Questionnaire Specification Database for Blaise Surveys

Lilia Filippenko, Valentina Grouverman, Joseph Nofziger, RTI International

1 Introduction

Developing large scale, complex Computer Assisted Interview (CAI) instruments is a team effort that presents many challenges. When a group of specification writers, translators, programmers, and testers are working in parallel, specifications can quickly get out of sync with program code. The Questionnaire Specifications Database (QSD) facilitates all facets of the CAI development life cycle and streamlines the process of creating Blaise instruments, especially those with a second language and/or Audio Computer Assisted Self Interview (ACASI).

More than a common code generator, QSD allows spec writers and translators to make iterative changes at any point in development, while maintaining the integrity of the instrument. Changes to wording, question fills, and response types are inserted directly into the Blaise instruments. QSD produces specification documents that are complete with question text, response options, fills, and routing comments. Since these specifications and the current Blaise code are both generated from QSD, they are always in sync. Extensive support for ACASI includes script generation, audio review, and synchronization of code with audio files. Issue tracking and change logging are built-in features.

2. Life Cycle and Changing Requirements

QSD supports the full life cycle of large scale instrument development, beginning with the specification of requirements. The questionnaires in such efforts may have multiple sections, a high degree of complexity, multiple languages, or a combination of these factors. Thus, instrument development requires a team effort. Each member or subgroup of the team has a stake in the outcome, and therefore influences the requirements.

Changing requirements are a reality of software development, and instrumentation is no exception. In a typical scenario, the primary source of requirements is the client, or possibly more than one client. Managers may be forced to demand cuts based on budget constraints or schedule. Survey specialists create programmer-ready specs, consulting methodologists for refinements and ensuring usability. Review boards, internal and sometimes external, may need to approve the specifications. This scale of development often demands multiple programmers, who discover inconsistencies or suggest changes to simplify coding. Thus begins an iterative cycle of spec writing, coding, review, testing, and revision involving all these parties and more, from quality assurance to translators.

Figure 1. Questionnaire requirements come from many sources.

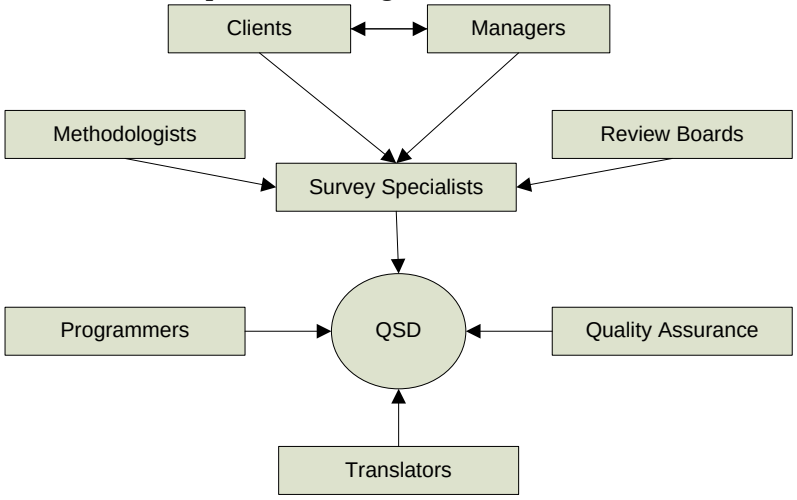
Stakeholder	Focus
Clients	Elicit complete, correct, consistent data
Managers	Balancing client needs with budget and schedule
Survey Specialists	Interviewer usability and training
Methodologists	Bias in wording; Visual representation; User interface
Review Boards	Respondent protections; Compliance; Sensitivity
Quality Assurance	Function versus specs
Translators	Language constructs
Programmers	Correct function; Adherence to specifications

QSD helps to keep the task manageable. The key to accomplishing this is the structure it imposes onto the program code. This structure, described below, facilitates quick incorporation of the most common changes we encounter: wording changes. These can be entered by non-programmers and incorporated immediately into a working instrument without fear of side effects or coding errors. This speeds the development cycle by quickly getting the new version to the testers or back out to production.

