ELAN → Flex → ELAN “round trip” workflow

John Mansfield, University of Melbourne, 29 May 2015

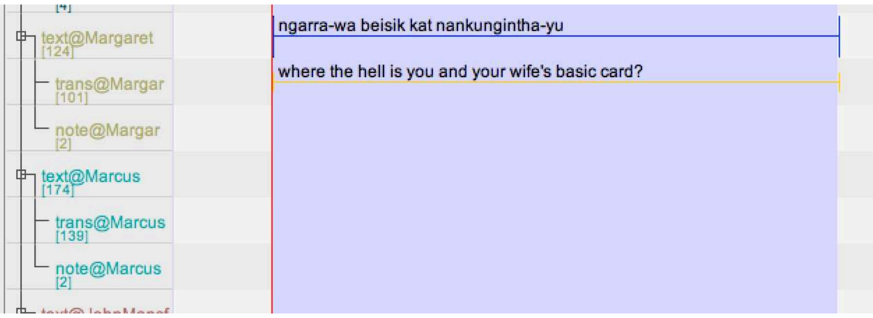
Main steps:

1. Transcribe recording in ELAN
2. Export to .flextext format
3. Import into Flex
4. Interlinearise in Flex
5. Export in .flextext format
6. Merge interlinear analysis with original .flextext file
7. Import .flextext back into ELAN
8. Run a tier-renaming script on the .eaf file
9. ELAN file is now ready for further transcription, contains morphological analysis, and can go on another round trip if further interlinearisation is required.

Note: this guide is based on ELAN 4.9, and SIL Fieldworks Language Explorer 8 (“Flex”).

1. Transcribe recording in ELAN

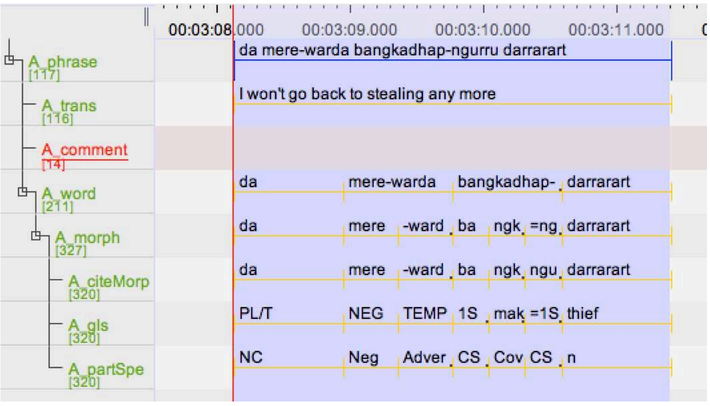
Transcribe as per normal using MPI ELAN. You can use whatever tiers you want, though some might have their names changed. But *note*, however, that an utterance > word > morph hierarchical structure is assumed for this entire process. I.e. the tier hierarchy should look something like this (for a simple transcript):



ngarra-wa beisik kat nankungintha-yu

where the hell is you and your wife's basic card?

Or like this, for a transcript with interlinearisation:



da mere-warda bangkadhap-ngurru darrarart

I won't go back to stealing any more

da	mere-warda		bangkadhap-				darrarart
da	mere	-ward	ba	ngk	=ng		darrarart
da	mere	-ward	ba	ngk	ngu		darrarart
PL/T	NEG	TEMP	1S	mak	=1S		thief
NC	Neg	Adver	CS	Cov	CS		n

2. Export to .flextext format

Use the ELAN menu option, File > Export As > FLEx File. This will prompt you through four steps:

Steps 1 and 2 don't ever seem to require deviation from the default settings. If there is a problem here, this guide will have to be updated accordingly.

