

Preface

This document contains the papers presented at the 17th International Blaise Users Conference held at the New Babylon Meeting Center in The Hague, the Netherlands, from October 4 - 6, 2016. The conference included three days of technical papers and presentations on the use of Blaise and related topics.

The Conference Program was organized and planned by the Program Committee, chaired by Hilde Degerdal, Statistics Norway. Members of the committee include:

Hilde Degerdal (Statistics Norway)

Mark M Pierzchala (MMP Survey Services, LLC, USA)

Rob Wallace (US Census Bureau, USA)

Gina Cheung (University of Michigan/SRC, USA)

Éric Joyal (Statistics Canada)

Mike Hart (Office for National Statistics, UK)

Leif Bochis Madsen (Statistics Denmark)

Jacqueline Hunt (CSO, Ireland)

Roger Linssen (Statistics Netherland)

Jane Shepherd (Westat, USA)

Table of Contents

Conference Program

Papers

Plenary Session 2

Blaise 5: Is Worth the Wait

Mark Pierzchala, USA 1

Presentation Session 1 - Modern Data Collection with Blaise 5

Experiences with Colectica

Felix Coleman – CSO, Ireland 14

Blaise 5 at Statistics Norway

Trond Båshus – Statistics Norway 19

Converting a CATI Instrument from Blaise 4 to Blaise 5 for Pilot Testing

Emily Caron & Vito Wagner – National Agricultural Statistics Service, USA & Kathleen O'Reagan – Westat, USA 27

Presentation Session 2 - Blaise 4 and Blaise 5

Chitwan Valley Family Study Household Registry Paper to Computer Exercise

Karl Dinkelmann & Stephanie Chardoul – University of Michigan/SRC, USA & Bishnu Adhikari – Institute for Social and Environmental Research – Nepal 37

Blaise Instrument Extensions with Alien Routers and Manipula

Lilia Filippenko, Chris Carson, Orin Day & Mai Nguyen – RTI International, USA 51

Statistical Surveys in Households and by Individuals in Slovakia – the Use of Blaise 4 / Blaise 5

Ľudmila Hadová & Danica Tureková – Statistical Office of the Slovak Republic 61

Dep Unlimited: Solving Complex Runtime Problems with Manipula

Rick Dulaney & Peter Stegehuis – Westat, USA 78

Presentation Session 3 - Tablets and Devices

Blaise on Touch-Screen Tablets

Matthieu Olivier & Alexandre Chevallier – Centre de Données Socio Politiques (CDSP), France 90

From CPI to CPIX <i>Gerrit de Bolster – Statistics Netherlands</i>	96
Converting a Blaise 4.8 Survey Instrument for Tablet Use <i>Michelle Keniry, Caroline Donegan & Conor MacDomhnaill – CSO, Ireland</i>	106
Capturing Signatures Electronically from a Blaise Instrument <i>Todd Flannery & Ed Dolbow – Westat, USA</i>	120
A Web Compatible Dep <i>Maurice Martens – CentERdata, The Netherlands</i>	128
Converting a Paper Collection Exercise to Collecting Data on a Tablet <i>Mike Hart – Office for National Statistics, UK</i>	136
<hr/> <i>Presentation Session 4 – Why Paradata is essential</i> <hr/>	
Using Survey Paradata <i>Gina Cheung, Andrew Piskowski, Lisa Wood & Hueichun Peng – University of Michigan/SRC, USA</i>	147
The Uses of Blaise Audit Trail Files at Statistics Canada <i>Éric Joyal – Statistics Canada</i>	156
Capturing Survey Client-side Paradata <i>Hueichun Peng & Jason Ostergren – University of Michigan/SRC, USA</i>	170
Paradata Transform and Report <i>Lisa Wood, Andrew Piskowski & Jennie Williams – University of Michigan/SRC, USA</i>	179
Using Audit Trail Data to move from a Black Box to a Transparent Data Collection Process <i>Jacqueline Hunt – CSO, Ireland</i>	194
<hr/> <i>Presentation Session 5 – Blaise 5</i> <hr/>	
Blaise 5 Development at University of Michigan <i>Youhong Liu & Max Malhotra – University of Michigan/SRC, USA</i>	203
Running Blaise 5 Instrument as a "Stand-alone" Environment <i>Roberto Picha & Mecene Desormice – US Census Bureau, USA</i>	218
Experiences with Blaise 5 Layouts and Templates <i>Sharon Johnson, Richard Squires & Michael Johnson – US Census Bureau, USA</i>	231
From Blaise 4 to 5: Systems in Transition – <i>Leif Bochis Madsen – Statistics Denmark</i>	246

Presentation Session 6 – Blaise 5 Data Handling

Blaise 5 Data In/Data Out

Andrew Piskoworski – University of Michigan/SRC, USA 253

Data-Out Experience/Challenges in Blaise 5

Mohammad Mushtaq & April Beale – University of Michigan/SRC, USA 265

Census Setup with Blaise 5 Manipula

*Michael K Mangiapane, Roberto Picha & Mecene Desormice –
US Census Bureau, USA* 276

Adventures of a Blaise 5 API Jockey

Rod Furey – Statistics Netherlands 285

Poster Session

The Blaise 5 Champion Instrument Suite

Mark M Pierzchala, MMP Survey Services, LLC, United States 290

Tuesday, 4 October: IBUC – Day 1	
Time	Activity
8:15 – 9:00	Registration and Coffee
9:00 – 10:00	<p>Plenary Session 1</p> <ul style="list-style-type: none"> • Opening remarks by IBUC Chair, Hilde Degerdal – Statistics Norway • Welcome by Bert Kroese – Deputy Director General of Statistics Netherlands • Keynote address by Jim Smith – CEO of Westat, USA
10:00 – 10:30	Break
10:30 – 11:30	<p>Plenary Session 2</p> <ul style="list-style-type: none"> • Blaise 5: Is Worth the Wait by Mark Pierzchala, USA • Colectica and Blaise by Dan Smith & Jeremy Iverson – Colectica, USA • Blaise Celebrates its 30 Year Anniversary!
11:30 – 13:00	<p>Lunch</p> <p>Starting at 12:00 there will be a Poster & Demonstration session</p>
13:00 – 14:30	<p>Presentation Session 1 - Modern Data Collection with Blaise 5</p> <p>Chair: Mike Hart – Office for National Statistics, UK</p> <ul style="list-style-type: none"> • Phoenix – Towards Modern Data Collection Systems and Processes by Joost Huurman – Statistics Netherlands • Experiences with Colectica by Felix Coleman – CSO, Ireland • Blaise 5 at Statistics Norway by Trond Båshus – Statistics Norway • Converting a CATI Instrument from Blaise 4 to Blaise 5 for Pilot Testing by Emily Caron & Vito Wagner – National Agricultural Statistics Service, USA & Kathleen O'Reagan – Westat, USA
14:30 – 15:00	Break

15:00 – 16:45	<p>Presentation Session 2 - Blaise 4 and Blaise 5</p> <p>Chair: Leif Bochis Madsen – Statistics Denmark</p> <ul style="list-style-type: none"> • Chitwan Valley Family Study Household Registry Paper to Computer Exercise by Karl Dinkelmann & Stephanie Chardoul – University of Michigan/SRC, USA & Bishnu Adhikari – Institute for Social and Environmental Research – Nepal • Blaise Instrument Extensions with Alien Routers and Manipula by Lilia Filippenko, Chris Carson, Orin Day & Mai Nguyen – RTI International, USA • Statistical Surveys in Households and by Individuals in Slovakia – the Use of Blaise 4 / Blaise 5 by Ľudmila Hadová & Danica Tureková – Statistical Office of the Slovak Republic • Dep Unlimited: Solving Complex Runtime Problems with Manipula by Rick Dulaney & Peter Stegehuis – Westat, USA • Transition of a Blaise 4 Web Survey to Blaise 5 by Bryan Bungardt – Statistics Netherlands
17:00 – 18:30	Welcome Reception
Evening	Informal groups for dinner

Wednesday, 5 October: IBUC – Day 2	
Time	Activity
8:15 – 9:00	Registration and Coffee
9:00 – 10:00	<p>Plenary Session 3</p> <ul style="list-style-type: none"> • Keynote address 'The Ever Changing Landscape of Survey Data Collection' by Jelke Bethlehem, The Netherlands. Jelke is the former head of Blaise, currently Professor in Survey Methodology at the Faculty of Social and Behavioural Sciences of the Leiden University (with a focus on online data collection) • Methodology: The ultimate design for Collection on mobile Devices by Jeldrik Bakker –Statistics Netherlands and Utrecht University
10:00 – 10:30	Break
10:30 – 12:15	<p>Presentation Session 3 - Tablets and Devices</p> <p>Chair: Gina Cheung – University of Michigan/SRC, USA</p> <ul style="list-style-type: none"> • Blaise on Touch-Screen Tablets by Matthieu Olivier & Alexandre Chevallier – Centre de Données Socio Politiques (CDSP), France • From CPI to CPIX by Gerrit de Bolster – Statistics Netherlands • Converting a Blaise 4.8 Survey Instrument for Tablet Use by Michelle Keniry, Caroline Donegan & Conor MacDomhnaill – CSO, Ireland • Capturing Signatures Electronically from a Blaise Instrument by Todd Flannery & Ed Dolbow – Westat, USA • A Web Compatible Dep by Maurice Martens – CentERdata, The Netherlands • Converting a Paper Collection Exercise to Collecting Data on a Tablet by Mike Hart – Office for National Statistics, UK
12:15 – 13:15	Lunch
13:15 – ...	Conference Excursion and Conference Dinner

Thursday, 6 October: IBUC – Day 3

Time	Activity
8:15 – 9:00	Registration and Coffee
9:00 – 10:30	Presentation Session 4 – Why Paradata is essential Chair: Jane Shepherd – Westat, USA <ul style="list-style-type: none">• Using Survey Paradata by Gina Cheung, Andrew Piskoworski, Lisa Wood & Hueichun Peng – University of Michigan/SRC, USA• The Uses of Blaise Audit Trail Files at Statistics Canada by Éric Joyal – Statistics Canada• Capturing Survey Client-side Paradata by Hueichun Peng & Jason Ostergren – University of Michigan/SRC, USA• Paradata Transform and Report by Lisa Wood, Andrew Piskoworski & Jennie Williams – University of Michigan/SRC, USA• Using Audit Trail Data to move from a Black Box to a Transparent Data Collection Process by Jacqueline Hunt – CSO, Ireland
10:30 – 11:00	Break
11:00 – 12:30	Presentation Session 5 – Blaise 5 Chair: Éric Joyal – Statistics Canada <ul style="list-style-type: none">• Introducing Device Sensor Data in Surveys by Ole Mussmann, Jeldrik Bakker, Jacqueline van Beuningen, Barry Schouten & Rob Warmerdam – Statistics Netherlands• Blaise 5 Development at University of Michigan by Youhong Liu & Max Malhotra – University of Michigan/SRC, USA• Running Blaise 5 Instrument as a "Stand-alone" Environment by Roberto Picha & Mecene Desormice – US Census Bureau, USA• Experiences with Blaise 5 Layouts and Templates by Sharon Johnson, Richard Squires & Michael Johnson – US Census Bureau, USA• From Blaise 4 to 5: Systems in Transition – by Leif Bochis Madsen – Statistics Denmark

12:30 – 13:30	Lunch
13:30 – 15:00	<p>Presentation Session 6 – Blaise 5 Data Handling</p> <p>Chair : Rob Wallace – US Census Bureau, USA</p> <ul style="list-style-type: none"> • Blaise 5 Data In/Data Out by Andrew Piskoworski – University of Michigan/SRC, USA • Data-Out Experience/Challenges in Blaise 5 by Mohammad Mushtaq & April Beale – University of Michigan/SRC, USA • Census Setup with Blaise 5 Manipula by Michael K Mangiapane, Roberto Picha & Mecene Desormice – US Census Bureau, USA • Adventures of a Blaise 5 API Jockey by Rod Furey – Statistics Netherlands
15:00 – 15:30	Break
15:30 – 17:00	International Blaise Users Group Meeting

Blaise 5 – Is Worth the Wait

Mark M Pierzchala, MMP Survey Services, LLC, United States

1. Abstract

There are 2 possible ways to understand the title of this paper.

- Blaise 5 – Is Worth the Wait?
- Blaise 5 – Is Worth the Wait!

In other words, is the title a question or is it a positive statement of fact? The reason to pose this question is that Blaise 5 has taken a long time to come into production. Additionally, many of the institutes that have tried Blaise 5 so far have struggled with its concepts. This is especially true for layout and devising templates that work across devices and screen sizes.

On the other hand, there are fundamentally important technical and questionnaire design issues with fielding governmental and scientific (*GovSci*) survey instruments in today's environment. The next section describes many of these challenges. Additionally, Blaise surveys are often lengthy and complex, and must be implemented in a methodologically sound manner. After a description of the challenges, the paper demonstrates how Blaise 5 addresses these challenges and how users can use the system optimally.

2. Fundamental Methodological and Technical Issues

MMP Survey Services, LLC (MMPSS) has worked with the Statistics Netherlands Blaise Team for several years to research, document, and validate Blaise 5 concepts. Investigations were also conducted into the types of questions and question structures that are found in *GovSci* surveys. Over the years, the research revealed how nine aspects of the survey environment impact electronic instrument design, especially screen design impact on question formulation. These are referred to as the 'nine multiples'.

The point of this section is to illustrate how modern-day challenges are so complex that (1) new instrument design and production methods are required, (2) no person has the knowledge to deal with all these aspects, and (3) an institute should plan ahead in order to know how to handle its survey world. The last point is explained further below. Some powerful solutions and approaches are also given.

2.1 The Nine Multiple Aspects of Computer-Assisted Instrument (CAI) Design

The list below enumerates the nine technological aspects that impact instrument design. The term *Multiple* derives from the prefix *Multi* as used in the following terms (in alphabetical order).

- **Multi-cultural:** surveys across human cultures
 - There may be different representations of the same underlying survey concepts between cultures. This may result in differently worded questions or response options, or even alternative structures for the same concept. These go beyond linguistic challenges.
- **Multi-device:** surveys on devices ranging from smart phones to tablets to computers
 - Differences in screen size, pixel density, whether there is a physical or virtual (pop-up) keyboard, and other differences between devices, can alter question display.

- **Multilingual:** surveys with two or more languages.
 - Translations must be timely and accurate and decisions have to be made about whether a user can change languages on the fly and how. Alphabetic, Asian, right-to-left, and left-to-right languages all must display properly.
- **Multimode:** surveys that are both interviewer- and self-administered (and variants)
 - Blaise 5 considers *Interviewing* and *Self-Administration* as the major mode division, but also formally accounts for variants such as CATI and CAPI (as interview modes) and web and paper (as self-administered modes). Question statement, instruction, and even question structure can differ between modes. One of the toughest issues is how to handle DK and RF options between modes.
- **Multi-national:** surveys in two or more countries
 - There can be country-specific attributes and references that have to be handled. These include currency, phone number formats, names of institutions and programs, languages to allow in each country, web links, terminology, and other items. Even same-language countries have to account for wording changes. For example, *surname* in the UK and *last name* in the United States refer to the name that comes after the first name. There is a Blaise 5 sample that includes three versions of the North American Industry Classification System (NAICS) as they are used in Canada, the United States, and Mexico.
- **Multi-operable:** surveys having to deal with different operational characteristics such as touch- versus mouse/pointer- devices; virtual versus physical keyboards
 - The more complex the question or question structure, the more design is impacted by how the device is operated. For example, a battery of scale questions will be operated differently between a computer browser screen, a smart phone, and a Windows computer.
- **Multi-platform:** refers to surveys running as native applications on a device or running across the internet in a connected mode. For a browser, it can also refer to whether JavaScript or pop-ups are allowed.
 - The performance of the instrument, and when the rules are executed, may impact how many questions are placed on a screen, as an example.
- **Multi-structural:** surveys with questions that are structured differently for some manifestations.
 - This may refer to the practice where survey sponsors insist on different question structures or wording between modes. The researcher does this in order to achieve *cognitive equivalence* between modes and/or devices, even in situations where the structure can be rendered identically between modes. But there are also cases where a question structure in one mode cannot be achieved in another (see section 4.2 below).
- **Multi-version:** refers to surveys with many questionnaires that are similar, but have differences.
 - For example, kinds of industries or agricultural regions each require custom questionnaires. The questionnaire versions, and there can be hundreds, are all variations on a theme.

Many of the above design aspects are related, but each can impact design by itself.

2.2 Other Important Design Considerations

While the nine multiple aspects of CAI design mentioned above are tough enough to deal with, they do not account for all design challenges. In addition to the *nine multiples* above, other design considerations include:

- Paper questionnaires that never went away. These are still heavily used for some kinds of surveys, and have their own design considerations (mostly limitations). These can result in differences in question wording and construction between paper and electronic modes.

- Making electronic instruments usable to blind, visually impaired, and moter impaired users presents additional design challenges. This challenge of dealing with this issue can be under estimated. One reason is that assistive technology is used in conjunction with the Blaise instrument. Another reason is to truly accomodate the needs of the disabled rather than merely complying with laws such as check-listing against Section 508.
- Handling the following edit hierarchy can present additional design issues:
 - Univariate range edits (user types a number that is not within the field definition)
 - Univariate soft edit (a suspicious, but possibly valid value is entered)
 - Soft or hard edits between two or more fields on the same page.
 - Soft or hard edits between fields on different pages, including situations where the fields are physically located many pages apart.
- Legacy issues arise in surveys that have been fielded for many years. Survey sponsors may be reluctant to change how questions are asked because they do not want to disrupt data series.
- Mobile devices give new opportunities for data collection. By definition, some collections with smart phones and tablets are not even possible for other platforms.

2.3 The First Overall Goal

The first overall goal of Blaise 5 development is to be able to handle all of these aspects with one Blaise instrument, one database, and one well-conceived specification.

3. Blaise 5 Features give New Opportunities

This section gives only a hint of Blaise 5 features that are designed to handle the above challenges. Enough description is given here in order to describe below how to bring all these features together.

3.1 Layout Features and Modules

There are two design tools and three language features worth mentioning with respect to achieving layout. When used properly, and with adequate pre-planning, one reward is achieving instrument layouts that work across modes, devices, and screen sizes. A second reward is the large savings of time and effort in implementing desired layout.

3.1.1 Two Layout Design Tools

The **Resource Database** allows the institute to define fonts and styles (and many other aspects of layout) and to design display templates that are suited for each mode, device, and screen size. The elements of the Resource Database are available as design options in the Layout Designer. The Resource Database is a stand-alone module separate from the Control Centre.

The **Layout Designer** is found in the Control Centre. Here, design elements from the Resource Database are applied to the questionnaire. Screens for all modes and devices can be seen in the Layout Designer.

3.1.2 Three Language Layout Features

A display keyword called **GROUP** indicates a multi-field display. A group can be handled similarly or differently between modes, devices, and screen sizes.

The keyword **MODES** can be used in the source code itself and can (should be used to) map to Layout Set Groups in the Layout Designer. This makes it possible to marry source code to layout.

Naming conventions for types, groups, and blocks can be defined in the Resource Database templates and used in the source code. The result is the automatic application of layout elements; a possibility that is especially useful, and even necessary, for large instruments.

3.2 Programming Language Features

New keywords in Blaise 5 address several of the nine multiple aspects listed above. These include:

- **MODES:** In addition to mapping to Layout Set Groups, this keyword can be used in the Rules.
- **LANGUAGES:** This keyword is also present in Blaise 4, but in Blaise 5 it is used for natural spoken languages. Thus it is liberated from tasks that are now assigned to ROLES.
- **ROLES** can be used for tooltip display, screen reader text, variable names for downstream statistical packages and so forth. ROLES text can be defined in all the LANGUAGES.
- **SPECIALANSWERS:** This keyword allows users to assign additional non-value statuses to fields. These are in addition to the still-standard DK and RF. For example, it is now possible to define NA for 'Not Applicable'
- **SPECIALANSWERSETS:** This keyword allows users to allow different special answers for each mode. This formalizes a practice often seen in multimode surveys, especially the differential handling of DK and RF and EMPTY between interviewing and self-administration.
- **ATTRIBUTES:** Assigns the desired SPECIALANSWERSET for the instrument.

Listing 1. Model Source Code showing some New Language Features

```
DATAMODEL B5Complex "Blaise 5 Complex"

LANGUAGES =
  ENG "English",
  NED "Nederlands"

ROLES =
  Help "Help",
  ToolTip "Tool tip hint",
  ScReader "Screen reader text",
  StatVar "Stat variable name"

MODES =
  Self "Self",
  Interview "Interview",
  Data "Data"

SPECIALANSWERS =
  NA "Not applicable",
  OoR "Out of Range"

SPECIALANSWERSETS
  EmptyAllowed = Self : EMPTY
                  Interview: DK, RF
                  Data : DK, RF, NA, OoR

ATTRIBUTES =
  EmptyAllowed
```

In Listing 1, three MODES are recognized. These are *Self*, *Interview*, and *Data*. These are used respectively for self-administered mode, interviewing mode, and post-collection processing. The SPECIALANSWERS include *NA*, and *OoR*. These are in addition to the standard DK and RF.

The keyword SPECIALANSWERSETS allows differential handling of special values by mode. For example, for the *Self* mode, EMPTY is allowed for all fields by default. For *Interview*, NOEMPTY is the default value, but DK and RF are allowed for all fields by default. The *Data* mode allows by

default *NA* and *OoR* in additions to *DK* and *RF*. All these defaults can be overridden on the field level if necessary.

MODES, *SPECIALANSWERS*, and *SPECIALANSWERSETS* are all defined independently of device, screen size, and settings. They allow different handling of special answers between modes. Thus *Blaise 5* formalizes survey taking as it has been practiced in governmental and scientific surveys.

3.3 Settings

Settings determine when rules are invoked, when special answers are displayed, how to enforce routing, and how to handle edits. That is, settings further adapt the operation of an instrument to different modes. Table 1 shows four model Settings, *Interview*, *Self*, *App*, and *Data*.

Table 1. Selected Settings for Setting Names Interview, Self, App, and Data

Setting Name	Letter	DK/RF & Special Answers Display	When Rules	Dynamic / Static Execution	Run Mode
Interview	A	Yes	Datum change	Dynamic	Thick client
Self	B	No	Page change	Dynamic	Client-server
App	C	No	Datum change	Dynamic	Thick client
Data	D	Yes	On demand ^{1/}	Static	Thick client

The settings shown above reflect typical uses. For example, *Interview* is intended for thick client instruments where rules execution is instantaneous. The rules are executed every time a data item is changed. The *DK* and *RF* are displayed for the interviewer.

On the other hand, *Self* is intended for a browser where the rules are executed remotely from the user. Here, it is appropriate to execute the rules on a request for new page. This often leads to fewer items per screen and it can slow down the overall instrument execution.

3.4 Other Configurable Aspects of Blaise 5

There are other aspects of modern survey taking that are accounted for by *Blaise 5*. Here is a short list.

- **Screen size:** Resource Sets in the Resource Database allow users to design templates for each screen size. These are applied in the Layout Designer.
- **Runtime parameters:** Allow users to match mode to layout set to settings to screen size.
- **Styles:** These allow users to easily change the look and feel of your instrument.
- **Screen layout:** Resource Sets in the Resource Database also allow users to decide between full-, split-, and entry- screen designs as shown in Figures 1 through 3.

Figure 1. Full-Screen Display for Self-Administration on a Web Browser on a Computer

Modestly Complex Questions
Language
English ▾

The challenge here is the high-level classification of choices. This structure comes directly from the paper questionnaire.

Please classify your principal employer.

SELF EMPLOYED or a BUSINESS OWNER

In a non-incorporated business, professional firm, or farm

In an incorporated business, professional practice, or farm

PRIVATE SECTOR employee

In a for-profit company or organization

In a non-profit organization (including tax-exempt and charitable organizations)

GOVERNMENT employee

In a local government (e.g., city, county, school district)

In a state government (including state colleges/universities)

In the U.S. military service, active duty or Commissioned Corps (e.g., USPHS, NOAA)

In the U.S. government (e.g., civilian employee)

OTHER

Other type of employee, specify

Back
Next

RDB Template = Default

3/18 PrincipalEmployer | Self-Large | Question | B5Modest

Figure 2. Split-Screen Display for CATI in Windows for a Computer Desktop or Laptop

Modestly Complex Questions
Language
English ▾

Next Mode DK RF Code

The challenge here is the high-level classification of choices. This structure comes directly from the paper questionnaire.

Please classify your principal employer.

<p><input type="radio"/> 1 In a <u>non-incorporated</u> business, professional firm, or farm</p> <p><input type="radio"/> 2 In an <u>incorporated</u> business, professional practice, or farm</p> <p><input type="radio"/> 3 In a <u>for-profit</u> company or organization</p>	<p><input type="radio"/> 4 In a <u>non-profit</u> organization (including tax-exempt and charitable organizations)</p> <p><input type="radio"/> 5 In a <u>local</u> government (e.g., city, county, school district)</p> <p><input type="radio"/> 6 In a <u>state</u> government (including state colleges/universities)</p>	<p><input type="radio"/> 7 In the <u>U.S. military</u> service, active duty or Commissioned Corps (e.g., USPHS, NOAA)</p> <p><input type="radio"/> 8 In the <u>U.S. government</u> (e.g., civilian employee)</p> <p><input type="radio"/> 9 Other type of employee, specify</p>
--	--	---

Intro <input type="checkbox"/>	Other principal employer <input type="text"/>	
Education code <input type="checkbox"/>	Kinds of employment <input type="text"/>	
Principal employer <input type="checkbox"/>	Other kind of employment <input type="text"/>	
Principal employer high <input type="checkbox"/>	Number <input type="text"/>	
Self employment <input type="checkbox"/>	Unit <input type="text"/>	
Private employer <input type="checkbox"/>	Number <input type="text"/>	
Government employer <input type="checkbox"/>	Unit <input type="text"/>	

RDB Template = Split

1/6 | Interview-Split | Question | B5Modest

Figure 3. Form-Pane Display for Post-Collection Processing

The screenshot shows a data entry interface with a header bar containing 'Modestly Complex Questions' and 'Language English'. Below the header are 'Next' and 'Mode' buttons. The main area contains two columns of fields. The left column includes fields like 'Intro', 'Education code', 'Principal employer', 'Principal employer high', 'Self employment', 'Private employer', 'Government employer', 'Other principal employer', 'Kinds of employment', 'Other kind of employment', 'Number', 'Unit', 'Number', 'Unit', 'Title', and '*'. The right column includes 'MiddleName', '*', 'Suffix', '*', 'Apartment', '*', 'Zip5', 'Zip4', 'KindEmploy2', 'OtherEmploy2', 'YearHSDip', 'NotFinishHS', 'State_', and 'Prov'. At the bottom, there is a navigation bar with '1/2', 'Data-Entry', 'Question', and 'BSModest'.

