Managing Translations for Blaise Questionnaires

Maurice Martens, Alerk Amin & Corrie Vis (CentERdata, Tilburg, the Netherlands)

Summary

The Survey of Health, Ageing and Retirement in Europe (SHARE) is a multidisciplinary and cross-national panel database of micro data on health, socio-economic status and social and family networks of more than 30,000 individuals aged 50 or over. SHARE is coordinated centrally at the Mannheim Research Institute for the Economics of Aging (MEA, Germany).

The first wave of data collection was in 2004 (Denmark, Sweden, Austria, France, Germany, Switzerland, Belgium, the Netherlands, Spain, Italy and Greece), Further data have been collected in 2005-2006 in Israel. Two 'new' EU member states - the Czech Republic and Poland - as well as Ireland have joined SHARE in 2006 and participated in the second wave of data collection in 2006-2007. The survey's third wave of data collection, SHARELIFE, has collected detailed retrospective life-histories in sixteen countries in 2008-2009.

The tools discussed in this paper were developed by CentERdata; an independent research institute specialized in data collection and data analysis, housed at the campus of Tilburg University (the Netherlands).

The process of translating the questionnaires and creating the various country-specific versions is aided by several tools. The CAPI instruments are implemented in Blaise and the generic texts (English) of these questionnaires are stored in an external database (MySQL). The different countries translate their version of the instruments using a web application called Language Management Utility (LMU). The questionnaire is broken down into elemental units that can be translated independently. These elements are question texts, interviewer instructions, answer categories, fill texts, error messages and other texts needed for a proper display in the language translated. They are joined into country specific survey instruments, based on the blueprint of the generic version. Translation is done in a cyclic process, making it possible to make changes to the questionnaire during translation. Whenever the generic blueprint changes, the system indicates translation issues with flags. The translated texts are automatically inserted into the Blaise questionnaire using the Blaise Generator application that was also developed by CentERdata.

The tools have also been used to manage the translation process in the UK Household Longitudinal Study, for which a full Unicode supported Interview Program was written (Amin et al., 2009)

1. Introduction

The Survey of Health, Ageing and Retirement in Europe (SHARE) is a multidisciplinary and cross-national panel database of micro data on health, socio-economic status and social and family networks of more than 30,000 individuals aged 50 or over. Interviewers interview household members at their homes with a CAPI questionnaire.

There is a multitude of potential problems with a project of this scale. All countries or even sometimes regions involved have a different fieldwork organization. They have their own systems and workflow. Since SHARE is a panel study some information is forwarded into future waves. Sometimes refresher samples are needed.

The respondents speak different languages and write them with specific character sets, or grammar. They also have different cultural backgrounds, what leads to differences in distinction of manners approaching people. Each wave has a cycle of two years in which the questionnaire is designed, translated, tested and the data are collected.

CentERdata; an independent research institute specialized in data collection and data analysis, housed at the campus of Tilburg University (the Netherlands), was asked to help to program the questionnaires and guide the translation process. In time it became clear that we could also help in automating the complete chain of data-handling. Several tools were programmed; sample management systems, management information tools and tools to transform the data into datasets.

This paper focuses on the ideas behind the methods and tools CentERdata developed to manage the questionnaire programming and the translation process.

First some choices were made: It was decided to use Blaise because of its proved ability for CAPI surveys, the reliable handling of the data, the widely reputation of the program and the picked up knowledge and experience with Blaise at CentERdata. The questionnaire was originally designed in English by teams from various countries. This version is called the generic questionnaire. Afterwards translators had to translate that version. Translators should never make their translations directly in the Blaise source code. Copy- and pasting in translations from a text file into the source code by programmers is also something that should be prevented. Furthermore filling in strings into a question text based on some routing may need different Blaise code for different languages. It would be an ideal task for a system rather than a person.

To make a start we chose to separate each question into elemental translation units and automate the process of joining them back together. Translation is done via a web interface, assuring us the translated text will be in a format we can support and giving us the possibility to stay on top of things.

It should be possible to translate parts of the questionnaire during the questionnaire design hence shortening total development time.

Since some of the languages we had to support need non-Latin character sets we should develop methods to display these languages in the Blaise DEP. The texts were stored on our server in Unicode. We agreed upon a set of basic rules we apply in programming the questionnaire to ensure the translations can be fit into the questionnaire nicely.

Next we will describe the building blocks we used and the constraints we defined on the freedom of programming the Blaise code and how this is displayed in the online translation management tool.

2. Getting the lingual building blocks

Because of the complexity of the questionnaire a very clear structure was chosen.

The questionnaire is composed from several sections with a theme, for instance “income” or “health”. These sections each are defined as a block. Each of these blocks is stored in a separate include file. The blocks can also contain sub blocks still stored in the same file.

