An End-to-End Solution for Using Unicode with Blaise to Support Any Language

Alerk Amin (CentERdata, Tilburg, The Netherlands)
Richard Boreham (National Center for Social Research, London, England)
Maurice Martens (CentERdata, Tilburg, The Netherlands)
Colin Miceli (National Center for Social Research, London, England)

1. Introduction

Understanding Society is a major new household panel study which has been commissioned by the Economic and Social Research Council (ESRC). Taken as a whole, it will be the largest study of its kind in the world interviewing people in a total of 40,000 households across the UK. It is led by the Institute for Social and Economic Research (ISER) at the University of Essex. The survey will also be known as the UK Household Longitudinal Study (UKHLS) among the academic community, but we will only refer to it as Understanding Society.

The survey will collect data from all members of a household aged 10 and above on an annual basis (10-15s by self-completion only, all those 16+ by CAPI and self-completion). Annual interviewing will allow us to track relatively short term or frequent changes in people lives, and the factors that are associated with them. As the years of the survey build up we will be able to look at longer term outcomes for people in the sample.

There is an ethnic boost in addition to the main general population sample. In the ethnic boost, a household is potentially eligible if anyone (including children) in the household has parents or grandparents from any of these groups: Indian, Pakistani, Bangladeshi, Black-African, Black-Caribbean, Sri Lankan, Chinese, Far Eastern, Middle Eastern and Iranian, Turkish, North African and African Asian.

Response rates among some ethnic groups tend to be lower than the general population, and this can partly be explained by the fact that most surveys exclude non-English speakers, which results in an interviewed population who are not representative of a particular ethnic group.

Providing a translated questionnaire and documents is one way to reduce this bias and to increase response rates by allowing non-English speakers to participate. The questionnaire needs to be translated centrally to ensure that all participants are being asked the same questions in the same way.

In order to determine which languages would be used for translations, we needed to determine the prevalence of people who used that language, but would not have adequate English to participate in an interview. Hence, although Hindi is one of the most widely spoken languages in the UK, it is not one of the languages used in Understanding Society because the majority of Hindi speakers also speak English. The nine languages chosen are

- Arabic,
- Bengali,
- Cantonese,
- Gujarati,
- Punjabi (Gurmukhi script),
- Punjabi (Urdu script),
- Somali.
- Urdu,
- Welsh (there is a legal requirement to provide translations into Welsh even though there are very

few people who speak Welsh who cannot speak English).

These languages include languages with non-Latin scripts (which Blaise cannot deal with) and languages which are written from right to left.

Any interviewer could potentially come across a household where someone speaks one of these nine languages, but where other members of the household can speak English. Therefore we need one questionnaire instrument with English and the other nine languages, and the ability to switch languages during an interview in a single household.

2. Solution Overview

Blaise is the interviewing software used by NatCen so we needed a solution which allowed us to program the main English questionnaire in Blaise. Blaise does not currently support a mixture of Latin and non-Latin scripts in the same instrument, so we needed an alternative to the Blaise DEP which would replicate most of the features of the DEP, but would allow the display of Unicode fonts. So we decided to develop a new data entry program called UNITIP (Unicode Translation Interviewing Program). CentERdata has already previously developed an LMU (Language Management System) for the SHARE survey to manage the translations of the SHARE questionnaire into different languages. We decided to adapt the LMU for use with Understanding Society, by taking the translations stored in the LMU and creating a Blaise questionnaire containing English plus the nine translated languages using the Multi Blaise Generator (MBG) program.

So the two main components to provide the solution to being able to interview in English plus 9 languages are an LMU to manage the translation process and UNITIP data entry program to control the interviewing.

2.1. Translation of Questionnaire

2.1.1. Translation process

There were four stages in the translation process:

- Translation
- Checking by a proof reader
- Checking by independent checker
- Adjudication

The questionnaire was translated online using the LMU using one translator per language. Each translator reviewed their own translation before they code that it is ready for proof reading and they were able to record queries about specific questions in the LMU.

Once the translator had coded that their translation is suitable for proof reading, the agency arranged for a second professional translator to re-read the job. The proof-reader would check the translation against the source text making sure that there are no grammatical mistakes, typos or formatting issues, as well as mistranslations or omissions, including style improvement.

Once the supervisor has coded that the translation is suitable for checking, the translated questionnaire was checked by independent third parties (i.e. checkers). The checkers were either NatCen researchers/freelance researchers, language students/teachers, or qualified linguists. Each language was checked by one checker. They worked by themselves and fed back comments on the translation and the appropriateness of the translation using the 'query' function in the LMU. The checkers checked the translation for accuracy, completeness, interpretation and consistency of meaning with the English questionnaire. All comments and reported errors were sent back to the translation agency for revision.

If the reviewer and translator were unable to reach agreement over the correct translation of an item, then an independent adjudicator resolved the disputed translation.

2.1.2. Questionnaire setup

A type of markup was used in the questionnaire source so elements in the source code could be linked to the LMU via the MBG program.

The first main task was to add a unique tag to each question which required translation. The tags had to be unique to match the unique fields in the LMU, so we added a two character prefix which reflected the block name followed by the question name, for example HH_BIRTH was the birth question in the household grid.

Each element of the question was required to be on separate lines, like this:

```
MvEver (DE_MvEver)
"Have you personally lived at this address your whole life?"
""
/ "LIVED AT ADDRESS WHOLE LIFE" :
TYesNo
```

All standard types which required translation had to be declared at the datamodel level so they could be picked up by the MBG.

We also had some types which didn't require translation and so these were placed in a separate file. This included things like date ranges, numeric ranges and but also questions that could have been translated but we took a decision not to such as country of birth.