RDB Template = Data

3.5 A Second Overall Goal

A second overall goal is to have the questionnaire designer focus on questions (Figure 4). This person should not worry about methodological and technical details; it is just too complicated and overwhelming. These details should already have been decided by the institute as described below.

Figure 4. One Specification handles Modes, Devices, and Screen Sizes

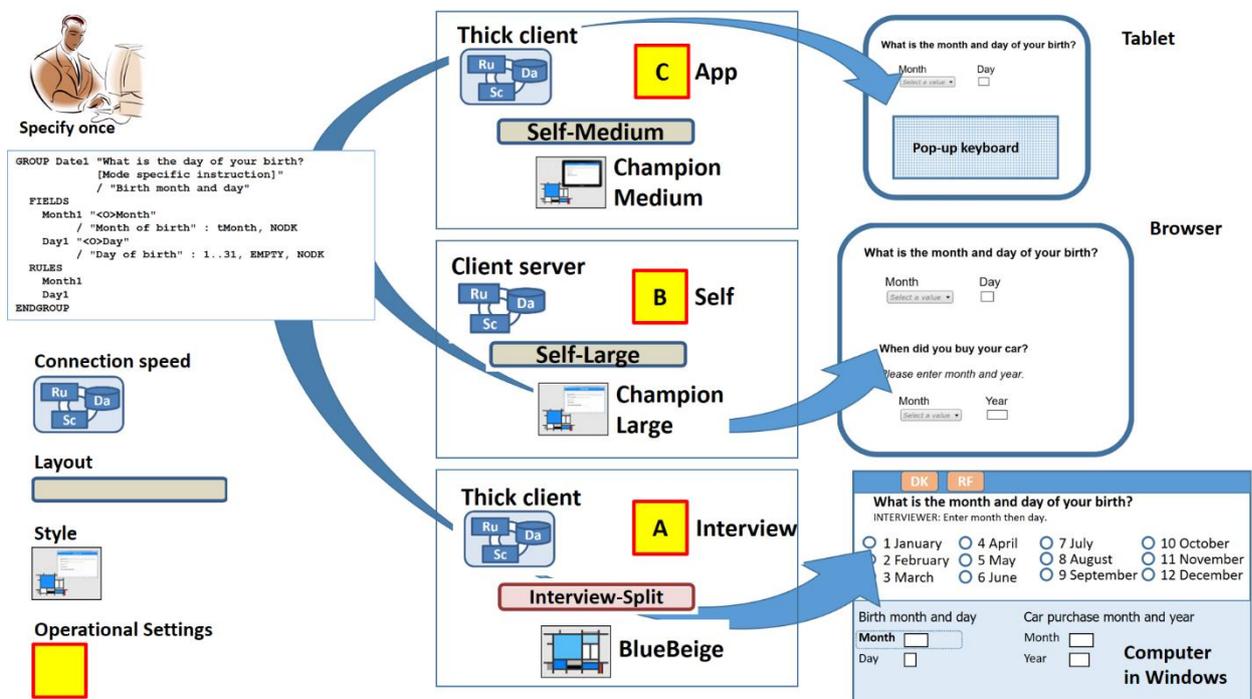
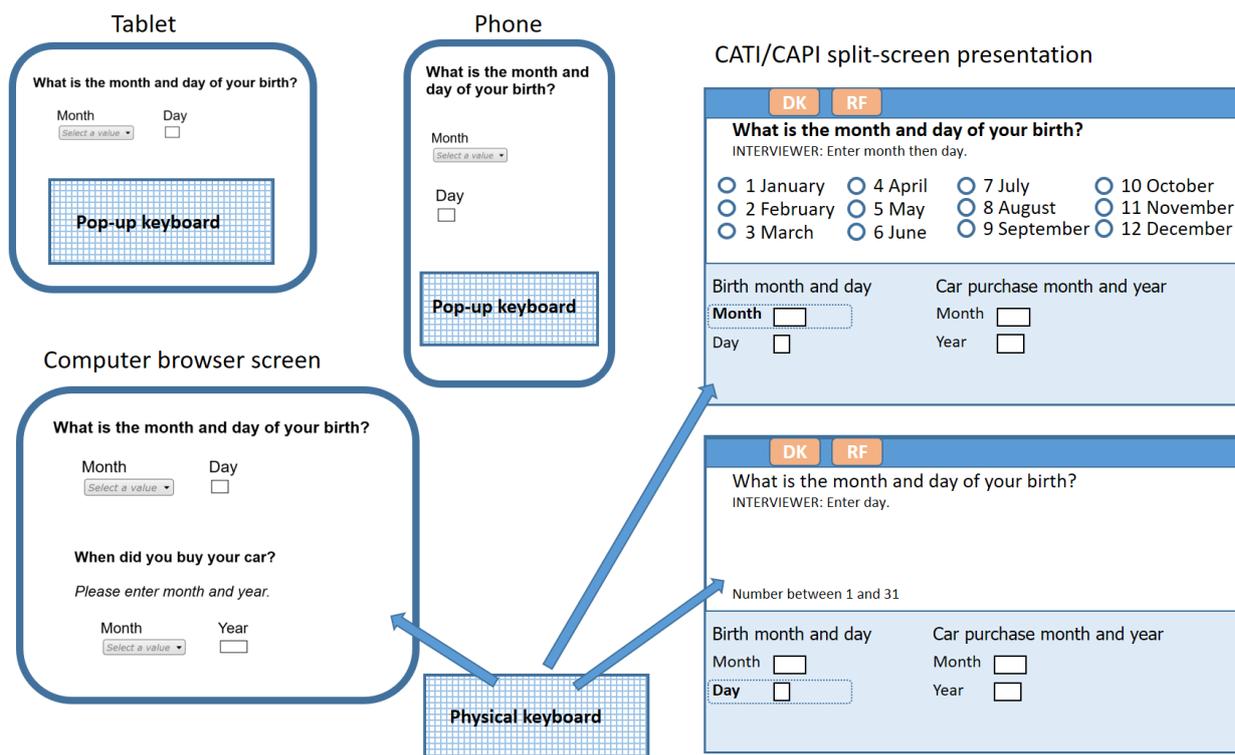


Figure 5 shows how a Quantity/Unit question (two fields) might appear on a tablet, a smart phone, a computer browser screen, and in the traditional Blaise 4 split-screen presentation. Note that the computer browser screen shows two Quantity/Unit pairs whereas the tablet of a similar screen size shows just one pair. This is because for the tablet, it might be necessary to reserve room for the pop-up keyboard.

In any case, the goal is to have all these displays (and more) appear, as if by magic, by one well-conceived specification, written by an individual who is concerned with just the questions.

Figure 5. A Quantity/Unit Pair displayed different Ways on different Devices



4. Institutes should Establish Layout and Operability Standards

This section shows how an institute can establish layout and operability standards for multi-device and multi-mode production of instruments. As explained above, there are so many details to think about that the institute should convene a standards-setting group instead of making up ad hoc survey standards.

By design, Blaise 5 has the capacity to pre-package these agreed institute standards. Layout standards are encoded in a standard Resource Database. Other standards are held as model source code, model settings, and model layout sets and groups. Additionally, the institute should establish type, block, and group naming conventions, that when used in the source code, automatically generate layouts.

4.1 Design Assertion

The methods described below rely on the following design assertion:

"The design of layout across modes and devices depends only on question format and structure."

This means that it is possible to conduct all necessary design and methodological research through investigations of question format and question structures. All resulting institute findings and standards are thus independent of subject matter, which is a strong asset.

4.2 Unimode versus Generalized Mode Design

In multimode surveys, there are two design traditions. The first is called *Unimode* design. This tradition holds that question statement and structure should be presented the same across all modes. This might mean that the question is presented less than optimally in one or more modes. The benefit is that the measurement of the concept is as same as possible across modes.

The second tradition is called *Generalized Mode* design. In this tradition, the aim is to achieve cognitive equivalence across modes. This often means that the question structure is presented differently for one or more modes in order to optimize the presentation for each mode.

4.2.1 Discussion of Design Traditions

A good description of *unimode* versus *generalized mode* design is given by de Leeuw (2004). Pierzchala et al. (2004) give an example of a survey that used generalized mode design for a multimode CATI, Web, and paper survey. For some questions there were three question structures across the three modes. Pierzchala (2006) concludes that as the number of modes goes up in a survey, the harder it is to maintain a unimode design. This paper defines the concept of *degrees of disparity* between modes on several design criteria. However, a recent volume by Dillman, et al. (2014), advocates for unimode design wherever possible, even in surveys mixing paper, CATI interviewing, and Web.

Most of the original discussion took place before smart phones and tablets emerged as popular survey devices. As Blaise 5 was developed, it became clearer that the nine multiple aspects of CAI Instrument Design (section 2.1 above) make it harder to consistently achieve unimode design. The strategies and techniques described (at a high level) in this paper allow the institute to methodically investigate these issues, and where generalized design is necessary, to achieve the necessary displays across devices.

4.2.2 Impact of Difficult Questions on Instrument Design

Blaise is used for the world's most difficult governmental and scientific surveys. These surveys often feature lengthy and complex questions. They also incorporate difficult question structures such as tables. Our research shows that lengthier and more complex questions also encourage generalized mode design.

In order to more fully understand the impact of question structure on instrument design, MMPSS put together a suite of instruments called the Blaise 5 Champion Instruments. These are described below and in Pierzchala (2016). These instruments hold many different question types and structures and are used to validate Blaise 5 layout. They are available for Blaise institutes to use in their own investigations.

4.3 Institute Display and Operability Standards

Blaise institutes often have display and operability standards for their survey instruments. As institutes adapt their questionnaires to the modern survey-taking world, it may be time to re-assess these standards. A few standards used in the production of the Blaise 5 Champion Instruments are given here.

4.3.1 Web Screen Display Standards

A few of the many web display standards are described below. These are meant for a browser on a computer. One aspect of a web-based instrument is that the rules are executed on a remote server and there can be a delay whenever rules are executed.

- Use a full-screen display (Figure 1 above).
- The base font is Arial.
- Question text is bold.
- Instruction text is unbolded italics.
- Choice text (for enumerated fields) is unbolded and not in italics.
- DK and RF are not displayed when the respondent first enters the screen.
- DF and RF are displayed if the respondent leaves a field empty and tries to go to the next screen.
- If one question is used as a condition to route a second question, the second question is placed on the next screen.
- When routing is not an issue, the use of multi-item displays is encouraged to speed up the survey.
- Horizontal scrolling is not allowed. Vertical scrolling is permitted but not liked.
- Edits involving 2 or more fields should be used only if all involved fields are on the same screen.

4.3.2 Interviewer Screen Display Standards

A few of the interviewer display standards are described below. These are meant for an interviewing instrument using MS Windows®. Here, the display and the rules reside on the same computer and rules execution is instantaneous.

- Use a split-screen display (Figure 2 above).
- The base font is Arial.
- Question text is in mixed case bold.
- Instruction text is mixed case, unbolded and is preceded with the text 'INTERVIEWER:'.
- Optional text is mixed case unbolded text.
- Choice text that is to be read is mixed case, unbolded.
- Choice text that is not to be read is upper case, unbolded.
- DK and RF choices are always available where allowed. The interviewer does not offer them, but can use them if necessary.
- Question routing has no impact on page breaks.
- High data density is encouraged in order to facilitate navigation.
- All edits are available to be used regardless of the location of the involved fields.

4.3.3 Discussion of Web versus Interviewing Display Standards

The standards above are illustrative of those used by many Blaise institutes, even if the details differ. On one hand, an institute should try to align display standards across modes. On the other hand, experience teaches that this is not always possible, or desirable. There really is a difference between aural versus visual presentation of questions. There is also a profound difference between the untrained one-time self-respondent and a trained interviewer who may conduct hundreds of interviews on the same survey.

Once the institute has re-assessed its presentation standards, its specification protocols and tools should easily be able to handle these differences.

4.4 Three Layout Design Strategies

One of the goals of Blaise 5 has been to establish model design standards. These are held in the default Resource Database and as model source code, settings, and layout sets and groups. These resources will also be used in the Blaise 5 Champion Instruments. Given these resources, there are three layout design strategies for a Blaise 5 Institute.

- Use Blaise 5 out-of-the-box defaults. This is easy.
- Modify Blaise 5 defaults to create your own standards. This is moderately easy.
- Start from scratch. This is difficult and time consuming.

Deciding on a strategy to use depends on many factors. A good first step is to assess how these standards appear when used in the Blaise 5 Champion Instruments.

4.5 Blaise 5 Champion Instruments

The next step is to establish a test bed of different question formats and question structures. To this end, ten so-called Blaise 5 Champion instruments have been established and will be part of the Blaise 5 distribution. These are described below.

The ten *Champion* instruments demonstrate a range of display capabilities. Six instruments demonstrate specific layout capabilities and four are survey instruments.

4.5.1 Champion Instruments that Demonstrate Layout Capabilities

These six instruments are called B5 instruments.

- **B5Basic**: basic question layouts
- **B5Modest**: modestly complex questions and structures
- **B5Complex**: group questions such as tables
- **B5Scales**: a variety of scale types and display possibilities
- **B5PanEuropean**: multicultural aspects of surveys with over twenty languages
- **B5CodeFrame**: Coding kinds of questions (Windows for interviewing only)

These B5 instruments demonstrate about 100 kinds of items and data-collection structures. These questions, screens, and structures come primarily from government and scientific surveys.

4.5.2 Champion Survey Instruments

Two of the four instruments are based on real surveys (Census and ASI). The other two are made-up but realistic surveys. Two of the survey instruments were programmed from scratch in Blaise 5 (Retail Trade and Census). The other two (NCSPerson and ASI) were Blaise 4 surveys converted to Blaise 5.

These two instruments were initially programmed in Blaise 5 using the latest multimode features.

- **Retail Trade Survey**: collects financial data from firms and features non-linear page navigation. This instrument is meant for PC browsers or tablets.
- **Census-type Listing Instrument**: This instrument is meant for self-completion on device or browser and for interviewing on a device such as a smart phone or tablet.

These two were converted from Blaise 4 which necessitated some code restructuring and broadening of perspective in order to operate on new platforms.

- **NCSPerson:** a self-completion instrument meant to be used on PC Browser and devices.
- **Annual Survey of Industry (ASI):** a complex CATI and PC Browser instrument that collects economic data from firms. There are differences between web and CATI display, edit handling, and wording. It features the use of metadata statements in external files in order to drive the instrument differently for each industry.

4.5.3 Use of the Champion Instruments by your Institute

Institutes should inspect the Champion Instruments to see if they represent the entire range of question types and structures used in its survey program. If there are additional question structures that are used, these can be added to the Champion Instruments. The goal is to establish a ready-to-used test suite of instruments to assess display and operability.

5. Improved Specification and Instrument Generation

In this conference there is a presentation by Colectica (2016), a company that is producing a Blaise generator. This system has a specification interface and upon pushing a button, a Blaise instrument can be produced. The generation of Blaise instruments has been demonstrated several times in the seventeen International Blaise Users Conferences. The first was by Pierzchala in 1992 when he demonstrated how the National Agricultural Statistics Service (NASS) could generate over forty versions of the same instrument. It could be, that in order to more easily handle the modern survey world, generating Blaise questionnaires may become more common or even necessary.

Regardless of whether a Blaise generator is used, or all source code is typed into the text editor, the Blaise institute should have considered all its design practices across modes. It is easy for a Blaise programmer to follow standards. It is very difficult to invent these on the fly.

6. Summary

The modern survey world forces instrument designers and developers to re-assess their practices and ways of working. It is such a complex world that no one person has all the answers. Best results for an institute is to plan ahead and standardize its way of handling the nine multiple aspects of CAI design.

Blaise 5, for its part, allows users to encode these standards in the Resource Database and in model source code, settings, and layout sets and groups. Then it is possible to easily bring together all these standards for each device, screen size and mode. The questionnaire designer should focus on the questions. With proper planning and preparation, it should be possible to easily produce instruments that (1) work across all devices, screens sizes, and modes, and (2) be methodologically defensible.

To finally answer the question posed in the abstract: **Blaise 5 – Is Worth the Wait!**

The new features, layout possibilities, language features, and settings, can be combined in any way that is necessary. However, the institute will yield much larger benefit if it re-evaluates its standards and practices as described above then encode its standards and practices in the Resource Database, model source code, model layout set definitions, and model settings. The Blaise 5 Champion instruments give the institute a good test bed to test out these encoded standards. At the same time, Blaise 5 gives you the flexibility to adapt these standards to new situations as they arise.

7. References

De Leeuw, E. D. (2005). To Mix or Not to Mix Data Collection Modes in Surveys. *Journal of Official Statistics*, 21, 2, 233-255.

Dillman, D. A., Smyth, J. D., and Christian, L. M. (2014). *Internet, Phone, Mail, and Mixed-Mode Surveys: The Tailored Design Method*, 4th Edition. New York: John Wiley & Sons.

Iverson, J. and Smith D. (2016). Blaise Colectica Visual Survey Designer. Presentation at the 2016 International Blaise Users Conference, The Hague, The Netherlands, 2016.

Pierzchala, M. (2016). The Blaise 5 Champion Instruments Suite. Poster Session presented at the 2016 International Blaise Users Conference, The Hague, The Netherlands, 2016.

Pierzchala, M. (2006). Disparate Modes and Their Effect on Instrument Design. Paper presented at the 2006 International Blaise Users Conference, Papendal, The Netherlands, 2006 (at <http://www.blaiseusers.org/2006/Papers/207.pdf>).

Pierzchala, M., Wright, D., Wilson, C., Guerino, P. (2004). Instrument Design for a Blaise Multi-Mode web, CATI, and Paper Survey. Paper presented at the 2004 International Blaise User's Conference, Gatineau, Canada, September, 2004 (at <http://www.blaiseusers.org/2004/papers/24.pdf>).

Pierzchala, M. (1992). Generating Multiple Versions of Questionnaires. Paper presented at the 1992 International Blaise Users Conference, Voorburg, The Netherlands, 1992 (at <http://www.blaiseusers.org/1992/papers/GENERATING%20MULTIPLE%20VERSIONS%20OF%20QUESTIONNAIRES.pdf>).

Experiences with Colectica

Felix Coleman, Central Statistics Office, Ireland

1. Abstract

There is currently a transformation program taking place in the Social statistics environment at the Central Statistics Office. A program is under way to develop a more process orientated approach to the production of national statistics. In order to facilitate this transformation, the social statistics questionnaire design team (QDU) is utilising the DDI metadata environment to integrate the transfer of statistical metadata through its statistical production environment. The transformation is designed to develop best practice principles around the creation and storage of statistical metadata. One of the key developments in the work is the systematic mapping of survey design in DDI to the Blaise survey instrument environment and then the "carrying through" and supplementation by Blaise of additional metadata as data exits data collection phase provided by Blaise. This short paper will describe the progress / direction being taken to date with this work and highlight some of the challenges and solutions that have been experienced with the transformation.

2. A brief context: How it was.

Traditionally CSO's household survey process has worked on the basis that its subject matter experts had the responsibility of designing and compiling the questionnaires for their statistical products. As part of this traditional approach to statistical production, the specialist statisticians applied their experience in survey design to the subject matter and produced a questionnaire specification for CSO's Blaise team to develop. This method of design has a number of benefits and drawbacks. Probably the most important perceived benefit is that the statistician responsible for the publication of a survey's results was also responsible / had a direct say over the design of the survey's questionnaire. This also meant that statisticians with the best understanding of the statistical subject matter were able to directly design questionnaires to fit the data requirements of the surveys.

Unfortunately, designing surveys in this way also has some significant drawbacks. These included

- Inconsistent survey design (e.g. the same data being requested from different statistical domains using different questions)
- Poor documentation of statistical process
 - Non standardised
 - Incomplete
- Inconsistent survey specifications provided to IT (Blaise)

So, CSO are developing solutions to some of these issues as part of their household survey transformation project while at the same time maintaining the important benefits that the more traditional approach to survey design provided in the past. To achieve this, we are moving towards the systematic storage of survey information and away from the use of standalone survey specific silos of metadata. This method of data management will allow us to automatically transfer this standardised metadata through the statistical production process to our users (analysis, dissemination, archiving etc.) The preferred method of storing this metadata is by using Data Documentation Initiative (DDI). DDI in simple terms is a standard method of tagging statistical data and metadata elements using XML. One of the leading software tools to apply this tagging is Colectica Designer. This paper explains how Colectica is helping with different parts of CSO's survey transformation project and how we hope to further develop the use of DDI into the future.

3. Survey Development: Collecting information for the Questionnaire Design Unit (QDU).

Part of the procedures being developed by the QDU within the CSO involves the development of and request for Input specification documents from upstream in the statistical process as defined by the Generic Statistical Business Process Model (GSBPM). Generally speaking the input specifications as you might expect contain the important information/metadata that a survey designer needs before they can go to work. More specifically, the lists of metadata items being requested are being drawn from Colectica Designer's design interfaces in order to simplify and highlight the relevant fields of metadata from the large and complicated DDI dictionary that exists.

The Information is being requested with the following principles.

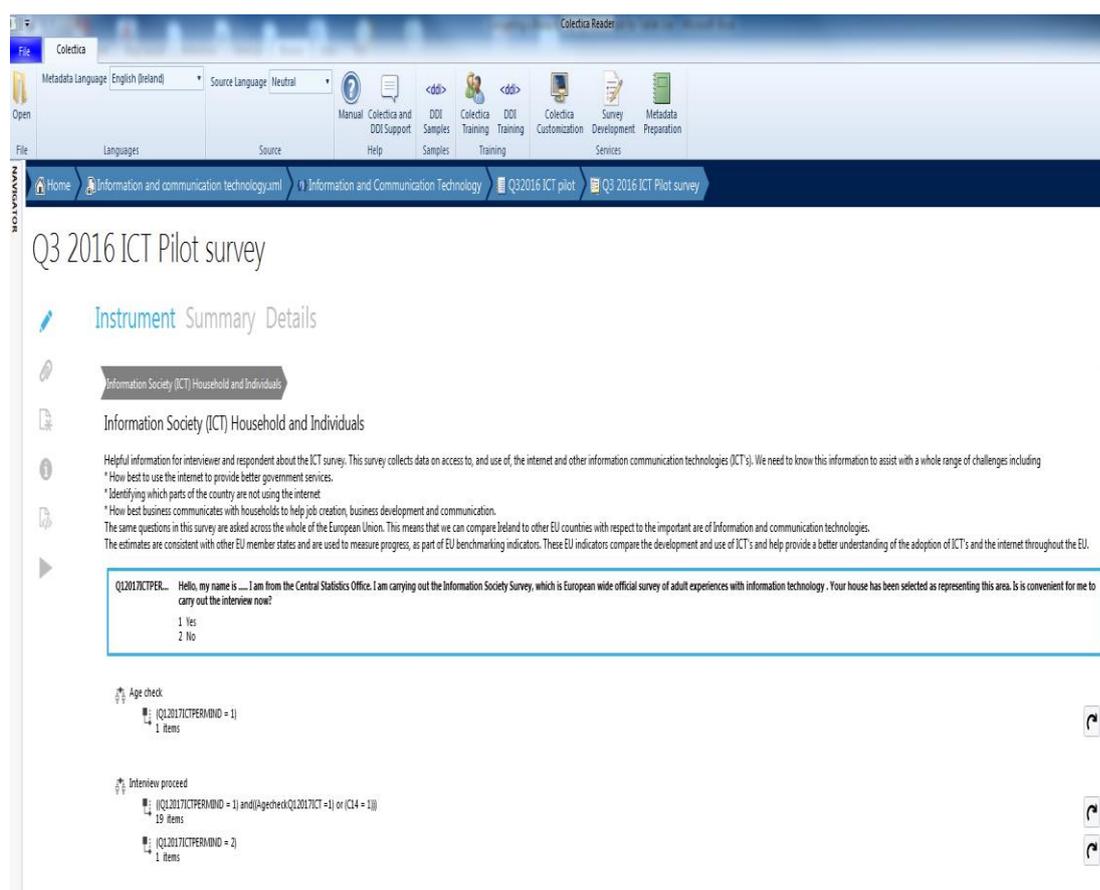
- The metadata that is required by the survey designers is created and transferred to QDU by the persons or departments that carry the responsibility or expertise of the management of these metadata elements.
 - e.g. If a household survey's sampling methodology is being developed specifically by a statistician who is specialised in sampling procedures, then the metadata explaining the sampling procedure is passed onto the QDU by the sampler. (Subject matter specialists will invariably be involved with sample methodologist to organise the sample selection)
- The critical information provided is expressed in standard metadata titles/tags extracted from the DDI dictionary.
 - This introduces a more precise use of the terminology in DDI and begins the stage of structuring the metadata around survey design into DDI compatible pieces.
 - By just initially using the titles of elements and removing the DDI tagging procedures we move away from the need initially for everyone to be familiar with DDI.

An example of one of our initial input specification requirements can be found in Appendix 1.

4. Questionnaire development

Once the input specification process has been completed and submitted the QDU then begins the process of incorporating the survey metadata from the early stages of the GSBPM into the question bank provided by the Colectica software. The process of metadata documentation being captured at the correct time during the production process begins while at the same time creating data in a structured (DDI) environment. In many cases, the metadata that is associated with the early parts of the GSBPM process has been designed over a long period and is already stored in either an NSI's metadata holdings or via Eurostat's documentation

Figure 1. Colectica Reader



5. Questionnaires and Blaise

At the moment CSO is linking questionnaires stored in its Colectica software with the Blaise team using basic output functionality.

- Word / PDF documentation
- XML DDI (which can be read using Colectica Reader)
- Blaise data models created by Colectica

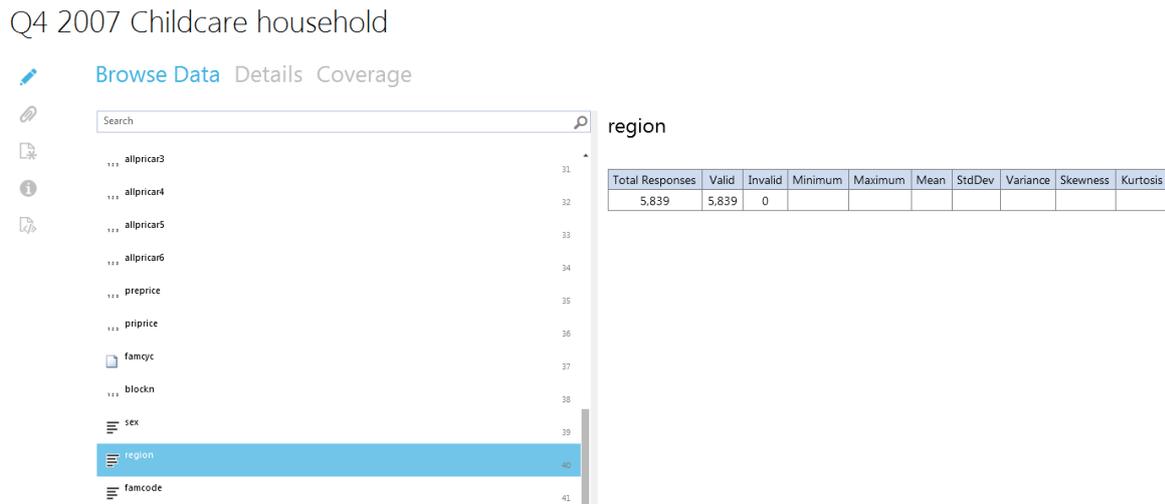
This approach initially creates a more flexible link between our DDI holdings and our Blaise survey instruments allowing the QDU space to develop more consistent metadata packages and a better/greater understanding of DDI and how to structure survey metadata in an organisational context. It is a longer term aim that the questionnaire design team will be able to develop a systematic relationship between the Blaise output from Colectica and the Blaise data models being used in our current social statistics surveys. It is important that the metadata being developed by our subject area specialists in the GSBPM's early stages can be linked systematically into the Blaise process and then output into our metadata and data servers for analysis or publication. At the moment, the Blaise data models being created by Colectica do not match the existing survey instruments we are using in Blaise however the code is useful to our Blaise team as a source for incorporating automatically generated code rather than writing programs from scratch.

