. Other modules are not being considered in this paper and presentation for this use case.

---

21 Also known as keyboard accelerators.

22 (Filip et al., 2013)
For this reason, the translator should be able to select the scope of categories they are interested in.

![Figure 22 Selecting available ITS Data Categories. Multiple items can be selected.](image)

### 3.2.6.1 Text Analysis

The Text Analysis data category is used to annotate content with lexical or conceptual information for the purpose of contextual disambiguation. This information can be provided by so-called text analysis software agents such as named entity recognizers, lexical concept disambiguators, etc., and is represented by either string value or IRI references to possible resource descriptions.

While text analysis can be done by humans, this data category is targeted more at software agents. (Filip et al., 2013) [https://www.w3.org/TR/its20/#textanalysis](https://www.w3.org/TR/its20/#textanalysis)

![Figure 23 XLIFF Document with ITS metadata. View focused on Text Analysis data category.](image)
Various recognized entities can provide additional context to translators. Such information could potentially improve translation quality and save time on investigation, mainly for documents containing proprietary information.

To prevent visual clutter, only the abstract of the information is available. Additional details can be accessed via the tooltip and its link to the data source. To minimize mouse usage, entities should be also made accessible via numbered hotkeys.

To further improve UX, the rows are collapsed by default with the exception of the active segment and those manually expanded by the user.

### 3.2.6.2 Localization Quality Issue

The Localization Quality Issue data category is used to express information related to localization quality assessment tasks. Such tasks can be conducted on the translation of some source content (such as a text or an image) into a target language or on the source content itself where its quality may impact on the localization process.

(Filip et al., 2013) [https://www.w3.org/TR/its20/#lqissue](https://www.w3.org/TR/its20/#lqissue)

![Figure 24 XLIFF Document with ITS metadata. View focused on the Localization Quality Issue data category.](image)

Localization quality issues can be created for the whole segment, its substring, or even multiple overlapping substrings, even spanning segments within a unit. All these cases can be expressed and stored within an XLIFF Document and should be therefore supported by the review environment with proper rendering.  

23 W3C ITS Localization quality issue encoding in XLIFF is also critical for transmitting Multidimensional Quality Metrics (MQM) 24 data effectively and in context.

### 3.2.7 In-Context-Preview

As discussed in 3.1.1, Extractors can process additional metadata based on their knowledge of the native format. Translation of strings within pictures can serve as an example. Provided the native format is capable of storing text data in separate layers that can be processed by the

---

23 (Bikmatov et al., 2013) described an attempt of early ITS 2.0 implementers to provide a preview of the ITS markup within a browser window, while the language specialist would continue to work in a separate CAT tool.

24 MQM is currently dual housed at ASTM Committee F43 [https://www.astm.org/COMMITTEE/F43.htm](https://www.astm.org/COMMITTEE/F43.htm) and a W3C Community Group [https://www.w3.org/community/mqmcg/](https://www.w3.org/community/mqmcg/)
Extractor, for instance Photoshop documents; the layer location, dimensions, font face, font size, and decorations, as well as the base picture layer can be Extracted into an XLIFF Document along with the strings.

This data is stored within the Size and Length Restriction (SLR) and the Resource Data modules. Once present, the CAT tool capable of processing the modules can not only validate, whether the target string does not violate the imposed limits, but also render the in-context preview.

The preview can help the user better understand the layout and any information conveyed by the picture itself.

An XLIFF Document with SLR and Resource Data modules opened in a web browser can be transformed into a meaningful visual representation of the background raster layer (the picture), overlaid with vector layers rendered in the correct position, with correct font face and font size. The translator can alternate between the source and target nodes using the radio buttons.

4 Conclusion

We argue for the creation of effective and efficient GUI and UX for human language specialists based on available open standards and based on new development proposals within the existing localization and translation standards ecosystem. We design and propose through various standardization venues, such as the OASIS XLIFF and XLIFF OMOS Technical Committees or the GALA TAPPIICC pre-standardization project, an ecosystem of highly interoperable, transparently documented, and easy to use technologies that will bring a

consistent and user friendly experience to tech savvy and less so inclined translators and other language specialists, providing them with all and just the necessary information and context as per their personal and task driven preferences.

The authors intend to collect additional feedback on the above introduced XLIFF Rendering Module proposal from relevant constituencies and communities such as the ASLING Translating and the computer conference.

Acknowledgements
This research was supported by the SFI ADAPT Centre at the Trinity College Dublin (SFI Research Centres Programme Grant 13/RC/2106) and its targeted Spokes project with Moravia, an RWS company.

References
Coleridge, S.T., 2018. THE RIME OF THE ANCIENT MARINER; PŘÍBĚH PRASTARÉHO MOŘEPLAVCE.
Filip, D., Husarčík, J. (Eds.), 2018. XLIFF 2 Extraction and Merging Best Practice, Version 1.0 [prd01], BP. Globalization and Localization Association (GALA).
TAPICC Steering Committee, 2017. TAPICC Charter.