XLIFF 2.0 and the Evolution of a Standard

Chase Tingley
Spartan Software, Inc
chase@spartansoftwareinc.com

Abstract

This paper is a write up of the FEISGILTT 2014 keynote on whether or not XLIFF 2.0 was a successful evolution of the XLIFF 1.x standard. This offers a higher level perspective on the chances that XLIFF 2.x is to become the common denominator of the localization industry. This is an honest and clear discussion of how XLIFF 2.0 has learnt from XLIFF 1.2 issues, which also suggests that the XLIFF 2.0 object model has potential to influence how localization data and metadata are exchanged at lower granularity through webservices, based not only on XML.

Keywords: *Interoperability, Standards, Feature Creep, Adoption, XLIFF*

"I find it rather puzzling," a technologist friend wrote to me an email, "that this small industry has such difficulties designing robust standards."

I agreed with his assessment.

As a consulting engineer in the localization industry, I spend a lot of my time developing solutions to help clients stitch together different pieces of technology into one coherent tapestry of automated process. In the course of this work, I spend a lot of time using — and complaining about — the various standards that define interactions across tools in our industry. In particular, I've spent a lot of time dealing with XLIFF 1.2 (Savourel et al; Eds.; 2008), a standard which I frequently utilize and almost as frequently find wanting.

XLIFF 1.2 is not a convincing standard, despite good intentions and flexible approach allows it to capture data for many different use cases. It suffers from a lack of interoperability between different implementations, characterized in particular by an inconsistent implementation of its feature set. Even mutually supported features are not always implemented in ways that are mutually intelligible. While the basic feature set can be relied upon, it is difficult to use the advanced features of XLIFF 1.2 across a tool chain containing more than than one or two different implementations.

With the recent finalization of the XLIFF 2.0 standard (Comerford et al; Eds.; 2014), it is worth taking another look at XLIFF 1.2, as well as the industry as a whole, and try to examine why producing a reliable interchange format for localization data is difficult. Using that analysis, we can look again at XLIFF 2.0 to see how it avoids pitfalls encountered in the past.

1. Difficulties in the Localization Ecosystem

1.1 Supply Chain Complexity

As an industry, localization is famously decentralized. Even a simple translation process may involve several parties, as a vast web of individual translators perform work for small local vendors, who in turn are hired by larger intermediaries who aggregate work across multiple languages. For a large customer engaging with multiple translation suppliers, the supply chain contains a large number of links, across which localization data and process information must be transmitted intact.

The problem of reliably communicating structured information through this complex chain has been compounded by the ambition of XLIFF itself. In addition to carrying enough data to serve as an interconnect, XLIFF 1.2 was designed to carry high-level process data from one end of a tool chain to another. This increased complexity by expecting different tools and organizations to agree on how and when this metadata should be processed. Any inconsistency in the tool chain in how these values are used and consumed could render the metadata incorrect or useless.

1.2 Competing Design Objectives

In order to serve this complex ecosystem, the development of XLIFF has been impeded by a bevy of competing design objectives. As with any standard, there is an inherent tension between a simple standard, which is easy to understand and implement, and a complex standard, which can provide additional features. The fluid nature of the localization industry also creates a tension between

the need for rigid standards, which provide strong guarantees of interoperability, and flexible standards, which allow for extension and customization. Similarly, it has been difficult to balance a descriptive approach which canonicalizes features of existing implementations against a prescriptive approach which demands certain functionality for implementing tools.

1.3 Why do Standards Fail?

The failure of standards is not a problem unique to the localization industry. In 2011, Carl Cargill, a Standards Principal at Adobe, published a paper called "Why Standardization Efforts Fail" (Cargill 2011) that analyzed common factors across industries that led to the unsuccessful development and adoption of standards. These covered all stages of standards development, including conceptual, developmental, and implementation failures. Of the six major categories Cargill identifies, three are of particular interest when thinking about XLIFF 1.2.

1.4 Feature Creep

"Feature Creep" in the standards world is analogous to its meaning in engineering: the standard contains such an abundance of functionality that it loses focus and becomes difficult to implement. Cargill cites an over-reliable on compromise by the standards body as one of the chief causes of feature creep in standards.