6. Non questionnaire data and metadata and Quality reporting metadata

Although the survey design team's main focus is on questionnaire design (question text, answer categories, routing, etc.), it is also important that the specification provided to Blaise includes the

details of additional non question metadata that is required by analysis statisticians in order to comply with statistical regulations and quality reporting. For example, data covering a household's location or the time that interviews took place needs to be included in the metadata specifications but not necessarily the questionnaire specifications. From this point of view Colectica Designer can provide the correct creation / tagging of these elements (as an output variable rather than a question element) but it is not the initial priority of QDU to have this/these in place? We know from experience that data of this nature is being extracted from our Blaise/ survey IT environment and provided as output variables to the results and analysis statisticians. The key part of the process is to be able to incorporate these variables into the documentation process.

Figure 2. Non questionnaire data holding in Colectica Designer



7. To Blaise and beyond.

The Questionnaire Design Unit (QDU) is developing a standard output from Colectica of their work in order to.

- Improve the articulation and precision of questionnaires and specifications being provided to Blaise.
- Better define relevant metadata that is needed to fulfil statistical requirements at the right moment in the statistical production process to work towards best practice principles of documentation and quality.
- Reinforce the principle that there should only be a single source of specific metadata elements in a statistical production process that can be used and reused as required in the overall statistical environment.

In time and in consultation with our IT expertise we hope to look forwards to removing the need for paper based specifications by providing electronic link up of the Colectica data holdings with our other metadata holdings. This will enable the metadata being captured by the questionnaire design team to be used and reused our statistical products move through the production system to dissemination and beyond.

8. Challenges and future improvements

In order to fully benefit from the use of DDI and tools such as Colectica, CSO and other NSI's will need to promote the interoperability of the use of DDI. Some suggestions that could help this process would be

- Early development and sharing of DDI documentation around European statistical products
 - E.g. Model questionnaires provided in DDI rather than traditional Word / Excel.

- (Colectica Designer allows for multiple language entries for each metadata element).
- Statistical production specialisation by member states and sharing DDI expertise and survey metadata development
- Promoting the interoperability of expertise in different stages of the statistical production process using DDI profiles.
- Eurostat's metadata server could become more of a two way system allowing NSI's to extract standardised metadata packages/ elements for statistical products.
- It would be useful to provide a source for standard classifications within DDI as a menu item. For example if a designer wishes to use a list of standard ISO countries as an answer category for a question, they currently need to import the classification. It would promote consistency if perhaps the UN classifications could be updated with the DDI tools environment.

9. Appendix 1: Input specification requirements template for Question Design Unit

Input specification documents / Input metadata requirements

- Survey NAME
- Survey area
- Creator - (RAP Statistician)
- Contributor
- Specification delivery date required
- Output variable listing
- Links to Previous specifications
- Abstract
- Purpose
- Analysis unit, (e.g. households, individuals, working individuals aged between x and x, enterprises)
- Subjects and keywords
- Temporal coverage (reference periods)
- Spatial coverage
- Sample population
- Sample size
- Sample procedure
- Intended frequency
- Mode of collection
- Required response rate
- Related legal regulation
- Testing requirement (quantities, cognitive)
- Cost restraints

10. References

Data Documentation Initiative (DDI) <http://www.ddialliance.org/>

Colectica <http://www.colectica.com/>

Blaise 5 at Statistics Norway

Trond Båshus, Statistics Norway

1. Abstract

Statistics Norway has used Blaise 5 since late 2014. We decided early on to gradually replace Blaise 4 for web survey since we felt it to be increasingly outdated, especially the handling of layout and support for different devices (although we have had success with CMoto from CentERdata). Our view was then that Blaise 5 was superior for web surveys, despite of several crucial features that were still missing.

An important feature which was not finished in 2014 was Manipula. We overcame this by using the supplied data conversion utilities which allowed us to process data with external tools i.e. Blaise 4 Manipula or SAS. Another sorely missed feature was support for other database engines, which made access to Blaise data rather troublesome.

For the last couple of years new features has been added to Blaise at an impressive pace, which is good, because we now have many of the tools we have missed. But this has also presented us with challenges, the most important one that it's not advisable to upgrade a production server with active surveys without doing extensive (and expensive) testing first. Our solution to this challenge has been to set up two parallel production environments.

Even though the setup survey layout has greatly improved in Blaise 5, it can still be a challenge. Especially if there is a need (which there often is) to make changes to the resource database. The resource database is large and be confusing, perhaps much because of the extensive dependability between elements in the database. One of the great horrors of Blaise 4 can still present problems: namely tables. In addition extensive testing with different browsers and devices is still very important.

Blaise 5 now has many features we have waited for: Manipula, support for MySQL (for survey and audit trail data). The next step is integration with our Case management system (just around the corner), and thereafter exploration the CATI-functionality in Blaise 5.

2. Introduction

Statistics Norway has used CAWI as a collection method for social surveys for many years (since at least 2008), using the Blaise 4 software. While Blaise 4 has worked very well for us, and is continuing to do so, it is mainly for web surveys that the software has shown its age. The layout tools are not very user friendly, and the resulting web forms are increasingly looking outdated, especially for respondents using mobile devices.

We have followed the development of Blaise NG, now Blaise 5, with interest for several years. After attending a Blaise 5 layout training course in September 2014, we decided to phase out Blaise 4.8 in favor of Blaise 5 for web surveys, despite several shortcomings in the version of Blaise 5 available at that time.

3. Timeline

Below I will track the development of our Blaise 5 infrastructure and Blaise features in relation the deployed web surveys

3.1 2014

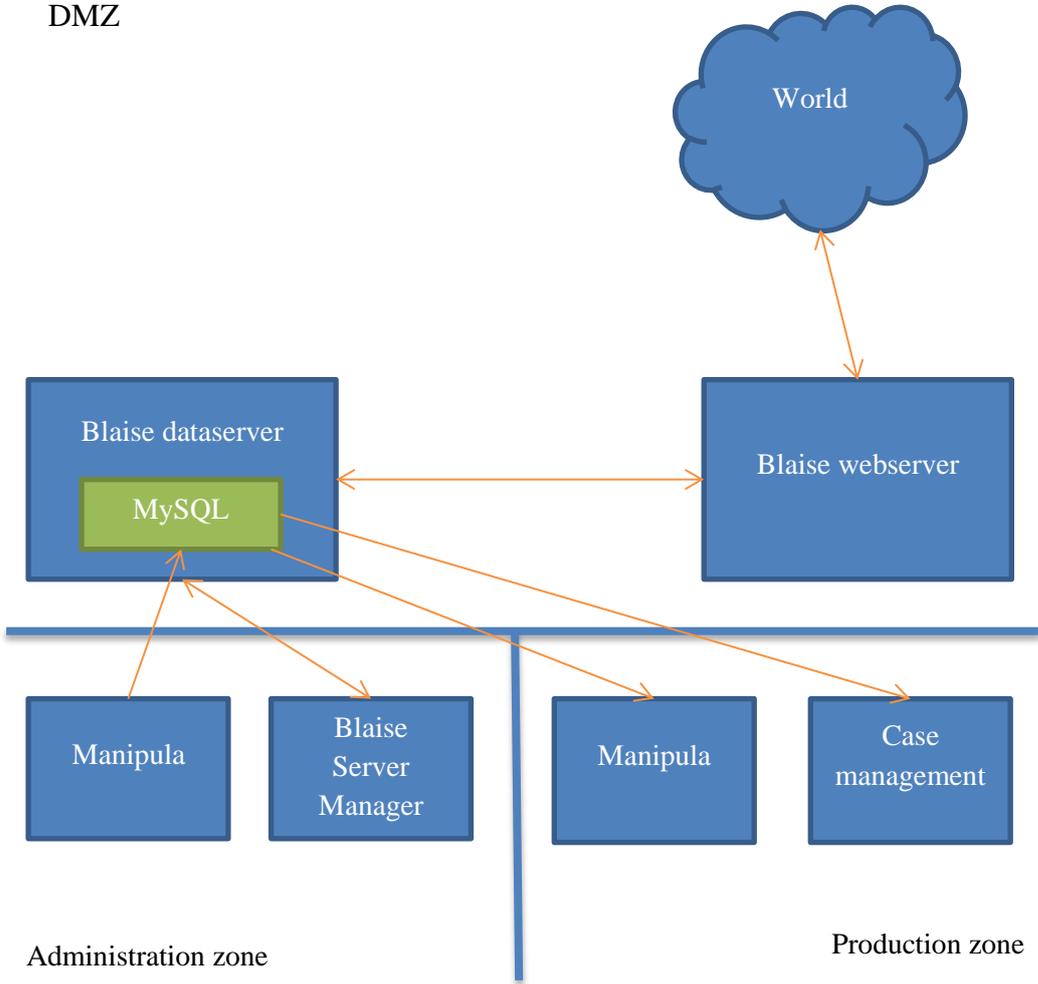
Our first Blaise 5 web survey, which was created when we came home from the Blaise 5 layout training course, was a small web survey aimed at international research institutions. At that time we had just a single server set up for Blaise, and it ran Blaise 4.8 and Blaise 5 in parallel. There were no

support for storage in external database systems, and no working Manipula. With the server located in DMZ, and only accessible through Remote Desktop, access to data was a real hassle. The instrument worked well, however

3.2 2015

This year we reached several important milestones. Firstly we set up two separate server environments, one for test and another one for production. Previously we use normal desktops machines for testing of new versions of Blaise, but this is not a satisfactory solution in the long run. Secondly, we split up the servers in a web-server and a data-server part for increased security. Blaise got support for storage in external database systems, so we set up MySQL on the data-server for storage of Blaise-data. Data on the servers were now accessible through a single port. A working Manipula also helped a lot. We also started testing of the Blaise app for Android

Figure 1. Simplified overview over the Blaise 5 environment at Statistics Norway



3.3 2016

Developments this year has been the introduction of a second production environment, which we introduced to handle upgrades of Blaise without disrupting running surveys. Because of this we avoid time consuming testing which would have been necessary to ensure that running surveys performed as expected on newer versions of Blaise 5. Additional improvements include storage of session- and audit trail data on MySQL-databases.

We had planned to utilize Blaise server events as an input to our case management system (SIV), but because of limitations in the implementation of server events in the then current version of Blaise 5, we decided instead use the audit trail database as a source for updating SIV. It works by monitoring the audit trail database for session start and session end events, and has been tested in a pilot of the Household budget survey.

4. Overview of Surveys

Table 1. Overview of Surveys

Survey	When	Multilanguage	Sample	Notes
<i>Research in Svalbard</i>	2014	No	250 (research organizations)	
<i>Price collection</i>	Yearly from 2014	No	Store managers	
<i>Rental survey</i>	Monthly from November 2015	Yes (2)	2500	
<i>Tourist survey</i>	Triannual from 2015	Yes (14)	3000	Mostly paper, but some with Blaise app
<i>Voters survey</i>	2015	Yes (6), including right to left	18000	
<i>Survey on ICT Usage</i>	2015	No		
<i>Members of council survey</i>	2015	No	428	
<i>Rental survey (for landlords)</i>	Approx. 4 times per year from 2016		Approx. 1500	
<i>Eurostudent</i>	2016	Yes (2)	24000	
<i>Statistics Norway's customers survey</i>	Quarterly	No	150	Customer experience survey
<i>CVTS</i>	2016 (to be conducted)	No		
<i>NorLag3</i>	2016/2017 (to be conducted)	No	7400	Mixed-mode (CATI Blaise 4.8, CAWI and paper)
<i>Housing and rent survey</i>	2016 (to be conducted)	Yes (2)	37000	
<i>Adult Education Survey</i>	2016 (to be conducted)	No	5000	Mixed-mode
<i>Household budget survey</i>	2016/2017 (to be conducted)	No	7500 households	Web survey part in Blaise 5

4.1 Examples

4.1.1 Voters survey

Following the 2015 municipal and county council elections Statistics Norway conducted the Voters survey. Previous voter surveys have usually been CATI-survey, but it was decided to primarily conduct the survey on the web, with paper as an option to selected groups. A specific concern was to examine voting behavior among immigrants and foreign citizens, and it was therefore important to present the survey in several different languages: Norwegian, English, Polish, Lithuanian and Somali. Translated texts were copied from a spreadsheet, and this worked quite well, including Urdu. We had some concerns that Urdu might present a problem, especially when setting up layout, but it worked well out of the box. As is usual for most of our survey, the Voters survey had a small and a large layout aimed at smartphones and PCs/tablets respectively (Lillegård & Torsteinsen 2016)

Figure 2. Voters survey (large layout, Urdu)

Statistisk sentralbyrå
Statistics Norway

Velgerundersøkelsen 2015

اب ہم آپ سے اس بارے میں کچھ سوال پوچھیں گے کہ کیا آپ نے اس سال انتخابات سے پہلے سیاسی مباحثوں یا بات چیت میں حصہ لیا ہے۔
آپ اپنے گھر والوں، دوستوں، واقفوں یا ساتھی کارکنوں سے کتنا اکثر انتخابات کے بارے میں باتیں کرتے رہے؟

- روزانہ
- ہفتے میں چند دفعہ
- مہینے میں چند دفعہ
- اس سے کم
- کبھی نہیں

بچھڑے اگے

Figure 3. Voters survey (small layout, Urdu)

Velgerundersøkelsen 2015

آپ کو بالعموم سیاست میں کتنی دلچسپی ہے؟ کیا آپ کو ...

- بہت دلچسپی ہے
- کافی دلچسپی ہے
- کم دلچسپی ہے
- بالکل دلچسپی نہیں

Figure 4. Voters survey (small layout, English)



4.1.2 Tourist survey

The Tourist survey is mainly a conducted on paper, but is also implemented as a Blaise Android app. Data is collected at hotels and camping sites, and the questionnaire is translated to 14 languages (Haslund 2015).

Figure 5. Tourist survey Blaise Android app



4.1.3 Screenshots from other surveys

Figure 6. Housing and rent survey (small layout with help text)

Bolig og husleie 2016

<
>

? **Kor mange andre opphaldsrom med vindaug har bustaden?**

Ta ikkje med kjøkken, bad og WC.

Andre opphaldsrom Med andre opphaldsrom meiner vi hovudsakleg stuer og arbeidsrom.

Figure 7. NorLag3 (large layout with table)

Statistisk sentralbyrå
Statistics Norway

Livsløp, aldring og generasjon

Nedenfor ser du en liste over egenskaper folk kan ha. Hvor godt stemmer dette for deg?

	Kun ett svar per linje				
	Stemmer ikke i det hele tatt	Stemmer nok så dårlig	Stemmer delvis	Stemmer nok så godt	Stemmer helt
Pågående	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Forståelsesfull	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Har lederegenskaper	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Hengiven	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Trøster gjerne andre	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Forsvarer mine meninger	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Viser medfølelse	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Villig til å ta sjanser	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Øm	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Selvhevdende	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Følsom for andres behov	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Viljesterk	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Forrige
Neste

5. Successes (and some obstacles)

Overall, the introduction of Blaise 5 at Statistics Norway has been a success and we have used Blaise 5 for many surveys since the modest beginnings in 2014. I will try to outline some of the advances which have been of use to us. Firstly, the most obvious change is the Blaise 5 Control Centre which is a considerable improvement. It took some time perhaps to get used to organizing everything into solutions and projects, but this a great help in keeping track of all the files belonging to a specific survey or project.

The editor is much better than before. Personally I have used external editors such as TextPad and Notepad++ for years, instead of the rather primitive Blaise 4 editor. The built in editor now has most of the features one has come to expect. Proper UTF support, for example, makes it quite a bit easier to deal with multi language surveys, including right to left languages such as Urdu.

While the Blaise language is more of less the same as before, there are new useful features here too, such as text roles, and groups. Both features we have used extensively, although it has to be said that groups must be used with care, as it can lead to code which is hard to read and follow.

The biggest change is of course one of the most obvious improvements, and also the main reason we began making the switch to Blaise 5 in the first place. Setting up satisfactory layout in Blaise 4 is not trivial, and the support for small screen sizes is poor. These factors were important when we made our decision to switch to Blaise 5. There are still some issues which can be troublesome, and I will come back to that later, but it is overall fair to say the layout handling in Blaise 5 is in a higher league altogether. I will not go into much detail here, but such a thing layout set, makes it quite easy to support small and large screen sizes. An increasing proportion of our respondents use smart phones, so it is quite essential that our surveys work well on small screens. We have found that it is always worthwhile to first create layout for small screen sizes, and then move on to a layout for larger screens. The reason for this is that generally, a survey which works well on a small screen, will also work well on a large screen, but not necessarily the other way around.

The server environment has on the whole been very stable. Storage of both survey and audit trail data in external databases is easy to set up, and also improved access to data. The ability to store audit trail in a common database opens up for numerous potential uses. Some performance issues have occurred, but these have largely been self-inflicted. We learned for example the hard way that simultaneously sending 24 000 SMS invitations *can* be a bad idea.

We have also encountered some obstacle during the last two years. Some problems has been due to missing features in Blaise 5, mainly related to storage and missing Manipula, but these has been known issues where there have been alternatives. The issues that have caused most headaches are related to differences in behavior and incompatibility between versions of Blaise 5, and some layout related issues.

We have found that it is not advisable to upgrade a server with a running server without doing very extensive testing beforehand. The problems were usually unpredictable changes on how layout is rendered. The fast development of Blaise 5, where every release contains important new features, has made it necessary to upgrade to new versions quite often, but this issue is especially critical if a serious bug has been found, forcing an upgrade within a very tight timeframe. Fortunately CBS has been able to backport fixes, when we have encountered such compatibility issues, although we hope that unpredictable incompatibilities between releases will be resolved or at least improved in the future.

The resource database which contains all layout elements, standard text, etc is also a source of some concern. Adapting the database to our needs, that is applying styles, logos, font sizes and translation of languages, can be quite time consuming. When switching to a newer version of Blaise, it is possible to export the resource database to XML and copy and paste the modified portions into the resource database of the newer version of Blaise 5. This process is not always successful, and it can be difficult

to figure out why it fails. Also, importing modified design elements and templates based on an older version of the resource database may not be desirable, because the original design elements and templates has been improved by CBS. Our approach now is to make as few changes to the resource database as possible, having in mind that any changes should be easy to implement in future releases of Blaise 5.

In our experience, layout is normally easy to set up in Blaise 5, and there have been made considerable improvements in the resource database during the past two years. One area which still is challenging however is tables. It still takes quite a bit of work to create satisfactory tables, but we are confident that this also will be solved in future releases.

When it comes to support from the Blaise support team regarding the issues we have encountered, the response have always been prompt, friendly and very helpful. This support has of course been essential for us.

6. Future

Currently, we are only using Blaise 5 for web surveys (although we have also used the Blaise Android app). In the short term will finalize status synchronization from Blaise 5 to our case management system, SIV. This will remove the need to do many time consuming manual operations, and will hopefully free resources to expand the use of Blaise 5

Looking a bit further into the future:

- Testing and implementation of Blaise 5 for CATI. We plan to do this gradually, and will probably run both Blaise 4 and 5 for some time still.
- Working towards a common portal with more secure authentication for all surveys running on Blaise 5.
- Possible replacement of CAPI-PCs with tablets and the Blaiseapp.

7. Conclusion

Replacement of Blaise 4 with Blaise 5 for web surveys at Statistics Norway has on the whole been a success, and has seen numerous improvements both of the software itself and the our internal infrastructure supporting in. We are impressed with the progress made during these last years, and are looking forward to expand the use of Blaise 5.

8. References

Haslund, Jan. Tourist survey from Blaise 4.8 to Blaise 5 in Norway. The 16th International Blaise Users Conference, 2015.

Lillegård, Magnar and Torsteinsen, Arnhild. Velgerundersøkelsen 2015, Dokumentsasjonsrapport. 2016.

Converting a CATI Instrument from Blaise 4 to Blaise 5 for Pilot Testing

Emily Caron, Vito Wagner – U.S. National Agricultural Statistics Service, and Kathleen O'Reagan – Westat, United States

1. Abstract

This project was collaboration between Westat and NASS. For an ongoing data collection we began to explore how to convert a Blaise 4.8 CATI instrument for multi-mode Blaise 5 data collection. The study involves collecting information about agricultural products. Respondents report on an annual basis so there are many complex edits between questions and a number of questions involving multiple responses and grids.

We describe our initial approach to converting this instrument and using the default layouts available in Blaise 5. A goal of the conversion was to test the Blaise 4 to Blaise 5 tools available, and determine if the code would run successfully as a connected web application as well as through an app on iOS tablets or phones. We present information about the conversion tools used and the process of creating software for a pilot test of this survey in Blaise 5.

2. Introduction

The National Agricultural Statistics Service (NASS) is an agency of the United States Department of Agriculture and is responsible for collecting, editing, and summarizing agriculture data. NASS is the sole agency for producing Agriculture Statistics for the United States. NASS is currently using Blaise build 4.8.4.1915 for CATI data collection and interactive editing, which handles over 120 distinct surveys per year conducted as over 350 separate survey instances. Other non-Blaise systems are currently being used for CAWI and CAPI data collection.

The Blaise development group at NASS contacted Westat in January 2016 to request a demo of Blaise 5 capabilities. Migrating from Blaise 4 to Blaise 5 was inevitable, but how long would it take? What all was involved with this conversion? NASS use of Blaise thus far involved CATI and interactive edit modes, but with Blaise 5 should this expand to CAWI and/or CAPI? More information was needed.

Westat staff visited NASS headquarters in February 2016 and provided an initial demo of Blaise 5. This was a very high level demo geared toward upper management which showcased Westat Blaise 5 instruments across various modes. The demo generated a good deal of interest and discussion at NASS for how Blaise 5 software would play into future data collection efforts. Westat was requested to take a look at one of NASS' shorter surveys to see what time and effort would be involved in converting it from Blaise 4 to Blaise 5.

The 2016 Mink Survey was selected as a prime candidate for this pilot test. This annual survey collects data from Mink operations across the U.S. regarding mink pelts taken and females bred across 10 different color classes. Westat began the pilot test by reviewing the Mink Survey hard copy instrument, which consists of over 30 data items across two pages. The main Mink table from the hard copy is shown in the image below.

Figure 1. Main Mink Table

	Pelts taken from 2015 crop for marketing	Females bred & to be bred to produce kits in 2016
	(Number)	(Number)
Of the total in Items 2 and 3, how many are in each of the following color classes?		
a. Black - (Standard, Pure Dark)	101	201
b. Demi/Wild - (Dark Brown, Ranch Wild, Demi-buff)	102	202
c. Pastel - (Dawn, Orchid)	103	203
d. Sapphire	104	204
e. Blue Iris - (Aleutian, Gunmetal)	105	205
f. Mahogany	106	206
g. Pearl	107	207
h. Lavender - (Lavender-Hope)	108	208
i. Violet - (Cameo, <u>Winterblue</u> , Glacial)	109	209
j. White	110	210
k. Palomino	112	212
l. Other (Specify: _____)	111	211

The 2016 Mink Survey was programmed in Blaise 4.8 and ran as a CATI instrument with interactive edit functionality. Even though the hard copy instrument appeared relatively simple, the source code was made up of over 30 modules due to NASS shell code being included in the instrument. Shell code holds administrative-type fields and rules shared across most NASS Blaise instruments. Some of the questions involved very complex skip and edit logic. In addition the data was monitored across years, so edits are executed related to prior information provided by the survey respondent. Westat received a preload database with prior year data which was needed to populate the data fills and other variables needed to support the interactive edits.

3. Goals of the Conversion Pilot Test.

The primary goal of the pilot test was to determine if the Blaise 4.8 code could be successfully converted into Blaise 5. NASS intends to utilize the multimode capabilities of Blaise 5 in the future and there is a significant amount of code developed in Blaise 4.8 that requires conversion. Blaise’s capabilities to use the same code base and execute in different modes is particularly advantageous for this work.

After converting the instrument, Westat converted the preload database. The conversion tool works for converting data in both directions: from Blaise 4 to Blaise 5 and also from Blaise 5 to Blaise 4. This is important because NASS has backend systems that use Blaise 4 database files.

Since the goals were focused on a proof of concept for larger production activities, Westat did not make any modifications to question wording or formatting to accommodate changes in mode from interviewer administered to self-administered. They also used default question formats, and did not make updates to presentation items such as fonts, logos, or color palate.

4. Converting Survey Instruments to Blaise 5

Blaise 5 includes several tools to support the conversion of existing survey instruments and data from Blaise 4. Source code is converted using the Blaise4to5Source.exe executable.

This is executed by clicking on the Convert Blaise 4 Sources button in the Control Center, filling in the information and clicking the Convert button.

5. After the Conversion

After the source files and database files were converted there were still some conversion coding items that needed to be addressed. Among those were the removal of Blaise 4 functions that are no longer valid in Blaise 5 such as the EnvVar. In addition, some Blaise 4 defaults needed to be stated in Blaise 5 such as setting the attributes of the instrument's Primary Key.

