

Next Generation XLIFF

Simplify - Clarify - and Extend

Asgeir Frimannsson , Red Hat
Christian Lieske, SAP AG

1st XLIFF Symposium
Limerick, 22 Sep 2010

Agenda

Next generation XLIFF presentation

- Preliminary for the panel discussion

Coffee break

Panel Discussion on minimal and modular XLIFF

About the Presenters

Asgeir Frimannsson

Red Hat

- Senior Software Engineer at Red Hat, within the Internationalization Group
- Now based in Oslo, Norway
- Developing Java-based open source tooling for L10N
- PhD candidate at QUT, Brisbane, Australia, supervised by A/Prof James M. Hogan
- L10N interests, Okapi contributor, nocturnal XLIFF TC member

Christian Lieske

Globalization Services
SAP AG

- Knowledge Architect
- Content engineering and process automation (including evaluation, prototyping and piloting)
- Main field of interest: Internationalization, translation approaches and natural language processing
- Contributor to standardization at World Wide Web Consortium (W3C) OASIS and elsewhere
- Degree in Computer Science with focus on Natural Language Processing and Artificial Intelligence

Why Red Hat doesn't use XLIFF 1.x

Ambiguous and overly broad

- Overly complex, bloated
- No defined processing expectations

Poor interoperability between tools

Poor or non-existing open source tool support

- Virtaal notable exception

We do limited work on a minimal subset of XLIFF

*“XLIFF is not really a standard – but
a file format developed within a
standards body ...”*

“... but we can fix this for XLIFF 2.0”

What's Behind XLIFF's Complexity?

XLIFF's comprehensiveness/complexity derives from its large scope: XLIFF covers a broad range of dimensions related to the process of localizing/translating content.

Sample dimensions are the following:

- generic representation of inline markup
- support for linguistic processing like segmentation
- support for internationalization (language identifiers, markers for non-translatable content, notes for process participants)
- rich meta data on the level of strings (example: length restrictions and string type)
- rich meta data to log workflow events
- extensibility features

Another way of looking at this is the following: XLIFF covers several domains or layers. Most notably, XLIFF covers an extraction domain and a translation domain.

XLIFF Promises

Common Container

- Enable processing of various file formats through a shared XLIFF data model
- Allow – through extensibility features – format-specific processing (e.g. native previews, rich editing) for various file format without breaking interoperability across tools

Interoperability

- Enables different tools to modify the XLIFF data structure in ways that can be validated and does not break interoperability

Processing

- Interoperable tools process XLIFF for a specific purpose or workflow step
 - Mark up spans of content within a translation unit for a specific purpose (e.g. Segmentation, Terms, Protected text)

Why doesn't the promises come true?

1. Too few implementations

2. Implementation covers only a subset of XLIFF

3. Implementation handles XLIFF incorrectly

As a consequence of 2 and 3, interoperability and exchange possibilities are limited.

We love XLIFF!

..but we'd like to see the promises
fulfilled

Mantra for Next Generation XLIFF

Simplify

- Make it dead easy to get started creating or manipulating a minimalistic XLIFF file
- Make it dead easy to create tools that annotate or process XLIFF files
- Make the standard understandable for your process

Clarify

- Clear processing requirements
- Provenance of data categories and values – Where from and Why?

Extend

- New Requirements
- Format specific data and processing
- Re-use existing standards

Making it Happen

Options for Reduced Complexity and Easier Implementation

1. Realize that XLIFF does not need cover all of the aforementioned dimensions in a certain context.

- Think of layers/domains.

2. Shy away from reinventing the wheel. Reuse!

3. Acknowledge that new approaches are available for describing resources/providing meta data

Approaches to Next Generation XLIFF (1/2)

The why defines
the how

- Lower threshold for adaptation
- Less steep learning curve
- Decrease efforts for implementation
- Ease of use
- Facilitate checking of conformance

The status quo
defines the how

- What's optional/mandatory
- Categories (e.g. inlines) define the modules

Some New Ideas
are allowed

- Make pointer/access to original mandatory (in order to allow use in certain workflow steps (eg. rendering))

Approaches to Next Generation XLIFF (2/2)

Top-Down

- Think about domains
- Think about data categories
- Discuss on macro layer

Bottom-Up

- Think about most frequently used XLIFF constructs
- Discuss on micro layer

Implementation aspects (eg. new schema) and definition of „minimal“ a separate story ...

Framework/Priorities

Backwards compatible wherever possible?

Cover new requirements (eg. advance leveraging involving MT)?

Introduce new ideas

- Allow access to original, or store copy (similar to skl)?
- Turn to Resource Description Framework (RDF)?