For multi-language surveys, not only question text must be translated, but also every type and fill. Translators see all question elements together and enter their translations, which are included in the generated code without any programming. A comprehensive view of untranslated questionnaire elements helps ensure complete coverage.

The stakeholders, with their various perspectives, benefit from customized views of the specifications. Since the question definitions are edited only through QSD, its contents are guaranteed to match the instrument, making production of accurate spec documents at any point in the process a trivial task.

Figure 2. Stakeholder input flows through the team into QSD.



All entered changes are logged. Problems found in testing are recorded directly in QSD.

In the next sections we explore in greater detail how users interact with the system.

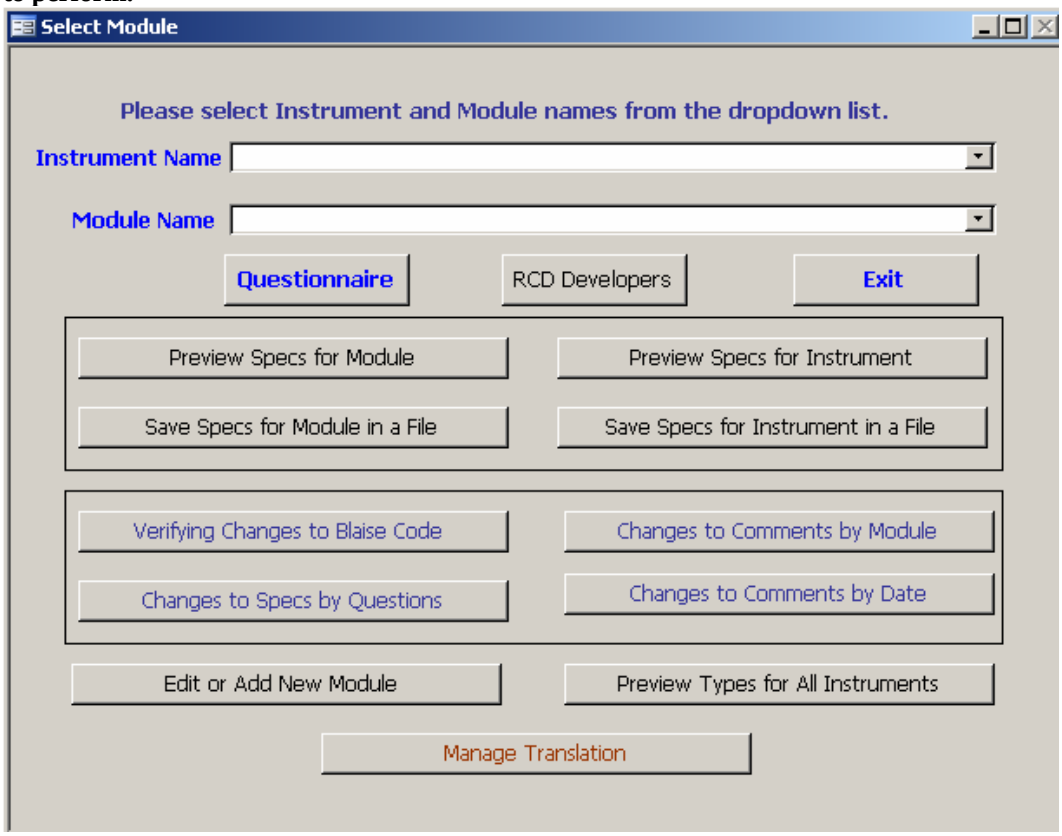
3 Using QSD

QSD consists of tiers for the user interface and business rules, and a back end relational database. Microsoft Access was chosen based on its simplicity, its familiarity among the programmers, and to ease the transition from working with specifications exclusively in word processing documents to using a central system.

Each survey team uses a dedicated instance of QSD. Specific information such as instrument names, languages, and use of ACASI, is saved in a survey configuration table. User access is managed internally.

Users log in to the system using their network username and a QSD-specific password. After being verified as authorized on the survey, the user is presented with several options.

Figure 3. From QSD's initial screen the user selects an instrument, a module, and a function to perform.

