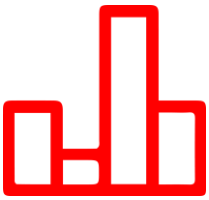
Lessons learned — XLIFF2 extraction and merging

www.moravia.com

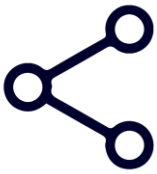
MORAVIA

Introduction

> Examples originate from real-world experience
> Focus is mostly on well-formed HTML and XML as native format
> Existence of native format's XML schema is presupposed
> Files are available in TAPICC GitHub
> Examples are discussed by thematic groups without particular order



Inline codes



Recommendations for processing standalone and spanning inline functional and formatting elements of localizable content.

> Perform complete extraction

> Represent spanning code using <sc> and <ec> (or <pc> where possible)

> Represent standalone code using <ph>

> Include inlines in extracted content

> XLIFF2 prose

M

- **Using <ph> to represent spanning code**
  - [spanning_as_ph] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/spanning_as_ph
  - Extractor could use knowledge of schema and only use does not use <ph> for codes that are declared as EMPTY. To further help the extraction process, following W3C recommendation could be followed: „The empty-element tag *SHOULD* be used, and *SHOULD* only be used, for elements which are declared EMPTY.“ (https://www.w3.org/TR/REC-xml/#sec-starttags), e.g. even without content would use as compared to
.
  - https://issues.oasis-open.org/browse/XLIFF-14
  - http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#ph
- **Excluding outermost tag pairs**
  - [outermost_inline_excluded] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples

/outermost_inline_excluded

- Both functional and formatting inline codes provide additional context for translator and could be linguistically significant.
- If they are important enough to be in native format, they should be present in extracted content.

- **Incomplete extraction of inline codes**
  - [CDATA] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/cdata
  - [inline_codes_plain_text] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/inline_codes_plain_text
  - http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#d0e8112
  - https://www.w3.org/TR/xml-i18n-bp/#AuthCDATA
  - Not using native XLIFF representation leaves inline codes unprotected and increases risk of roundtrip corrupting them.
- **Using single inline element to represent multiple subsequent codes**
  - [multiple_codes_represented_as_single] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/multiple_codes_represented_as_single
  - Grouping several independent inline codes into single representation could prove challenging with negative impact on
    - Translation quality
    - Fluency
    - Functionality
    - Automated actions
    - Validation
  - Some codes needs to be removed, copied, added or reordered.
  - If any of the above actions is to be prevented, it can be controlled using editing hints with finer granularity.

Target content

Populating content of <target> element during Extraction or Enriching.

> Leave <target> empty unless adding value

> Use Translation Candidates Module to store suggestions instead of <target>

> Implement XLIFF state machine

- **Inserting unmodified source content into <target>**
  - [source_in_target] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/source_in_target
  - Source language in <target> generally does not provide any advantage during roundtrip.
  - Existing target could complicate segmentation modification.
  - It needlessly increases size of the XLIFF file.
  - Presence of <target> requires @trgLang to be defined, which necessitate knowledge of languages implemented in Extractor, which is redundant in most cases.
  - Omission of both <target> and @trgLang enables Extractor to create language agnostic XLIFF file. Localization scope can be defined later in the roundtrip.
- **Inserting possible translation into <target>**

- [pre-populated_target] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples
- Enricher can use Translation Memory, Machine Translation engine or other means to obtain suggestions for translators.
- Using <target> for storage limits the number of suggestion and the metadata available (unless the state machine or some other XLIFF feature is overloaded). Interoperability issues with Agents without prior knowledge of the workflow can be expected.
- Recommended way is to use **Translation Candidates Module** http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#candidates

- **State machine**
  - Using the linear state machine based on the @state in combination with native validation artifacts introduced in XLIFF2.1 enables enforcing existence of <target> as child for states different from *initial*.
  - https://issues.oasis-open.org/browse/XLIFF-11

Editing and context hints

> Protect non-deletable codes
> Preserve code order where needed
> Control segmentation modification
> Provide context
> Consider default behavior of canOverlap

- **Non-deletable inline codes**
  - [editing_hints_canDelete] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/editing_hints_canDelete
  - Removal of some of the inline codes from translated content could cause issues during Merging or functional problems in translated product. Placeholder replaced during runtime are one example of such codes.
  - Attribute canDelete can be used to control validation mechanism available in XLIFF.
  - https://issues.oasis-open.org/browse/XLIFF-10
  - https://issues.oasis-open.org/browse/XLIFF-13
- **Preserving order of codes**
  - [editing_hints_canReorder] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/editing_hints_canReorder

- Schema could prescribe nesting of inline codes.
- Changing order of the codes in translated content would cause issues discoverable only during of after Merging.
- Usage of canReorder allows to validate the order anytime during the roundtrip.

- **Controlling segmentation modification**
  - [mapping_to_unit] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/mapping_to_unit
  - Depending on the mapping decisions, list items or cells of a table row could be extracted as segments of single unit.
  - While beneficial for other reasons, segmentation within unit can be modified, with possibly negative impact on Merging.
  - Attribute canReorder on XLIFF structural elements can be used with care to control Modifier's behavior. It's value can be devised from logic based on role of structural and inline codes of native format, e.g. set to "yes" for:
    - Lists (<ol>, <ul>, <dl>, …)
    - Table cells
    - Alt text

    depending on the content.

