**Face to Face Meeting - London**

**Participants: Yves Savourel, Joachim Schurig, Kevin O’Donnell, David Filip, Fredrik Estreen, Ryan King, Bryan Schnabel, Lucia Morado.**

**Date**: Monday, 10 June 2013, 04:00am to 09:00am EDT  
**Location**: London

B: This is the seventh XLIFF FTF meeting. Big agenda in front of us, we will make the best of our time. Thanks Kevin and people from Microsoft for accommodating us. Thanks David for our your work in the P&L SC. The good news is that we cannot do substantial changes; otherwise we will have to start over. [B explains the agenda for today]. There will not be a public session; we can use that time to continue working.

**First Topic: Re-segmentation 9:30 - 10**

F: If we do not have information in these new modules about segmentation, what should we do about resegmentation? The other option would be simply remove the modules.

B: And the processing requirements would not help?

F: we do not have a standardised way to resegment. I cannot see from the modules we have now that information. It does not really work.

J: the boundary of information is difficult of establish. In general, we need to retain the information, even it is separated. Seg is a specific domain of translation.

F: We are adding too much stuff at the unit levels that makes segmentation difficult. We have the matches that we have discussed. The only simple that we can do is to is to add something to the unit that would indicate whether that element could or could not be segmented.

Y: I do not think it is not possible either; any type of information is already on a segment base. It seems to me that the flag would help. But at the same time I would not prohibit to segment.

F: If you put a module or a namespace where we say that we want to preserve segmentation.

R: I see the point of having the flag for saying whether to segment something or not. At the tool developer level makes sense.

F: We should think where to put that and when. Whether to put it in the unit or in the segment, if you put it on the segment, that might be lost. That it is also related if we think whether is a process format. If XLIFF is not supposed to have processing information then I do not care.

D: I think you are right. It is up to you if you want to have the segments in the same shape.

F: the problem would be if somebody has some validation rules.

D: that’s up to you how you do your validation roundtrip.

R: Which modules would affect that?

F: matches, notes. If I use a marker I can identify. The subsegments would be still in the segments. But it is a very big change at this stage. Do we have the time to work on that?

D: The question is, if we have too much metadata the segmentation becomes more difficult. The whole thing about having segments is to allow segmentation.

F: during the translation, if I translate at the unit level, and I would like to change something that would be done at the unit level, not at the segment level.

D: We should have a way of adding md in markers.

F: I would always have them in markers.

D: I agree with that statement.

Y: It has some advantages, e.g. matches should match subsegment.

Y: That would mean implying many changes in the current schema. It could be easy to do it for tools. It seems to me that we are getting closer and closer to a binary format [irony?].

F: The possible solutions:

Option 1: don’t allow if module or extension (make core depend on non-core)

Option 2: throw away (if you don’t understand, throwaway –disrupts modules)

Option 3: set flag, (if it is a yes, it is undefined behaviour)

Option 4: Move meta to unit (lots of changes to spec; can still have flag (or different use)

[Email from Fredrik:

Non-core Features blocking segmentation

On <segment> elements mtc:matches, mda:metadata, ctr:changetrack, val:validation Attributes fs:fs, fs:subFs. There is also the notes element which is core so less complicated.

On <ignorable> element mda:metadata

Attributes fs:fs, fs:subFs

On <source> and <target> attributes fs:fs and fs:subFs

Problem; we must create processing requirements that can be implemented by tools not supporting anything except the core features. Since we do have references to the module data in he core specification it would be possible to use different requirements for different modules already at core level. But that would to a large degree remove the benefits of modules and would not scale if the number of modules grow. Instead any processing requirement in core should apply equally to all modules.

The only simple solution at this time is to not allow any segmentation changes if there are unsupported elements or attributes present on or in the above four elements. A module defining elements or attributes in these places should either provide sufficient processing requirements to allow segmentation changes or the single rule that segmentation changes are not allowed if the module is used in these places. That would allow a module to re-enable segmentation if the document is processed by a tool supporting that module as long as no other blocking issues exist.

An option would be to allow an agent to just remove offending unsupported elements and attributes. But as far as I remember that option was rejected.

A larger solution would be to rework the modules to not interfere with segmentation. Seems a bit late in the process to do that now.

To avoid misusing modules just to block segmentation I suggest that we add a unit level attribute that signal weather segmentation changes are allowed or not. With such an attribute we could instead require that any tool adding elements or attributes that would pose a problem for segmentation must set this flag. The idea of this attribute was originally put forward by Oracle (Jung?).

End of email]

R: The option 3 is basically what we have today, but just saying whether it can be segmented or not.

Y: the problem is with the processing requirements, if you are a tool developer what should you do.

D: It seems to me that it would be viable (option 3).

B: We would be talking to many people on the following days about this.

Y: I think Option 4 it is the smartest way to do it. But it will imply loads of rework. I do not agree with option 1, because it will mean making something that is not core, look like core. Option 4 is not really loads of work, but loads of changes.

B: Who is the owner of the segmentation?

Y: I think Rodolfo.

B: O1 and O2 would not mean much work. O3 really touches the schema.