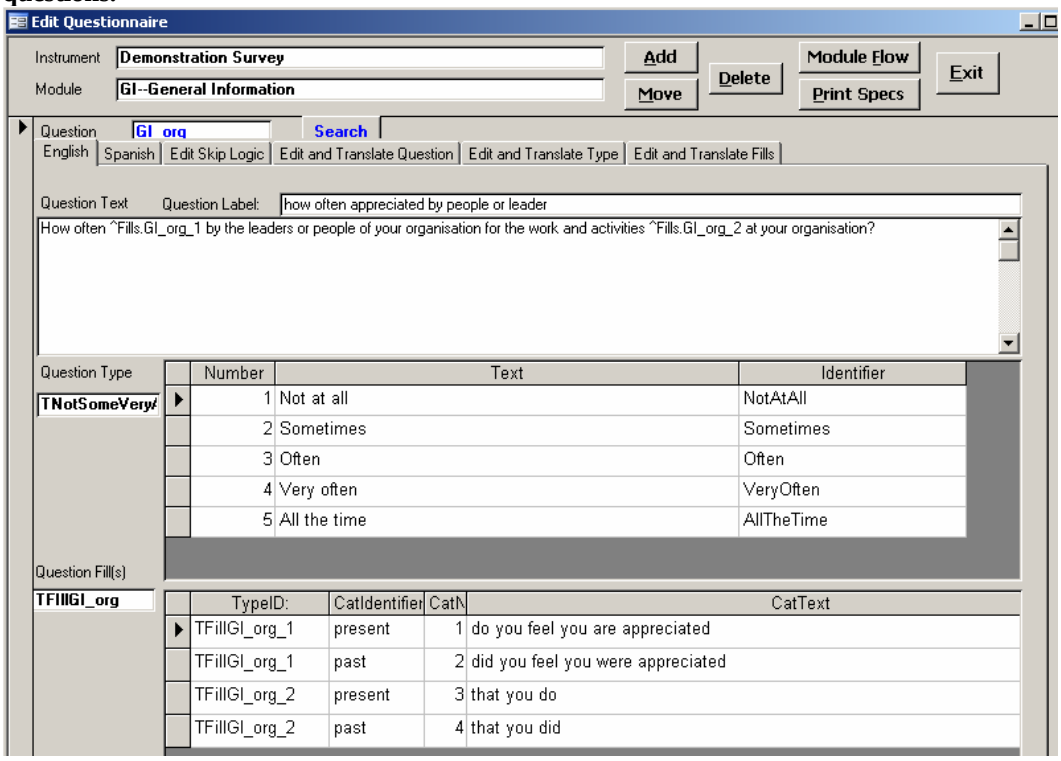


On the "Select Module" form, various actions are available depending on the user's access level. From here the specs can be viewed as a Microsoft Access report, or exported to a Word document. Entered spec changes can be reviewed directly in the changes table or printed as reports in a more readable format. The "Manage Translation" option leads to additional functions for translators.

When requirements for a module are changed or a module is ready for translation, the user selects "Questionnaire" from the "Select Module" form. The resulting "Edit

Questionnaire" form is where all changes to the specs for the selected module are entered.

Figure 4. On a single screen, users can change wording and translate, add, move or delete questions.



3.1 User Levels

Since many people with different needs have access to QSD, we want to be sure that only appropriate options are available to each user. There are three levels of users in QSD:

- Standard users;
- Questionnaire spec writers;
- Questionnaire programmers.

Translators and Testers are given access as "Standard user". Survey specialists have "Spec writer" access.

Translators can see all pages on the form "Edit Questionnaire" but they can make changes only on the pages "Edit and Translate Question", "Edit and Translate Type", and "Edit and Translate Fills".

In addition to standard functions, survey specialists can modify the text description of the skip logic for a question and add or change question attributes. If there is a need to add a new response option to a type or a new fill to a question, a request is added to a field named "Comments" on the "Edit Skip Logic" page. This field holds information about all changes made to the question during development of the instrument. Programmers will use this information to complete the implementation.