- **Providing context**
  - [context_hints] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/context_hints
  - Role of the inline code might have impact on the roundtrip.
  - Actors need context to make correct decision whether to remove, reorder or add the codes and how their occurrence impacts the accuracy and fluency of the translation.
  - Such context can be provided using attributes disp(Start/End), equiv(Start/End), type and subType

- **Considerations for using <sc>, <ec>**
  - [editing_hints_canOverlap]

https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/editing_hints_canOverlap

- <sc>, <ec> can be considered more universal for representation of spanning codes. They handle segmentation changes better and can even span across units.
- Their ability to overlap can create problems for well formed spanning codes
- Default value of their @canOverlap is “yes”
- Inline spanning codes can be malformed during roundtrip without native validation to be able to detect the problem in the default mode.
- @canOverlap should be set to “no”

XLIFF structure

> Make use of <group> element

> Choose appropriate mapping of native format structures to <unit>

- **XLIFF file structure**
  - [group] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/group
  - Native format can contain structural elements dividing the content into parts, such as title, body, header and footer, or tables, lists and divs for markup languages, or windows, dialogs, menus for software resources.
  - Representing native structural elements in XLIFF using nested <group> elements can be useful for correctly scoping:
    - Additional context (@name, @type, @subType, attributes from Format Style Module)
    - Restrictions (@canResegment, @translate, attributes from Size and Length Restriction Module)
    - Whitespaces handling (@xml:space)
    - Information from modules:

- Metadata
- Validation
- ITS

Most of the above can still be achieved without the <group>s at a cost of high redundancy and overloading some XLIFF features.

- **Role of <unit>**
  - [mapping_to_unit] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/mapping_to_unit
  - Extractor sets the XLIFF structure, which cannot be modified during the roundtrip above the <unit> level
  - Appropriate relationship between structures of native format and <unit> can make the difference between hindering the roundtrip and making the most of XLIFF features
  - Problems can be caused by both too many and not enough <unit>s

Miscellaneous

> Do not overload id attribute
> Preserve whitespaces only where necessary
> Use correct representation for non-localizable content
> Consider all valid modifications when merging
> Choose correct language tags
> Perform sanity check on extracted content

M

- **Value of @id**
  - [id_and_name] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/id_and_name
  - Implementers could be tempted to store values of resource Ids in the id attribute of XLIFF structural elements
  - XLIFF's @id value can be only NMTOKEN, while native format may not have such restrictions
  - XLIFF's @name is designed to store the original identifier of the resource. Value is not restricted to NMTOKEN.
  - @original can be used on <file>
  - Usually not an issue for native formats not using resource Ids
- **Whitespace handling http://docs.oasis-open.org/xliff/xliff-core/v2.1/xliff-core-v2.1.html#Preserve_Space**
  - [xml_space_preserve] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/xml_space_preserve
  - @xml:space can be set on different elements
  - Setting its value to "preserve" indiscriminately on high level structural

elements, such as <xliff> or <file> can cause localizability issues and loss of leverage depending on configuration of Enriching from TM, MT or other sources.
    - Whitespaces should be preserved only where appropriate, such as HTML <pre> element
- **Protecting non-localizable content**
    - [ph_and_mrk] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/ph_and_mrk
    - <ph> is preferable for code with programmatic purposes
    - <mrk> is preferable for linguistically significant text
- **Merging translated content**
    - [merging] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/merging
    - Modifiers are free to:
        - change segmentation (unless restricted by canResegment)
        - enrich the content with annotations
        - add inline codes
        - change CDATA into escaped content
        - use <pc> and <sc>, <ec> interchangeably (for well-formed spanning codes)
        - perform other changes allowed by Processing Requirements

      XLIFF compliant Mergers needs to be able to correctly handle any valid modification of the file.

    Improper extraction shift the problem further downstream with unpredictable results.
    - https://issues.oasis-open.org/browse/XLIFF-12
- **Selecting language tags**
    - [language_tags] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/language_tags
    - All language tags in XLIFF must be valid BCP 47 values
    - Specific enough values should be used to prevent interoperability issues
    - Possible future expansion of language set should be considered along with subset-superset implications
    - Leveraging from related languages can be influenced by language tag value
- **Validation of extracted content**
    - [sanity_check] https://github.com/GALAglobal/TAPICC/tree/master/extraction_examples/sanity_check
    - Native formats use various reserved characters or sequences for structure

and inline markup as well as programmatic purposes

- While not violating XLIFF constraints and PRs their incidence in extracted content could point out issues in extraction process
- Failed sanity check would ideally interrupt the roundtrip as early as possible and prevent problems further downstream

XLIFF2 Validation

> Validate after modification

> Consider implementing more than one way of validation

Currently available ways to validate XLIFF2 files:

- Okapi Lynx (http://okapi-lynx.appspot.com/validation, offline version available too)
- Microsoft (https://github.com/Microsoft/XLIFF2-Object-Model)
- Native Validation Artifacts — since XLIFF2.1 (https://tools.oasis-open.org/version-control/browse/wsvn/xliff/trunk/xliff-21/schemas/#_trunk_xliff-21_schemas_)

Conclusion

- Extractor does not have to implement segmentation
- Extractor does not have to have language knowledge
- Inline codes should be properly extracted
- Context for roundtrip actors should be provided
- Merging should account for all valid changes during roundtrip
- Validation is vital

Q&A

M