Whether or not compromise was in fact at fault, there is clear evidence of feature creep in XLIFF 1.2. Some concepts, such as inline tags, have multiple representations when fewer would have sufficed. Some pieces of metadata, particularly related to process information, have no clear semantics and do not identify the problem they are meant to solve. And some inclusions, particularly related to software localization metadata, seem unrelated to the rest of the specification.

The impact of this has led to a large set of functions without any clear guidelines from the specification about what is truly necessary. Tool vendors have tended to implement arbitrary subsets of the available feature set, as documented by Micah Bly (Bly 2010), among others.

1.5 Incompatible Implementations

The problem of incompatible implementations is simple to understand, but its causes can be subtle. In addition to the incompatibility that results from partial implementations of the standard, Cargill also recognizes ambiguity and omission in the language of the standard itself as a critical problem. Developers, he argues, prefer to pick the simplest solution that can be labelled "standard-compliant", and so any linguistic wiggle room that they find may be exploited.

Incompatible implementations may be the single greatest problem with XLIFF 1.2. It is generally understood that only the barest subset of XLIFF 1.2 features can be interchanged and correctly processed across a heterogeneous tool chain. Consistent implementation of even common features like the use of the alt-trans element for memory proposals can be difficult to achieve without tightly controlling the set of tools that are allowed to touch the file.

The causes for this incompatibility across XLIFF 1.2 implementations are varied, and extend beyond simply an inconsistent feature set. The overloading of some features, such as the `<alt-trans>` and `<mrk>` elements, led to confusion amongst tool vendors, many of whom support these features only partially. The ambiguity in the plain language of the standard that Cargill cites is also present: does the match-quality attribute support decimal points? Does it require (or allow) the presence of a percent sign? An over-broad extension mechanism allowed for the development of many tool-specific variants of the format that further hinder interoperability.

Perhaps most importantly, the standard provides little assistance in proving that a given implementation is correct or incorrect. XLIFF 1.2 lacks processing expectations, a set of test cases to verify correct operation, or even a set of requirements for compliance. The meaning of "processing XLIFF 1.2" is left up to the implementor.

1.6 Market Indifference

Cargill describes market indifference as a situation where a standard is completed, but not widely adopted in the market. Most commonly, this happens when the market has already moved on to another solution to the same problem, and no longer has a need of the standard.

For XLIFF 1.2, this has not been a problem. XLIFF 1.2 has been, and continues to be, widely supported, albeit in inconsistent ways. The more interesting question is whether market indifference could hamper the adoption of XLIFF 2.0.

There are several factors working against XLIFF 2.0.

The size and complexity of the standard may make tool vendors reluctant to invest the engineering resources to support it when the market for it is not already developed; this could create a chicken-and-egg situation. XLIFF 2.0 is also not backwards-compatible with XLIFF 1.2, which provides a migration hurdle for existing implementations. Lastly, the rise of different forms of data exchange, in particular web services, provide alternatives to XLIFF for the exchange of localization data.

2. Can XLIFF 2.0 Learn from Past Mistakes?

The development and standardization of XLIFF 2.0 spanned several years of effort by the OASIS XLIFF Technical Committee, and they made a conscious effort to address many of the shortcomings of XLIFF 1.2. How do the changes in the new version of the standard work to avoid the problems seen in the past?

2.1 Feature Creep

XLIFF 2.0 streamlines many aspects of XLIFF 1.2, but it also adds several sizeable new features, including the ability to embed terminology data and enforce size and length restrictions on content. Overall, the number of available features has increased. Will this exacerbate the complexity problems found in the previous version?

I am optimistic that it does not, thanks in large part to the addition of the module mechanism in XLIFF 2.0. Modules are a meta-feature, a system for organizing other aspects of the XLIFF format into functional clusters that must be implemented or ignored as a whole.

This change has a profound effect on the overall complexity of the standard. Although the number of elements has increased, whether an implementation now supports a given "feature" is now a discussion of whether or not it supports a "module", a much coarser distinction. This simplifies compatibility discussions and helps developers prioritize within their implementations. Additionally, modules provide a regular way for implementations to *not* support a feature, thanks to the requirements regarding the handling the markup of unsupported modules. Restrictions against re-implementation of core features provides an important check against abuse.