Figure 2. Converting Files

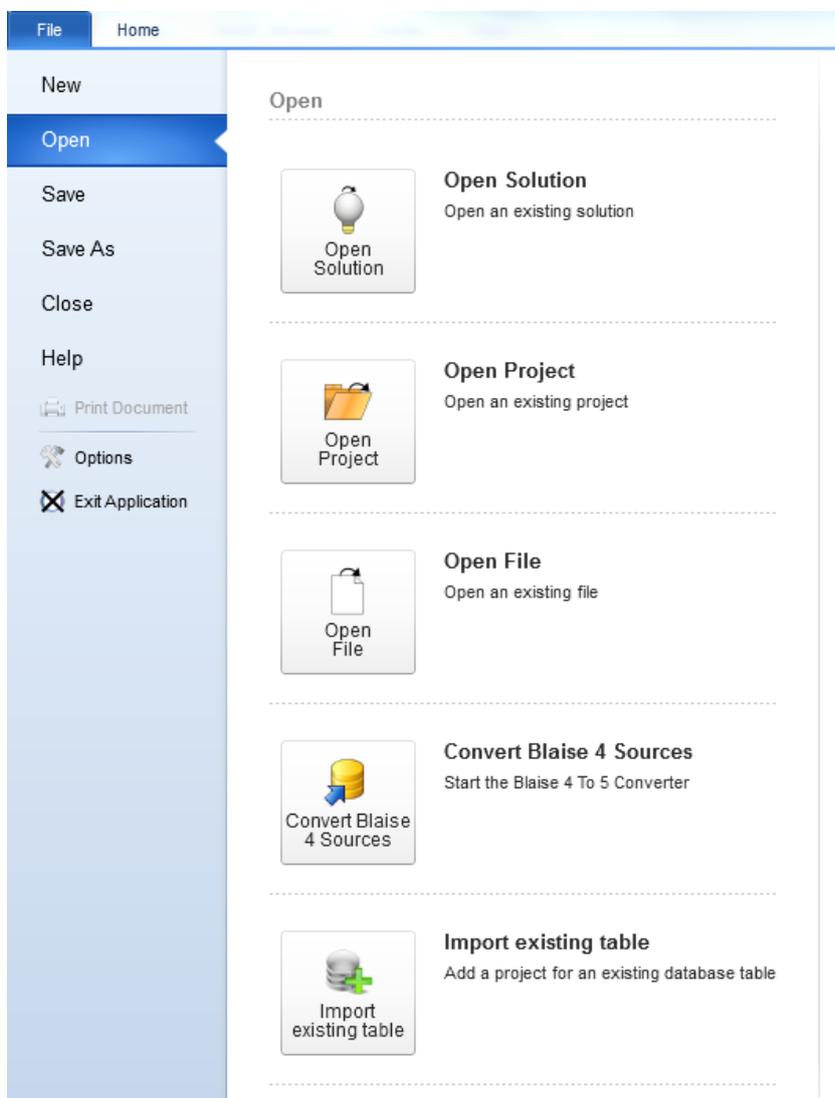
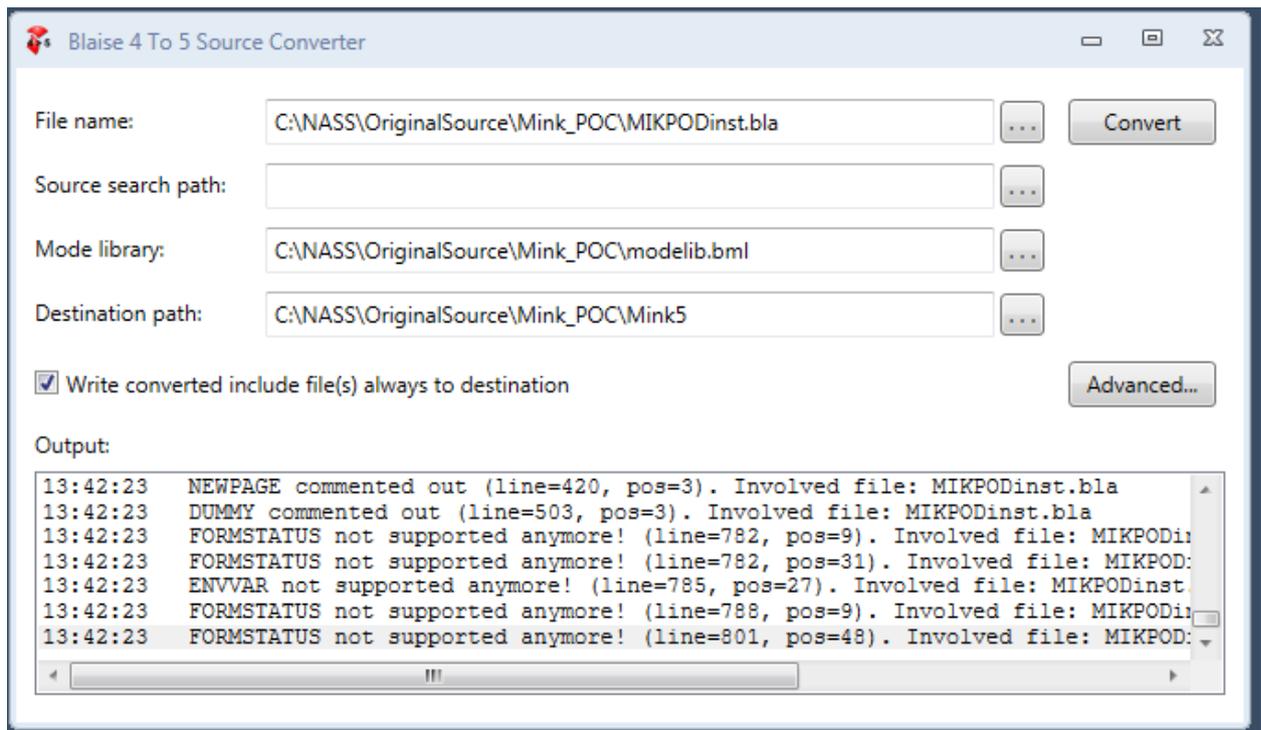


Figure 3. Converting Files



Westat converted the main instrument (MIKPODInst.bla), the master instrument (Master15.bla), and the library (ShellType.lib).

A second tool, Blaise4to5Data.exe, is used to convert data. Data files are converted from Blaise 4 (whether stored as a bdb or through use of a .boi) to the Blaise format specified within the Blaise interface file (.bdix). The default format is the Blaise 5 database file .bdbx. Other database platforms such as MySQL or SQLServer may be configured for use with the .bdix. Once the bdix is configured, the database conversion tool is easy to execute and moves the Blaise 4 data to the database format of choice.

Figure 4. Converting Files

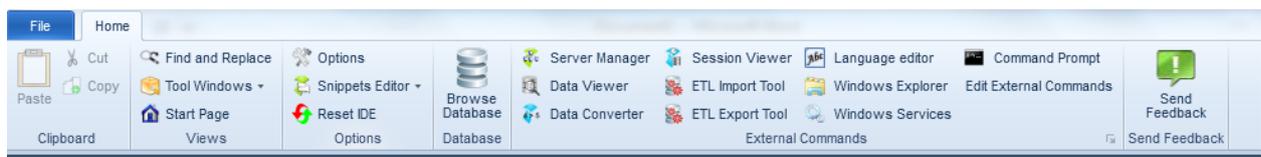


Figure 5. Converting Data

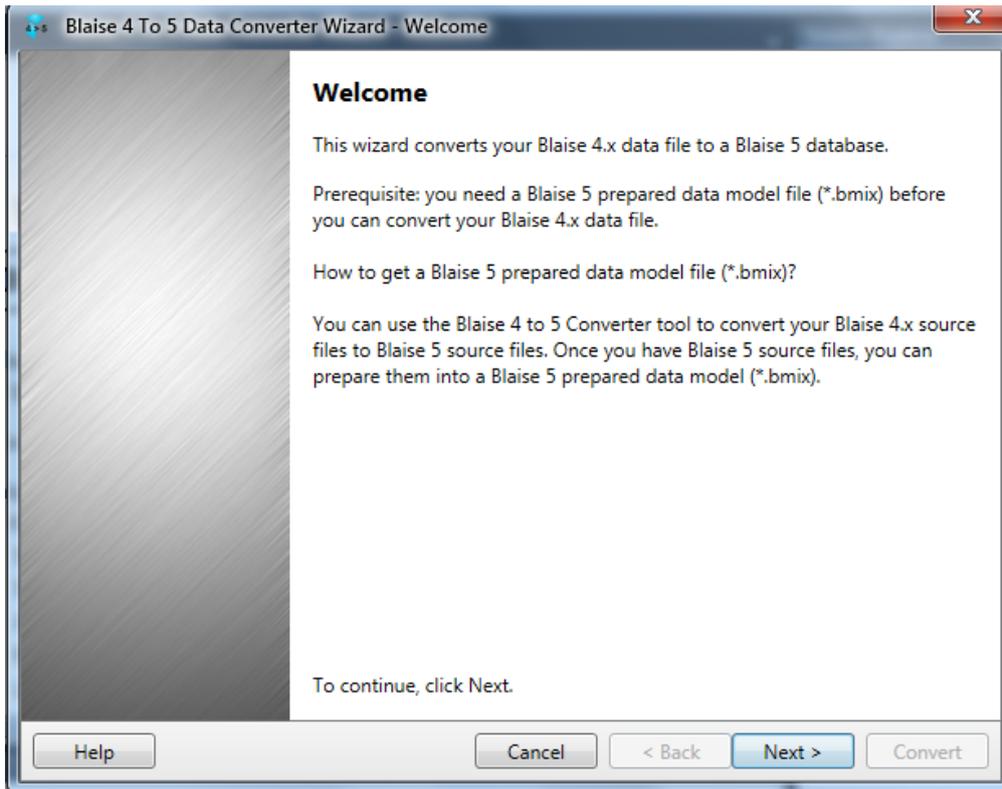


Figure 6. Converting Data

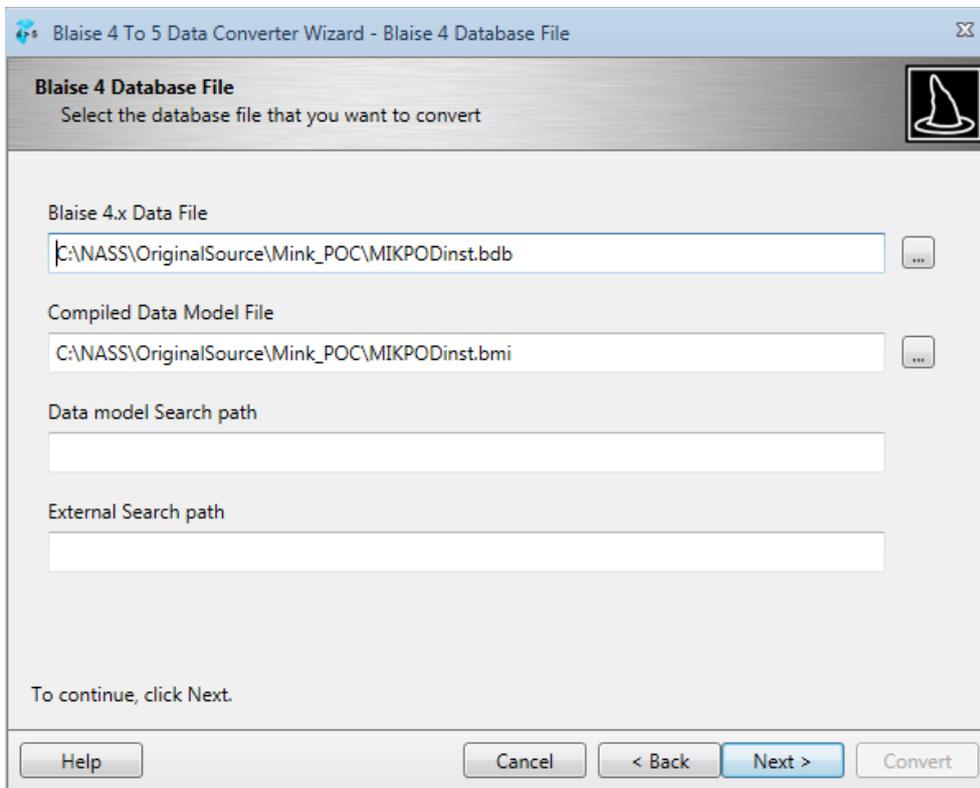


Figure 7. Converting Data

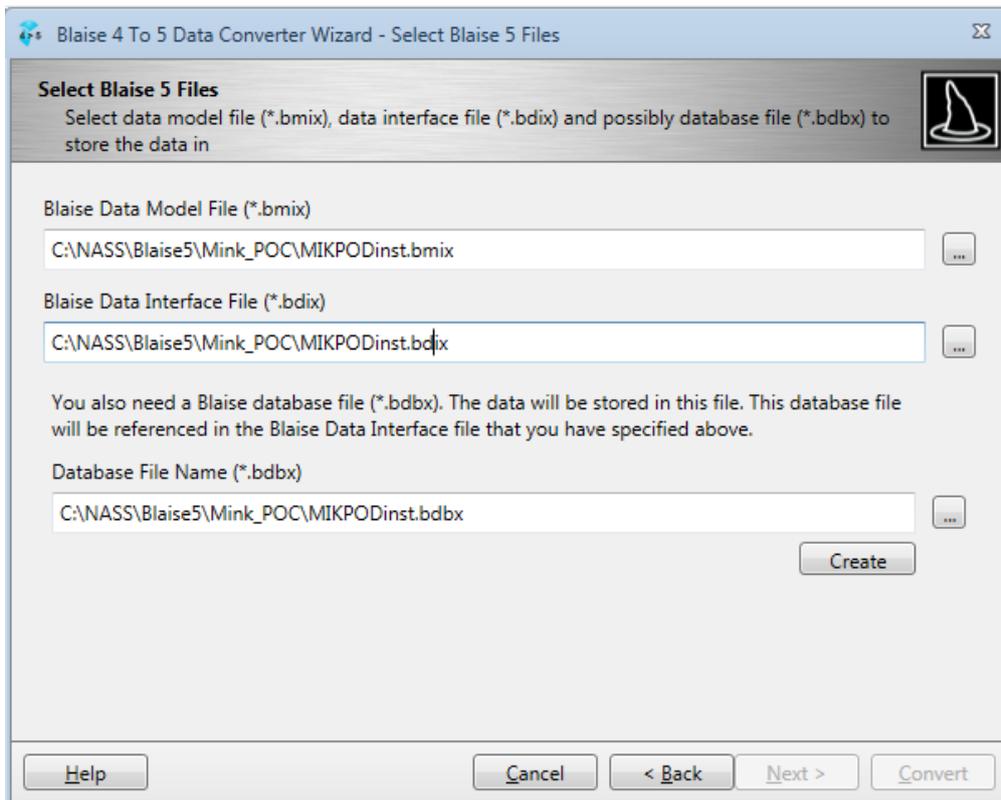


Figure 8. Converting Data

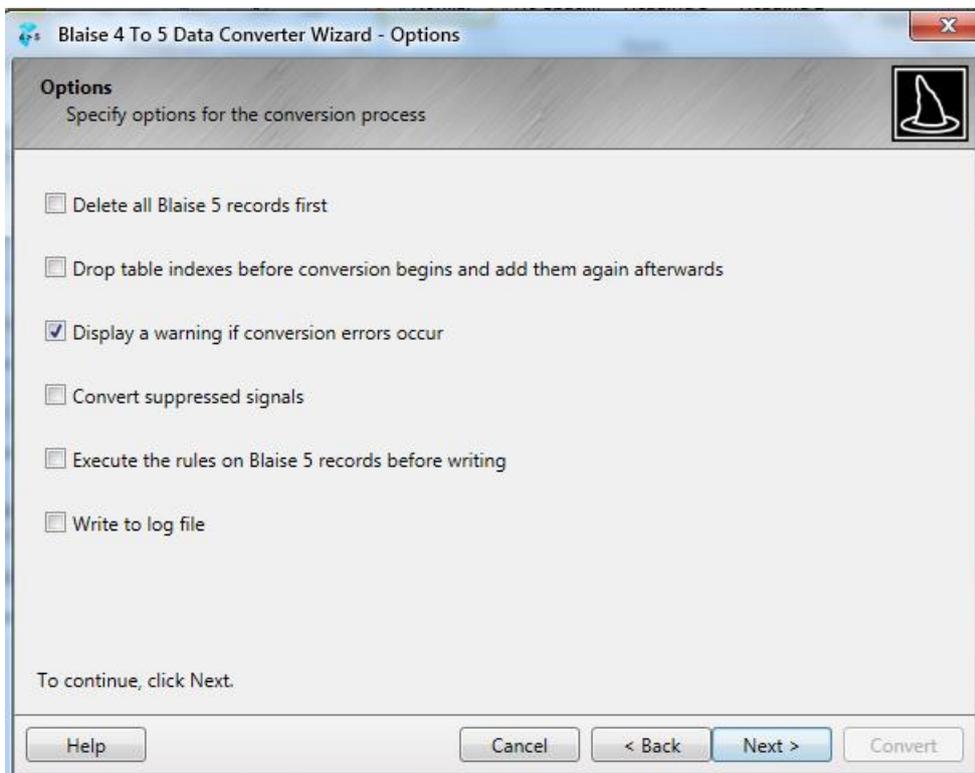
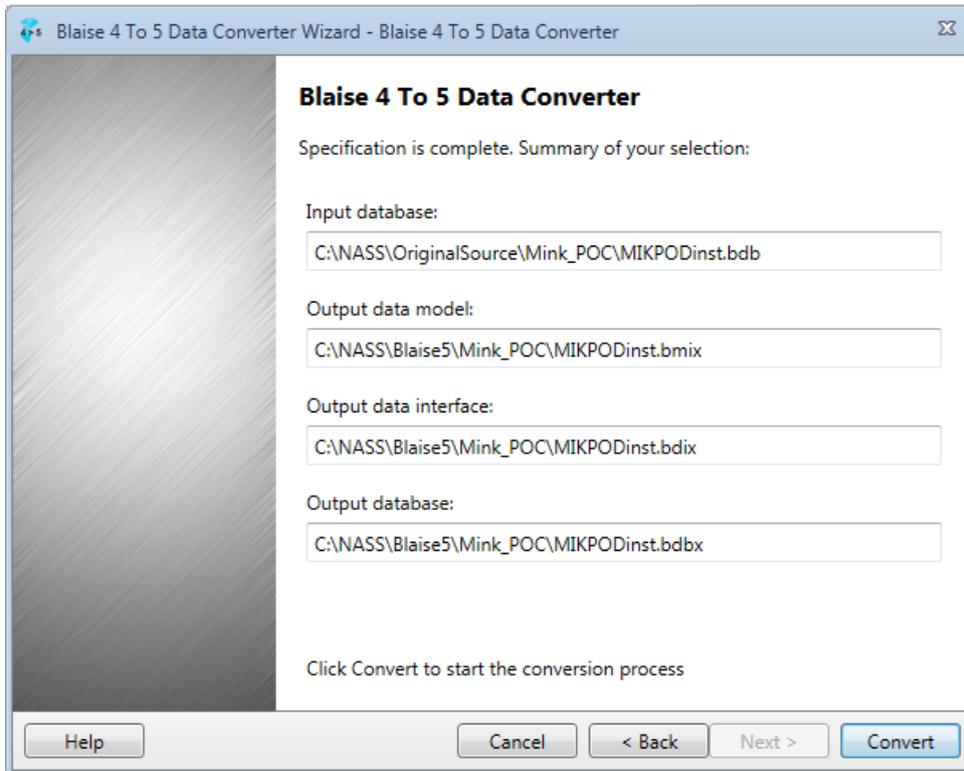


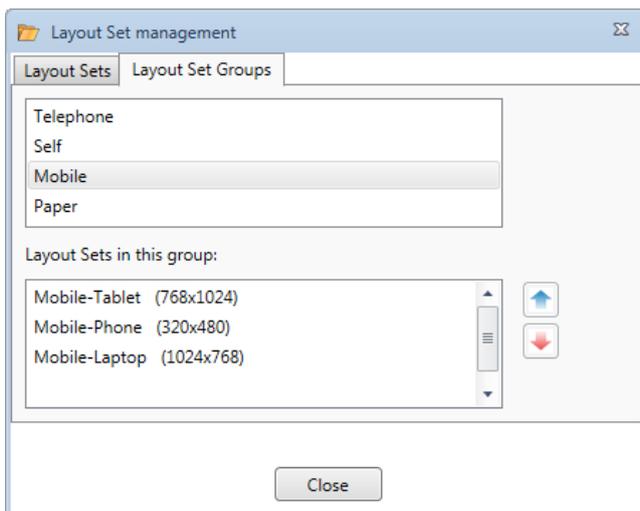
Figure 9. Converting Data



6. Adding Modes

The pilot required demonstrating a Blaise 5 multi-mode capability for the CAWI, CATI and CAPI modes. This was done in the coding by adding Modes and Attribute statements; using the Layout Set Management feature; and making changes to the Resource Database.

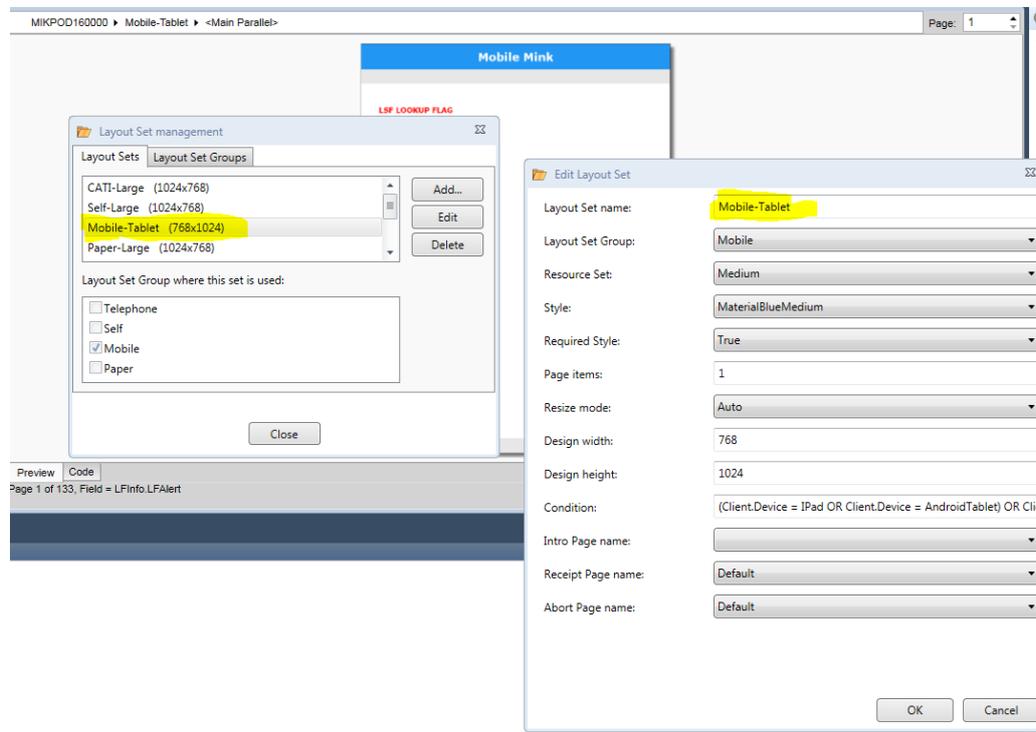
Figure 10. Example listing of Layout Set Groups created based on the Modes statement.



The list of Layout Sets for Mobile is created under the Layout Sets tab (see next screen snapshot.)

Each Layout Set has a set of properties. These properties reflect what is available in the Resource Database.

Figure 11. Layout Sets and Properties



The presentation of the on a device for a given mode is tied to layout set being used. In particular, each Layout set ties to a Resource Set found in the Resource Database. Below are some changes made to Resource Set(s) based on the listed mode.

CAWI

- Copied the Default Row Header template and named it Default1.
- Changed the groupDescription cell's TextSource from "Text or Name" to "Name". This changed the layout from displaying the field's descriptive text instead of the name of the field.
- Copied the FieldRows Table template and named it FieldRows1.
- Deleted the groupText cell and added the fieldText cell. This changed the layout so it displayed the question text for the different columns on the table.
- Phone and Tablet
 - Copied the Data Value template AnswerList and named it AnswerList1.
 - To the AnswerListArea cell, added to the default template collection a Category button and a Special Answer button.
 - Copied the Data Value template SpecialAnswerGrid and named it SpecialAnswerGrid1.
 - To the SpecialAnswerArea cell, added to the default template collection a Special Answer button.
 - Copied the Data Value template SpecialAnswerGridAbreast and named it SpecialAnswerGridAbreast1.
 - To the SpecialAnswerArea cell, added to the default template collection a Special Answer button.

CATI

- Copied the Default Table template and named it MinkTable.
- Added Field Question Text as the first row.

6.1 Code Modifications for Layouts

In Blaise 5 you can group fields together so they can be displayed cohesively by applying a template in the Layout Set (such as Other Specify, address, or tables).

Changes Westat made:

- Enumerated field and “Other Specify” field – Declaration and logic for both fields were moved to a Group statement.
- Tables - Since tables were already declared as a separate blocks there was no need to make any changes in the code.
- Address - Declaration and logic for City, State and Zip fields were moved to a Group statement.

6.1.1 Layouts

- Default templates are applied during the code compilation process. However, in order to improve the look of the instrument, the default templates can be modified or different templates can be applied. For example, table questions in Blaise 5 by default are displayed as individual questions. Blaise 5 has various templates that can be used for displaying a table.
- Different templates were applied for the Other Specify, address, and table fields.
- Fields were made “critical” on the tables so that the display fields would resolve when moving from row to row.
- Pagination changes were made.

6.2 Modifications to Project’s Resource Database

Changes Westat made to templates:

- Copied the Field Pane template QuestionTextOnly and named it “LabelTextOnly”.
- Changed the font to “W”.
- Changed the Field Text Property TextSource to “Value”.
- Added an Applicability Condition: POSITION(‘Label’, FieldDefinition.LocalName)>0
- Moved LabelTextOnly above the Vertical template in the stack.
- This ensured that this layout was automatically assigned to Auxfields containing the word ‘label’ and they would appear green.
- Copied the Table template Abreast and named it CityStateZip.
- Removed “groupText” from its cell.

7. Summary of Results

The conversion exercise met the goal of successfully converting Blaise 4 CATI instrumentation into Blaise 5. The Blaise 5 instrument executes in CATI, CAWI and CAPI modes. The conversion tools are easy to use and this exercise laid the groundwork for increasing NASS’ understanding of how existing Blaise 4 code can be converted to Blaise 5, and how additional data collection modes can be added at the same time.

Figure 12. Web Mink

Web Mink

Please update any field that is incorrect.

The street address/route number of this operation is: 1234 SOMEWHERE LANE

Rather not answer
 Don't know

The additional street address/route number of this operation is:

Rather not answer
 Don't know

The city/town of this operation is: ANYWHERE

Rather not answer
 Don't know

The state abbreviation is: ID

Rather not answer
 Don't know

The zip code for this operation is: 22222

Rather not answer
 Don't know

Figure 13. Web Mink

Web Mink

MAKE UPDATES HERE

Please update any field that is incorrect.

The operation name for this operation is: SUNNY ACRES

Rather not answer
 Don't know

The person name for this operation is: IMA FARMER 10

Rather not answer
 Don't know

8. What the Future Holds

NASS and Westat continue to communicate on Blaise conversion topics, such as how to best implement Blaise 5 in NASS' very unique network structure, further consideration and advice on how to run CAWI and CAPI modes in Blaise, and others. NASS has created an in-house Blaise 5 team to head up the conversion effort. Actual implementation dates have yet to be determined.