Our goal is to partition responsibility in instrument development by allowing the spec writers and translators to make wording changes directly, thus freeing programmers to concentrate on programming the instrument flow. The button "RCD Developers," on the "Select Module" screen referring to RTI International's Research Computing Division, is available only to programmers. A programmer can use QSD as follows:

- Load modules (sections), question types with responses, and question fills into QSD from a Microsoft Word document;
- Add new response options (question types) and new question fills;
- Export changes made by specs writers and translators into Blaise code;
- Create scripts for audio files and generate Blaise code to play them.

3.2 Monitoring Changes

A key feature of QSD is its ability to track all changes made to the specifications. Every time a change is saved, information is recorded about who made the change, the date and time, and the kind of change made. By selecting the button "Changes to Specs by Questions" from the "Select Module" screen, users can filter, sort, and prepare custom change reports.

This is very helpful for translators to check for updates after revisions are made by the spec writers. Programmers use it to determine which modules need fields updates, or there is a need to re-populate types and/or fills.

Central logging of changes and change requests streamlines communication, reducing the confusion and wasted time characteristic of poorly coordinated email exchanges.

It also allows monitoring the history of all changes that were made to any given variable of the questionnaire. By looking at the corresponding version of the variable in Microsoft Visual Source Safe, we can tell exactly what was changed. While all previous versions could be saved within QSD, we prefer to require the use of source code control for our instruments.

3.3 Monitoring Problems in Testing

During the testing phase, testers and programmers use QSD to log problems as they are discovered. After a problem is corrected, a programmer marks the problem as fixed. When the whole module is updated, the programmer records this using the button "Verifying Changes to Blaise Code" from the "Select Module" screen. QSD has reports which show the list of modules that need verification by programmers and by testers. Before a new version of the instrument is released for testing, all modules are verified by programmers. When many programmers are working on the instrument, this report helps the build manager to be aware of the statuses of modules. Testers note the test outcomes for each module. All modules are verified as having passed testing before being released into production.

3.4 Producing Targeted Specifications

Since QSD is the single repository for the specification from which the code is generated, it is not necessary to check consistency of wording, question order, and so on, between specifications and program. It is very convenient for testers to do their jobs and for questionnaire designers and clients to review specifications at any point of the development process, with confidence that what they are viewing is in step with the current instrument.

Figure 5. Specification documents show pre- and post-logic, label, tag, fills, and response options.

Logic before:
IF GI_work=yes then ask GI_org

GI_org *[GI2] how often appreciated by people or leader*

How often ^Fills.GI_org_1 by the leaders or people of your organisation for the work and activities ^Fills.GI_org_2 at your organisation?

Question Type: *TNotSomeVeryAll*

NotAtAll	1 Not at all
Sometimes	2 Sometimes
Often	3 Often
VeryOften	4 Very often
AllTheTime	5 All the time

Question Fill: *^FillGI_org_1*

present	do you feel you are appreciated
past	did you feel you were appreciated

Question Fill: *^FillGI_org_2*

present	that you do
past	that you did

Logic after:
IF GI_org = notAtAll skip to GI_why

QSD produces a specification document in a format that shows all elements of each field. Specifications can be printed in English only, Spanish only, or both languages at once for a selected module or an entire instrument. They may be exported in any format supported by Microsoft Access.