Questions which contained a textfill were still a requirement on this project. We had to we had to create a procedure for each question that required a textfill. This technique was used to give the MBG a way to find the textfill in the source code. Here is an example of a procedure before translation:

Note the comment in curly brackets after the assignment of the textfill, this was used as the tag by the MBG to find the rule.

```
PROCEDURE txtLACal
  PARAMETERS
    import imLACSx : TBoyGirl
    EXPORT exSFLACal_TFGender2 : STRING
  RULES
    IF (imLACSx = A1) THEN
      exSFLACal_TFGender2 := 'he'  {SFLACal_TFGender2[1]}
    ELSEIF (imLACSX = A2) THEN
      exSFLACal_TFGender2 := 'she' {SFLACal_TFGender2[2]}
    ENDIF
ENDPROCEDURE
```

And here it is after being processed by the MBG.

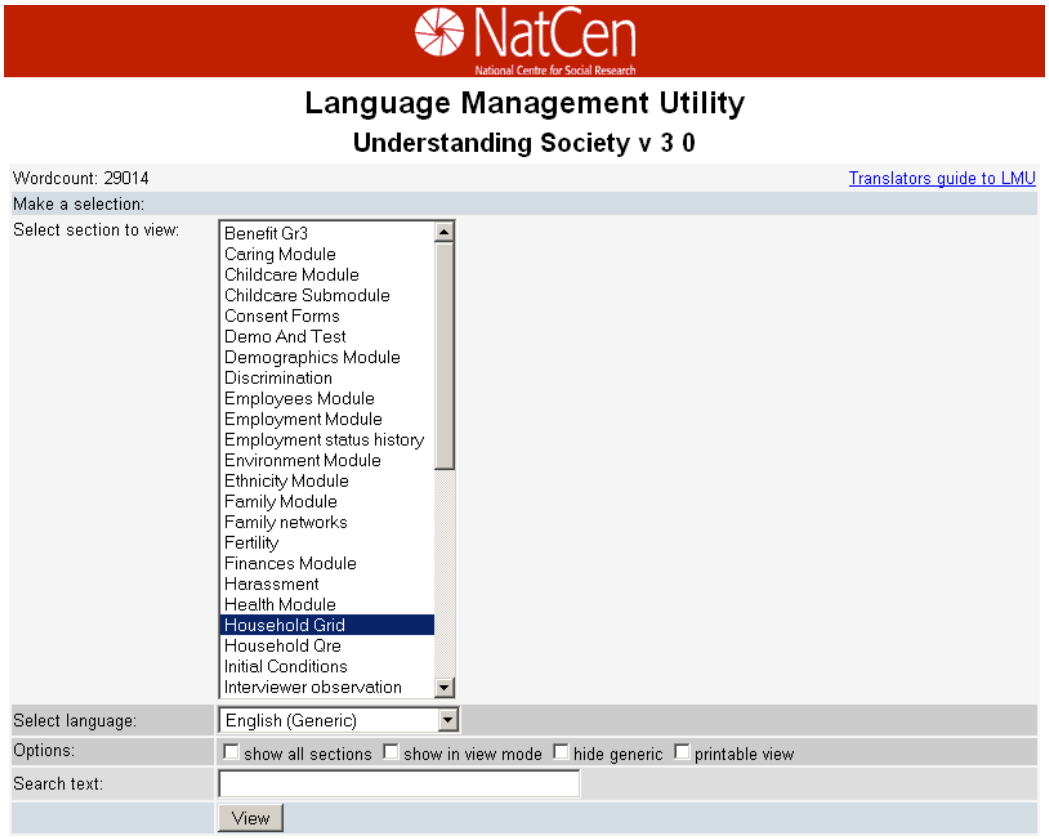
```
PROCEDURE txtLChal
  PARAMETERS
    IMPORT imLACsx  : TBoyGirl
    EXPORT exFTLChal_TFSheHe : STRING
  RULES
    IF (imLACSx = A1) THEN
        IF UT.CurrentLanguage = L1 THEN
          exFTLChal_TFSheHe := 'he'   {FTLChal_TFSheHe[1]}
        ELSEIF UT.CurrentLanguage = L2 THEN
          exFTLChal_TFSheHe := 'ਸ'   {FTLChal_TFSheHe[1]}
        ELSEIF UT.CurrentLanguage = L3 THEN
          exFTLChal_TFSheHe := ''    {FTLChal_TFSheHe[1]}
        ELSEIF UT.CurrentLanguage = L4 THEN
          exFTLChal_TFSheHe := 'e'   {FTLChal_TFSheHe[1]}
        ELSEIF UT.CurrentLanguage = L5 THEN
          exFTLChal_TFSheHe := 'ਨੇ'   {FTLChal_TFSheHe[1]}
        ELSEIF UT.CurrentLanguage = L6 THEN
          exFTLChal_TFSheHe := 'ਉਹਦੀ'   {FTLChal_TFSheHe[1]}
```

```
        ELSEIF UT.CurrentLanguage = L7 THEN
          exFTLChal_TFSheHe}  'اس لڑکے' =: FTLChal_TFSheHe[1{[
        ELSEIF UT.CurrentLanguage = L8 THEN
          exFTLChal_TFSheHe}  ' هو' =: FTLChal_TFSheHe[1{[
        ELSEIF UT.CurrentLanguage = L9 THEN
          exFTLChal_TFSheHe := '他  {FTLChal_TFSheHe[1]}
        ELSEIF UT.CurrentLanguage = L10 THEN
          exFTLChal_TFSheHe := 'ayay'  {FTLChal_TFSheHe[1]}
        ENDIF
    ELSEIF (imLACSX = A2) THEN
        IF UT.CurrentLanguage = L1 THEN
          exFTLChal_TFSheHe := 'she'  {FTLChal_TFSheHe[2]}
        ELSEIF UT.CurrentLanguage = L2 THEN
          exFTLChal_TFSheHe := "  {FTLChal_TFSheHe[2]}
        ELSEIF UT.CurrentLanguage = L3 THEN
          exFTLChal_TFSheHe := "  {FTLChal_TFSheHe[2]}
        ELSEIF UT.CurrentLanguage = L4 THEN
          exFTLChal_TFSheHe := 'hi'  {FTLChal_TFSheHe[2]}
        ELSEIF UT.CurrentLanguage = L5 THEN
          exFTLChal_TFSheHe := 'ਨੇਉਹੀ'  {FTLChal_TFSheHe[2]}
        ELSEIF UT.CurrentLanguage = L6 THEN
          exFTLChal_TFSheHe := 'ઉહદી'  {FTLChal_TFSheHe[2]}
        ELSEIF UT.CurrentLanguage = L7 THEN
          exFTLChal_TFSheHe}  'اس لڑکی' =: FTLChal_TFSheHe[2{[
        ELSEIF UT.CurrentLanguage = L8 THEN
          exFTLChal_TFSheHe}  'هي' =: FTLChal_TFSheHe[2{[
        ELSEIF UT.CurrentLanguage = L9 THEN
          exFTLChal_TFSheHe := '她  {FTLChal_TFSheHe[2]}
        ELSEIF UT.CurrentLanguage = L10 THEN
          exFTLChal_TFSheHe := 'ayuu'  {FTLChal_TFSheHe[2]}
        ENDIF
    ELSE
        IF UT.CurrentLanguage = L1 THEN
          exFTLChal_TFSheHe := 'they' {FTLChal_TFSheHe[3]}
        ELSEIF UT.CurrentLanguage = L2 THEN
          exFTLChal_TFSheHe := " {FTLChal_TFSheHe[3]}
        ELSEIF UT.CurrentLanguage = L3 THEN
          exFTLChal_TFSheHe := " {FTLChal_TFSheHe[3]}
        ELSEIF UT.CurrentLanguage = L4 THEN
          exFTLChal_TFSheHe := 'nhw' {FTLChal_TFSheHe[3]}
        ELSEIF UT.CurrentLanguage = L5 THEN
          exFTLChal_TFSheHe := 'ਨੇਅਲੋ' {FTLChal_TFSheHe[3]}
        ELSEIF UT.CurrentLanguage = L6 THEN
          exFTLChal_TFSheHe := 'ઉહનાં દી' {FTLChal_TFSheHe[3]}
        ELSEIF UT.CurrentLanguage = L7 THEN
          exFTLChal_TFSheHe} 'ان' =: FTLChal_TFSheHe[3{[
        ELSEIF UT.CurrentLanguage = L8 THEN
          exFTLChal_TFSheHe} 'هم' =: FTLChal_TFSheHe[3{[
        ELSEIF UT.CurrentLanguage = L9 THEN
          exFTLChal_TFSheHe := '他們 {FTLChal_TFSheHe[3]}
        ELSEIF UT.CurrentLanguage = L10 THEN
          exFTLChal_TFSheHe := 'ayay' {FTLChal_TFSheHe[3]}
        ENDIF
    ENDIF
ENDPROCEDURE
```