Y: We could have a flag preventing re-segmentation. That is a separate problem that has nothing to do with the MD. Toolmakers would not like it in the beginning. They might like to work with what they have right now or might not have a mechanism. It is a lot of changes to map to what we had before. It might be the smartest way to do it, but it implies work.

J: would it be ok to have one of these three options, or do we stick with what we have?

D: we can do a ballot later on by presenting the four options.

B: What do you think about the options?

D: 3 or 4.

R: 3 or 4.

J: 4

K: 3 or 4.

L: I abstain.

Y: 3 or, but 4 for the long run. We work with annotations, and tools would finally have to work with that. It seems more logical to go that way.

B: I would go for 4.

F: I would go for 4.

**Extensibility: Ryan’s requests (segment and glossary) 10 – 10:30**

R: The glossary module today is not extensible. In some discussion the ITS idea also came up. But I think we should not have a module that can replace the functionality that already exits today. We should make the glossary module extensible.

D: I think the idea is that modules can be extended. But the issue here that the module isn’t extensible. You cannot add functionality rather than the one that is already defined.

B: we had an earlier discussion on how to reconcile extensibility and metadata.

R: the only issue we have is the glossary extensibility or the extensibility of modules in general. So if we extend the glossary module with a md module we could store all our md module, but the problem would be with the interoperability. It is odd that the matches module is the only one that can be currently extensible with the md module.

J: I agree with should have more extensibility.

D: Extensibility, but not other module. There is an interchange track, for anybody interested. It seems to be that the modules are protected; I can interpret the module as a replace.

J:If you don’t specify in the core, where the module should appear, it loses it meaning.

R: you imply that if it appears in any other part than the extension point. It cannot be protected. The point of this session, is that if we can have the md module extensible.

D: The glossary

J: I was working to have the glossary module as simple as possible. If we have loads of containers, people might not use.

R: I actually have the same question about the matches module. In 1.2 the alt-trans was not used that match. I wonder if that is it going to be the same direction with the matches module.

J: For us, as service providers, all these data does not have value. It would make more sense an URI.

Y: alt-trans is different than terminology, which is a list of terms. For alt-trans we are having a list of matches. There is information that you have in alt-trans that you cannot have in TMX (e.g. Fuzzy-match percentage).

D: Regarding the glossary, some proposed to drop the glossary module and have TBX. There have been some comments about the difficulty of adding the tbx, in our schema.

B: I think Joachim is right, we could a lite version (tbx) and we could go to something more specific if needed.

R: So that means, that if we stick to the lite version, would it also be extensible? Could we use it as another module?

J: From toolmakers perspective, the biggest issue is to segment languages. If we keep a mechanism to identify terms.

D: That was the issue with the glossary module, that it does not allow you to identify inline terms. I think we should have a reference mechanism.

J: We have internal information that we found it more valuable than other information.

R: As a content provider, the glossary module is not good enough for me. Do we want to add terminology data through a namespace mechanism? The main point is if I can replace the glossary module with something else. Either we extend the glossary module or we have an alternative solution.

D: When you want to tie information, the glossary just needs to have an external and an internal reference.

R: What does that change from what we have today?

Y: You could put already a reference to a document. But the problem is that it is quite complex.

D: How do you point from?

Y: You could do with ITS. It is doable, people have already implemented it. We should not prevent people to use something more sophisticated.

D: How would it work?

Y: You can literary add a TBX document within a XLIFF document, which is not the smartest idea. I do not think you should have it at the unit level. If we add a TBX extension, what would that be? Would be a module?

J: I think it is better not to have a profile. I am not doing the same thing as the glossary module at the unit module.

R: I understand the point.

J: I don’t think that there would be a conflict; TBX would not be competing with the glossary module. So, should we define a TBX basic basic?

:Mda or extensibility?

D: I think that apart from extending it. The concept should have an id.

Y: And then you can add a marker to that one that you can reference. We would add an id to the glossary entry and them in your unit you can add a marker that references to that id.

D: I think we should stick to this ID idea and avoid the use of extensibility point.

R: in extensibility, do we mean attributes or elements from another namespace?

D: the danger is to overlap the TBX function. Do we still want to allow the TBX to be embedded?

Y: They can also be referenced externally.

D: Don’t you think we should be explicit about the relation between TBX and the glossary module? In the glossary, we should be legal to point to the file or not.

Y: the question is where you put your reference material.

D: I would put a normative note about this.

D: would we a value to say something normative to have the location of the reference material?

R: You can have tbx or anything you want.

Y: that’s true, it can be a lite database for example.

D: I think we have consensus on what to do with the glossary module.

Y: White pointer. Add id on (top point from <mrk> type term) on <glossentry>, and add extensibility (elements, and attributes). – not on children

R: One concept can have more than one entry, so I can have more than one element from one concept.

J: I think having in Glossary to have it.

R: Any objections?

All: no.