Chitwan Valley Family Study Household Registry: Paper to Computer-Assisted Interview

Karl Dinkelmann, Stephanie Chardoul – University of Michigan Survey Research Center, United States, and Bishnu Adhikari – Institute for Social & Environmental Research, Nepal

1. Abstract

The Chitwan Valley Family Study (CVFS) is a longitudinal data collection effort conducted by the Institute for Social & Environmental Research – Nepal (ISER-N). The CVFS has been conducted since 1996 and includes full life histories for more than 10,000 individuals. The study investigates the influence of social contexts on population processes and the relationships between the environment and population processes (<http://spe.psc.isr.umich.edu/research/cvfs.html>). For more than 15 years, the ISER-N staff used complex paper questionnaires, where monthly household data were collected for individual members living, marital, pregnancy, and education status. The paper instrument included extensive hand transferring of data between forms and from wave to wave.

In 2013, a team at the University of Michigan’s Survey Research Center began a collaborative effort with an ISER-N team to convert their CVFS Household Registry’s paper instrument to a Computer-Assisted Interview (CAI) application. This paper discusses the challenges and complexities of converting a multi-member multi-component longitudinal household registry paper instrument into a fully dynamic CAI application. This paper will include:

- An overview of the CVFS Household Registry paper questionnaires and the advantages and disadvantages of using paper vs. moving to CAI.
- Unique CAI features including: 13 Blaise parallel blocks with a household status tab offering the flexibility to update family composition and collect individual monthly member information seamlessly while bringing survey components online or offline as eligibility criteria are met.
- The ability to “spawn” up to three additional Blaise data models on the fly from the main Blaise application to collect an individual’s away status, marital status, or childbirth within any given month.
- Obstacles encountered and review of areas where efficiencies could be applied to the CAI application to aid in the administration of the CVFS.
- Future developments based on the success of the CVFS Household Registry conversion to CAI and lessons learned from this complex multi-member and multi-component application that can be applied to future projects.

2. The Chitwan Valley Family Study

The Chitwan Valley Family Study (CVFS) is a longitudinal study conducted in Nepal by the Institute for Social and Environmental Research – Nepal (ISER-N) since 1996 and includes full life histories for more than 10,000 individuals and tracks continuous measurement of community change both within the original study neighborhoods as well as with respondents who move out of those neighborhoods regardless of where they go. The study investigates the influence of social contexts on population and the relationships between the environment and population processes.¹ ISER-N and the CVFS (along with other studies) are directed by researchers within the Population Studies Center (PSC) and the Survey Research Center (SRC) at University of Michigan’s Institute for Social Research.

For over 16 years the ISER-N staff used complex paper questionnaires to collect monthly household roster data as well as the living, marital, pregnancy, and education status of each individual household member. This “paper and pencil interview” (PAPI) included extensive editing and manual transferring of information from form to form, and from wave to wave. In 2013, a team at the

University of Michigan's Survey Research Center's Survey Research Operations (SRO) unit began a collaborative effort with ISER-N to convert their CVFS Household Registry's PAPI instrument to a Computer-Assisted Interview (CAI) system. This paper discusses the challenges and complexities of converting a multi-person multi-component longitudinal household registry paper instrument into a fully dynamic CAPI system – in a non-U.S. setting.

As in other longitudinal studies, interviewers visit the panel households regularly to update the family composition and collect contact information on individuals who have moved out or in. Following CVFS's complex rules, the SRO team's task was to develop a multi-module Blaise application to allow the interviewers to seamlessly move from the family roster module to customized follow-up interview modules with eligible household members. In addition to the Blaise application, SRO used its proprietary "SurveyTrak International" system, an electronic sample management system modeled after the domestic SurveyTrak. Standard SurveyTrak functionality allows the interviewers to manage their caseload, transfer cases between interviewers, record the outcome of every contact attempt, access the Blaise questionnaire, and transmit completed survey data and process data. SurveyTrak International has the added feature of selecting language for display and simplifies some of the structure and code. For the CAPI CVFS Household Registry project, interviewers in Nepal collect the interviews on laptops and then transmit the encrypted data via the internet to a server designated for international projects housed at SRO in Michigan.

Preparing the PAPI CVFS Household Registry interview required transcribing by hand all the previous waves' household members and relationships into the paper questionnaire. This is similar to how one would programmatically preload data into a CAPI instrument, and would be an entire operation from the field office before data collection begins. The pre-filled paper questionnaires were given to interviewers, and interviewers would take the correct questionnaire to each household. The interviewer begins the interview by updating the Household Registry, updating the columns with each person's status, and capturing information on newly added members, and their individual status. Depending on each individual's situation, various follow-up modules may be asked (e.g., if there were a marriage or birth). Where some parts of the interview are completed by a household-level reporter, others are completed by the individuals within the household.

The main PAPI Household Registry instrument is represented below in Figure 1. This is where the interviewer would hand-transfer the data on the right-side and capture the current wave's information about each household member's living, marital, pregnancy, and education status by month.

to CAPI for the CVFS Household Registry was the cost of printing the PAPI instrument – paper is extremely expensive in Nepal, and regular interviewing of approximately 5000 households was very paper intensive. The alternative was to print the questionnaires at the University of Michigan and then ship them to Nepal – also very expensive! A trade-off was the need for hardware to conduct CAPI interviews. An easy solution was to re-purpose laptops that had been fully amortized and retired from other SRC studies; it was more cost-effective to wipe them clean and re-load them with required software then ship them to Nepal than to keep printing paper questionnaires.

Many of the other reasons are consistent with the experiences of other organizations and studies that have already made the transition from PAPI to CAPI in the last two decades. Chief among them: the reduction of interviewer-introduced error within an extremely complex instrument, and increased speed of data processing and the ability to eliminate logistical steps of shipping questionnaires to the main office in Chitwan and then data entry. Another important factor is the ability to use an electronic sample management system to increase the availability of timely paradata to effectively manage and report on production. Perhaps the most compelling is ISER-N striving to be a leader in Southeast Asia for adopting best practices and techniques, and the ability to explore multi-modal options for the study moving forward (for example, ISER-N is currently experimenting with mobile and text messages for data collection).

Ultimately, the process of transforming the PAPI instrument to the present CAPI system took several paths. We began by collecting and understanding all the PAPI materials, and assimilated their content and objectives to begin to create the CAPI and sample management specifications. We also were fully aware of needing to do as much ex-ante harmonization as possible to assist in the PAPI and CAPI data compatibility. In many ways, the CAPI instrument attempts to mirror how the PAPI instrument was administered.

In the beginning, a proof of concept was created and we held a series of prototyping meetings to review if the overall designs would work for the project. Once the overall design was approved by the research team and the look and feel of the instrument was set, we began programming the instrument - a process that took two separate rounds. The result of the first round was version 1 of the instrument and was used to collect one full wave of data and was considered a six-month pilot. The second round resulted in some significant structure changes to address some features that we not able to incorporate into V1 and to add functionality based on feedback received from the ISER-N interviewing and supervising teams. Version 2 has been in production since March 2014. Work on version 3 will begin in the fall of 2016 and will go into production in 2017. The major change in version 3 is to extend the months available in the system so that data collection can continue through January 2022.

In addition to developing the technical systems, the SRO team provided extensive training to the ISER-N technical lead during a two-week visit in Ann Arbor prior to releasing V1. The training sessions covered the use of Blaise and SurveyTrak, importing translated text into the systems, understanding the structure of the database, how to train their own interviewers on the systems, how to monitor production using our reporting structure, etc.

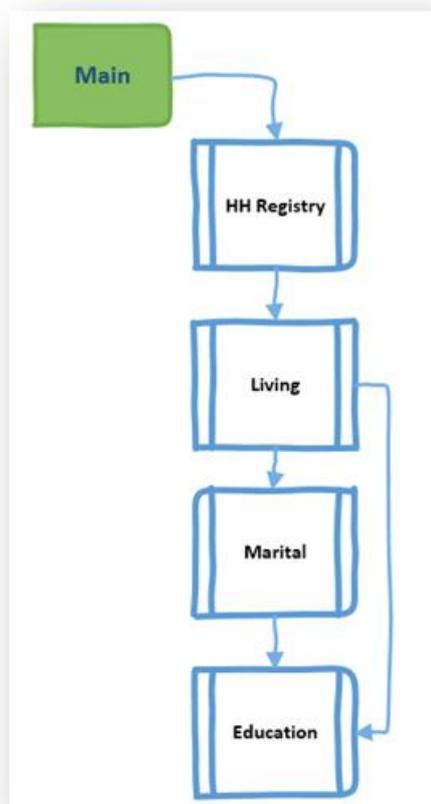
CAPI instrument was designed to mirror the PAPI process, both to maintain the original PAPI data structure and to help the interviewers adapt to the new technology. This design involved using 13 Blaise parallel blocks where one of those blocks served as the household status tab (main household roster). This tab is the first thing the interviewer sees upon entering the instrument and the last thing they see upon completing the survey. The roster tab offers the flexibility to update family composition and collect individual monthly member information seamlessly while bringing survey components (questionnaire supplements) online or offline as eligibility criteria are met.

One of the unique features of the Blaise software is the ability to use parallel blocks. However, not all programmers know about this feature or its uses. The Blaise online help says blocks are taken out of the “sequential processing order as specified in the rules” and “enables one to process one or more

blocks separately from the current route in the Data Entry Program (DEP).” The online help goes on to say this can be used for many situations, including: concurrent interviewing, appointments blocks, nonresponse blocks, notes from previous surveys, etc. Parallel blocks can be displayed on individual tabs, within the same window, accessed from the menu, function keys, using custom onscreen buttons, or via hyperlinks imbedded in text display in the infopane. Any defined block can be defined as a parallel. However, the important difference is once the block is defined as a parallel block, it is taken out of the typical vertical logic presentation and placed in what could be seen as a horizontal logic presentation.

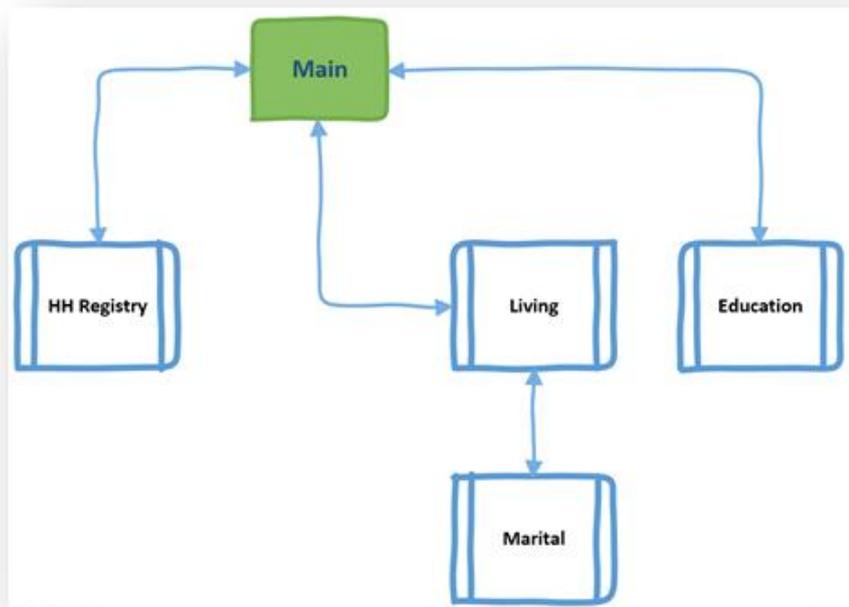
Looking the traditional vertical presentation of the logic within Blaise instruments, sections are asked in in the sequential order they are presented in the logic. Using the CVFS as an example, figure 3.1. below shows this traditional vertical presentation where the sections are asked in successive order until one reaches the end of the instrument.

Figure 3.1. Traditional Blaise Vertical Presentation



However, when we begin to think of Parallel Blocks within Blaise, we have the opportunity to take this vertical logic presentation and transpose it by presenting these blocks in a horizontal structure. Figure 3.2 below shows the same example and how that might look if the modules blocks were defined as Parallel Blocks.

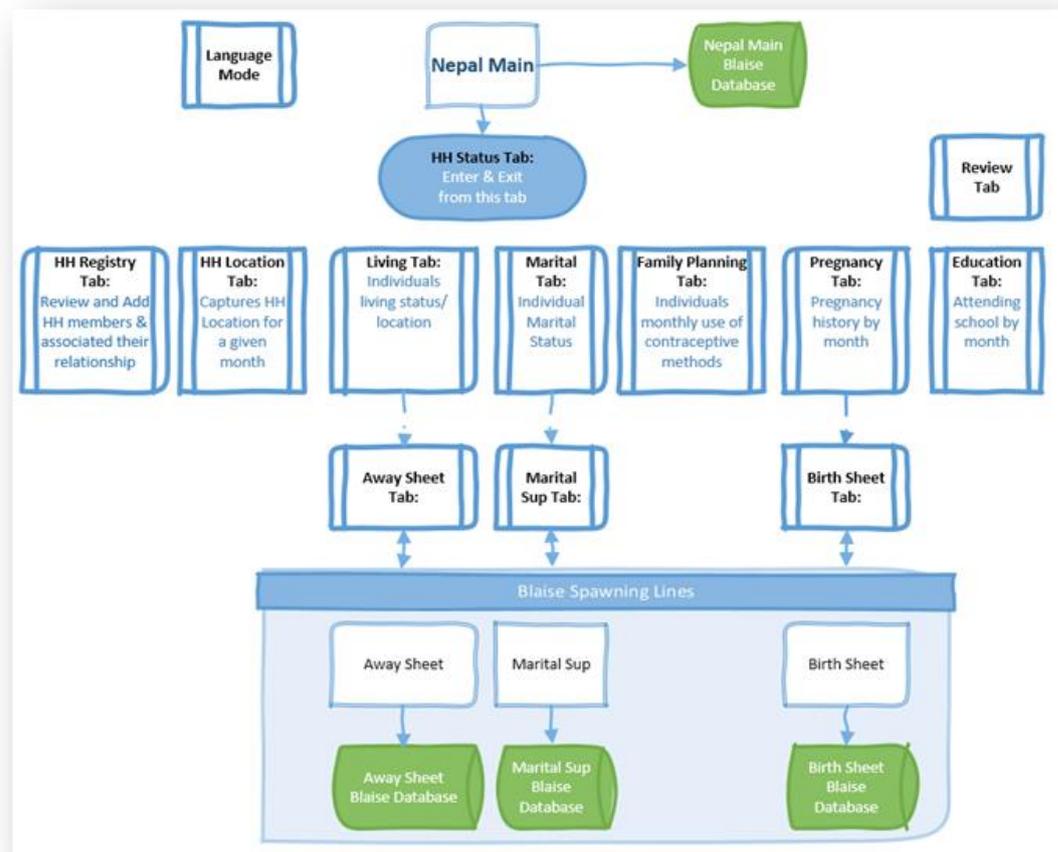
Figure 3.2. Parallel Blocks Presentation in Blaise



Essentially, the horizontal presentation of parallel blocks allows one the ability to bring blocks online or offline as needed within the datamodel. Using the same example, the Household Registry, Living, and Education sections are all online and accessible at the same time. This transposed horizontal structure forms the foundation of how the CAPI Household Registry was created for the Chitwan Valley Family Study Household Registry in Nepal. The use of parallel blocks allowed the best way to replicate how things were done in PAPI for over two decades. While designing the CAPI version of the instrument it was felt users' transition and assimilation to new CAPI instrument would happen faster and their experience would be more enjoyable. Additionally, we needed to allow the interviewer the flexibility to capture the data as needed within a given household (I.e. with the household lever reporter or the individual respondents who were available at a given visit).

Overall the Chitwan Valley Family Study Household Registry Blaise instrument was designed to accommodate several years of data collection where the ISER-N team would reuse the same instrument from wave to wave. Any particular wave asks retrospective information of up to 30 household members for the current month and the previous five months. In addition to "traditional" households, the roster also needed to expand to fit extended families, and larger group quarters. It is designed to accommodate 47 months total. All this flexibility means there are a more than 27,000 variables but only a fraction of them are used in any given survey. Reusing the same application on successive wave's means that logic needs to be dynamic and the instrument makes use of arrays where the household members loop through months using the data collection month index as the array index. Figure 4 below shows an overview of the various parallel blocks with in the CVFS's Blaise instrument.

Figure 4. CVFS Blaise Instrument Overview



There are 14 parallel blocks in total and all but three are displayed in parallel tabs. The three that are not displayed on their own tabs are the main Nepal block, the global status, and language mode blocks. The language block is accessed via a menu command. The only way to complete the instrument is to proceed past the global status block (or the HH Status Tab). All the remaining parallel blocks are routed back to the global status block at the end of the block.

The user always enters the instrument through the HH Status tab, where they see a snapshot of the sections that are on route or have recently become on route as well as the status for each of those sections for each member within the given household. Below are a series of images (in figures 5.1 – 5.5) that represent the look and feel of the CVFS CAPI instrument.

Figure 5.1a. Household Status tab in Nepalese

परिवारको सदस्यको स्थिति:

क्र०	नाम	घरघुरी घटना दती	बसाइ-बसाइ स्थिति	बैवाहिक स्थिति	सर्वांगण स्थिति	शिक्षा
001		1	1	1	1	1
002		1	1	1		1
003		1				
004		1				
005		1	1			1
006		1	1			

• घरघुरीको स्थिति: घरघुरी

• घरघुरीको टेमान Status: योग्य

• कृपया घरघुरी घटना दती मा कप परिवारका सदस्य भर गर्नुहोस् ।

0 = समाप्त 1 = योग्य 2 = शुरु 3 = मृत्यु भयो 4 = घर बाहिर

AS = घर बाहिर गएको जन्मकारी
MS = विवाह सम्बन्धी जन्मकारी
FP = परिवार नियोजन सम्बन्धी जन्मकारी
BS = बच्को जन्म सम्बन्धी जन्मकारी

मौड-बाटो/यो घरघुरीको स्थिति

GStatus.GlobalStatus 203090 9/8/2016 Version Date: 3/1/2014 Version Time: 11:11AM Version: 2.11 2.0.11.2179

Figure 5.1.b. Household Status tab in English

Household Member Status:

SN	Name	Registry	Living	Marital	Pregnancy	Education
001			Eligible	Eligible	Eligible	Eligible
002			Eligible	Eligible		Eligible
003			Eligible			
004			Eligible			
005			Eligible	Eligible		Eligible
006			Eligible	Eligible		

• Household Type: intact

• Household Location Status: Eligible

• Outstanding! The Registry tab has been completed, please finish any items that are not completed.

Navigation / Current HH Status

GStatus.GlobalStatus 203090 9/8/2016 Version Date: 3/1/2014 Version Time: 11:11AM Version: 2.11 2.0.11.2179

Figure 5.4. The Living Status Block

SN	Name	Update	Status	Ask	AS	AS #	200	201	202
001		5		0	0		0	0	0
002		5		0	0		0	0	0
003		5		0	0		0	0	0
004		5		0	0		0	0	0
005		5		0	0		MO	MO	MO
006		1		1	1		042	042	042

After every section has been completed, the interviewer returns to the HH Status tab where the status of the sections is updated. The top of figure 5.5 shows what this looks like as the living section is completed and person 6 is away and has supplemental Away Sheets to fill out (indicated by the “(AS)” which is hyperlinked to jump to the Away Parallel tab. The bottom of figure 5.5 shows what this looks like after completing the Marital section and two follow up family planning sections are needed (indicated by the newly inserted “(FP)”

Figure 5.5. Example of Supplemental Blocks Coming On-Route

Top Screenshot:

Living	Marital	Pregnancy	Education
Registry	Eligible	Eligible	Eligible
Complete	Eligible		Eligible
Complete			
Complete			
Complete			
Complete	Eligible		Eligible
Complete (AS)	Eligible		

Bottom Screenshot:

Living	Marital	Pregnancy	Education
Registry	Complete (FP)	Eligible	Eligible
Complete	Complete (FP)		Eligible
Complete			
Complete			
Complete	Complete		Eligible
Complete (AS)	Complete		

Some new features added to Version 2 of the CVFS was the addition of Blaise Alien procedures that makes a call using Manipula to up to three external Blaise data models. These procedures “spawn” up to three additional Blaise datamodels on-the-fly from the main Blaise application to collect an individual’s away status, marital status, or childbirth within any given month. This “spawn” process creates a zero-to-many situation and turns the CVFS Blaise Household Registry instruments into a relational database. If we were to add all the data locations for the 30 members across all the months in the instrument we would be adding in excess of 67,000 variables to the instrument. Since the three supplemental questionnaires are based on a given month for a subsection of individuals, we decided to make them their own Blaise datamodels; where Blaise Manipula creates a unique sample id (secondary key in the main database) and spawns to a secondary Blaise instrument to collect information for the month in question.

One of the more interesting cross-cultural challenges this project faced was the fact that in Nepal they use a different calendar than is used in most of Europe and the United States. The Nepali calendar is based on an official Hindu calendar known as the Bikram Sambat (B.S.) calendar. This is approximately 56 years, 8 months and 16 days ahead of the typical Gregorian calendar date. In figure 6 below, we see a mapping of how the instrument determines dates within the CVFS instrument. The nature of the Household Registry project is extremely date dependent, so understanding the Bikram Sambat calendar and being able to map it to the Blaise structure was critical.

Figure 6. Gregorian and Bikram Sambat Calendar Crosswalk

Year	HHR Round	Month Index	Gregorian Month & Year	G-CM	B.S. Month & Year	N-CM
CVFS Year 17	33	193	February 2 2013	1358	Magh 10 2069	2038
		194	March 3 2013	1359	Fagun 11 2069	2039
		195	April 4 2013	1360	Chaitra 12 2069	2040
		196	May 5 2013	1361	Baisakh 1 2070	2041
		197	June 6 2013	1362	Jestha 2 2070	2042
		198	July 7 2013	1363	Asar 3 2070	2043
	34	199	August 8 2013	1364	Saun 4 2070	2044
		200	September 9 2013	1365	Bhadra 5 2070	2045
		201	October 10 2013	1366	Asoj 6 2070	2046
		202	November 11 2013	1367	Kartik 7 2070	2047
		203	December 12 2013	1368	Marg 8 2070	2048
		204	January 1 2014	1369	Poush 9 2070	2049
CVFS Year 18	35	205	February 2 2014	1370	Magh 10 2070	2050
		206	March 3 2014	1371	Fagun 11 2070	2051
		207	April 4 2014	1372	Chaitra 12 2070	2052
		208	May 5 2014	1373	Baisakh 1 2071	2053
		209	June 6 2014	1374	Jestha 2 2071	2054
		210	July 7 2014	1375	Asar 3 2071	2055
	36	211	August 8 2014	1376	Saun 4 2071	2056
		212	September 9 2014	1377	Bhadra 5 2071	2057
		213	October 10 2014	1378	Asoj 6 2071	2058
		214	November 11 2014	1379	Kartik 7 2071	2059
		215	December 12 2014	1380	Marg 8 2071	2060
		216	January 1 2015	1381	Poush 9 2071	2061
CVFS Year 19	37	217	February 2 2015	1382	Magh 10 2071	2062
		218	March 3 2015	1383	Fagun 11 2071	2063
		219	April 4 2015	1384	Chaitra 12 2071	2064
		220	May 5 2015	1385	Baisakh 1 2072	2065
		221	June 6 2015	1386	Jestha 2 2072	2066
		222	July 7 2015	1387	Asar 3 2072	2067
	38	223	August 8 2015	1388	Saun 4 2072	2068
		224	September 9 2015	1389	Bhadra 5 2072	2069
		225	October 10 2015	1390	Asoj 6 2072	2070
		226	November 11 2015	1391	Kartik 7 2072	2071
		227	December 12 2015	1392	Marg 8 2072	2072
		228	January 1 2016	1393	Poush 9 2072	2073

Some issues faced during the CVFS transition to CAPI were situations inherent to Blaise 4.8 and being ANSI (American National Standards Institute) based. An ANSI character set is an extension of the ASCII character set-encoding standard and includes an additional 128-character codes². The problem with ANSI is its inability to handle all languages. In recent versions of Blaise, significant

Figure 7.3. Blaise 4.8 Unicode Source in the Control Center

```

name      (name)
Eng      "What is this household members name?"
Nep      "आफ्नै घरका सदस्यको नाम के होला ?"
         / "Name" "नाम" : Type_NameStr50, NODK, NORF

gender    (Gender)
Eng      "What is ^name's gender?"
Nep      "आफ्नै घरका सदस्यको लिंग के होला ?"
         / "Gender" "लिंग" : Type_Gender2, NODK, NORF

age       (Age)
Eng      "And how old is ^name?"
Nep      "आफ्नै घरका सदस्यको उमेर के होला ?"
         / "Age" "उमेर" : Type_Age, NODK, NORF

marital   (marital)
Eng      "What is ^name's marital status?"
Nep      "आफ्नै घरका सदस्यको विवाह के होला ?"
         / "Marital" "विवाह" : Type_Marital2
    
```

Figure 7.4. Blaise 4.8 Unicode Source in a Unicode Aware Application

```

name      (name)
Eng      "What is this household members name?"
Nep      "आफ्नै घरका सदस्यको नाम के होला ?"
         / "Name" "नाम" : Type_NameStr50, NODK, NORF

gender    (Gender)
Eng      "What is ^name's gender?"
Nep      "आफ्नै घरका सदस्यको लिंग के होला ?"
         / "Gender" "लिंग" : Type_Gender2, NODK, NORF

age       (Age)
Eng      "And how old is ^name?"
Nep      "आफ्नै घरका सदस्यको उमेर के होला ?"
         / "Age" "उमेर" : Type_Age, NODK, NORF

marital   (marital)
Eng      "What is ^name's marital status?"
Nep      "आफ्नै घरका सदस्यको विवाह के होला ?"
         / "Marital" "विवाह" : Type_Marital2
    
```

Another factor that can alter the presentation of Unicode text can be any application that interfaces with the text, resulting in changing the underlying encoding and corrupting it. In addition, when authoring instruments with Unicode text, one must consider that textual fills in the instrument take up as much as three times more space than what would be required for traditional ANSI based character sets. This means that the length of defined fills within an instrument need to be increased and tested to ensure the fills are not truncated and the display is not corrupted.

Though the transition of the CVFS Household Registry data collection from PAPI to CAPI has been successful, and we were able to take advantage of the fact that ISER-N in general already followed the standard protocols of SRC data collection – terminology around the survey task itself was familiar to the ISER-N team --, the team has nevertheless encountered some obstacles beyond the expected challenges of implementing a new technology with a staff very experienced with the previous method. Here is one example. As described in detail above, the questionnaire uses parallel blocks to create a flexible structure, with individual questionnaire supplements dependent on data entered into the household roster tab. Each individual member of a household is assigned a unique identification number, and all information about each individual needs to tie back to both the household and the individual. There are times when a household change has occurred that should launch a questionnaire supplement, but the individual needed to answer the supplement questions is not available. The supplement then has missing data, and the protocol on PAPI was to attempt to collect this missing

data during the next data collection. This is much more difficult in a CAPI system, where all contingencies need to be programmed and allowed. The ISER-N team ended up creating new paper-based work-arounds for these unusual situations, and we are thinking through how to better accommodate them within the CAPI system.