2.1.3. LMU/MBG to manage translation process

The Language Management Utility (LMU) is a web-based tool that can be used to translate a specially structured Blaise questionnaire. The questionnaire is divided into several modules. For each of these modules, two separate include files are used. The first file contains a block, with questions and routing. The second file contains the declaration of the fills and the set of procedures. The same modules are also inserted in the LMU.

Figure 1. Sections overview LMU



Every question consists of 7 elements:

- Tag; The tag is used to link questions in the Blaise Questionnaire with their counterparts in the LMU-database
- Name; The first part of every fieldname is uniquely coded by the tag making it easier to group the questions
- Description; Short description of the question text
- Question text; The text the interviewer should read to the respondent
- Help text; The text the interviewer can read for background information on the current question
- Answer; Several types of answers are possible
- Fills; The text variables that are used in the question text to rephrase questions based on certain settings in the environment

The elements are stored in separate fields in the online database. For each language we make the same fields available, but then empty.

The figure below shows the interface that defines a generic question. The elements all are visible in this interface. Also there are options for marking the texts with colors and putting in remarks. This will help us communicate with the translators better. The colors are used to indicate what parts of the question text are new, or what part of a text should not be translated. It is possible to formulate complex answer structures and add fills.

Figure 1. Generic interface for question LvAg14

Section: Family Module
Language: English (Generic)
Version: Understanding Society
main > section Family Module > LvAg14

Question

name	tag	description
LvAg14	FY_LvAg14	Who living with at 14

Remark

1.

question text [n] [m] [o] unmark

[m]SHOWCARD F3[/m]
Please look at this card and tell me [n]the number that describes [/n]who you were living with when you were [n]@B[/n]aged 14[n]@B[/n].

interviewer instruction [n] [m] [o] unmark

'STEPFATHER/STEPMOTHER' INCLUDES COHABITING PARTNERS OF NATURAL PARENT

help text [n] [m] [o] unmark

showcard: SHOWCARD F3

answer type

- Text — Max length: 0
- Number — Decimals: 0
- Memo
- Range ..
- Enum:
- Set, max:
- Array: 0 .. 0