+++++++++++++++++++++++

**=== Topic: Agents and Processes**

DF: having a set of defined agents is good for various reasons

.. for example one of the reviewer noted that we have almost no conformance clauses for application

.. test suite is needed

.. we need examples as well

.. static example are not enough, we need to see what happened before/after

.. segmentation is another case when we need example/tests

RK: example with segmentation: resegmenter should put back segments back

FE: disagree: as long as the integrity of the unit is preserve we should be ok

.. otherwise resegmenting is never possible

DF: the point is knowing the agent can do is necessary

JS: those definitions don't help necessarily, we always end up with input and output files

DF: they help to provide conformance profiles

.. set of tests different per type of agents

.. this is like a normative layer to extract conformance profiles for specific type of tools

FE: agree that we need to define what is to be used for test

.. for each modifier is capable to roll back:

.. too complicated

DF: meaning of modifier is different here, it's not for every steps

FE: example maximize use of pc element for example

DF: pc is not structure element

FE: should be able to go back to initial state

RK: by legal transformation you men the ones defined?

FE: yes

.. not sure I see difference between TE and modify

FE: would id allow to change grouping

DF: agree, but not in specification

.. should have a list of the allowed transformations

FE: simple thing is to say what is allowed.

FE: know a few use cases for source editing

.. revision of source, but should be done with target=source

.. other case is to improve source for the duration of project

DF: XLIFF also used as a source format

.. for example Oracle

FE: those cases are really target=source

RK: not in all cases: some content provider do not want source changed

DF: could have flag with do not accept source change

.. other may be able to allow it

FE: but treating source as target answer all those use cases.

.. can't merge it back (its' the source)

RK: allowing source editing changes nature of XLIFF

YS: not sure, this is true. we still change source/target

KO: flag would allow to control this

FE: could get many different 'fixed' sources then

YS: think we go at a too fine level.

DF: validator is another categories

FE: merger could be doing validation

.. many files for example: we validate and merge at the same time

DF: idea is validator would not need to pass the merging tests

YS: seems this is too granular again

DF: seem: extractor, modifier, enricher, merger.

LM: can see the blur between modifier/enricher

FE: maybe this 4 types are enough

.. modifier changes only content of units

.. enricher: is adding new info without changing existing info

BS: idea is to change the wording of the PRs to allow testing profiles

DF: some tools can be composed

FE: translation editor should pass tests of the modifier: if it changes only a sub-set for what a modifier can change

.. seems we would test that a tool do only some tasks

.. not if it does not fail the PR

RK: if an editor doesn't create a given target it doesn't break XLIFF

FE: this is not testing interoperability

.. more a test of functionality

.. see PR as a mean to tell what changes are allowed

JS: were close to consensus at some point.

DF: with the 4 categories we should be ok to create profiles

JS: see only 3 things: start, modifications, end

.. workflow chain that allow interoperability

FE: would be fine with or without enricher

.. extraction: valid output

.. modification/enricher: should perform only valid changes

.. merger must accept back-modified XLIFF

DF: thing the difference is in talking about application or document

BS: so something to decide later

**=== Topic: Timeline**

BS: maybe we can move this for later (not much time left)

DF: Think we could start the timeline now

BS: here is the timeline as I see it for now:

o Goal for reconciling each comment: [02 July]

o Statements of Use [identify by 16 July]

[Note: Tom will join us in the afternoon dial-in session to discuss potential solutions he‚’s working on]

ÔÇß Must have a statement of use for each major feature?

ÔÇß Test Suite (1 application vs. ecosystem of tools)

ÔÇß Reference Implementation [identify by 16 July; roll out by 06 Aug?]

ÔÇß One implementation that touches each feature

ÔÇß Candidates?

o Re-approve Committee Draft (that reflects resolved comments, https://www.oasis-open.org/policies-guidelines/tc-process#committeeDraft ) [06 Aug]

o Second Public Review, 15-day [12 Aug ‚ 30 Aug]

o Approve Committee Specification (https://www.oasis-open.org/policies-guidelines/tc-process#committeeSpec ) [17 Sep]

o Approve OASIS Standard (https://www.oasis-open.org/policies-guidelines/tc-process#OASISstandard ) [17 Sep ‚Äì 09 Dec]

ÔÇß Submit Candidate Specification [17 Sep]

ÔÇß Public Review of Candidate Specification (60 days) [24 Sep ‚Äì 22 Nov]

ÔÇß Ballot for OASIS Specification approval [25 Nov ‚Äì 09 Dec]

FE: this schedule rules out major changes in modules

.. also not easy to have tight deadlines during the summer.

DF: we should be able to dispose of substantive changes now

YS: in second review we can change only things modified in the first review

"Changes made to a committee draft after a review must be clearly identified in any subsequent review, and the subsequent review shall be limited in scope to changes made in the previous review. Before starting another review cycle the revisions must be re-approved as a Committee Specification Draft and then approved to go to public review by the TC."

(https://www.oasis-open.org/policies-guidelines/tc-process#publicReview)

FE I think we may have some un-caught inconsistencies.

YS: We have many comments but from few only

BS: Some of my comments are from others

.. depending on if we find or not a show stopper we may or not be able to move to 2.0 and fix those later.

FE: Had some comments on overriding behavior. Think we may have other like this.

- close for the morning