Additionally, the new functional capabilities of XLIFF 2.0 generally reflect conventional wisdom regarding things that were missing from XLIFF 1.2, rather than an attempt to unify disparate implementations from existing tools. For example, the lack of terminological support in XLIFF 1.2 is well-known, which has led to the common practice of bundling a standalone TBX file (or other term format) with XLIFF.

2.2 Consistent Implementations

One of the most noticeable changes when reading the XLIFF 2.0 specification is its size — it's much longer than the previous version. To a developer, the XLIFF 1.2 specification is terrifyingly slim, and this more verbose style is a welcome change. The Technical Committee has consciously focused on improving the number and quality of available examples in the text, as well as clearly describing processing instructions for a number of features.

Consistency of implementation should also improve thanks to the approach taken with some of the more problematic features from XLIFF 1.2. The functionality of `<alt-trans>`, which was used both as match proposals and as a form of segment history, has been split into two separate features in dedicated modules, Translation Candidate and Change Tracking.

2.3 Pushing for Success

Even with an improved specification, widespread acceptance of XLIFF 2.0 is no sure thing. The Technical Committee, as well as parties that wish to further the spread of XLIFF 2.0, will need to make a concerted effort to drive its adoption.

It is vital to push for the implementation of the XLIFF 2.0 core in as many places as possible, as a stepping-stone towards more advanced functionality. Open source can be a valuable tool for new standards, as it allows for other implementers to quickly embed the functionality and build on it. Investing in high-quality open source implementations, such as the Okapi Framework, should be a priority.

Translation buyers also have an important role to play in the adoption of any standard, should they so choose, as they ultimately have the most to gain from improved interoperability between tools and among vendors.

It is also worth considering the applicability of the XLIFF model to other scenarios where it has not traditionally been applied. XLIFF is an XML-based, document-oriented format, but an industry focus on web services is increasingly dealing with translation

granularity smaller than a document. A web service may exchange a segment or small group of segments, and may use a non-XML format such as JSON to exchange the data. These formats are simpler, but also encode far less information; many of them ignore inline markup entirely. An abstract version of the XLIFF 2.0 data model would be valuable for these implementations, by providing a set of common structures for exchanging rich segment data.

3. Conclusion

Although technology and business demands change, there is little evidence that the basic structure of the localization industry is poised for imminent change. Translation and its associated activities will continue to depend on a multitude of organizations and individuals spread across the globe. The challenges in tying this web together in a way that satisfies the requirements of modern translation buyers is serious.

I believe, however, that we are moving in the right direction. From an implementor's perspective, XLIFF 2.0 offers clear technical benefits over its predecessor that both strengthen the standard and should address some of the critical interoperability problems that weakened its predecessor. In many ways, the next challenge is non-technical, as the XLIFF community pushes for the broad adoption of the standard.

References

Savourel, Y., Reid, J., Jewtushenko, T., Raya, R.M. (Eds.) (2008) *XLIFF Version 1.2* [online], OASIS Standard. ed, Standard, OASIS, available: http://docs.oasis-open.org/xliff/v1.2/os/xliff-core.html [accessed 10 Mar 2015].

Comerford, T., Filip, D., Raya, R.M., Savourel, Y. (Eds.) (2014) *XLIFF Version 2.0* [online], OASIS Standard. ed, Standard, OASIS, available: http://docs.oasis-open.org/xliff/xliff-core/v2.0/os/xliff-core-v2.0-os.html [accessed 10 Mar 2015].

Cargill, C.F. (2008) "Why Standardization Efforts Fail", *The Journal of Electronic Publishing* [online], 14(1), available: http://dx.doi.org/10.3998/3336451.0014.103 [accessed 10 Mar 2015].

Bly, M. (2010) "XLIFF: Theory and Reality" [online], presented at XLIFF Symposium Limerick, University of Limerick, 22 Sep 2010, available: http://www.localisation.ie/oldwebsite/xliff/resources/presentations/xliff_symposium_micahbly_20100922_clean.pdf