Defined

	Description	Label
1.	Biological mother and father	
2.	Adoptive mother and father	
3.	Mother and stepfather	
4.	Father and stepmother	
5.	Mother/no father figure	
6.	Father/no mother figure	
7.	In [m]Local Authority [/m]care/foster hom	
97.	Other	

Add at position: 98

attributes: (none)

rules description

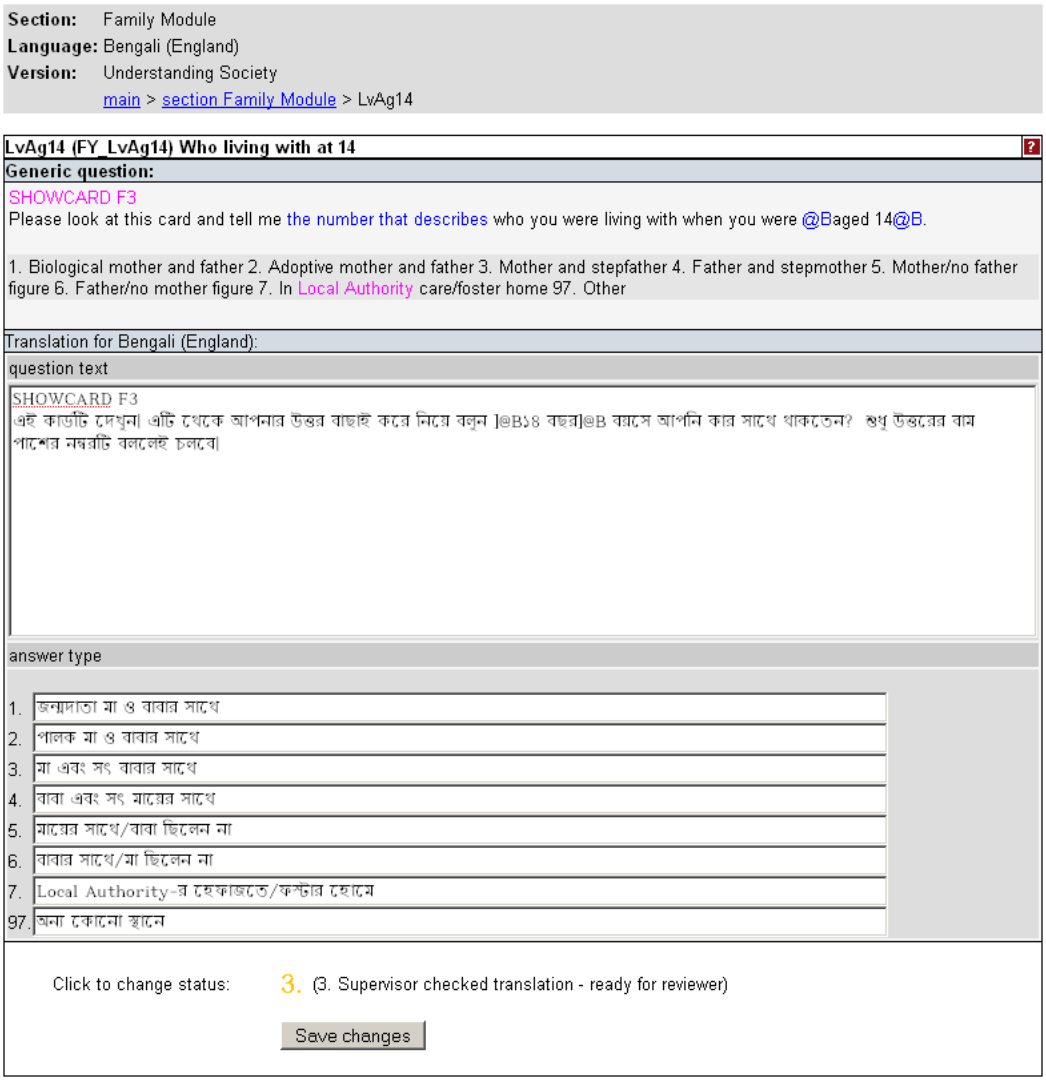
Available fills for this question:

Add new fill for this question:

Save changes | Delete question | Move question to: Benefit Gr3

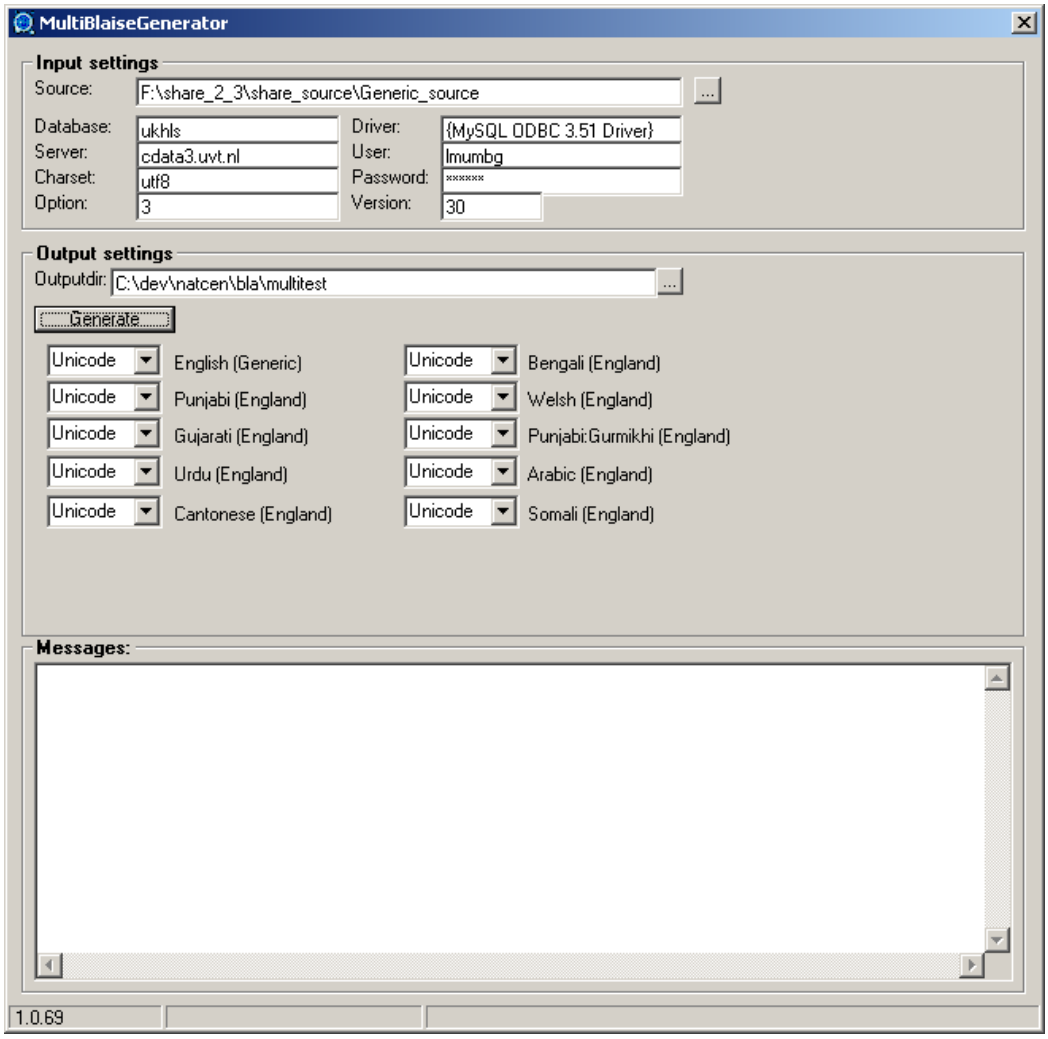
The translator has an easier interface. It shows only the texts that needs to be translated. The translator changes the status of a question from 1 to 2 when he is ready.

Figure 3. Interface for the Bengali translator for question LvAg14



When the translation is completed, we run the Multi Blaise Generator (MBG). It takes the elements from the online database and replaces them in the generic questionnaire. This will give us the source code of our original questionnaire, with all the selected languages added to it.

Figure 4. Screenshot of Multi Blaise Generator



2.2. Interview Instrument

2.2.1. Placing Unicode Text Inside a Blaise Questionnaire

Blaise uses an extended ASCII character encoding, with 1 byte per character. This system works fine for alphabets such as Latin, which have a limited number of characters. However, it is insufficient for languages with many characters, such as Arabic or Cantonese.

Unicode is the industry standard for representing text in most of the writing systems in the world. There are several mechanisms for implementing Unicode. Our solution is based on UTF-8, which uses 1-4 bytes for each character. UTF-8 was chosen because it is backwards compatible with ASCII, which makes it compatible with Blaise.

In a Blaise questionnaire, identifiers (such as field names) and text (such as question/answer text) are limited to ASCII characters. In our solution, we keep this restriction for identifiers, so the routing and text fills in Blaise work as normal. However, we replace the ASCII text for questions/answers with UTF-8 strings. This allows us to place multi-byte characters into the Blaise questionnaire, but since UTF-8 is backwards compatible with ASCII, Blaise treats them as ASCII strings.

With this setup, text processing works perfectly. Blaise treats all of the strings as ASCII, so it is able to compile the BLA source file into a BMI, and run the questionnaire. Because the field names are still ASCII, fills and other text processing work properly while running a questionnaire.

Even though text processing works, rendering does not work. Blaise cannot display the UTF-8 strings properly. This affects the usage of the Blaise Editor and the DEP. For the Blaise Editor, any other Unicode-compatible editor can be used, such as Notepad.

For example, the word "No" in Bengali is written as না, which is a combination of ন (pronounced "na") and া (pronounced "aa"). In UTF-8, the 3-byte code for ন is E0 A6 A9, and the code for া is E0 A6 BE. In ASCII, the codes E0, A6, A9, and BE map to à, ¦, ¨, and ¾, respectively. Therefore, the 6-byte code for this word, when rendered in ASCII, looks like "à¦¨à¦¾". Indeed, this is what the string looks like in the Blaise Editor.