Step 3 does require some setting: basically you have to map your ELAN tiers to the Flex data types. The basic tier types map as follows:

transcription → txt

translation → gls

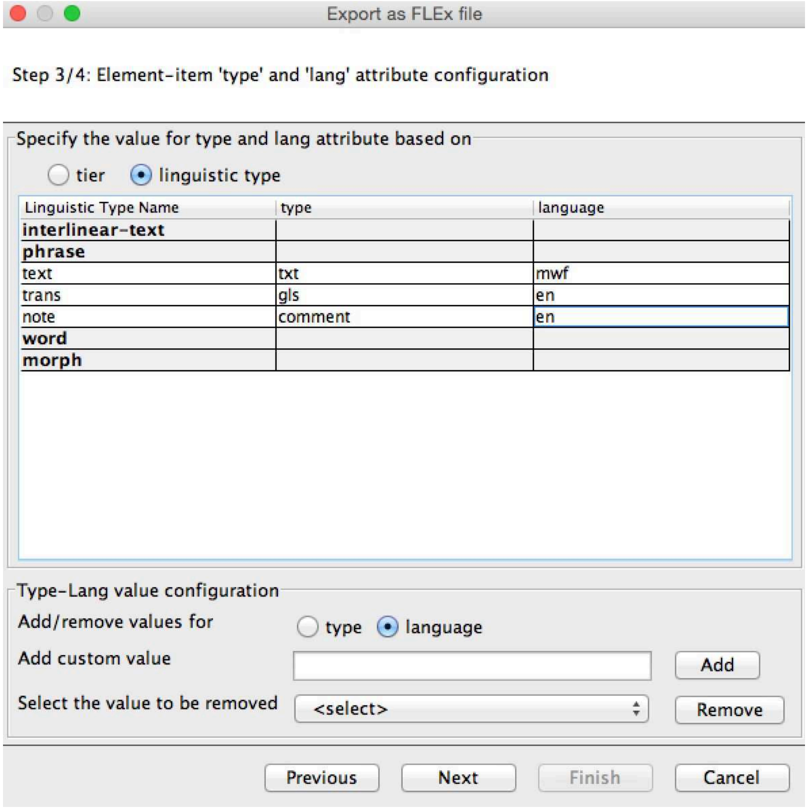
comments → comment

word/morph parsing → txt

word/morph glossing → gls

The dialogue lets you map each ELAN tier *type* as generalisation (use this if you have multiple transcript tiers, multiple translation tiers, etc), or map individual

ELAN tiers. Both options seem to work fine. Here's the mapping for a simple transcription file, with no interlinearisation or specialty tiers:



Export as FLEx file

Step 3/4: Element-item 'type' and 'lang' attribute configuration

Specify the value for type and lang attribute based on

○ tier ● linguistic type

Linguistic Type Name	type	language
interlinear-text		
phrase		
text	txt	mwf
trans	gls	en
note	comment	en
word		
morph		

Type-Lang value configuration

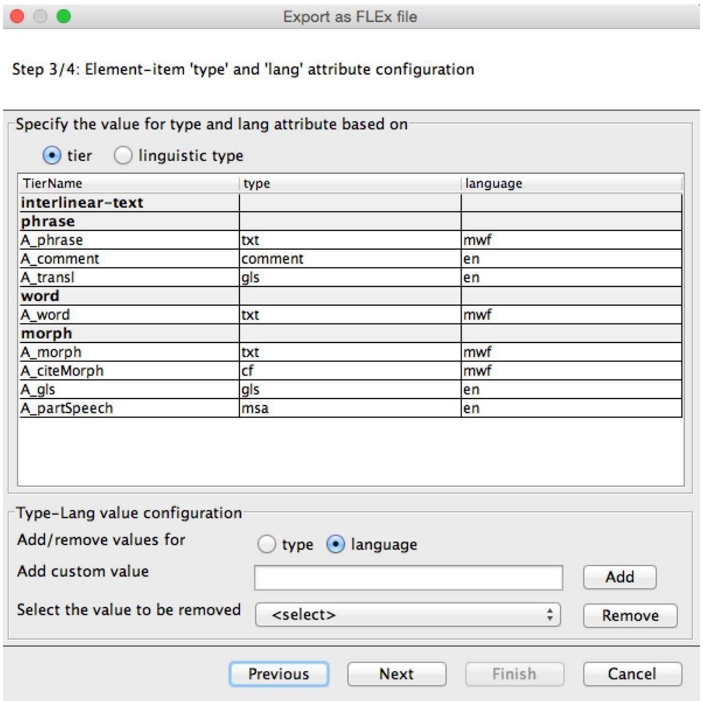
Add/remove values for ○ type ● language

Add custom value [ ] Add

Select the value to be removed <select> Remove

You also need to specify for each tier whether it is the language being transcribed (e.g. Murrinhpatha (mwf)) or English (en). These language codes are not already in the ELAN system, so you add them down the bottom of the dialogue box with "Add custom value (language)".