Another example of how the CAPI instrument's design resulted in adding unintended burden was not realized until we began working through the post data collection processing phase. This issue surrounds the Family Planning block, where a very small subset of respondents are required to answer this module. However, the instrument was designed to have the ability to capture this data for all 30 respondents, throughout the 48 months, and is captured as a set of 5 enumerated responses (or an "enter all that apply" question). This means there are more than 7,000 variables in the Family Planning section alone and takes up 27% of the total amount of variables within the second version of the CVFS's Blaise HHR application. This adds to a considerable amount of burden on the post data collection cleaning process and is one of the reasons it is being removed from the forthcoming Version 3 of the CVFS Blaise instrument.

In conclusion, the CVFS Household Registry's PAPI instruments conversion to a Computer-Assisted Interview (CAI) system was a success. We continue to look for areas of future enhancements and hope to transition the CAPI instrument to Blaise 5 in the future.

3. References:

¹ <http://spe.psc.isr.umich.edu/research/cvfs.html>

² <http://net-informations.com/q/faq/encoding.html>

Blaise 4.8 Online Assistant (Version 4.8.4.1915). (2016).

Blaise Instrument Extensions with Alien Routers and Manipula

Lilia Filippenko, Chris Carson, Orin Day, and Mai Nguyen, RTI International, United States

1. Abstract

RTI International has conducted many studies involving field data collection on laptop computers. Most of these Computer Assisted Personal Interviews (CAPI) are conducted in respondent's homes and then data is transmitted to RTI. While collecting data for researchers, sensitive personal information is often provided by respondents. This data should be protected on the laptops and during transmission.

There are different ways to protect collected information. This paper describes an easy way to do that by using Blaise instrument with alien routers and procedures in Manipula setup to:

- Enter sensitive data outside of Blaise interview;
- Call external application to electronically obtain proof of respondent consent or other sensitive information;
- Download GPS data from a GPS-tracking device at the end of the interview.

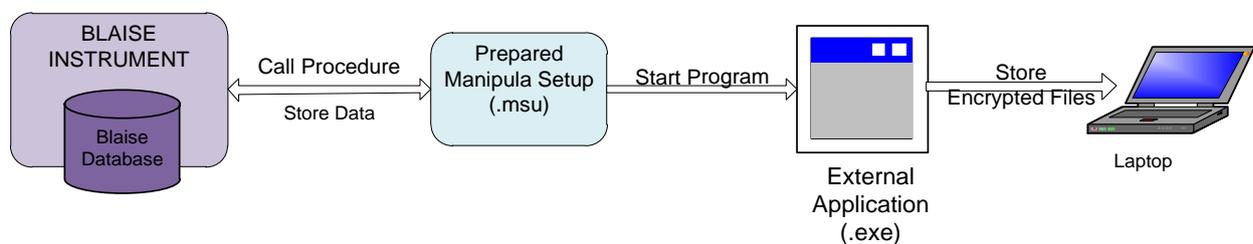
Examples of code in Blaise instrument and Manipula setup are provided in this paper along with description of the external applications used during Blaise interviews.

2. Data Protection on Field Interviewer laptops

The Check Point Full Disk Encryption Software is installed on all RTI field interviewer laptops. So all data, operating systems and temporary files are automatically encrypted by default without relying on the user. Full Disk Encryption performs the encryption transparently to the user, who never needs to bother about what to encrypt and when.

In addition to whole hard drive encryption, files that are ready to be transmitted to RTI are encrypted by the Case Management System. When encrypted zip files with collected data are received at RTI and decrypted, we still want to have some sensitive information encrypted at rest and ready for review only when they are needed. To achieve this goal, such files are encrypted at the very moment when they are created. To satisfy clients' requirements, FIPS 140-2 compliant encryption method is used by the external application called during the interview. Figure 1 shows the process of creating special files during the interview using Manipula setup.

Figure 1. Encryption of Sensitive Data during Interview



We've successfully used all available features of Blaise to call external programs and collect data outside of Blaise. Previously, we used alien routers created as DLL in .NET. However this approach had a drawback during the development stage for clients' testing because alien routers were required to be registered on the client's machines and that was not always permitted.

When a new feature was added to Manipula to use a parameter to set variable meta names at run-time, we found that implementation was very easy and preparation of installation packages for testing and deployment on field interviewer laptops were much simpler. The following section has a detailed explanation of what needs to be done in a Blaise datamodel and a Manipula setup to achieve this.

3. Collecting Data Outside of Blaise Interview

One example of sensitive data requiring immediate encryption after entry is a social security number (SSN). Collecting and storing the SSN separately from the survey data ensures respondents that their data will be available only for limited research and that it will be specially protected before it is delivered to researchers. A description of how this is done is shown below.

3.1 Implementation in Blaise Instrument

In the Blaise datamodel, fields where data will be stored in Blaise database are defined along with an alien router that redirects execution of a procedure in a prepared Manipula setup:

- block BSSN4 with field SSN4;
- router DataEntry (keyword BLOCK is used to apply the same router for another block; name of the meta file is passed as parameter);
- field SSN4 as BSSN4.

Figure 2. Define Fields and Alien Router

```

BLOCK BSSN4 {Used to ented SSN outside Blaise }
  FIELDS
    SSN4 ""/"The last 4 digits of Social Security number": STRING[4]
  ENDBLOCK

ROUTER BLOCK.DataEntry
  ALIEN('AHPSRouter.msu /KMyMeta=$dictionaryname', 'DataEntry')

FIELDS
  SSN4
  ENUS
  "We are committed to protecting the confidentiality of all the information you
  give us. What are the last four digits of your Social Security Number?"
  : BSSN4

```

In the RULES section of the Blaise datamodel, a defined router DataEntry is used instead of ASK method to handle the field in the block SSN4. Also, when data is saved outside of Blaise, we need to have information about how the respondent answered the question to monitor collection of SSN data and ask that question only once if data was collected and encrypted. That is achieved by checking the value of SSN4 - "0000" which means that SSN was collected.

Figure 3. Define a Routing Method

```

RULES
SSN4.keep
IF SSN4.SSN4 <> '0000' THEN
  {call Router DataEntry}
  SSN4.DataEntry
CHECK
  SSN4.SSN4 = NONRESPONSE OR SSN4.SSN4 = '0000'
  "PLEASE RE-ENTER THE LAST 4 DIGITS OF SOCIAL SECURITY NUMBER."
ENDIF
SSN4.keep

```

During the interview an audit trail file with all answers is created to help troubleshoot some problems and to monitor flow of the interview. The audit trail file is an ASCII file that can be opened in any text editor. By collecting the SSN outside of Blaise, the audit trail file is bypassed and security is enhanced by keeping this sensitive information out of the audit trail file.

3.2 Implementation in Manipula Setup

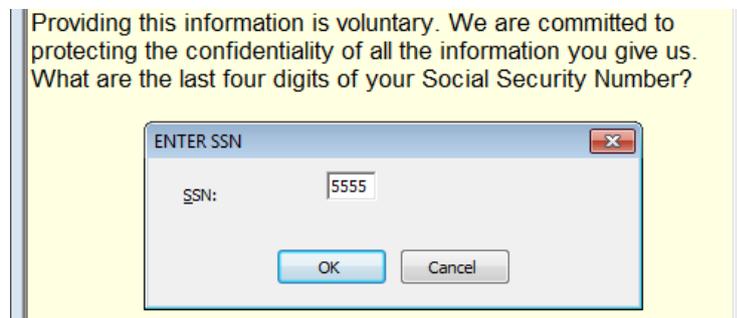
When the Manipula setup AHPSRouter is called during the interview, all information in the current interview is available through “INTERCHANGE =SHARED” setting at the beginning of the Manipula setup. The reserved word “(VAR)” is used instead of the meta identifier so that Manipula setup knows that the prepared datamodel that corresponds with the meta identifier is only known at run-time.

Figure 4. Define Meta and Settings in Manipula Setup

```
PROCESS AHPSRouter
SETTINGS
  DATEFORMAT = MMDDYY
  DATESEPARATOR = '/'
USES
  mymeta (VAR)
TEMPORARYFILE mydata:mymeta
SETTINGS
  INTERCHANGE=SHARED
```

A procedure DataEntry and a dialog to collect last four digits of the SSN are defined in the Manipula setup AHPSRouter as shown in Figure 5.

Figure 5. Entering Data in Manipula Setup



When the OK button is pressed and the SSN is validated to contain only digits, an external application developed for a study is called to encrypt it with a specified encryption key.

Figure 6. Abbreviated Version of the Procedure to Collect SSN in Manipula Setup

```
PROCEDURE DataEntry
AUXFIELDS
    CaseID : STRING[8]
INSTRUCTIONS
    IF (ROUTERSTATUS=BLRSPREEDIT) THEN
        FieldName := uppercase(ACTIVEFIELD)
        CaseID := mydata.GETVALUE('main_case.zrid')
        {Show Dialog in DEP}
        SSNDialog4
        {... more code to validate data entered ...}
    IF BUTTONs = Okay THEN
        {Call external program and pass case ID and SSN for encryption}
        Reslt:= RUN('AHPStools.exe Crypto' + ' ' + CaseID + ' ' + UserSSN4)
        IF Reslt = 0 THEN
            {File with encrypted data is created - save in Blaise with zeros}
            IF (FieldName = 'SSN9.SSN9') THEN
                mydata.PUTVALUE('SSN9.SSN9', '000000000')
            ELSEIF (FieldName = 'SSN4.SSN4') THEN
                mydata.PUTVALUE('SSN4.SSN4', '0000')
            ENDIF
            {Move to a next field in the interview}
            SETALIENROUTERACTION (BLRANEXTQUESTION)
        ELSE
            {... more code ...}
        ENDIF
    ELSE {return to the same field }
        SETALIENROUTERACTION (BLRAEDITQUESTION)
    ENDIF
ENDIF
ENDPROCEDURE
```

The application saves the encrypted value in a text file with a name that has the primary key, Case ID, as part of it. This will allow RTI to decrypt data from the file later, when it is ready for researchers to use. The file is saved on a laptop in a folder from which it will be transmitted along with other collected data for the case.

4. Obtaining Respondent Signatures with RTI's DocMan

Another example of collecting sensitive information is obtaining respondent signatures on different forms. In an effort to protect the confidentiality of survey participants and to ensure the security of all data that field interviewers use and collect while working cases, RTI has created an electronic document management system named DocMan.

4.1 DocMan Description

DocMan is the collective name for a set of systems developed by RTI to eliminate paper documents from field data collection. DocMan provides field interviewers a facility for electronic collection of signed forms or documents, via legal electronic signatures using a USB signature pad or through scanning paper forms using a USB portable scanner. Such forms could include informed consent, incentive receipts, and so on. Once collected and verified by the interviewer, DocMan encrypts the forms and provides secure transfer of those forms back to the RTI home office. DocMan also provides secure transfer of read-only documents, such as locating information, as well as the capability to key or scan additional information regarding the case. DocMan was first used in 2007 and has been used on many RTI field studies.

Notable components of DocMan include:

- DocMan Executable – .NET application,
- DocMan Information Files – set of PDF files,
- DocMan Forms – set of signed or scanned documents,
- DocMan Case Notes – notes in electronic format used to replace paper case folders.

DocMan Executable resides on the field laptop that the interviewer uses as an interface to access three primary features:

- 1) To view DocMan Information Files,
- 2) To collect signed forms as Forms, and
- 3) To record DocMan Case Notes.

DocMan can be deployed with any or all of the features above. The effectiveness of DocMan was enhanced by enabling it to be called directly from Blaise at the correct point in the interview. An example of how DocMan is used with Blaise is provided in Section 4.2.

DocMan Information Files – PDF files with contents and format specified by the project that are included in case preloads to provide case information to the field interviewer beyond what is available in the Case Management System. PDF files are encrypted, password secured, and can include security features to make them unprintable.

DocMan Forms – described above, these are signed or scanned documents digitally acquired and encrypted, signed via electronic signature pad (ePad) or scanned with a portable scanner (equipment examples below).

Figure 7. Equipment used with RTI DocMan: Interlink ePad (left) and Brother DS-620 Mobile Color Page Scanner (right)

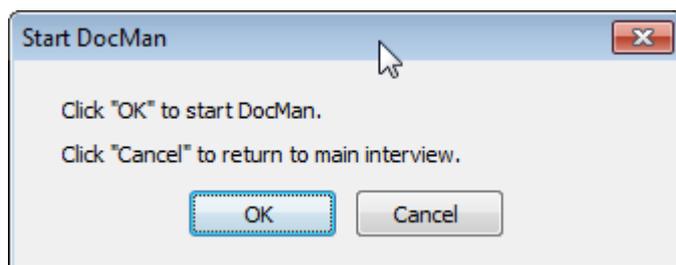


DocMan Case Notes – The option to collect interviewer notes on the laptop and then securely transmit those notes and event/status/comment information to RTI. Once received the information is decrypted and periodically collated and compiled into secure PDF documents. Those documents are released to RTI staff and transmitted to field supervisors. Implementing this system involves configuring the laptop application and Case Management System, and customizing the format of the Case Notes file to match project needs. For longitudinal studies the Case Notes from past rounds of data collection are included as DocMan Information Files, securely replacing paper case folders.

4.2 Blaise and DocMan

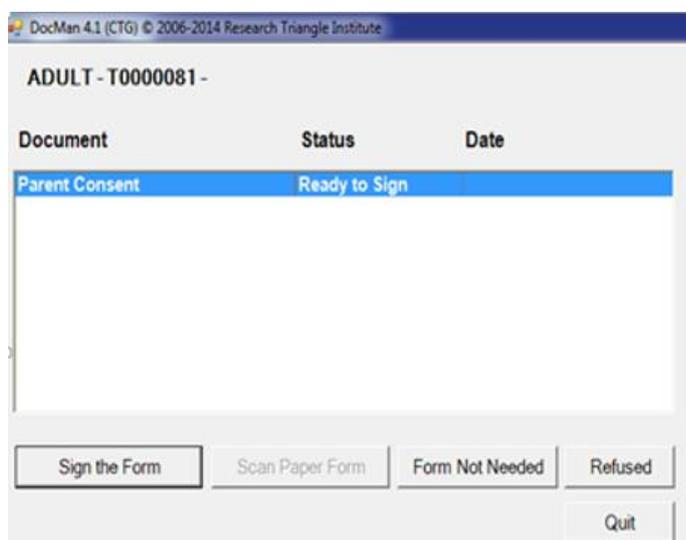
DocMan is used during the Blaise interview to electronically obtain proof of respondent consent to the interview or other information. The Blaise implementation of a call to DocMan is almost the same as described above in section 3. When an interviewer reaches a field where DocMan is called, a dialog is defined in a Manipula setup to confirm that the field interviewer has everything ready to start collection of a signature or to scan a document. While our example below includes the process for signing a document with the ePad, the process flow for scanning a document is similar.

Figure 8. Call to DocMan from the Blaise Interview



Clicking “OK” will automatically launch DocMan and display a list of usable forms for this portion of the interview. DocMan will automatically present the field interviewer with the appropriate options. Figure 9 shows the window displayed in DocMan during the informed consent portion of an interview. The available document has a status of “Ready to Sign” meaning the document had not yet been completed.

Figure 9. DocMan Selection Screen and Document Status



Clicking “Sign the Form” will open the DocMan template selection screen where the field interviewer will be asked to select from prepared templates that can be used in the current situation. The Word document is then ready for the respondent’s electronic signature. The field interviewer will ask the respondent to review the form on the computer, and will then hand the electronic signature pad (ePad) to the respondent for her/his signature.

After all required signatures have been collected and the field interviewer confirms that form was completed correctly, DocMan recognizes that the document has been closed and returns back to “DocMan Selection Screen and Document Status”. The status of the document is then changed from “Ready to Sign” to “Complete – e-Pad”. At this point the field interviewer will press “Quit” to return to the Blaise interview.

Control is then passed to the Manipula setup where the completion status of the signed form is determined by checking existence of the encrypted file created by DocMan in a specified folder. Fields defined in the datamodel to save the status of the document (completed or not) and some timing data to monitor field interviewer performance are calculated in a Manipula setup. An example is shown in Figure 10.

Figure 10. Run DocMan from Manipula Setup

```
{Create name of encrypted file - PDF or Word document}
IF (DocID = 'S1') OR (DocID = 'S2') THEN
    strFileName := strPathName + DocID + '\' + CaseID + '_' + DocID + 'E.pdf'
ELSE
    strFileName := strPathName + DocID + '\' + CaseID + '_' + DocID + '.docx'
ENDIF
{Start DocMan}
strRun := '"c:\Program Files\DocMan\DocMan.exe"' + ' /CASEID=' + CaseID
        + ' /' + strTrProd + ' /LAUNCHDOC=' + DocID + ' /INSTLANG=B'
Result:= RUN(strRun, wait)
IF (Result <> 0) AND (Result <> EMPTY) THEN
    DISPLAY('Problems with starting DocMan. Please report to FS.', WAIT)
ELSEIF Result = 0 THEN
    {Check that file was created and set completion status}
    IF FILEEXISTS(strFileName) THEN
        Status := '491'
    ELSE
        Status := '309'
    ENDIF
    mydata.PUTVALUE(BlockName + '.Status', Status)
    {... more code to save data in Blaise ...}
    {Move to a next field in the interview}
    SETALIENROUTERACTION(BLRANEXTQUESTION)
ENDIF
```

Protecting subjects' confidentiality and maximizing data security are increasingly critical objectives of field research. DocMan eliminates the need to collect and return paper documents from the field while sufficiently collecting electronic documents during an interview by using an alien router in Blaise interview with a Manipula setup. This approach allows the field interviewers to do their job more efficiently.

5. Download GPS Data on Laptop

Global Positioning System (GPS) is used by many people every day. One of our recent study clients requires the use of a GPS unit to capture geo-coordinates of the place where interview is being conducted. Collected data is supposed to be entered into a special Blaise instrument. We decided that having geo-coordinates will also provide us with an additional way to verify the field interviewers' work. Following sections describe how collection of geo-coordinates is implemented in the field and two ways to use collected data for verification.

5.1 Field Implementation

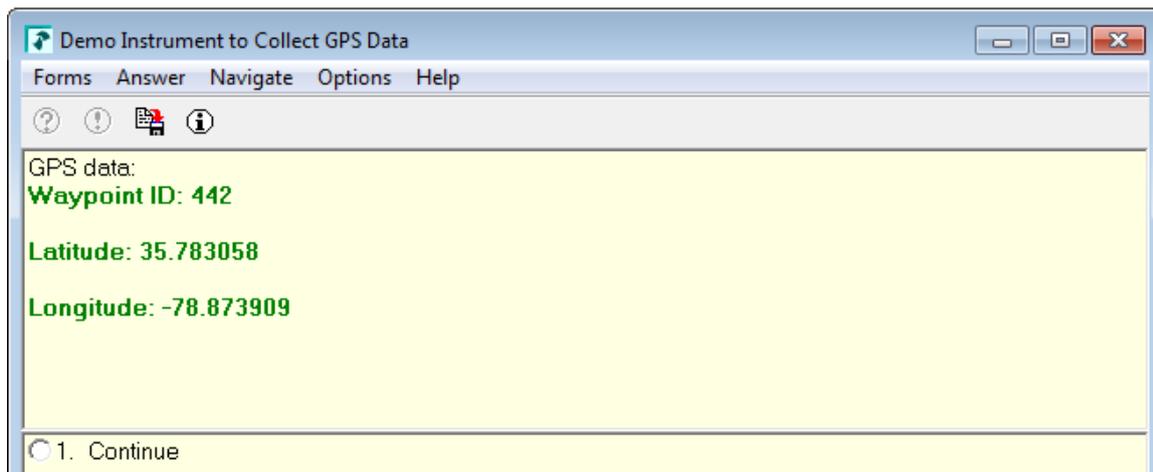
We selected the GlobalSat DG-100, a GPS data logger that records track data, because of its simplicity and low cost (approximately \$22 each). The DG-100 records time, date, speed, altitude and GPS location at preset intervals. It comes with a USB plug to download readings to any USB capable laptop or desktop and with Windows based software utility. There is no LED display. Field interviewers are simply instructed to turn it on, wait for a few minutes until the green light flashes, and then press the silver button.

Figure 11. GlobalSat DG -100



After the coordinates have been captured, the field interviewers launch another interview called the Field Interviewer Observations Module where they are asked to plug the DG-100 into their laptop. After it is confirmed that the device is plugged in, the Manipula setup is called to execute an external application “DataLogger” to download data from the GPS device and pass geo-coordinates to the Blaise database. The result of this process is shown in Figure 12. Implementation in the Blaise interview and the Manipula setup are the same as described above in 3.1 and 3.2.

Figure 12. Data downloaded from DG-100 in Blaise Database



5.2 Data Logger Application

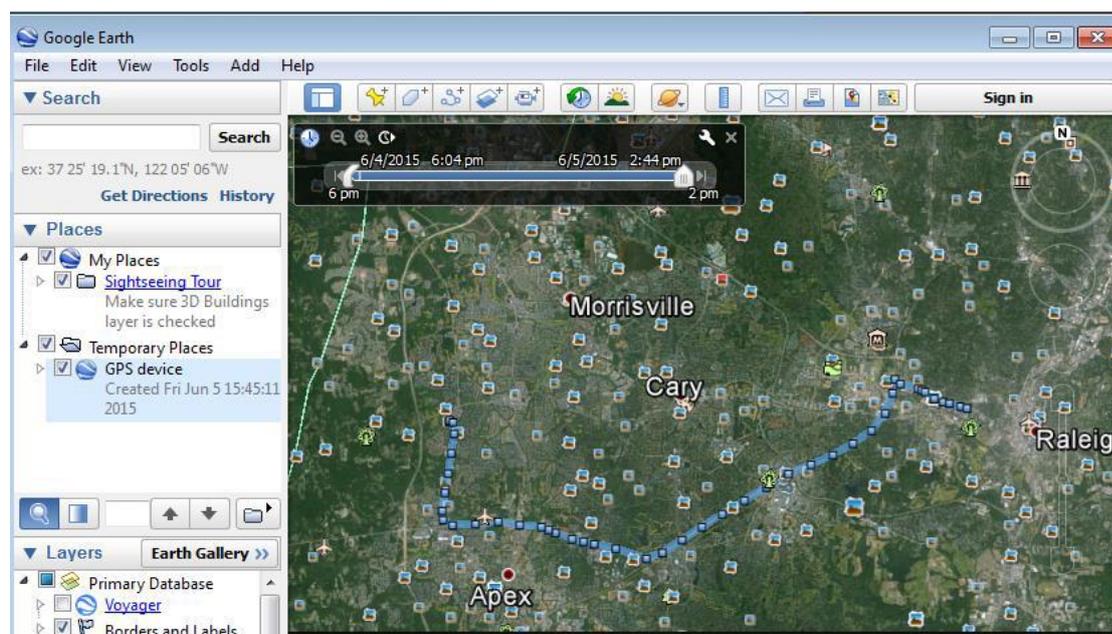
For the field interviewers, the process of downloading GPS readings looks seamless. But Data Logger application developed in .NET does a lot of work:

- it identifies the COM port to which DG-100 is connected on laptop,
- it checks that DG-100 is ready to upload data,
- it executes application gpsbabel.exe to save data on laptop in two files - <CaseId>.csv and <CaseId>.kml.
- it reads a .csv file to find geo-coordinates with timestamp as close as possible to the CAPI interview timestamp,
- it creates a temporary file with data that is used by Manipula setup to save geo-coordinates in the Blaise database.

Although DG-100 comes with its own utility to download data, it can only work in interactive-mode. So we decided to use free software called GPSTabel (www.gpsbabel.org) in a batch mode.

Both files saved on laptop are also transmitted to RTI along with the data from the Field Interviewer Observations Module. Excel can be used to review the .csv file and the .kml file can be viewed in Google Earth as seen in Figure 13 below. In this .kml file, the DG-100 was left running and the whole route captured.

Figure 13. kml File viewed in Google Earth



5.3 Field Interview Verification

GPS capture is one of the most recent advancements in field interview verification. Verifying that the Field Interviewer was present at the respondent's address at the date and time the interview took place is a very good indication that the interviewer actually conducted the interview with the respondent. However, it is not an absolute certainty. Other complimentary methods of interview verification should also be used. These include an analysis of timing data and listening to recorded segments of the interview (CARI).

5.3.1 SAS® Proc Geocode

The street-level geocoding to an address is done using SAS® Proc Geocode. This procedure can be used to look up physical street addresses and return their corresponding latitude and longitude coordinates. These coordinates can then be compared to the coordinates captured on the DG-100 and some simple calculations can then be employed to compute the distance between the two points. Reports in SAS are created every night for the project staff to review. Ideally, these comparisons will show a very small distance between the captured coordinates and the physical address. When this does not happen, the case can be prioritized for further scrutiny through the other methods used for interview verification.

There are several good examples on the SAS website how to use SAS® Proc Geocode. The default database SAS/GRAPH uses for street level geocoding is SASHELP.GEOEXP and includes addresses from Wake County in North Carolina, where the SAS headquarters is located. To get the data needed

to geocode throughout the United States, SAS provides a free download from its SAS Maps Online website:

<http://support.sas.com/documentation/cdl/en/graphref/65389/HTML/default/viewer.htm#n02y3yabtlqatsn16gp2fo51yo7p.htm#n1nmfmy3wf1kc8n0zw5651b41mjt>

5.3.2 Google Maps API / Google Geolocation API

As an alternative to SAS® Proc Geocode the Google Maps API can be used. The Google Maps API provides multiple comprehensive services, including Mapping, Directions, Places and Geolocation. The APIs support desktop and mobile devices, and in all popular programming languages: Java, C#, JavaScript, Python, etc. Our code samples will be focused on the Geocoding service using Python. An excellent introduction to the Python language can be found on this website:

<https://www.python.org/about/gettingstarted/>

Figure 14. Google Geolocation API example

```
from pygeocoder import Geocoder

address = '1600 Pennsylvania Ave., Wanshington, DC'

# Geocoding
result = Geocoder.geocode(address)

# output results
print(result.formatted_address)
print(result.coordinates)
```

```
1600 Pennsylvania Ave SE, Washington, DC 20003, USA
(38.8791981, -76.9818437)
```

The Google Geolocation API also includes reverse geocoding (address lookup) functionality that can be used for looking up address from a geo-coordinate. The Google Geolocation API is straightforward to use. A unique functionality supported by the Google API that is not available in the SAS procedure is the ability to do reverse geocoding. Further documentation about Google Maps API is available at this website:

<https://developers.google.com/maps/web-services/overview>

6. Conclusion

External applications have become a common requirement on many studies and Blaise has more than one way to implement them. However, we find that the use of an alien router and a Manipula setup that redirects execution is the simplest and most efficient solution: it reduces development time and is easier to debug. RTI has used this approach extensively for projects that need new innovations to accomplish their goals and to accommodate increased levels of data security. We hope that this option continues to be available in Blaise 5 for future studies.