Figure 5. TYesNo answer type, as rendered by the Blaise Editor

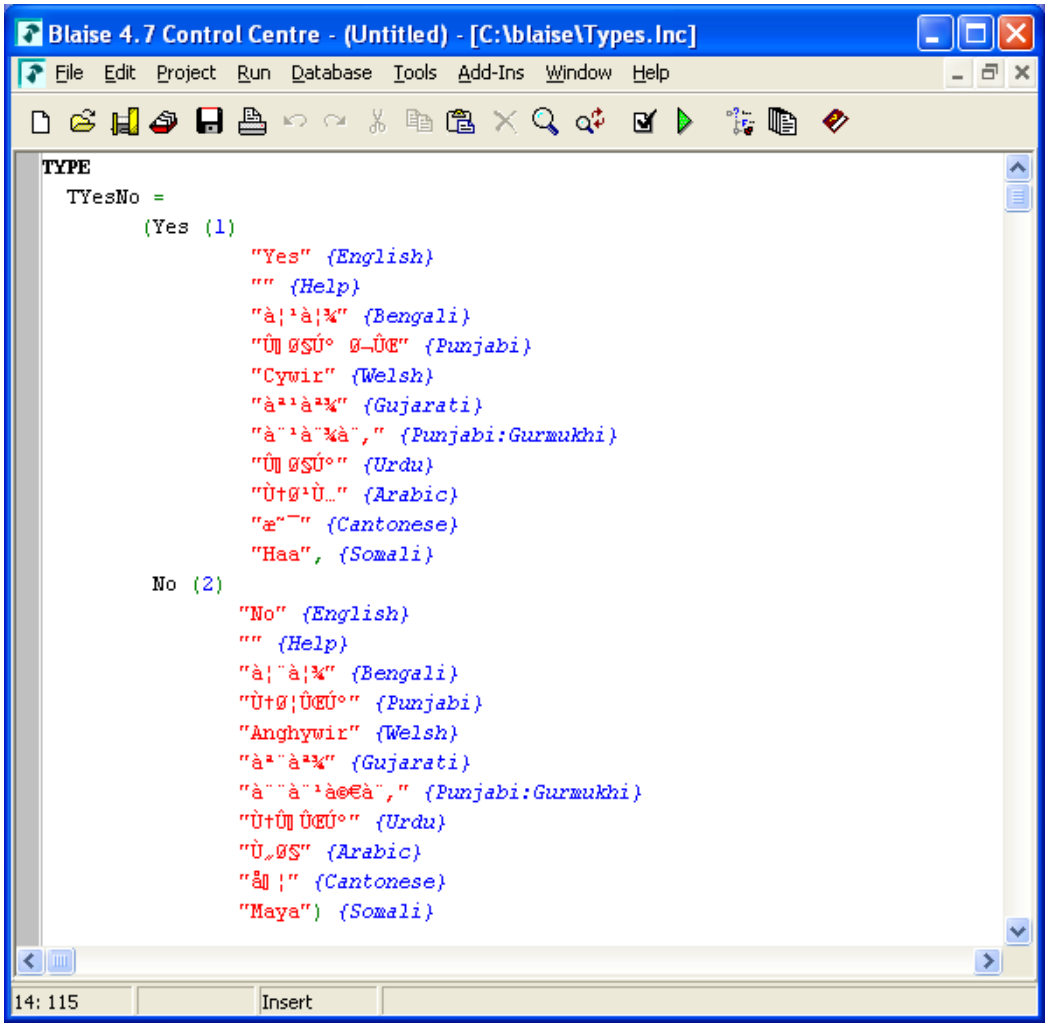
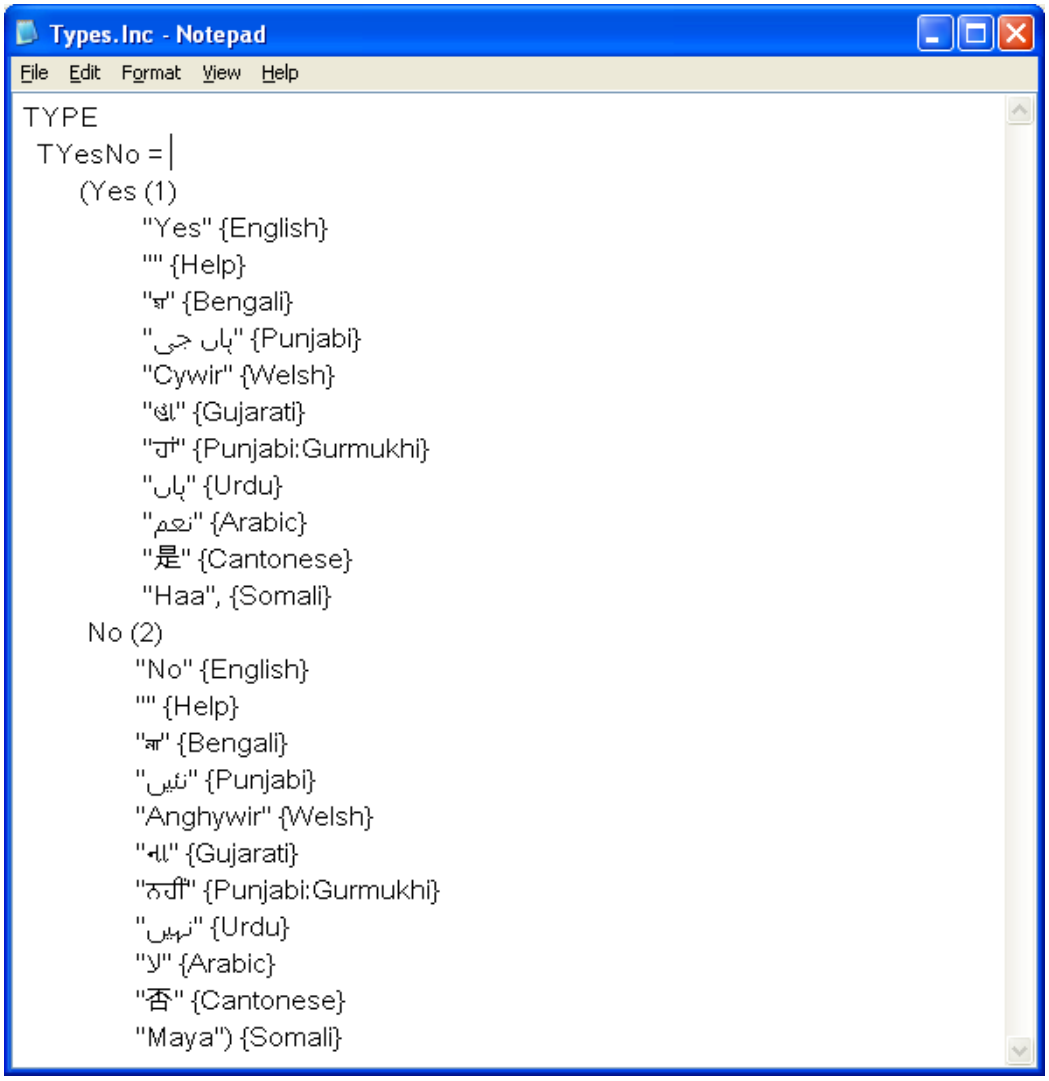


Figure 6. TYesNo answer type, as rendered by Notepad



A Unicode-compatible editor is fine for editing the questionnaire, and creating a BLA file. This file can be compiled with the Blaise Control Center, generating a BMI and BDB. However, when running the file, the Blaise DEP will display the strings as ASCII characters, instead of as Unicode characters. To overcome this program, the UNITIP program was created.