Solution Approach Proposal

Look at domains

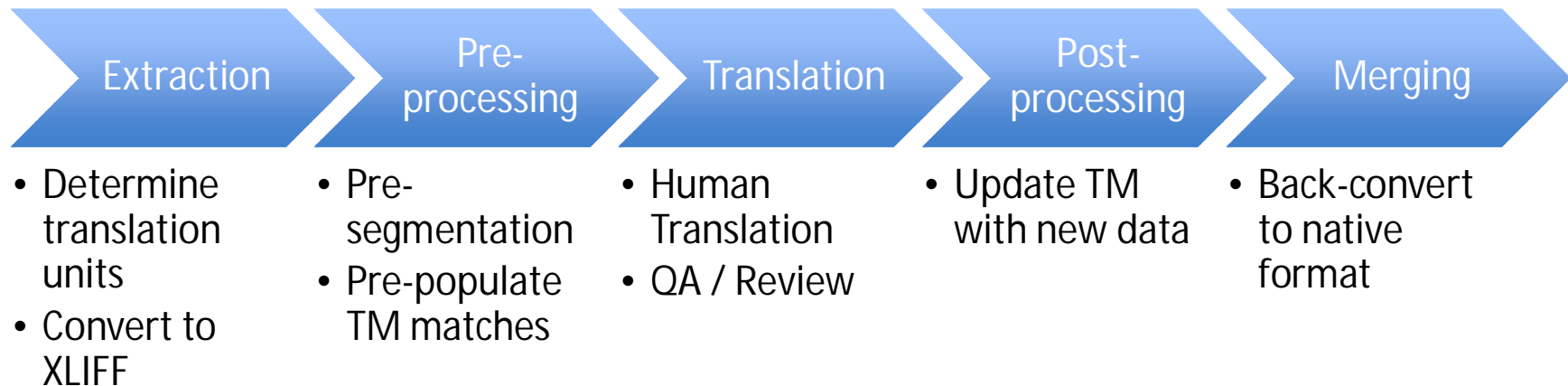
Determine framework

Select data categories

Domains/Concerns/Feature Sets (1/3)

Do not think about XLIFF in terms of XML elements and attributes

Determine how various tools/actors manipulate XLIFF through the various stages of a localisation process



Domains/Concerns/Feature Sets (2/3)

Extraction/Filtering

- (including reverse ie. insertion/merging)

Constraint Setting

- Determine size requirements
- Determine character set requirements

Internationalization

- Mark "not translation relevant"
- Create note

Automated Linguistic Processing

- Segmentation
- MT & TM Leveraging
- Terminology identification/extraction
- Automated linguistic checking (spelling, grammar, style, terminology)

Human Translation

- Creation of target text
- Display, creation or annotation of notes
- Special operations (eg. cloning)

Localization

- Adaptation of viewports
- Adaptation of URLs

Domains/Concerns/Feature Sets (3/3)

Reviewing

- Insertion of annotations
- Display, creation or annotation of notes

Inclusion of reviewing results

Workflow Events

- Status updates

Tool-specific Events

- Trigger rendering

Technical QA checks

- Missing "target" after translation
- Check size requirements
- Check character set requirements

Frequently Used Data Categories

Payload (ie. „source“ and possibly pre-filled „target“)

String length constraints (min. or max. length)

Resource type (e.g. different User Interface controls – label, button, ...; overlap with internationalization)

Inlines („ph“, „x“, ...)

Identifiers (for processing)

Names (ie. the identifier used in the native format – key in a Java property file ...)

Notes (e.g. explanations and other annotations; overlap with internationalization)

Internationalization (e.g. „translate“)

Domain/subject area

Relationships (e.g. between strings belonging to the same User Interface menu)

Creation (e.g. generator, and creation date)

Enrich with "Classic" Resource Description Framework

```
1 <!DOCTYPE xliif:xliif [
2   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
3   <!ENTITY vcard "http://www.w3.org/2001/vcard-rdf/3.0#">
4   <!ENTITY mime "http://www.iana.org/assignments/media-types/">
5   <!ENTITY domain "http://www.sap.com/mlt/domain/1.0#">
6   <!ENTITY collection "http://www.sap.com/sls/collection/1.0#">
7   <!ENTITY ppms "http://www.sap.com/ppms/1.0#">
8   <!ENTITY lang "http://www.sap.com/mlt/lang/1.0#">
9 ]>
10 <xliif version="1.2" xmlns="urn:oasis:names:tc:xliif:document:1.2" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
11   <file original="hello.txt" source-language="en" target-language="fr" datatype="plaintext">
12     <header>
13       <!-- We use RDF to describe meta data, and work with existing vocabularies/namespaces (see
14         http://dublincore.org/documents/dcmi-terms and http://www.w3.org/TR/vcard-rdf) -->
15       <rdf:RDF xmlns:dc="http://purl.org/dc/terms#" xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:v="
16         http://www.w3.org/2001/vcard-rdf/3.0#">
17         <rdf:Description rdf:about="http://dtr2010/DasExecutionCompView.wdview.xlf">
18           <dc:identifier xml:lang="en-US" rdf:datatype="xsd:string">XSR Test Object 1</dc:identifier>
19           <dc:format rdf:datatype="&mime;">application/xml</dc:format>
20           <dc:language rdf:datatype="&lang;">en</dc:language>
21           <dc:collection rdf:datatype="&collection;">TEF Collection</dc:collection>
22           <dc:subject rdf:datatype="&domain;">BC</dc:subject>
23           <dc:creator rdf:nodID="creator"/>
24           <dc:created rdf:datatype="xsd:date">2010-03-22T14:49:19+01:00</dc:created>
25           <dc:title xml:lang="en-US" rdf:datatype="xsd:string">A test file</dc:title>
26           <dc:description xml:lang="en-US" rdf:datatype="ppms;">Edition/version/release information (e.g. related to PPMS)</
27         </rdf:Description>
28         <rdf:Description rdf:nodID="creator" rdf:type="vcard;">
29           <v:UID>d025418@sap.com</v:UID>
30         </rdf:Description>
31       </rdf:RDF>
32     </header>
33     <body>
34       <trans-unit id="hi">
35         <source>Hello world</source>
36       </trans-unit>
37     </body>
38   </file>
39 </xliif>
```

Enrich with RDFa Attributes

```
<file xmlns:dc="http://purl.org/dc/elements/1.1/">  
  <author property="dc:creator">Alice</author>  
  ...  
</file>
```

This allows you to use XLIFF pretty much as today, but you provide a bridge into other meta data worlds (for example Dublin Core). In addition, you enter the world of the Semantic Web since many tools exist for technologies such as RDF. In addition, you can already reap benefits such as improved search results: Google Rich Snippets makes use of RDFa during crawling.

Modules of Data Categories