Statistical Surveys in Households and by Individuals in Slovakia – the use of Blaise 4 / Blaise 5

Ludmila Hadová and Danica Tureková, Statistical Office of the Slovak Republic

1. Abstract

Blaise has been a data collection and processing platform for social statistics, demographics and consumer surveys in Slovakia. In the year 2016 this includes such as The Labour Force Survey (LFS), The Survey on Information and Communication Technologies in Households and by Individuals (ICT), The Household Budget Survey (HBS), The Consumer Survey (CS), Statistics on Income and Living Conditions of Households (EU SILC) and more. With respect to questionnaire content, both - wording and methodology comply with European regulations. Data obtained in surveys are sent to Eurostat, while they are also used in national statistics.

The interview process realized by PAPI / CAPI methods by tablets (without possibility of on-line web access) - requires a special management and survey progress monitoring and places heavy demands on the software. The program has to be user friendly and conform for using Windows tablets and CAPI method for face-to-face interview. Among the many objectives, the main, and most significant, is the quality of the data. That is why many multiple checks are already incorporated at an early stage of data collection. The data have to be verified and subjected to various controls and comprehensive protocols (Manipula programs). Checks of some surveys are extended to include data obtained in the previous period for comparison with the new values.

This paper discusses the topics mentioned above. The concluding part provides some practical exhibitions of the Blaise 4 solution and the Blaise 5 development version of our the most difficult survey SILC, including selected identical illustrations (comparisons).

2. Introduction

Statistical Office of the Slovak Republic has used Blaise survey solutions since about 1992. The personal Blaise experiences of the authors of this paper have started at the end of 2013 together with organizational changes.

Initially household surveys were carried out in the Slovak conditions at 2 levels: centralized and decentralized. Centralized level represented central Statistical Office of the Slovak Republic, decentralized level comprised of 8 regional offices. Nowadays Statistical Office of the Slovak Republic is still the main coordinator and data processor of data for EU surveys. At decentralized level some changes happened in terms of responsibilities, so one of the regional offices is now coordinator of data collection, data recording and creation of regional databases within surveys. Within statistical organization structure it became part of the central Statistical Office now as *Section of Industrial Data Collection and Processing and Field Surveys in Banská Bystrica* (hereinafter referred to as “Section of Field Surveys”) which plays substantial role for realization of household surveys.

The authors of this paper are members of the department team for data collection supervision which includes activities such as a software development, training of interviewers, control activities and sending output data ASCII files to the central office.

3. Overview of Blaise Survey Implementations

Blaise data collection and processing program for each survey has to be created in two versions – for interviewer and administrator.

3.1 Interviewer Program Package

The basic program version for interviewer allows working exclusively with a household sample individually assigned to him/her. The structure of the survey sample, its range and the update possibility depends on the kind of survey. Using a simple start-up Visual Basic Script (VB script) the interviewer can record a new questionnaire during face-to-face interview or data from the paper form in recording phase as well as update them. Each interviewer has his/her own unique code which is required in each questionnaire record and update and is stored into the database together with a current date and time systemically (used USERNAME, SYSDATE, SYSTIME). These elements are very significant for our management control process.

Blaise questionnaire is complemented by several control and balance protocols as well as export function (via Manipula). Data export has to be executed on a monthly basis (or how often required) – at the interviewer local device. Subsequently, it must be transferred to the central section network server for data processing.

Table 1. Overview of Selected Blaise 4 Survey Solutions

Survey / Questionnaire	Number of Paper Questionnaire Sections	Maximum number of Interviewed Persons	Number of Blaise 4.8 Solution Screens	Number of surveyed Fields			Number of AUXFIELDS	
				Total	per Household	per Person (12 Persons)		
Monitoring of EHIS	-	-	1	14	14	-	4	
Monitoring of HBS	-	-	1	19	19	-	3	
Consumer Survey (CS)	1	-	4	45	45	-	6	
Household Budget Survey (HBS) - the small variant	1	12	11	675	39	53	416	
Household Budget Survey (HBS) - the full variant	3	12	Parts A, B	45	1 408	772	53	10
			C - Daybook	57	6 270	6 270	-	6
Survey on Information and Communication Technologies in Households and by Individuals (ICT)	1	-	28	80	80	-	8	
Statistics on Income and Living Conditions of Households + Grant Project (SILC GP)	3	12	175	5 413	133	440	136	
Statistics on Income and Living Conditions of Households (SILC)	3	12	173	5 280	132	429	136	

Table 2. Number of Used Classifications and Manipula Files

Survey / Questionnaire	Number of	
	used Classifications	Data Checking and Monitoring Manipula Files
Monitoring of EHIS	2	5
Monitoring of HBS	1	9
Consumer Survey (CS)	4	11
Household Budget Survey (HBS) - the small variant	4	9
Household Budget Survey (HBS) - the full variant	5	8
	8	10
Survey on Information and Communication Technologies in Households and by Individuals (ICT)	3	9
Statistics on Income and Living Conditions of Households + Grant Project (SILC GP)	9	6
Statistics on Income and Living Conditions of Households (SILC)	9	6

3.2 Administrator Program Package

Blaise interviewers data export files already saved at the central section network server have to be put together into one complete file containing data from all regions within the Slovak Republic. Data errors, abnormalities and remarks of particular files have to be preserved for subsequent checks at the central section level.

The administrator program package contains the administrator start-up VB script of all basic functions extended to control arrangements, summarization and evaluation protocols (rich using of Manipula). Afterwards, administrator can create final Blaise and ASCII export data files.

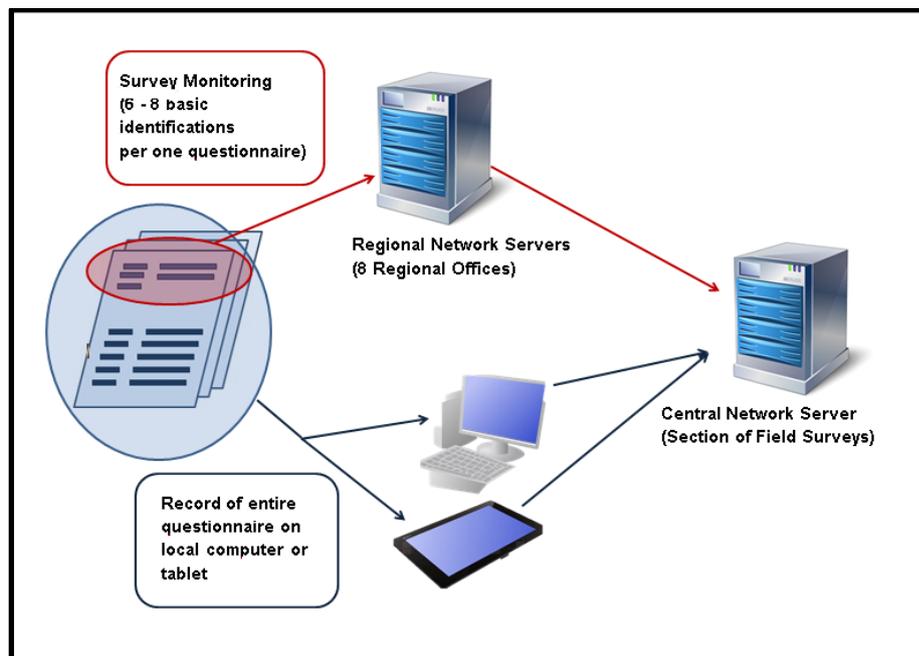
4. Data Collection Progress Monitoring

Most of household surveys use an address based sampling. Because of using only PAPI and CAPI data collection methods following questions are raised:

- How is data collection going on?
- Where, why and whether has the process been interrupted or delayed?
- How successful is interviewers work?
- What is a development of sample compliance in terms of multiple criteria?

During the off-line data collection period we are interested in the real structure of the sample and the interviewer's way towards data, too. Interviewers report about their work by inputting several basic data into the local regional monitoring system, thereafter they make data export to the central section network server (Figure 1).

Figure 1. Survey Progress Monitoring Scheme



4.1 Program and Reports

Figure 2 illustrates a screen of the monitoring program with inputted basic information about visiting of the specific household. We can see how many visits (1) were realized by interviewer (of code 133), which interview method was used (PAPI for part A of the questionnaire, CAPI for part B), how many members the household has (3) and more.

Figure 2. Blaise 4 HBS Monitoring Program - Input of Interview Basic Data

EVIDENCIA - MONITORING - Rodinné účty 2016 Opýtavateľ: 133 - Alex Clever Kód domácnosti: 11001201

Spôsob získania dotazníka RÚ/B

Záznam dotazníka RÚ/B na papierový dotazník alebo na tablet

Konečný výsledok zisťovania:

1. Papierový dotazník
 2. Záznam na tablet

ID_DOM: 11001201 Pocat_nav: 1 Pocat: 3
 IKODOP: 133 Vysledok: 11 DSp: Pohlavie: 2 Žena
 KRAJ: 1 Dat_nav: 5.12.2014
 OKRES: 108 PAPIA: 1 A_PAPI: Dat_nar: Vek: 62
 NazObec: Blatné PAPIB: 8 B_CAPI: Poznamka: číslo domu nie je viditeľné

By using the monitoring program we can control multiplicity of households, age structure of respondents and even more before they will be reached in the real survey data file.

As can be seen in Figure 3, there is something wrong in records of the interviewer of code 135:

- A respondent of the first household (of ID 11000101) is younger than allowed (<18).
- Recording of the second questionnaire (of ID 11000201) was probably interrupted due to EMPTY variables (NE).
- Logical totals do not match.

Only the third record of ID 11000301 is completed and OK (a respondent is a man of three-person household and between 55 and 64 year of age). Seven days later the interviewer of code 135 has inputted seventeen new questionnaire records. A data error is still remaining in his/her control protocol (a little star in Figure 4). We expect a comment or explanation of why a control was left without correction or confirmation.

Figure 3. Blaise 4 HBS Monitoring Program - Output Report Snippet No. 1

```

Bilancia výberu vyšetrených domácností podľa opyt.-podrobne: 3.3.2016 15:20[B]
      (za celý dostupný súbor evidencie RÚ 2016)
-----
M: muži  Ž: ženy          v1: 18-24 r.  v5: 55-64 r.
NE: prázdny údaj        v2: 25-34 r.  v6: 65-74 r.
<18: zadaný nížky vek  v3: 35-44 r.  v7: 75 a viac r.
x?: zadaný vek bez zadania pohlavia
v4: 45-54 r.
-----
kr  kop  č.d.      Členná domácnosť  | M  Ž  | veková skupina
-----
      1  2  3  4  5+  |  |  | v1  v2  v3  v4  v5  v6  v7
-----
L 133 11001201  |  |  x  |  |  |  |  |  |  |  |  |
Spolu opyt. 133: | 6  8  2  4  3  | M : 8  | 1  3  0  2  0  1  1
      spolu dom.: | 23  |  Ž : 15 | 2  4  3  3  1  2  0
-----
L 135 11000101  |  |  |  |  x  |  x  |  |  |  |  |  |  |  |
L 135 11000201  | NE  |  |  |  |  |  |  |  |  |  |  |  |  |
L 135 11000301  |  |  x  |  |  |  |  |  |  |  |  |  |  |  |
Spolu opyt. 135: | 0  0  1  0  1  | M : 2  | 0  0  0  0  1  0  0
      spolu dom.: | 3  |  Ž : 0  | 0  0  0  0  0  0  0
    
```

As shown in Figure 4, there are 6 situations to express an outcome of the visit. The result of the visit can be:

- A cooperating household (11) – only this code is desired
- Refusal to cooperate (21)
- Nobody at home (22)
- Nobody able to answer (23)
- Uninhabited apartment or house (24)
- Another reason (25).

In general, there can be made three attempts to visit a household. The report snippet shows how successful interviewers are from both personal and regional view (the example of region 1). This kind of report is especially important for us to issue guidelines for further action.

Figure 4. Blaise 4 HBS Monitoring Program - Output Report Snippet No. 2

Počet dotazníkov v evidencii RÚ podľa opytovateľov: 10.3.2016 11:15 [Ad]										
(za celý dostupný súbor evidencie RÚ 2016)										
Dsp (11): Spolupracujúca domácnosť					DNezp (23): Domácnosť nespôsobilá odpovedať					
Dodm (21): Odmietnutie spolupráce					DNeob (24): Byt je neobývaný					
DNez (22): Domácnosť nezastihnutá					Inyd (25): Iný dôvod					
P o č e t podľa výsledku návštevy										
Kraj	Kop	Spolu	vyšetrených Dsp (11)	Nevyšetrených (21 - 25)	Dodm (21)	DNez (22)	DNezp (23)	DNeob (24)	Inyd (25)	Meno opytovateľa
1	131	50	23	27	7	20	0	0	0	Peter XXX
1	132	40	23	17	3	14	0	0	0	Anna YYY
1	133	50	23	27	27	0	0	0	0	Alex Clever
1	134	50	24	26	12	11	2	0	1	Dana KKK
1	135	40	20	20	12	8	0	0	0*	Jana LLL
1	136	40	23	17	9	8	0	0	0	Maria MMM
za [1]:		270	136	134						
2	226	13	8	5	4	0	0	1	0	Martin NNN

4.2 Monitoring Log File

Some of Blaise 4 solutions include a supervision of an interviewer work. Such program monitors activities of the interviewer in the background while running the application or associated data control activities. A log file is created (keyword DAYFILE) and results are only available to a project manager (Figure 5).

Figure 5. Blaise 4 Monitoring Log File

```

20160506,13:14:44 *****
20160506,13:14:44 MANIPULA 4.8.4.1880, @ Statistics Netherlands 1989-2014
20160506,13:14:44 Setup: SB16n_exp.man
20160506,13:14:44 Export záznamov SB16 - 1/2 - BLAISE to BLAISE
20160506,13:14:44 Parameter:
--- *n 122 6.5.2016 13:14 5 gordis
20160506,13:14:45 File: INPUTFILE1 (d:\SB16n\SB16n)
20160506,13:14:45 Records read: 9
20160506,13:14:45 File: OUTPUTFILE1 (d:\SB16n\SBExp\SB16n)
20160506,13:14:45 Records written: 9
20160524,11:38:44 *****
20160524,11:38:44 MANIPULA 4.8.4.1880, @ Statistics Netherlands 1989-2014
20160524,11:38:44 Setup: SB16n_del.man
20160524,11:38:44 Prechod na nové obdobie
20160524,11:38:44 Parameter: ANO
20160524,11:38:50 --- *n 122 24.5.2016 11:38 9 gordis
20160524,11:38:50 --- Nasleduje v ý m a z dát za minulé obdobie!
20160524,11:38:50 File: SB (d:\SB16n\SB16n)
20160524,11:38:50 Records read: 9
20160524,11:38:50 Records written: 0
20160524,11:38:50 Records deleted: 9
20160524,11:38:50 File: SBDEL (d:\SB16n\PROTOKOLY\SBdel.rtf)
20160524,11:38:50 Records written: 11

```

Following findings can be identified in the log file:

- whether check protocols were executed by the interviewer or not
- whether he/she actually used CAPI method as declared
- where (physically) the application is located
- and more (in Figure 5 we can also see that the interviewer of code 122 has not worked with the software for eighteen days).

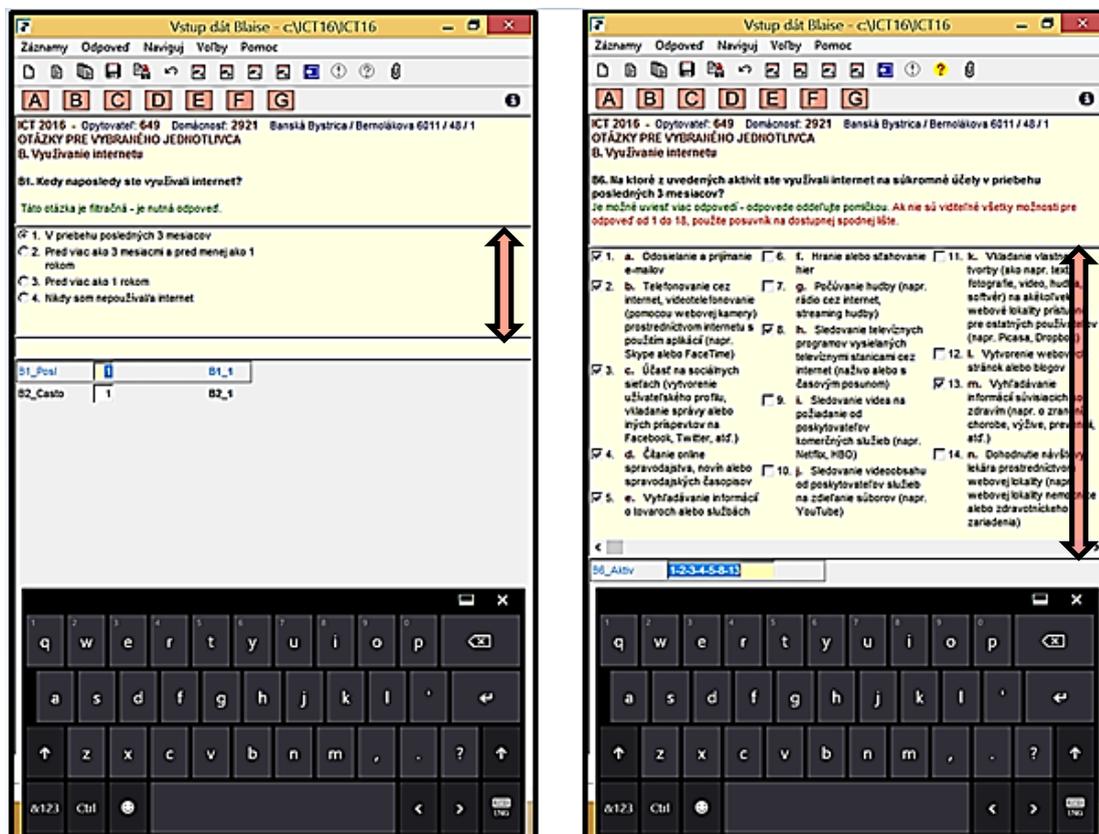
5. Blaise Program Layout Requirements

Our Blaise 4 programs are designed for use on computers and tablets. The interview process places heavy demands on the software which has to be user friendly. We recommend setting of the tablet font to a larger one (the best setting is 150 %) and to switch to portrait mode (“to the height”). Tablet touch screen keyboard may be positioned at the bottom of the screen at a free place if necessary. There is also allowed to use a simple Blaise keypad toolbar to enter numerical data.

Requirements for the Blaise 4 questionnaire layout:

- A floating height / width of an info panel so that a whole extensive question text is visible
- A floating height / width of an info panel so that all or as many as possible response options are visible (Figure 6) and there is no need to use a vertical or horizontal scroll bar, at all
- Fields location in a such way that corresponds with the questionnaire paper form proposals (an order of fields, a grouping of several fields related to a slot on a screen)
- To enable fast moving to another place of a program structure according special requirements (skipping at selected questions or modules; see “A, B, C, ... , G” in Figure 6)
- To enable fast moving into a block of questionnaire questions of other person (depending on the number of household members)
- To enable the answer input by typing an option number using a keyboard and more.

Figure 6. Blaise 4 - Two Questions in Two Different Layouts on Screens with Used Tablet Touch Screen Keyboard



6. Quality of the Data as the Most Important Output Requirement

Depending on a survey complexity our solutions contains numerous error checks that are performed directly at the questionnaire recording. Blaise has a very well invented principle of a field definition in a data model. Already by field definition – errors in recording are eliminated. However, the program needs to ensure compliance of certain relations and links among items. It is important to define conditions for evaluating error checks correctly. We must respect practical experiences which require incorporation of many data checks (Table 3). This ensures a significant increase of final data quality.

The input values are necessarily validated already during the interviewing. If the data value is extremely different from the average one, it can usually be caused by errors in the recording. Another time, an anomaly can occur that need to be assessed and after the verification corresponding check may be suppressed. Many built-in checks are intended to assess interdependencies among questionnaire items. Furthermore, special year-over-year (longitudinal) checks have been incorporated into our Blaise solution.

Table 3. Blaise 4 Solutions – Number of Included Checks

Survey / Questionnaire	Number of Checks							
	Total	of which		of which		of which		
		per Household	per Person (12 Persons)	HARD	SIGNAL	stored by EDITTYPE	not stored by EDITTYPE	
Monitoring of EHIS	8	8	-	2	6	8	-	
Monitoring of HBS	8	8	-	2	6	8	-	
Consumer Survey (CS)	14	14	-	1	13	14	-	
Household Budget Survey (HBS) - the small variant	1 221	33	99	19	1 202	753	468	
Household Budget Survey (HBS) - the full variant	Parts A, B	880	112	64	19	861	597	283
	C - Daybook	10	10	-	5	5	3	7
Survey on Information and Communication Technologies in Households and by Individuals (ICT)	59	59	-	26	33	59	-	
Statistics on Income and Living Conditions of Households + Grant Project (SILC GP)	3 946	130	318	216	3 730	3 058	888	
Statistics on Income and Living Conditions of Households (SILC)	3 906	138	314	216	3 690	3 054	852	

6.1 Classical Kind of Checks

Blaise questionnaire includes designed checks in such range and amount not to slow down (to go smoothly) data entry and data correction, especially if the interview is done by face-to-face. Checks are conditions that have to be satisfied. The check instruction defines a hard error ((keyword CHECK) and the signal instruction is for a soft error definition (keyword SIGNAL). If the values of the fields involved do not satisfy the statement, an error is invoked. Hard errors must be fixed and a clean record does not contain them. Soft errors have to be evaluated and subsequently suppressed or changed. We prefer to use signal checks in our solutions if possible.

Examples of data checks:

- An age of parents should be at least 16 years.
- If a person has a grandson, this grandson should be at least 32 years younger (to prevent contrary coding of relations between two members of the same household).
- If there is not a child in the household, why any household's member receives family allowances?
- If a person is employed, he/she should have an employment income.
- And more.

Special classical checks are those which are done for all ordered pairs of household members (“B” in the Table 4).

We often use enumerated fields of type EDITTYPE to gain access to the results of the checks what are particularly interesting in the final stage of output data validation.

Figure 7. Blaise 5 SILC after Conversion – How to Evaluate and Preserve the Edit Result

```

161 B2911;
162 IF B2911=Ano THEN
163     SIGNAL Er_B21 | (pocV15+pocV17)>=1
164     "<M>KB21: </M><Z>V domácnosti nie je nezaopatrené dieťa vo veku 15 - 19 rokov a poberá štipendium</Z>" ;
165     B2912;
166     B2913;
167     IF (B2912<>DK AND B2912<>RF) THEN
168     SIGNAL Er_B29 | ((B2912>=22.61) AND (B2912<=B2913 *452.1))
169     "<M>KB29: </M><Z> Overté výšku štipendií v domácnosti - výška štipendií v roku 2015 bola v intervale 22,61 až 45,21 EUR </Z>"
170     ENDIF
171     SIGNAL Er_B22 | (pocV15+pocV17)>=B2913 INVOLVING (B2913, A_A.PocOD)
172     "<M>KB22: </M><Z>V domácnosti je menej nezaopatrených detí vo veku 15 - 19 rokov, ako na ktoré domácnosť poberá štipendium</Z>" ;
173     ENDIF;

```

6.2 Year-Over-Year Kind of Checks

Year-over-year (YOY) comparisons are an effective way to evaluate data. This method facilitates the cross comparison of sets of data. YOY checks can be seen in our SILC survey solution. The basic principle of the SILC survey is a sample repetition with file variations rules. The specific case of check occurs when the household or the respondent are in the survey sample repeatedly. Then current answers are compared to previous year values and if they are not consistent they must be verified. For this purpose several previous data files have to be imported to the solution then have to be adapted for key searching (Table 4).

Compared to previous year data, we have defined many new checks in the current solution. As a result some kinds of errors do not occur in the output file now, at all.

We can see the definition of several classical and YOY checks in the Table 4

As Figure 8 denotes, the signal check of KBM28 has to be verified if the current specified payment for electricity differs by more than 20 % compared to previous year value.

Figure 8. Blaise SILC Solution - Program Snippet of YOY Checks

```

{Kontroly B_B - medziročné}
IF (B_B.B21[2].B21b<>EMPTY AND (PData_BF.SEARCH(A_A.ID_DOM))) THEN PData_BF.READ;
IF (PData_BF.B21b2<> EMPTY AND B_B.B21[2].B21b<>DK AND B_B.B21[2].B21b<>RF) THEN
    SIGNAL Er_BM27 | ((PData_BF.B21b2*1.2>=B_B.B21[2].B21b) AND (PData_BF.B21b2*0.8<=B_B.B21[2].B21b))
    "@MKBM27: @M@Z Náklady na bývanie - úhrada za teplo a teplú vodu sa líši o viac ako 20% oproti
    minulému roku ( ^PData_BF.B21b2) - overte situáciu. @Z" ;
    ENDIF;
ENDIF;

IF (B_B.B21[3].B21b<>EMPTY AND (PData_BF.SEARCH(A_A.ID_DOM))) THEN PData_BF.READ;
IF (PData_BF.B21b3<> EMPTY AND B_B.B21[3].B21b<>DK AND B_B.B21[3].B21b<>RF) THEN
    SIGNAL Er_BM28 | ((PData_BF.B21b3*1.2>=B_B.B21[3].B21b) AND (PData_BF.B21b3*0.8<=B_B.B21[3].B21b))
    "@MKBM28: @M@Z Náklady na bývanie - Elektriina sa líši o viac ako 20% oproti minulému
    roku ( ^PData_BF.B21b3) - overte situáciu. @Z" ;
    ENDIF;
ENDIF;

```

Table 4. Blaise 4 and 5 SILC Solutions – Example of Error Checks Definition List

Name	Kind of check ¹	Type	Meaning of check	Description of check
G01	C	Hard	Nedovolený vek respondenta! Vek musí byť vyšší ako 11 rokov.	$G1_Vek \geq 12$ OR $G1_Vek = DK$ OR $G1_Vek = RF$
G02	C	Signal	Pre osobu vo veku do 16 rokov môže byť v otázke na rodinný stav uvedené len slobodný/á	IF NOT ($G1_Vek = DK$ OR $G1_Vek = RF$) AND $G1_Vek <= 15$ THEN $G1_RodSt = Slob$
KAN48	B	Signal	Vek dieťaťa má byť aspoň o 16 rokov menší ako vek rodiča	IF $OsTAB[02].Vzt01=OtMa$ THEN ($OsTAB[02].Vek-OsTAB[01].Vek$) ≥ 16
KAN50	B	Signal	Vek vnúčaťa má byť aspoň o 32 rokov menší ako vek starého rodiča	IF ($OsTAB[02].Vzt01=Vnuca$ AND $OsTAB[01].Vek <> EMPTY$)
KAN56	B	Signal	Osoba ^ $OsTAB[i].Ido$ v domácnosti má viac manželov/manžieliek a druhov/družiek ako 1	($A_B.OsTAB[i].manz1+A_B.OsTAB[i].druh1+ \dots +A_B.OsTAB[i].manzj+A_B.OsTAB[i].druhj$) ≤ 1
KBM01a	Y	Signal	Náklady na bývanie celkom	(($(B_B.B21m+PData_BF.B22)*1.2 >= B_B.B21t$) AND ($(B_B.B21m+Data_BF.B22)*0.8 <= B_B.B21t$))
KBM28	Y	Signal	Náklady na bývanie - Elektrina	$B21[3]b21b$ (Year) = $b21b3$ (Year-1) +- 20%
KBM29	Y	Signal	Náklady na bývanie - Plyn	$B21[4]b21b$ (Year) = $b21b4$ (Year-1) +- 20%
KCM07	Y	Signal	Starobné dávky	if $C8011(Year-1) = C8011(Year) = 1$ then KONTROLA: väčšia z ($C8012, C8014$ - minulý rok) +- 20% z (menšia $C8012, C8014$ - tento rok)
	Y	Signal		if $C8012(Year-1) = C8012(Year) = 1$ then KONTROLA: väčšia z ($C8022, C8024$ - minulý rok) +- 20% z (menšia $C8022, C8024$ - tento rok)
	Y	Signal		if $C8013(Year-1) = C8013(Year) = 1$ then KONTROLA: väčšia z ($C8032, C8034$ - minulý rok) +- 20% z (menšia $C8032, C8034$ - tento rok)

¹ Kind of the error check: C – classical, B – between two household members each other, Y – year-over-year

6.3 Tools for evaluation of checks

During the data collection phase the interviewer has to check his/her collected and inputted data from inside a record (Figure 9) or by a view to the data browser (Correctness Status and Error Counter columns, Figure 10). Our solutions also include several kinds of control protocols for interviewer to be able to evaluate errors and other required criteria by only one view to his/her records set.