2.2.2. UNITIP

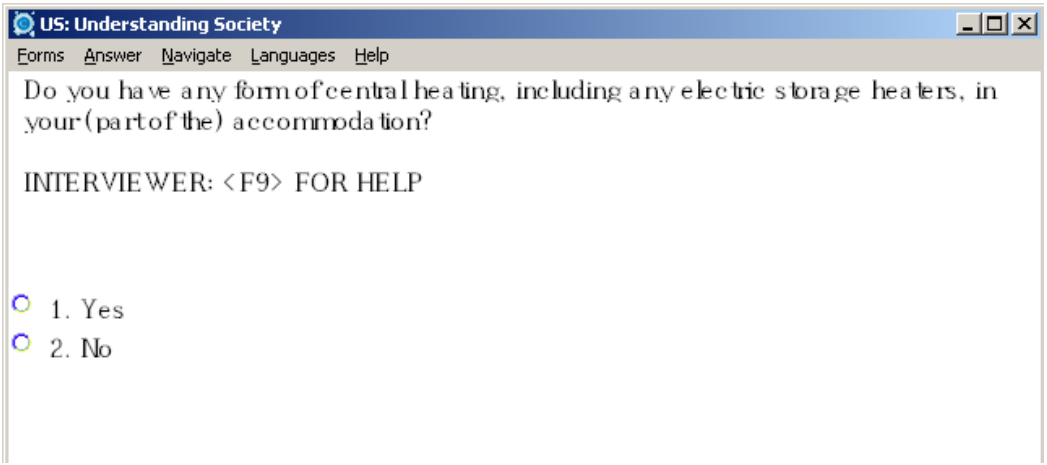
The Unicode Translation Interview Program (UNITIP) was developed to use the Blaise engine to handle questionnaire routing and text processing, but to render Unicode strings.

UNITIP uses the Blaise API to run the questionnaire. All routing and processing is handled by Blaise, to determine which question to display on the screen. Blaise will also handle fills, substituting the appropriate text as needed. While these text fills are really UTF-8 string, Blaise processes them as though they are ASCII. The Blaise API then returns these ASCII strings (that really contain UTF-8 strings) for the question and answer text.

The UNITIP application then renders these strings as UTF-8 strings. It handles left-to-right and right-to-left, as needed.

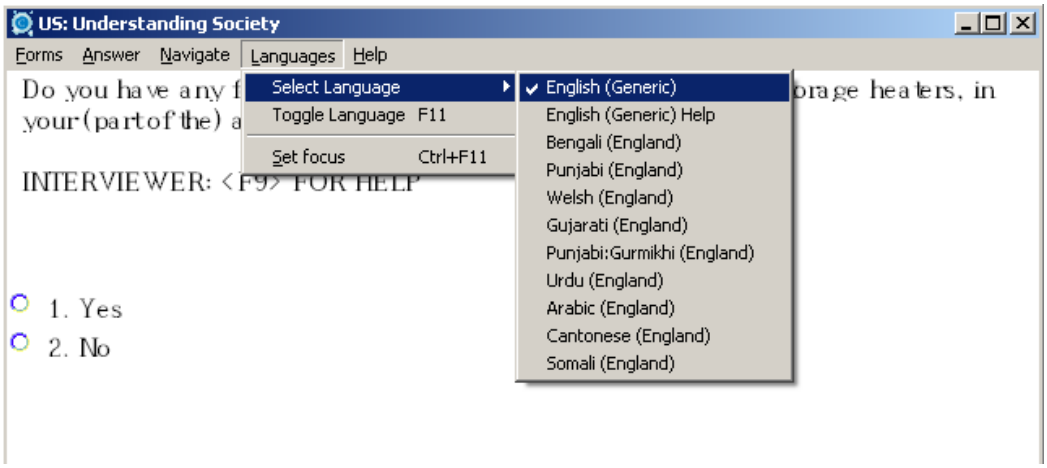
The screenshot below shows the UNITIP program showing a question in English.

Figure 7. An English question in UNITIP



Through the Languages menu, the interviewer can switch to any other language, at any time during the interview.