Questions and variables all have a name that has a reference to the name of the block. They also have a number, uniquely identifying the question, and a name-part to keep the code readable. These parts together make up the *name* of the question. Furthermore all questions have a *tag*. This tag is unique and can’t change. In Figure 1 an overview of one section is given. It shows the structure of question names, tags and how they fit into the section.

Section: ST Starting Section
Language: English (Ireland)
Version: Final Version Main
main > section ST

Questionname	Tag	Description	Action
ST001a_Proxy	(ST001a)	CHECK IF PROXY	
ST001b_Proxy	(ST001b)	VALIDATE PROXY	
ST002_Strt	(ST002)	START OF INTERVIEW	
ST003_name	(ST003)	NAME OF RESPONDENT	
ST004_namechk	(ST004)	CHECK IF NAME IS CORRECTLY RECORDED	
ST005_name	(ST005)	NAME OF RESPONDENT	
ST006_mnthob	(ST006)	MONTH OF BIRTH OF RESPONDENT	
ST007_yob	(ST007)	YEAR OF BIRTH OF RESPONDENT	
ST008_dobchk	(ST008)	CHECK IF DATE OF BIRTH IS CORRECTLY RECORDED	
ST009_mnthob	(ST009)	MONTH OF BIRTH OF RESPONDENT	
ST010_yob	(ST010)	YEAR OF BIRTH OF RESPONDENT	
ST011_gender	(ST011)	GENDER OF RESPONDENT	
ST012_strtcal	(ST012)	START THE CALENDAR	
ST013_introcal	(ST013)	INTRODUCTION OF THE CALENDAR	
ST016_proxycheck	(ST016)	PROXY CHECK	

View all questions for section ST

Figure 1; overview of section ST

The use of tags is essential because it links the database with translation texts to the questionnaire. All questions consist of these same basic elements. They are depicted in Figure 2.

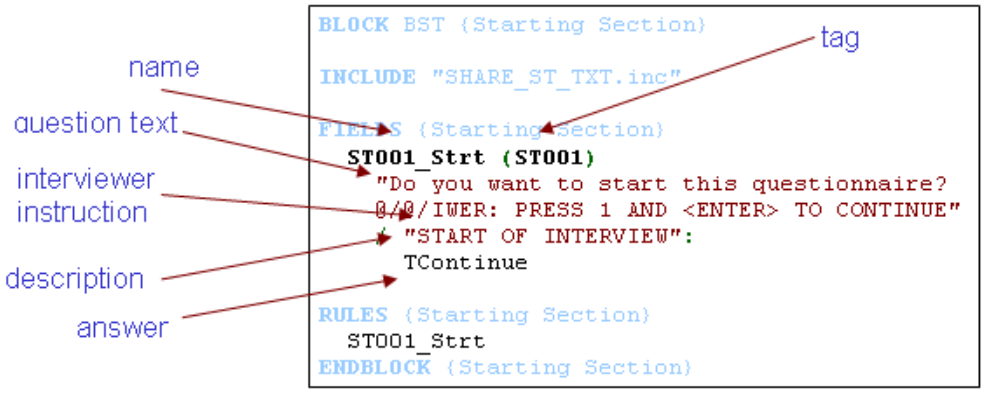

Figure 2; translation elements in Blaise source code

The same building blocks can be found in the online tool:

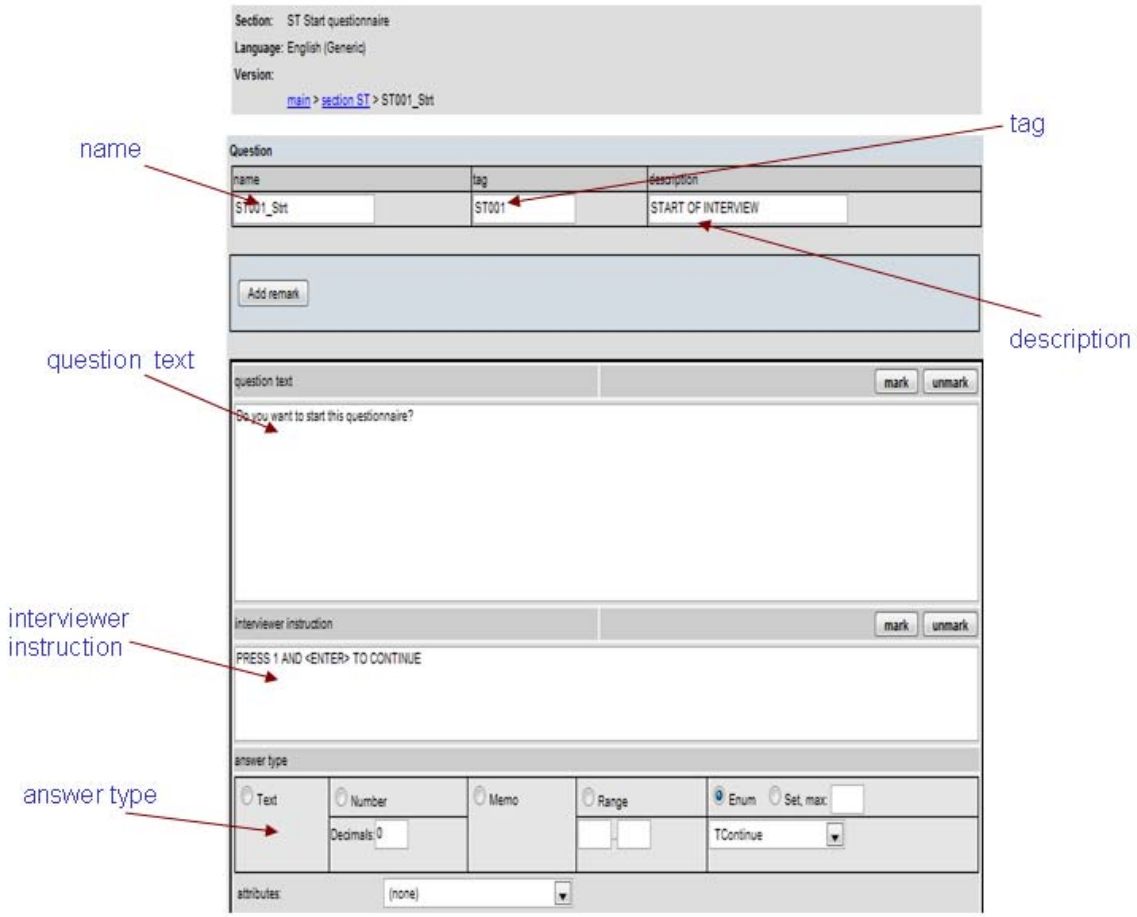

Figure 3; translation elements in the LMU

Each block consists of a set of questions, fills (text variables), a set of procedures for fills and rules. In this context we refer to fields as questions. The *question text* is the text the interviewer should read to the respondent, the *interviewer instruction* is for the interviewer to read, there is a short *description* for each question and finally each question has an *answer type* associated with it.

Answers can be structured in various ways. They can be globally defined answer types or a locally defined enumerated or set type. In general we define answer types global if they are used more then one time.

Some questions are dynamic. This means textual parts of the question change depending on certain environment settings. These variables are called fills and are the solution to many of our language specific problems. We make use of four different types of fills.

- Global fills; are variables that can be used throughout the questionnaire. They include for example birth year
- Answers to closed and open questions;
- Question specific fills; depending on the routing the question should be different in the generic questionnaire
- Translation fills; because of language specific reasons the question text should change based on an environmental setting. For instance based on the gender of the respondent the question should be phrased differently in certain language.

Figure 4 shows the interface of the form a translator should fill in to translate a question.