Here's the mapping I've used for exporting a rather complex transcript that already has interlinearisation. The ELAN tier names I've been using here are designed to match the Flex data structure in an intuitive way:



Step 3/4: Element-item 'type' and 'lang' attribute configuration

Specify the value for type and lang attribute based on

● tier ○ linguistic type

TierName	type	language
interlinear-text		
phrase		
A_phrase	txt	mwf
A_comment	comment	en
A_transl	gls	en
word		
A_word	txt	mwf
morph		
A_morph	txt	mwf
A_citeMorph	cf	mwf
A_gls	gls	en
A_partSpeech	msa	en

Type-Lang value configuration

Add/remove values for ○ type ● language

Add custom value

Select the value to be removed <select>

You might have to add custom data types for any specialty tiers you've used. ELAN will let you add any data type you like, but the names of these types should adhere to some kind of standard naming for the corpus you're building, otherwise they won't gel with other tools down the track.

Export file should have same name as original, but with .eaf changed to .flextext. And they should sit alongside each other in your corpus directory structure.

3. Import into Flex

Straightforward, no options to set.
But if this transcription is already in your Flex project, you should take some steps to avoid duplicating it. I would suggest 1) rename old version of transcript in Flex; 2) import new version; 3) delete old version if satisfied with import.

4. Interlinearise in Flex

Do it.

5. Export in .flextext format

When you are sick of interlinearising, export it back out again. You must save it back into the relevant corpus directory, now with the file ending .postflex.flextext

6. Merge interlinear analysis with original .flextext file

The biggest limitation with Flex interlinearisation is that most (or all) custom tiers in your .flextext file are not retained when it goes in and out of the program. So essentially you just want to get the interlinearised analysis that Flex has produced, and insert it into your original .flextext file. Do this using the XSL script merge-interlinear.xsl.
Also you will often have fixed some transcriptions or translations while interlinearising in Flex. Therefore this merge process overwrites the originals of these tiers, unless you ask it not to by using the option overwrite=no.

Examples of usage:

```
java -jar -Xmx1024m /Library/SaxonHE9-4-0-4J/saxon9he.jar -t
Magultje-test.flextext merge-interlinear.xsl interlin=Magultje-
test.postFlex.flextext overwrite=no > Magultje-test.merge.flextext
```

or

```
java -jar -Xmx1024m /Library/SaxonHE9-4-0-4J/saxon9he.jar -t
../archival/1990_Bible/01_Zechariah.flextext merge-interlinear.xsl
interlin=../archival/1990_Bible/01_Zechariah.postflex.flextext
>../archival/1990_Bible/01_Zechariah.merge.flextext
```

The two earlier flextext files should be moved to the /old-versions folder that you keep for every corpus session. The merged version will be the file used for corpus analysis, because it's a much more elegant data structure than ELAN's .eaf format.

Note that the .postflex file must be merged with the same .flextext file from which it originated. I.e. you cannot re-export from ELAN and use that to merge with the .postflex. This is because the merge process depends on recognition of element identifiers created in the ELAN export.

7. Import .flextext back into ELAN

The merge script will have produced a file that you should import into ELAN. I've just been leaving all the options at default here.

Re-imported file should have file extension .merge.eaf

Note that where there are many files being processed as part of a task, the ELAN "Import multiple files" option can be used to do, say, a whole directory at a time.

8. Run a tier-renaming script on the .eaf file

The file back in ELAN now has awkward (but highly logical) tier names, derived from the Flex data structure. You could change all these. To convert these into nicer, shorter tier names, use the script rename-tiers.py. (Also does the accompanying .pfsx file.)[1] Output of this will appear with an .edit.xml extension. If the output is okay, replace the original with it.

Usage:

```
python rename-tiers.py ../PATH/TO/FILE.merge.eaf
```

The output will have the same filename, but extension just .eaf

[1] In an earlier version of the script, the input ELAN file was sent into the associated /old-versions directory, and the output appeared in place of the input. This has now been changed to output reducing the .merge.eaf extension to simply .eaf

NB this can **only** happen after re-importing merged flex data into ELAN. Because it is the import process that creates the cumbersome tier names.

9. Done

ELAN file is now ready for further transcription, contains morphological analysis, and can go on another round trip if further interlinearisation is required.