Figure 8. The Languages menu in UNITIP



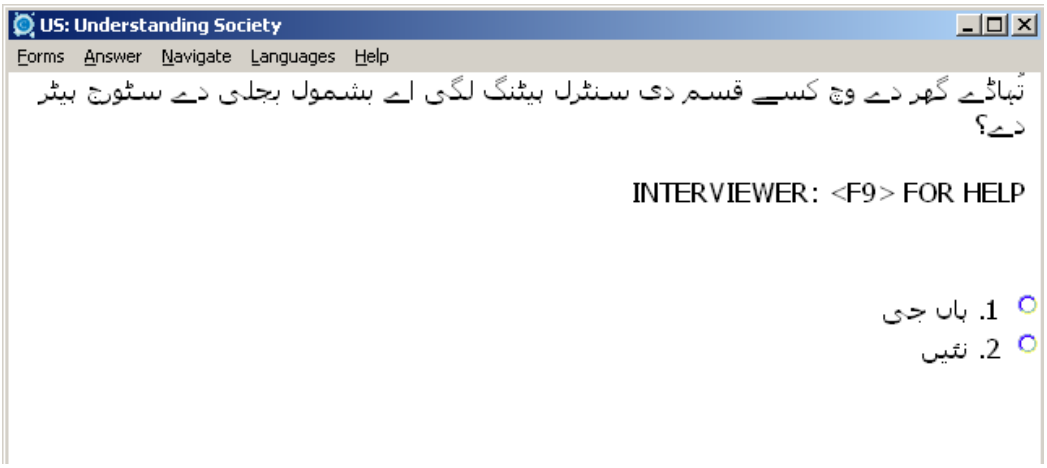
If the interviewer changes the language, the view switches to the new language. The following screenshot shows the same question in Cantonese.

Figure 9. A Cantonese question in UNITIP

If the new language is a right-to-left language, UNITIP will change the layout to right-to-left. Additionally, a

different font can be specified for each language. The following screenshot shows the same question in Punjabi, with a right-to-left orientation and a new font.

Figure 10. A Cantonese question in UNITIP



2.2.3. Fonts

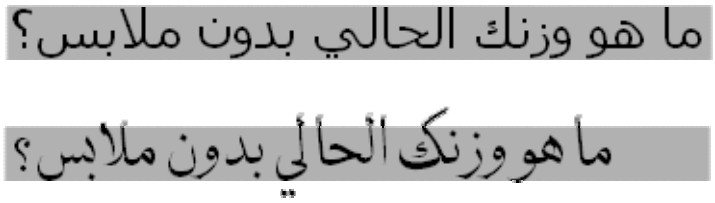
Once everything is set up for displaying a question, the final rendering is ultimately controlled by the font. Choosing a font that is easy for interviewers to read is an extremely important aspect of the questionnaire. However, there are several technical issues that must be considered.

UNITIP can render multiple languages in a single question. This is especially important in the Understanding Society study, as English text is mixed with the other languages for things such as interviewer instructions, names of cities/countries, and more. For each language, a font was chosen that also supports English. For a questionnaire that need to show even more languages in a single question, a font that contains all languages (such as Arial Unicode MS or Code2000) can be used.

The next issue to consider is the size of the font. UNITIP uses information from the Blaise BMI file for hints about the layout of the page. The size of many components, such as Grids, Field Panes, and Info Panes, is based on the size of the default font. The font chosen for each language must be the same height (or smaller) than the default language. This restriction caused a problem for Arabic.

The figure below shows the question "What is your current weight without clothes?" in Arabic. The top is rendered with the Tahoma font, and the bottom is rendered with a Nafees font. In general, Arabic readers prefer the Nafees font, as the varying line thickness and varying heights of characters looks better. However, the gray boxes show the visible area of each font, based on the standard height of the default font. Tahoma fits perfectly, but the Nafees font extends above and below the visible area. These parts of the character will not be visible in UNITIP, thus making the Nafees font unsuitable.

Figure 10. An Arabic question rendered with Tahoma and Nafees



Another issue to consider is bugs within fonts. In Unicode, accents and other diacritic marks are applied to a base characters. In some languages, several diacritics can be applied to a single character. Fonts have layout tables to determine how these should be rendered. We have discovered that some fonts have incorrect layout tables, causing characters to display incorrectly. For example, we discovered that a font used for Bengali was transposing 2 characters, thus rendering as an incorrect word.

Satisying all of these technical requirements, while also finding fonts that are acceptable for interviewers, is a task that should not be underestimated. There are many considerations to balance, but the result is one of the most important aspects of the solution.

3. Results

3.1. Interviewer Feedback from Translation Pilot

The software was remarkably successful. There had clearly been some glitches but many of these had been resolved. An outstanding issue was the F6 key (used to access more codes if not all appear on one screen) – both the fact that it didn't appear for all questions where it was relevant and that it didn't always work. Despite some freezes or glitches, it was well received by both bilingual interviewers and interviewer-translator pairs. Those undertaking interviews in English seemed to sense no difference from Blaise. Toggling was also clearly very successful – and very extensively used. CAPI translations clearly have the potential to increase the quality of the interview and the responses – and to reduce the time taken by interviews in one of the translated languages.

On the issue of time taken: the timings given by the bilingual interviewers suggested that they were not much longer than interviews undertaken in English. Any extra length in these interviews seemed to be a consequence of

- unfamiliarity with the translated questionnaire
- partly as a result of that extensive (even maybe excessive) toggling
- glossing of questions, where it was felt they were not immediately comprehensible (and sometimes to give a Sylheti version), which was again sometimes linked to toggling to check back what the question was about
- reading out of show cards, where people were not literate in the translated language.

Although there was some feedback that the data entry was not as fast as Blaise, even with the glitches in a pilot version the software was felt to be a considerable improvement on the old paper based system of translations.

3.2. Future Development

Although the overall project went fairly well, we have still some room for improvements. Some parts of the LMU should be made more dynamic. We like to improve our management layer. It should include the option to define user rights, workflow and give a better overview of the current status of the translations. The current version of UNITIP has not yet included the possibility of inserting media, but first experiments show this will not be that difficult.

4. Conclusion

The LMU/MBG/UNITIP solution can help create a multi-language questionnaire, with several languages that, in the past, have been difficult to implement in Blaise. This provides a great number of benefits to the interviewers, interview process, and the data collected for the Understanding Society study.