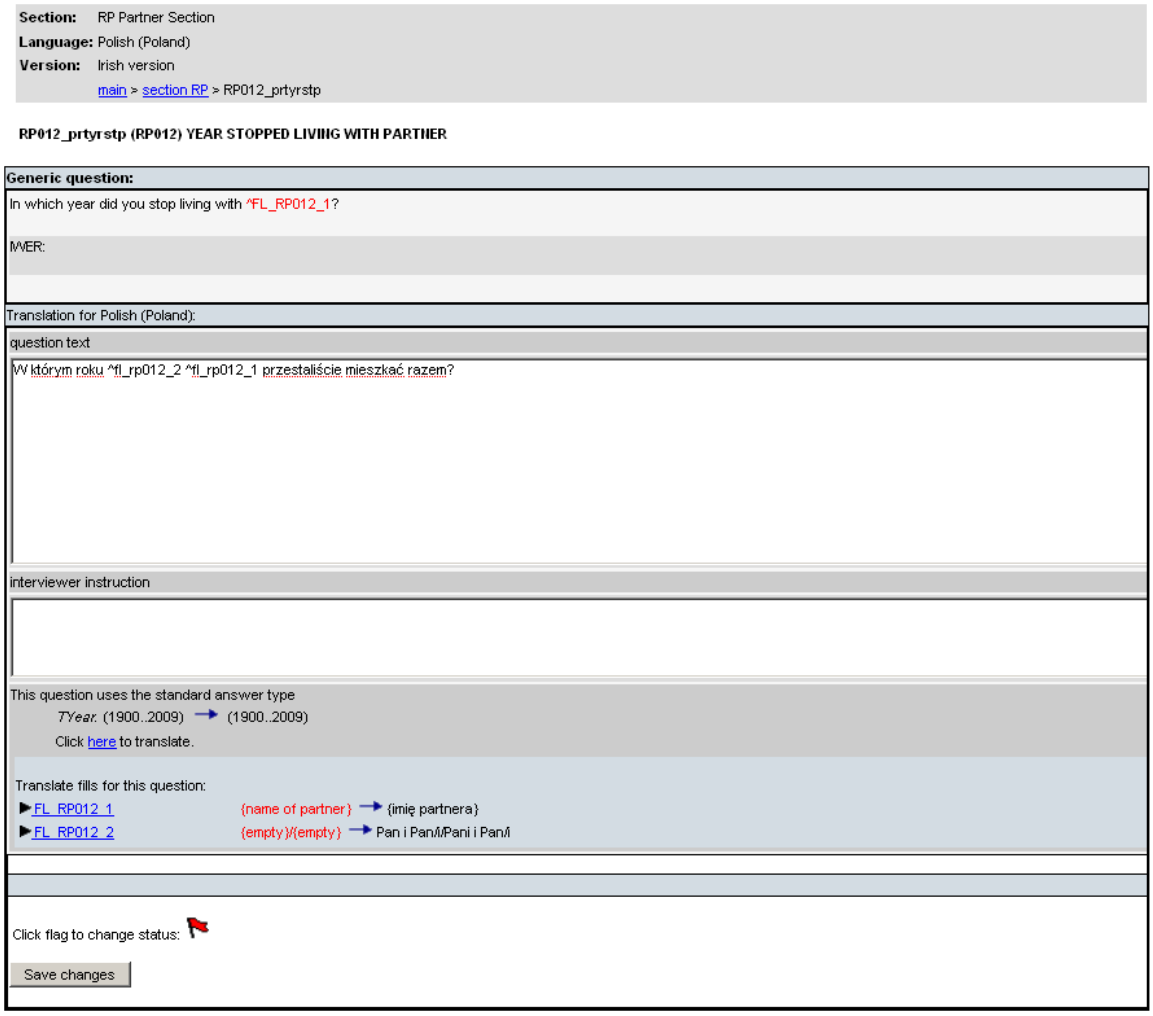

Figure 4; interface for the translator

On the left-bottom in this interface there are links for fills (FL_RP012_1 and FL_RP012_2). Most names of the fills start with “FL_” then the name of the section and the question number, underscore again and the serial number of the fill itself. Making it easier to identify what question a fill belongs to. The global fills and fills that reference other questions directly differ in this respect. The global fills also start with “FL_” but lack a reference to a specific question (e.g. FL_FirstName). The fills that reference questions use the question name. Clicking both links shows a form where the translation of the fill texts can be entered. “Rules” are shown to give the translator the conditions for the different texts in order to get the same sequences in all languages (Figure 5).

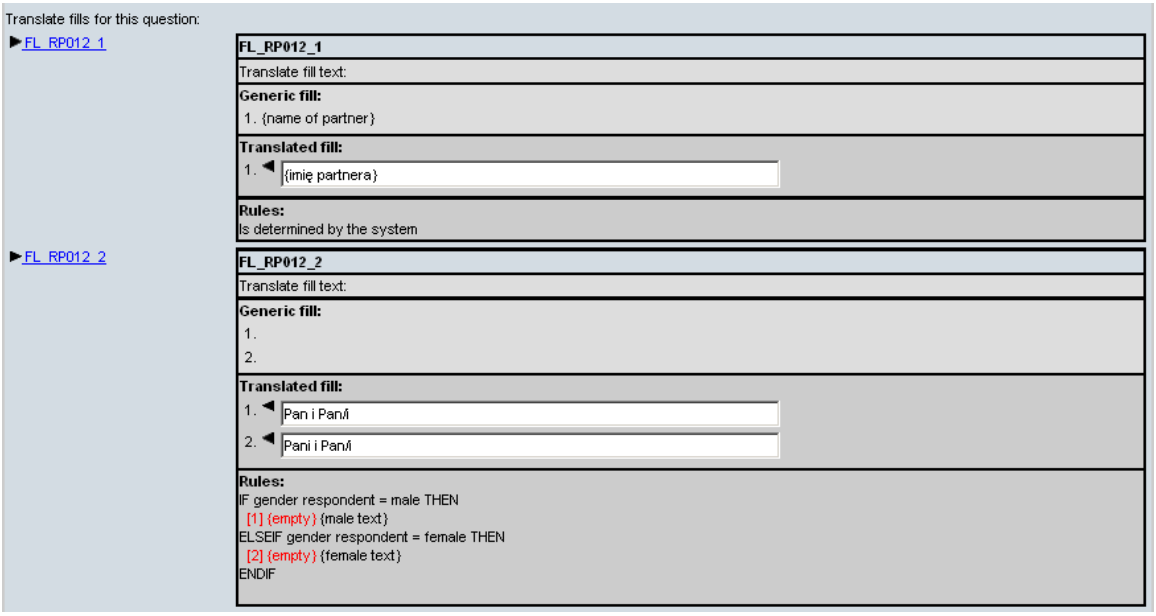

Figure 5; expanded fill interface

In the language management application all fills that can be used in a question are declared for that questions specifically, even the global ones. Translators are only allowed to use the fills that are defined for the generic question. However, translators don't need to use those fills. If a question can be translated without making use of the available fills they are free to do so. An example of how the translated fills of Figure 5 are loaded into the Blaise source code is shown in Figure 6.