4. Generating and Updating Blaise Instruments from QSD

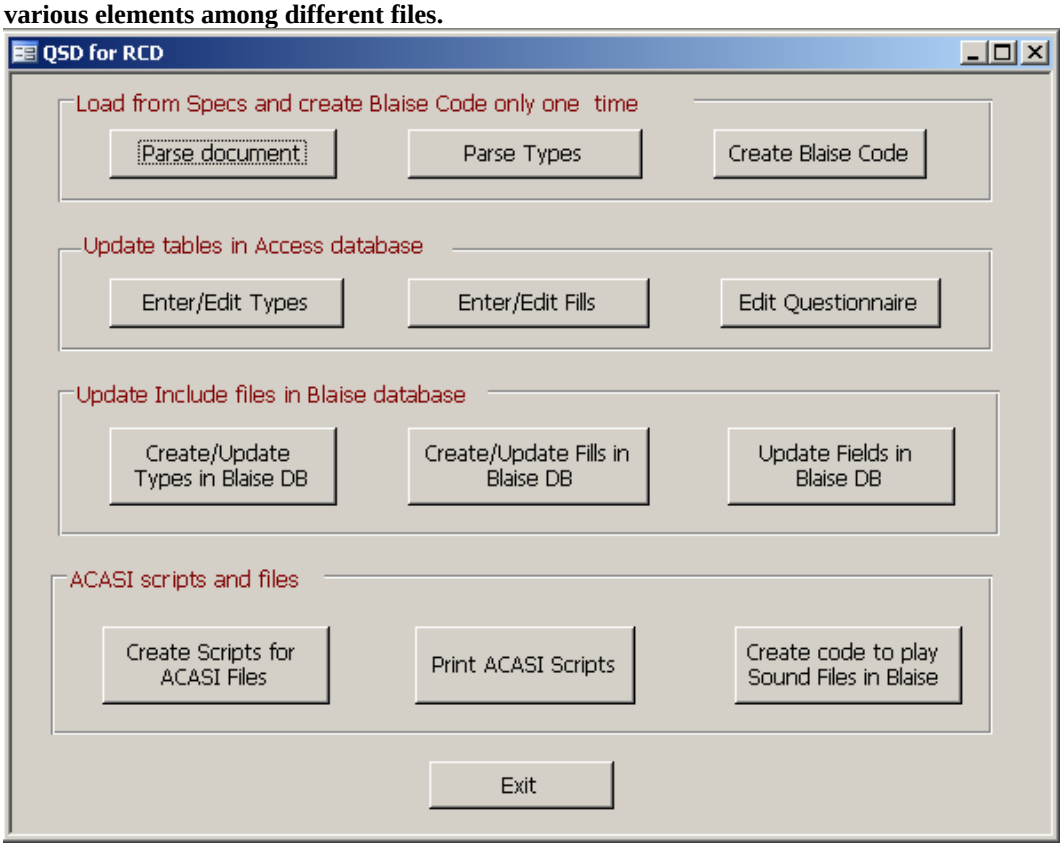
Blaise promotes modular questionnaire structure. A modular structure delivers efficiencies in code maintenance, division of labor, and data model preparation (compilation). In addition to these advantages, a modular structure facilitates the code generating and updating capabilities of a tool such as QSD.

Each section in our Blaise instrument is programmed as a separate block. All these blocks are incorporated into the instrument as included files. All types used in the instrument are compiled into one INCLUDE file. Fills declarations and their types are also separate INCLUDE files.

Since Blaise allows the separation of fields and rules at the block level as well, each block has an INCLUDE file which contains all fields associated with the specific block.

This important feature of Blaise allows dividing responsibilities between programmers and survey specialists.

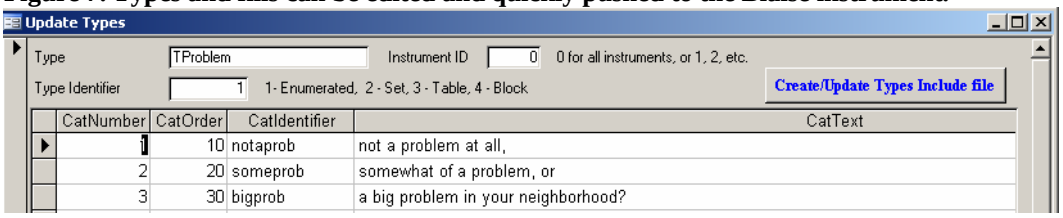
Figure 6. The interface for programmers reflects the distribution of the Blaise instrument's various elements among different files.



QSD is capable of simultaneously loading all user-defined types needed by the Blaise instrument. In order to do this, the programmer uses the "Parse Types" option and selects a properly formatted Microsoft Word document.

Types may also be added or edited at any point during the development process by using the "Enter/Edit Types" option.

Figure 7. Types and fills can be edited and quickly pushed to the Blaise instrument.



The same is applicable to fills, using an interface similar to the one shown. The programmer's goal is to use an approach where there is no hard coded text in the Blaise instrument.

At any point in the life cycle, the programmer can update types, fills, and fields for the whole instrument.