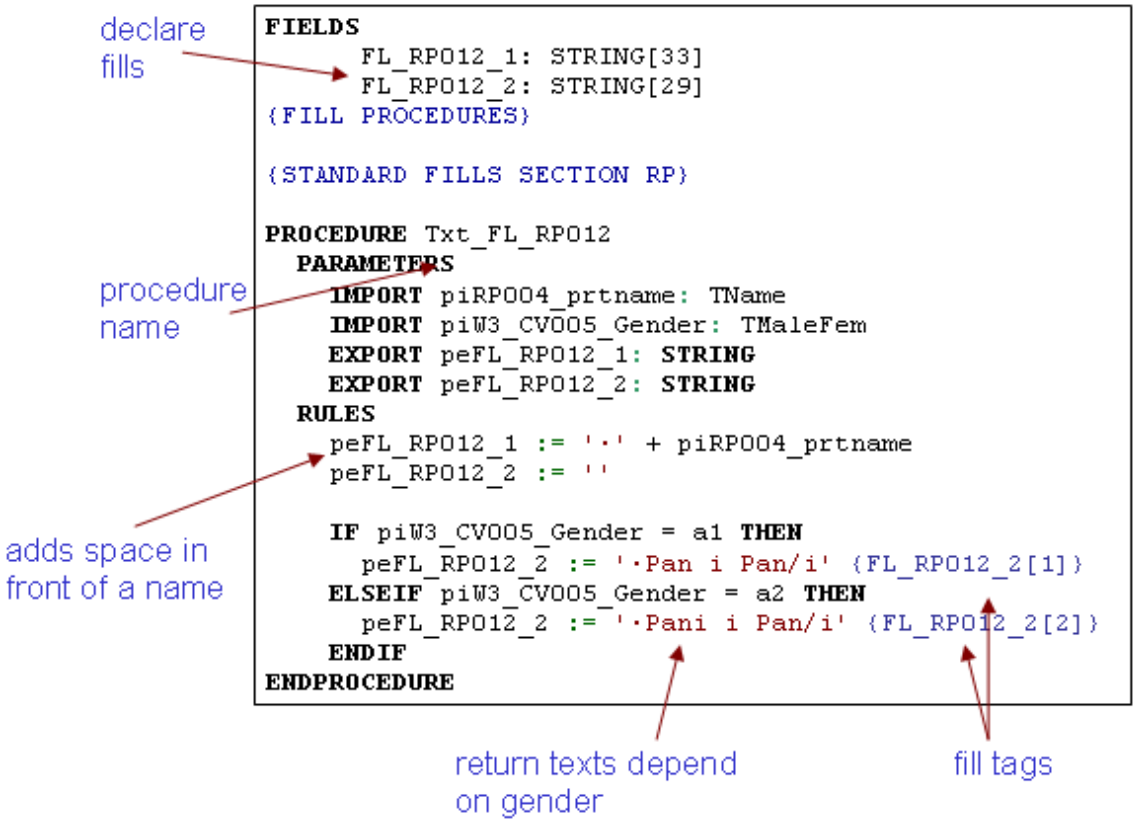

Figure 6; example of translated procedure

Note that the translation for FL_RP012_1 is not used in this source code. The code simply adds a hard space in front of the variable that is given to the procedure. The translation for "{name of partner}" is used to make documentation better readable.

There are other texts that are displayed during an interview while using the DEP. They should also be translated but don't need this special structure. We have made them available in the LMU. These include:

- Error texts
- Special global variables like days of the week, name of months, country names.
- Standard types; they can be accessed at question level, but there also is an overview available where all standard types can be seen. See Figure 7.

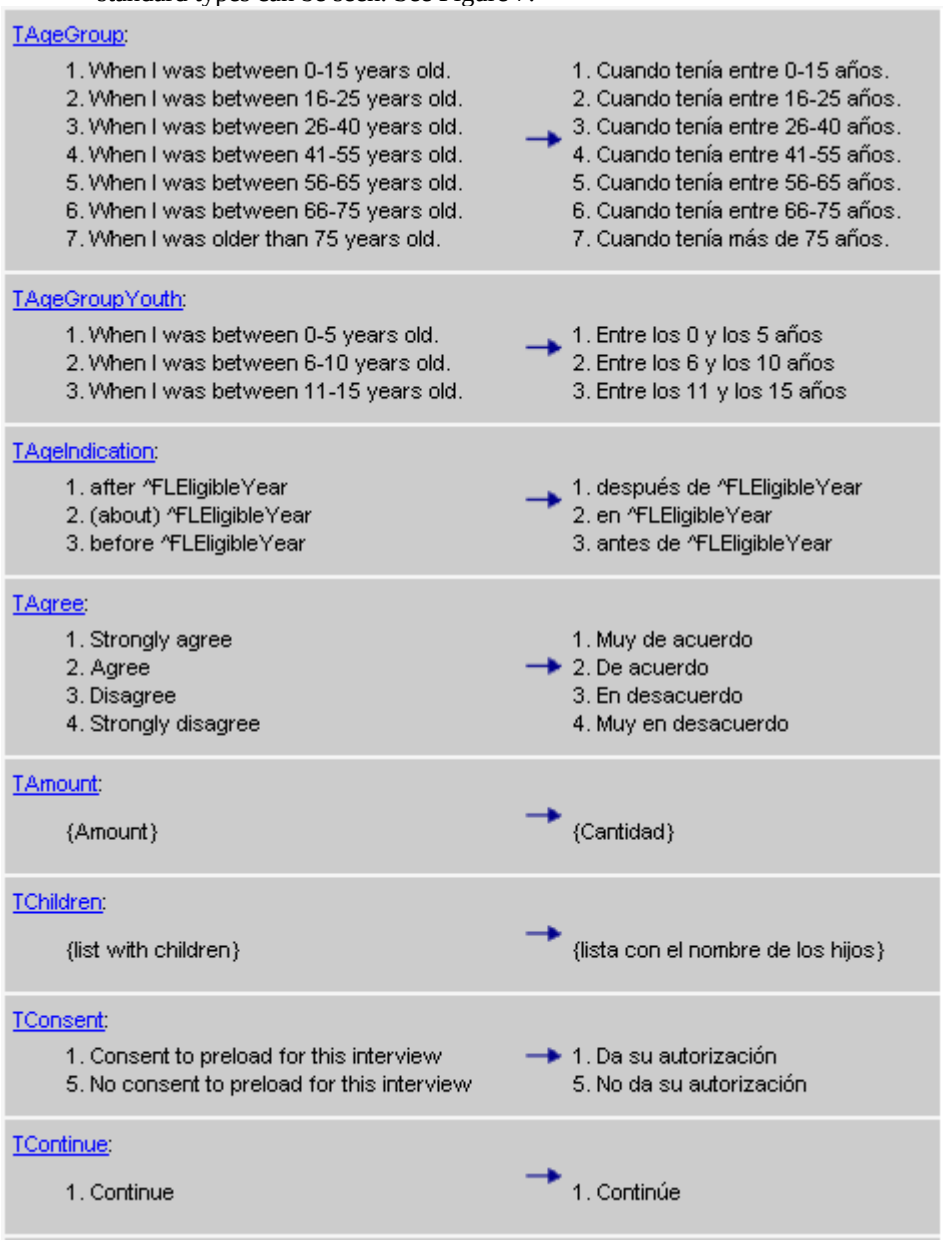

Figure 7; Some standard types in the LMU for a Spanish translation

The LMU can also be used for the labels and texts we use in the tool that work with the DEP:

- Labels for sample management
- A calendar tool was implemented for SHARELIFE
- Help texts are exported to XML

Through the LMU we were able to 'shield' all programming tasks from the translation process. It also provided us with tools to monitor the translation process. Each question that is new or has changed is flagged red. The translators change the status of the flag to yellow (meaning: working on, have some question about it or ready for revision) or green (meaning; this question is translated). The flags made it easier to apply changes to an already translated questionnaire; translators would see immediately what questions were changed. A remark field gives them some extra information on why the change was made.

Based on the status of the flags an administrator interface was build that gave insight in the current status of the translation process.

3. Inserting the translation in the questionnaire

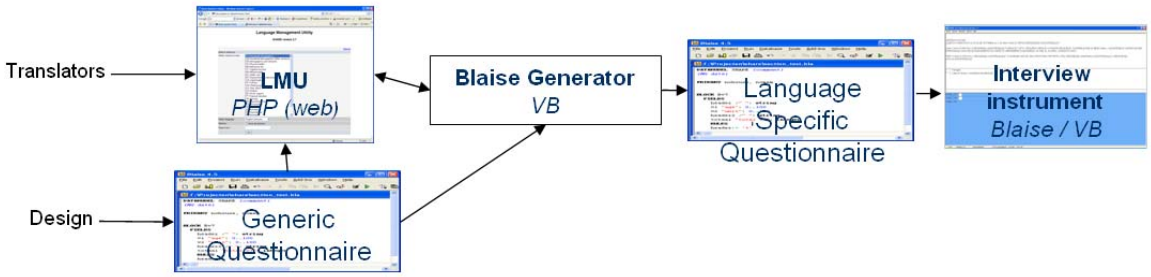

Figure 8; the translation process

To load questions from the LMU into the Generic Blaise Questionnaire a tool named "Blaise Generator" was developed. It takes the building blocks and replaces the texts in a copy of the generic questionnaire. Because some rigid coding constraints were taken into account it can find these texts by simply scanning the text for the tags and replacing the question-elements piece by piece.