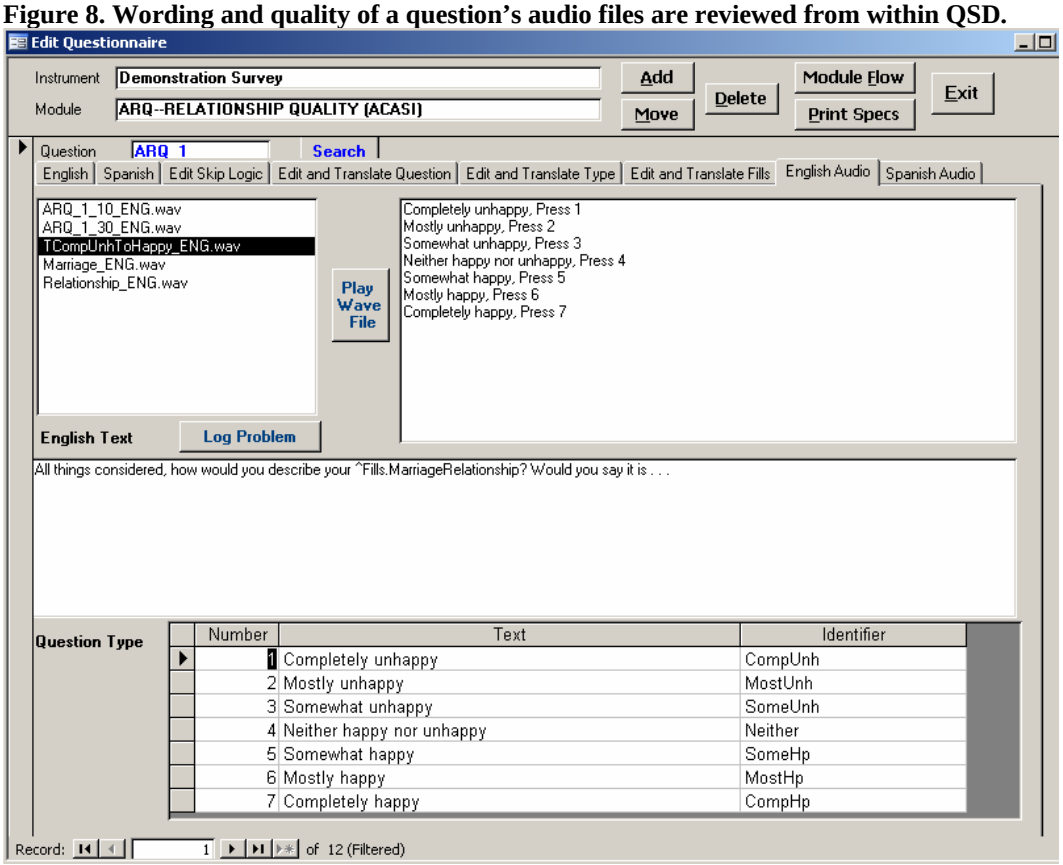
5. QSD and ACASI

For studies that require use of audio files, QSD helps to produce scripts for recording. Each question that is to be recorded is marked in QSD. Marked questions cause additional interface elements to be visible on the "Edit Questionnaire" form, and an entry for an additional language to be added to the Blaise code.

5.1 Testing Audio Files in QSD

Due to fills and types with response options, there are almost always several audio files for each question. These need to be tested carefully to provide the respondent a smooth listening experience and to ensure audio coverage of fills under various scenarios. It is time consuming to do such testing solely through interactive use of the Blaise instrument. Testers can use QSD to check that all necessary files are recorded for the question, that the pronunciation is good, and there is smooth transition from one file to the next.

Figure 8. Wording and quality of a question's audio files are reviewed from within QSD.



Testers log any problems with audio files into QSD. Programmers then update the scripts for re-recording.

5.2 Implementation in the Blaise Instrument

Some studies require that the respondent have a choice to listen to the questions only without seeing the question text on the screen. We satisfy this requirement by adding a so-called "Private" language to the Blaise instrument for each natural language used. If

the instrument has two languages, two “Private” languages are added. To maintain the multiple languages, we use an auxiliary field LNG.