It then walks through the fills in the online database and replaces the instances that are defined in the original questionnaire. While doing this, the maximum size of a fill text is determined and the length of the string adapted accordingly.

The global fills and standard types are generated into separate files. The other special translations go to other formats depending on how the tool was written to handle text.

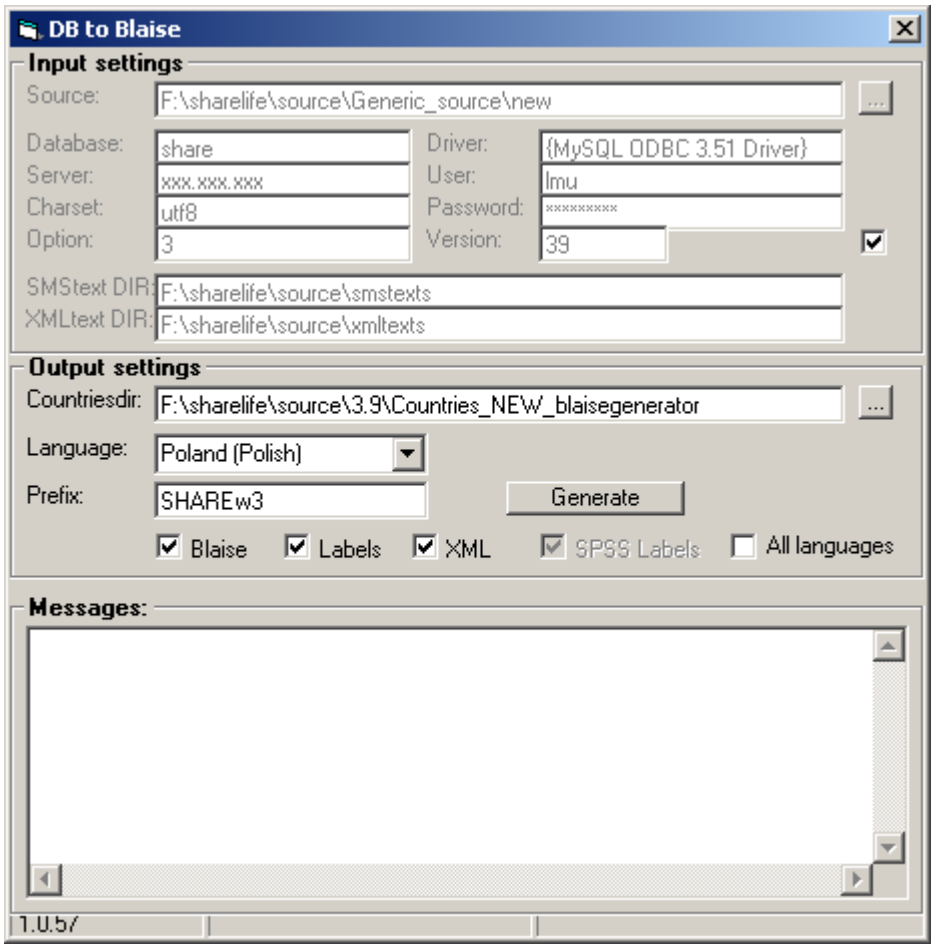

Figure 9; Blaise Generator interface

Since the translations are written in Unicode and stored as UTF-8, we need to convert the texts to display them in the DEP properly. The DEP only displays extended ASCII.

The information and references in the paper Rob Groeneveld presented at IBUC 2003 named “Using non-Latin alphabets in Blaise” was used to determine what format the text in the Blaise code should have.

In general the Blaise Generator has a formula per supported character set of how to translate a given UTF-8 encoded character. For most SHARE-countries this formula is very easy, since the UTF-8 encoding is equal to Latin-1. For Latin-2 (Czech Republic, Poland), Greek (Greece), Hebrew (Israel) and Cyrillic (Israel) the formula could either be subtracting a value from the UTF-code or using a matching table.

For Arab (Israel) the situation is more complex. The way a character displays on screen depends on the position of that character in a word and the surrounding characters. Based on finding in the paper “Using Arabic in Blaise” written by Shani Abel and Shifra Har there was decided to use multiple fonts and let the DEP change fonts when a character is available in another font. The translated questionnaire was analysed and based on the occurrences of characters the fonts were developed, minimizing the number of font switches. The formula for matching the right Unicode character to the right extended ANSII character and the right font was very difficult, but in the end it worked.

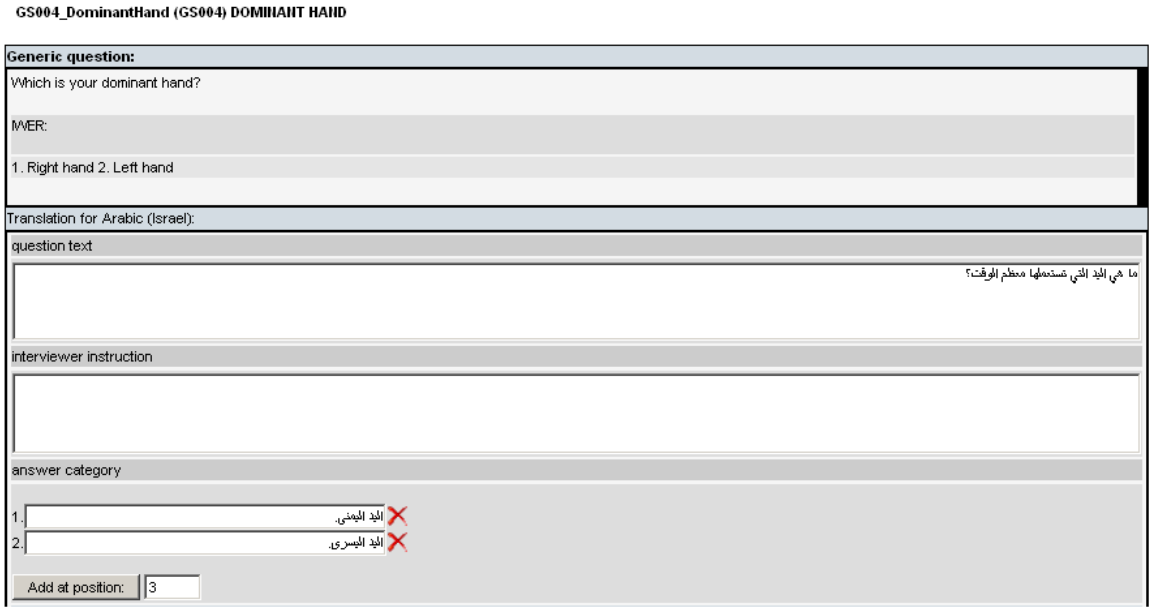

Figure 10; Arab example in the LMU

Will be placed in the questionnaire as:

```
GS004_DominantHand (GS004)
    "@>@k @k@kÄ@kÖíÞ@k¯@k áÇËâ @k°@këßãË@kÌ@k³@kË@k ñ@kÌ@kÞ@k¯@k ®óÞ@k¯@k ñê @k°@kâ"
    /"DOMINANT HAND":
      (a1 (1) ".ïçãóÞ@k¯@k ®óÞ@k¯@k .1",
       a2 (2) ".î-³óÞ@k¯@k ®óÞ@k¯@k .2")
```

Figure 11; Arab text exported into Blaise Source

In Figure 9 the translation is imported into the questionnaire, some characters have @k surrounding them. This means that for this character a different font is used.

The Blaise generator gives us feedback during its process. It indicates whether there are missing tags or fills, and checks for missing answer types. By default it will keep the generic version of a questionnaire element whenever it finds a problem.

When the questionnaire has been pasted in we have a new version of the Blaise source code in another language.

4. Conclusion

The tools that are developed for the SHARE-study have proven to be very valuable. It shortened the total questionnaire development and translation time of these large-scale projects. Furthermore, since we had our building blocks residing in a relational database, we were able to build other tools that generated paper versions of the questionnaire, and it was even possible to build a data-dissemination interface on it.

We have been continually further developing the tools. In future waves of SHARE the complexity of the questionnaire will grow because of changes in the sample, new respondents joined, households split, and respondents die. We remodelled our Sample Management System several times, and every version had some specifics to translate. Feed forwarding of previously gathered information had a huge impact on the structure. In the most recent wave a calendar application was included in the questionnaire. We managed to adapt our tools to these changing circumstances within the project.

The tools have been further developed for the “Understanding Society”-study. This study is run by the National Center for Social Research (London, UK). We had to rebuild our tools to their environment, and leave it for their

people to manage. This resulted in a major update to the LMU regarding the roles that are allowed in a translation process and what phases can be determined. A new program called the “Multi Blaise generator” was built that places multiple languages in one questionnaire. Together with the associated Unitip program it now fully supports all questionnaires in all languages and scripts, making the struggle we had with getting the Arab texts in the questionnaire a nice experiment but obsolete. For the next wave of Share we are a planning to use Unitip.

We plan to keep updating our tools, making them easier manageable by third parties. New interfaces will shorten the support time the IT department has to put in. We think about improving on versioning and porting the translations to an online interview interface.

5. References

Rob Groeneveld, Using non-Latin alphabets in Blaise, 2003

Shani Abel, Shifra Har, Using Arabic in Blaise, 2004

Alerk Amin, Richard Boreham, Maurice Martens, Colin Miceli, An End-to-End Solution for Using Unicode with Blaise to Support Any Language, 2009