Figure 9. In this example of QSD-generated code, the value of an auxiliary field W_AC_2 holds information about audio files associated with the question AC_2.

```
{***********************************************}
{**********Rules**FOR**ACASI**********************}
{***********************************************}
        W_AC_1 :=
            ' SOUND('+ WaveDir + '\AC_1_10_' + LNG + '.wav)'
        W_AC_2 :=
            ' SOUND('+ WaveDir + '\AC_2_10_' + LNG + '.wav)'
           + ' SOUND('+ WaveDir + '\TMonth_' + LNG + '.wav)'
```

Languages are defined for an entire Blaise instrument, so we cannot direct some modules to use a “Private” language and others to not use it. Thus, for questions that lack sound files, text for the “Private” language is generated from QSD with the same text as the corresponding natural language. This way the correct text is always presented on the screen. For questions where only audio should be played, the text is replaced by the special character "·". Blaise will recognize it as space to be displayed on the screen.

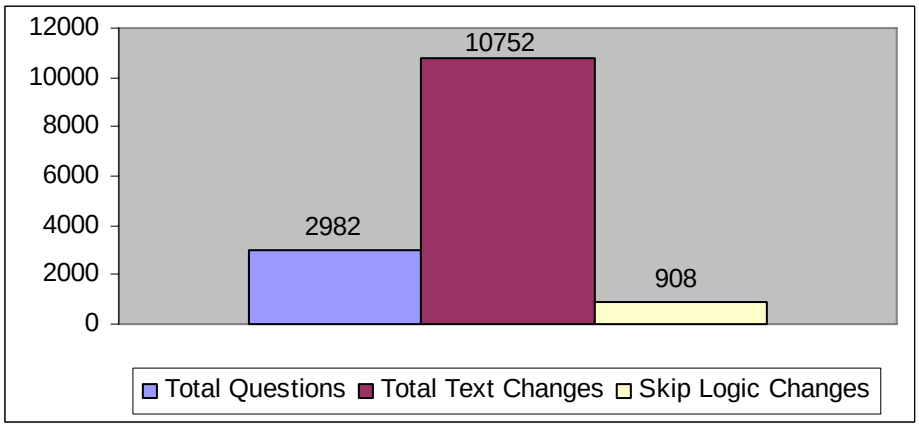
Figure 10. Audio-only questions generate a specially formatted field definition.

```
        AC_2
        ENG
        "In what month were you born?"
        SPN
        "¿En qué mes nació usted?"
        MML "^W_AC_2"
        PRV "···"
        PSP "···"
        /"Month of birth (practice)"
        : TMonth ,noDK, noRF
```

6. QSD at RTI International

In using QSD on a number of studies over the last two years, we at RTI International have found that it reduces questionnaire development time. The number of questions maintained in QSD has exceeded 9000 among 26 instruments across several studies. As Figure 11 shows, most changes are to text, and QSD gives us the flexibility to accommodate a high volume of these changes in a short period. On one recent study, 442 changes – mostly to wording – were logged in a single day late in the development cycle. A new version was ready for testing that evening.

Figure 11. QSD log data from four recent bilingual studies conducted by RTI International illustrate that great opportunities for efficiency lie in accelerating text changes.



We continue to add new features to QSD and improve the existing ones to satisfy requests from users. Future plans include a web interface and the option to use SQL Server for the back end.

7. Conclusion

QSD is a central place to enter and refine questionnaires, from organizing sections to adding and changing questions and logic, to keying translations, to associating audio files for ACASI. It has built-in support for problem tracking, change tracking, and version control.

Advantages of QSD include:

- Specs-Instrument synchronization;
- Programmer focus on logic rather than question wording;
- Addition of a second language without programmer intervention;
- Extensive support for ACASI.

QSD helps speed up development and simplify maintenance of questionnaires. It promotes the creation of error-free Blaise instruments.

Since internally QSD stores all questionnaire elements in a relational database, alternate code generation modules could be developed and plugged in so that the same definition could be used to create a questionnaire in a language other than Blaise, or to take advantage of new features in future versions of Blaise.

8. Acknowledgements

The authors would like to thank members of the Research Computing Division and Survey Research Division at RTI International for their use of QSD, as well as for their invaluable comments and suggestions related to its further improvement.