Figure 9. Blaise 4 ICT Solution - Window of All Errors of the Current Recording Questionnaire

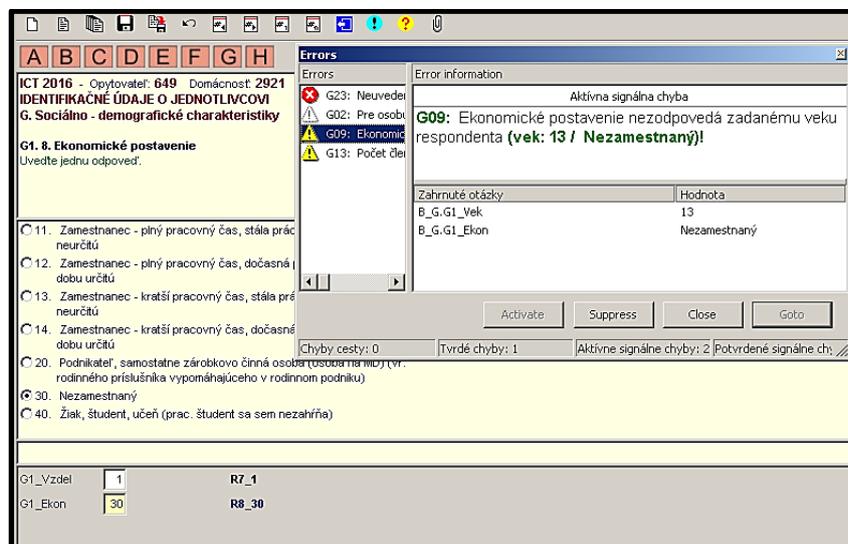


Figure 10. Blaise 4 ICT Solution - Data Browser with the Correctness Status and Error Counter Columns

Status	Error Counter	ID_DOM	KOp	PAPI_S	Kraj	Okres	KodObc	Typ	NazObec	Ulica	SupCD	C
Dirty	2	2921	649	_CAPI	6	601	508438	6	Banská Bystrica	Bernolákova	6011	4
Clean	0	2922	649		6	601	508438	6	Banská Bystrica	Bernolákova	6129	3
Clean	0	2923	649		6	601	508438	6	Banská Bystrica	Bernolákova	6011	4
Suspect	2	2924	649	_CAPI	6	601	508438	6	Banská Bystrica	Bernolákova	6011	4
Dirty	18	2925	649		6	601	508438	6	Banská Bystrica	Bernolákova	6011	4
Clean	0	2926	649		6	601	508438	6	Banská Bystrica	Bernolákova	6129	2

If a survey is simpler and its questionnaire contains fewer questions, the internal design of Blaise solution is easy to navigate. To ensure easy going recording of paper form the EMPTY attribute can be left in a field definition. However, many “non-response” answers are not desired in the final phase. That is why this kind of errors is shown in the error protocol, too.

As can be seen in Figure 11, interviewer of code 649 has to solve indicated problems (G02, G09 ...) in his three inputted questionnaires. Figure 12 as output protocol snippet denotes the recapitulation of records correctness.

Figure 11. Blaise 4 ICT Solution - Control Protocol Snippet No. 1

SIGNÁLNE CHYBY				ZÁVAŽNÉ CHYBY			v ý p i s c h ý b					
Č.D.	KOp	Supp.	Aktív.	Cesty	Hard	Spolu						
2921	649	2	1		1	4	G02	G09	G13	G23		
2924	649	1	2			3	A04	B06	B09			
2925	649		5	12	1	18	G01	G02	G03	G04	G05	G06

Počet záznamov v súbore:
 Spolu: 2 3 1 2 3
 =====
 - počet záznamov s nevykonanou kontrolou: ... 0
 - počet čistých bezchybných záznamov: 17
 - počet všetkých záznamov: 20

Figure 12. Blaise 4 ICT Solution - Control Protocol Snippet No. 2

1: OK - bez chýb
 2: OK - s potvrdenými chybami
 A: aktívne nepotvrdené chyby
 H: hard chyby (závažné chyby)
 P: prázdne (nenatypované) záznamy
 C: chyby cesty

Č.D.	KOp	[1]	[2]	[A]	[H]	[P]	[C]	
2921	649		x	x	x			11/ 0/ 0
2922	649	x				x		21/ 0/ 0 nespoľup.
2923	649	x				x		24/ 0/ 0 neobývaný
2924	649		x	x				11/ 0/ 0
2925	649			x	x		x	11/ 0/ 0
2926	649	x						22/22/11 bez chýb

7. Statistics on Income and Living Conditions of Households (SILC)

Project EU SILC was launched as the pilot survey in 2003 in 6 member states (Belgium, Denmark, Greece, Ireland, Luxemburg and Austria) and in country out of European Union in Norway. Initial year for its implementation was 2004, when survey was carried out in 12 member states, in Norway

and Iceland and in case of new members in Estonia. There was derogation for 10 new member states (including Slovakia), so survey was conducted for the first time in 2005.

SILC is the survey whose aim is to obtain information on income distribution, on level and structure of poverty and on social exclusion. This source of data and information enables not only international comparison of Slovakia within EU, but likewise lays foundations for analyses of living standards of population, conceptual planning and taking measures towards improving quality of life of Slovak population. Basic unit of this survey is private jointly managed household. It consists of the people who permanently occupy together one dwelling and reimburse together the household expenditure. Member of household is also a person who is not relative with other members of household, but he/she permanently shares one dwelling and all household expenditure with them.

The result of statistical processing is a database with well-defined outputs which are then used to calculate the indicators and to analyze the poverty and social exclusion. One of the main outcomes of this survey are common indicators of poverty that are used to measure progress towards the objective of the EU 2020 strategy and which are the basis for evaluation and mutual comparison of the level and structure of poverty in the countries of European Union.

7.1 Structure of Blaise SILC Project

The Blaise 4 SILC solution has been created for two similar projects (SILC GP, SILC) which only differed by the number of questions, the range and structure of the surveys samples (Table 5). Programs are made in accordance with defined checks and requirements for all collected variables, which Eurostat update every year. And in addition there are controls included in accordance with our national specifications (for example fixed amount of some social benefits, checks in terms of respondent's age and etc.).

Table 5. Blaise 4 SILC Solution – Summary Overview

Indicator	SILC GP	SILC
Number of Blaise 4.8 Solution Screens	175	173
Number of Surveyed Fields in Total	5 413	5 280
of which: per Household	133	132
per Person (Number of one interviewed Person)	440	429
Number of AUXFIELDS	136	136
Number of used Classifications	9	9
Number of attached previous period Data Files	3	3
Number of attached previous period Data Files Fields	191	191
Number of Checks in Total	3 946	3 906
of which: per Household	130	138
per Person (Number of one interviewed Person)	318	314
of which: HARD	216	216
SIGNAL	3 730	3 690
of which: captured and stored by using EDITTYPE	3 058	3 054
not using EDITTYPE	888	852
Number of Data checking Manipula Files	6	6

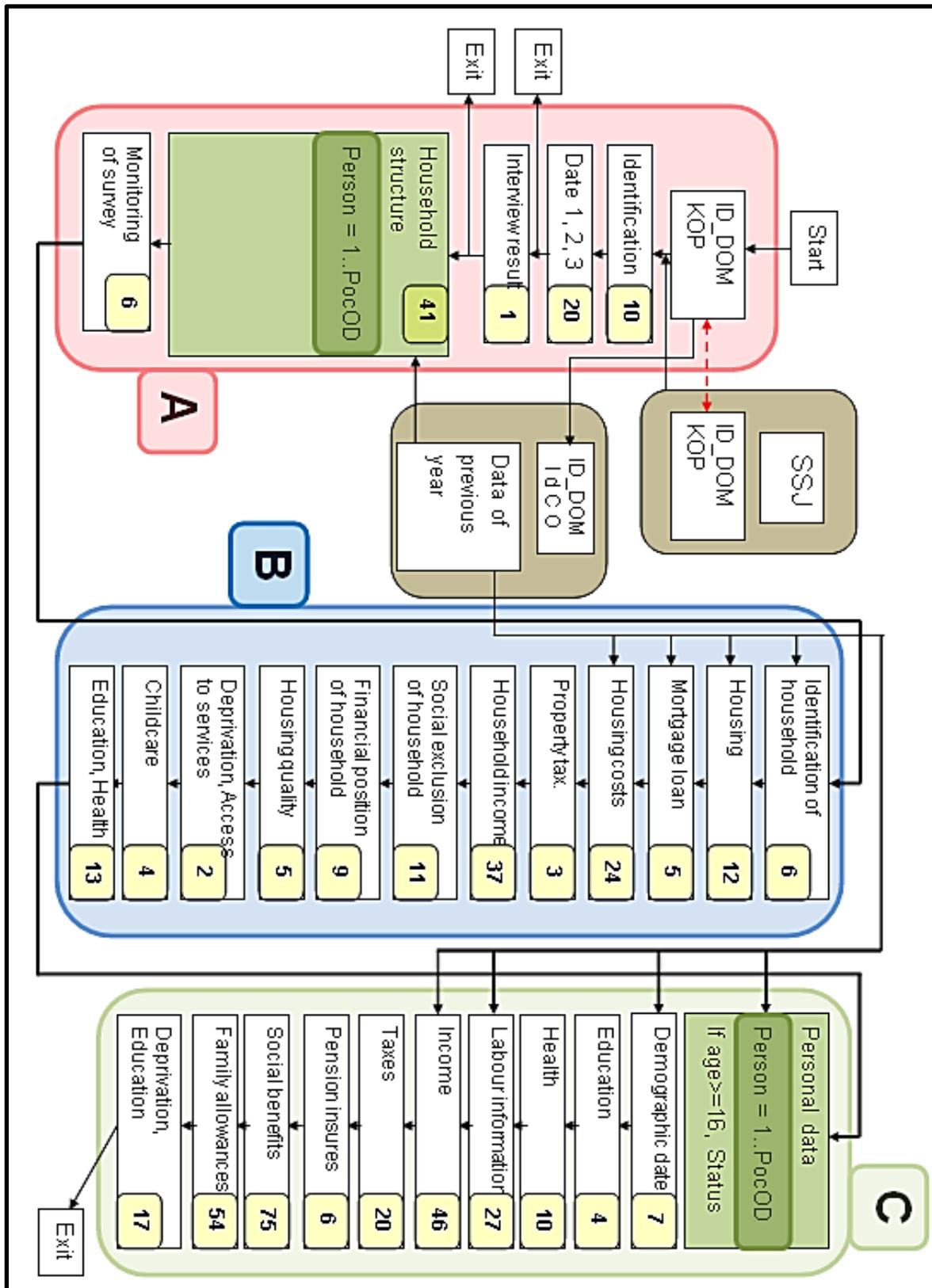
Interviewers acquired the required data based on the information directly from persons in households and filled them in the questionnaires.

As can be seen below (Figure 13), there are 3 types of them:

- SILC 1-01/A – Household structure
- SILC 1-01/B – Household sharing of expenditures data

- SILC 1-01/C – Personal data, which also served as a basis for input processing.

Figure 13. SILC Survey Questionnaire Structure Scheme



7.2 Selected Examples and Illustrations of Blaise SILC Solutions

SILC survey project means very large, difficult and time-consuming survey. Our Blaise 4 solutions (for last 2 years) were successfully used in the practice. They mainly allowed to work by PAPI method with a tablet testing possibility. We made use Blaise 4 software for data recording with fully benefit.

Conversion of the Blaise 4 SILC questionnaire into Blaise 5 equivalent was done on several attempts due to a lot of external data files and classifications. The new solution has been successfully born. It became clear that it is really only the beginning and much work lies ahead of us. Nowadays we continue in this development process. The following figures illustrate our work from the conversion phase to the most recent design with looking back to Blaise 4.

Figure 14. Blaise 5 SILC Layout – Experiment to Generate Critical Fields

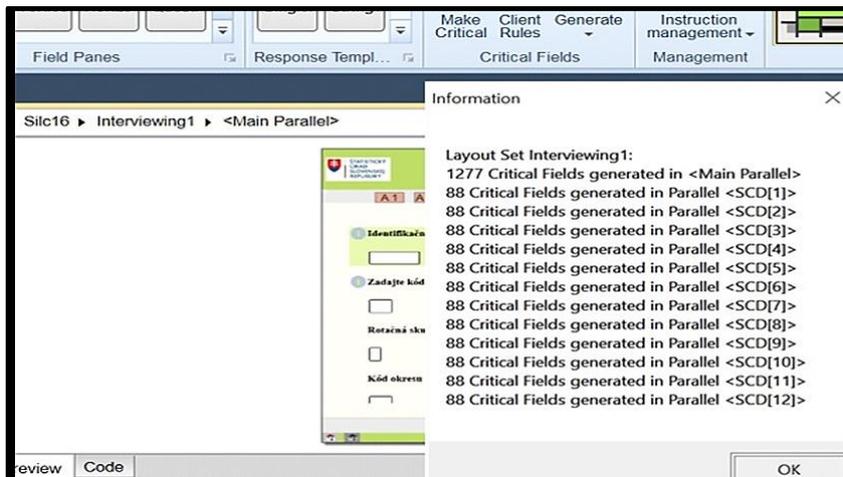


Figure 15. Blaise 5 SILC Resource Editor – Experiment to Insert Image



Figure 16. Blaise 5 SILC – Used Classifications and YOY Data Files (the Source Code)

```

22  USES
23
24  PsilcSSJ  'SSJ\ssj_silc',
25  PCC0086  'silcCis\CC0086',
26  PData_A  'silcCis\DataReg_A',
27  PData_A2 'silcCis\DataReg_A2',
28  PData_A3 'silcCis\DataReg_A3',
29  PData_B  'silcCis\DataReg_B',
30  PData_C  'silcCis\DataReg_C',
31  PISCO08  'silcCis\isco08',          { PC015S  '\SILC14\silcCis\c015S',}
32  PC003S  'silcCis\c003S',
33  PC016S  'silcCis\c016S'

```

As shown in Figures 16 to 18 the solution uses classifications for inputting or searching required data. They are the International Standard Classification of Occupations (ISCO-08), Statistical Classification of Economic Activities (SK NACE Rev. 2), Statistical Nomenclature of Countries and others.

Figure 17. Blaise 4 SILC – Inputting of Citizenship Value (Searching in Classification)

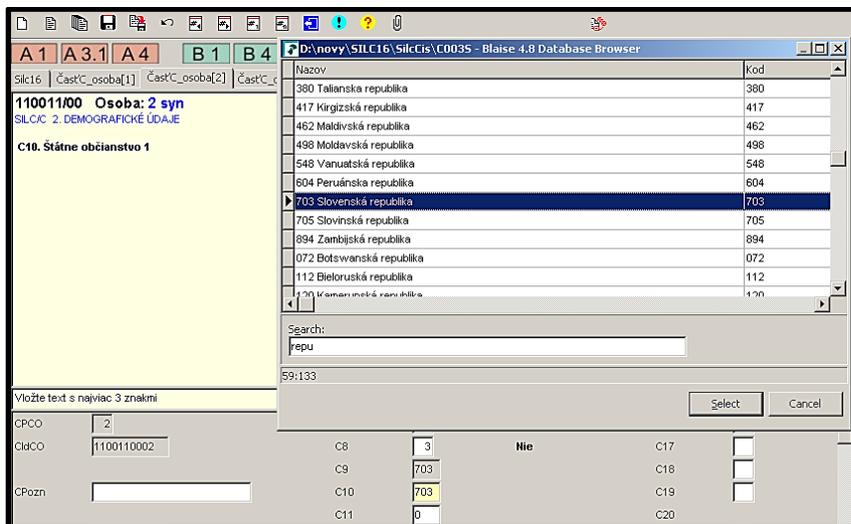
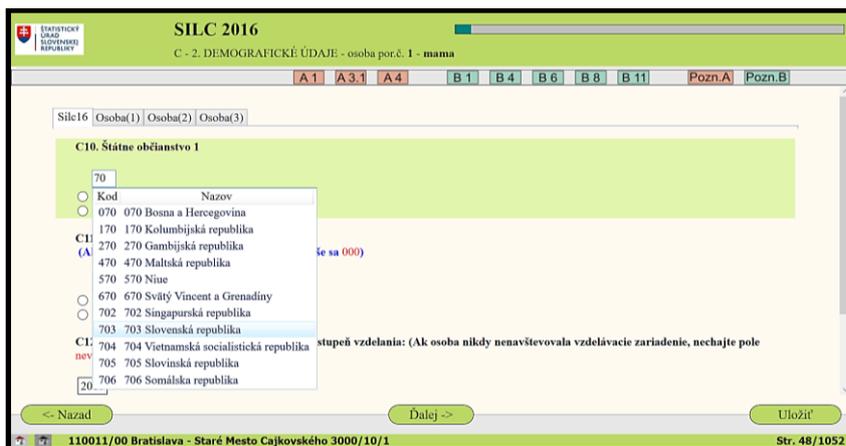


Figure 18. Blaise 5 SILC – Inputting of Citizenship Value (Searching in Classification)



The occurrence of the YOY error can be seen in Figure 19 and 20.

Figure 19. Blaise 4 SILC Solution - Error Window with YOY Check of KBM01a

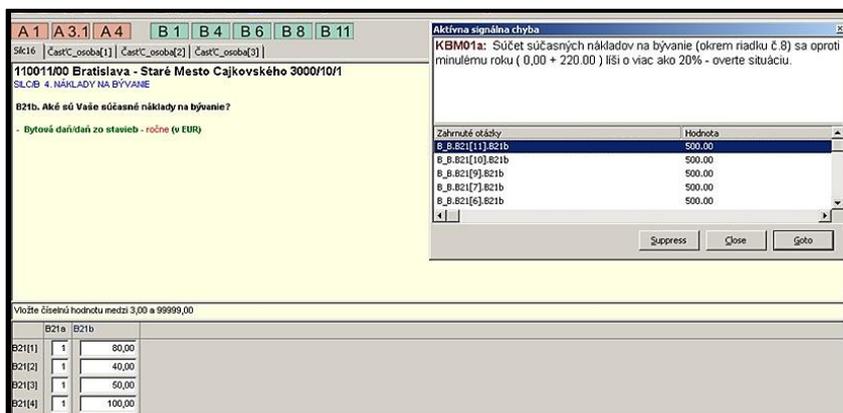


Figure 20. Blaise 5 SILC Solution - Error Message with YOY Check of KBM01a

In Blaise 5 SILC solution we have created a table of household members in another way as in Blaise 4. We have decided to use a rotated table layout due to comments and requests of our interviewers (Figures 21 and 22).

Figure 21. Blaise 4 SILC Questionnaire – Table of Household Members (1 – 12 Persons)

PCO	IJO	IdCO	MNar	RNar	Vek	Pohl	VybOs	SIOa	SIOb	StatG	Modst	MOU	ROU	M2015	HEA15	MPY	RPY	SIByd	RPVz	ZEA16
OsTAB[1]	1	mama	5	1972	44	2	1	1	1	1								1		1
OsTAB[2]	2	syn	2	1996	20	1	1	1	1	1								1		4
OsTAB[3]	3	príbuzná	7	1970	46	2	2	3	3	3				4	2015			1		1
OsTAB[4]	4	dcera príbuznej	11	2015	1	2	2	4	4	4								1		4
OsTAB[5]																				
OsTAB[6]																				

Figure 22. Blaise 5 SILC Questionnaire – Table of Household Members (1 – 12 Persons)

PCO	OsTAB[1]	OsTAB[2]	OsTAB[3]	OsTAB[4]
1	Paradové číslo osoby 1	Paradové číslo osoby 2	Paradové číslo osoby 3	Paradové číslo osoby 4
IdO	mama	syn	príbuzná	dcéra príbuznej
IdCO	1100110001	1100110002	1100110003	1100110004
MNar	A11. Mesiac narodenia 5	A11. Mesiac narodenia 2	A11. Mesiac narodenia 7	A11. Mesiac narodenia 11
RNar	1972	1996	1970	2015
Vek	44	20	46	11
Pohl	<input type="radio"/> Muž <input checked="" type="radio"/> Žena	<input checked="" type="radio"/> Muž <input type="radio"/> Žena	<input type="radio"/> Muž <input checked="" type="radio"/> Žena	<input type="radio"/> Muž <input checked="" type="radio"/> Žena
VybOs	<input checked="" type="radio"/> Vybraná osoba <input type="radio"/> Spolužívajúci (nová vybraná osoba vo vzorke)	<input checked="" type="radio"/> Vybraná osoba <input type="radio"/> Spolužívajúci (nová vybraná osoba vo vzorke)	<input checked="" type="radio"/> Vybraná osoba <input type="radio"/> Spolužívajúci (nová vybraná osoba vo vzorke)	<input checked="" type="radio"/> Vybraná osoba <input type="radio"/> Spolužívajúci (nová vybraná osoba vo vzorke)
	Bol v domácnosti v predchádzajúcich vlnách a je jej súčasným členom	Bol v domácnosti v predchádzajúcich vlnách a je jej súčasným členom	Bol v domácnosti v predchádzajúcich vlnách a je jej súčasným členom	Bol v domácnosti v predchádzajúcich vlnách a je jej súčasným členom

Figure 23. Blaise 5 SILC Questionnaire – Screen Design with Parallels

SILC 2016
C - POZNÁMKA K OSOBE DOMÁCNOSTI - osoba por.č. 2 - syn

A.1 A.3.1 A.4 B.1 B.4 B.6 B.8 B.11 Pozn.A Pozn.B

Silc16 Osoba(1) Osoba(2) Osoba(3)

Poradové číslo osoby 2

C2. Identifikačné číslo osoby 1100110002

Poznámka - C
Sltži na uvedenie nejakých zvláštnych okolností, ktoré môžu spôsobiť nejaké neštandardné situácie a aktivovať kontroly, ktoré v prípade potreby treba tu skomentovať (prípadne pri aktuálnej otázke) a poukázať
Uveďte krátky text, ktorý bude identifikovať osobu a prípadne jej nejaké anomálie
Osoba je prístahovaná z iného mesta a v domácnosti bude bývať minimálne 4 roky

C3. Dátum opytovania pre formulár C - Deň, Mesiac, Rok
5 15 2016

Dalej -> Uložiť

110011/00 Bratislava - Staré Mesto Cajkovského 3000/10/1 Str. 1/84

As Figure 23 illustrates (and other of Blaise 5 SILC Figures above), our screen design includes:

- A progress bar
- A current section title (depending on a Role definition of a current field),
- Image buttons for skipping at a specific question or field of a large questionnaire
- Parallels depending on the number of household members
- Field definitions as Help, Interviewer and own specific roles
- Buttons for moving one page backward or forward and for saving a record
- A bottom line of a screen which allows to skip at the first and the last page (by clicking on the little white and dark houses in a left bottom corner)
- A place for current ID household and its address in the middle of the bottom line of each page
- A current page position (from all pages in solution / parallel) in a right bottom corner

8. Summary

Statistical Office of the Slovak Republic has used Blaise 4 household survey solutions for data collection and processing. Nowadays we study Blaise 5 and develop pilot program solutions as a test projects with a vision of their future implementation in practice.

Topics covered in this paper include the overview of the several Blaise 4 statistical projects. We introduce solutions for the data collection progress monitoring as well as Blaise program layout requirements. We describe the way to ensure the quality of the data as the most important need of outputs. The last part deals with the real Blaise 4 SILC program in comparison to the Blaise 5 SILC development version.

The authors describe their personal experiences from data collection and processing as well as from developing and programming activities. The used illustrations and examples selected from the Blaise 4 HBS, ICT and SILC solutions are those of the authors own work.

The authors would like to thank the staff of Section of Field Surveys for their help and comments to the paper.

9. References

Statistical Office of the Slovak Republic, Statistical Yearbook of the Slovak Republic, 2014

Statistical Office of the Slovak Republic, Regional Statistical Yearbook of Slovakia, 2014

Blaise 5 Help Reference Manual, Statistics Netherlands

<http://help.blaise.com/>

Blaise 4.8 Online Assistant, Statistics Netherlands

<http://www.blaise.com/support/blaise-4-x>

DEP Unlimited: Solving Complex Run-Time Problems with Manipula

Rick Dulaney and Peter Stegehuis, Westat, United States

1. Abstract

The Blaise 4.8 Data Entry Program has the capability to shell out to Manipula, which allows us to solve some difficult problems during data collection. We will show three examples: (1) When selecting dates, it can be desirable to use a graphical date-picker rather than the Blaise Date type. We use Manipula to establish the allowable date range, to display and manage the date-picker, and to store the resulting data in the active datamodel. (2) There are times when it is desirable to add “off-path” data during the course of an interview. We describe some advantages of using Manipula rather than parallel blocks. (3) The visible components and controls available in Manipula can provide a very effective tool for managing lists or rosters. In addition to displaying and selecting items from the list, we can use Manipula to allow additions and modifications to the list as well.

2. Introduction

Blaise 4.8 introduced significant new capabilities in Manipula. The traditional use for Manipula prior to version 4.8 was for batch manipulation of data between Blaise datamodels, or for extract-transform-load operations between Blaise and external software systems. In version 4.8, Manipula functionality is part of the DEP executable, which means Manipula can operate on the DEP record in memory, with simultaneous access to other datamodels and external data files. Manipula now also has direct access to the Blaise metadata, complementing and sometimes replacing Cameleon programming.

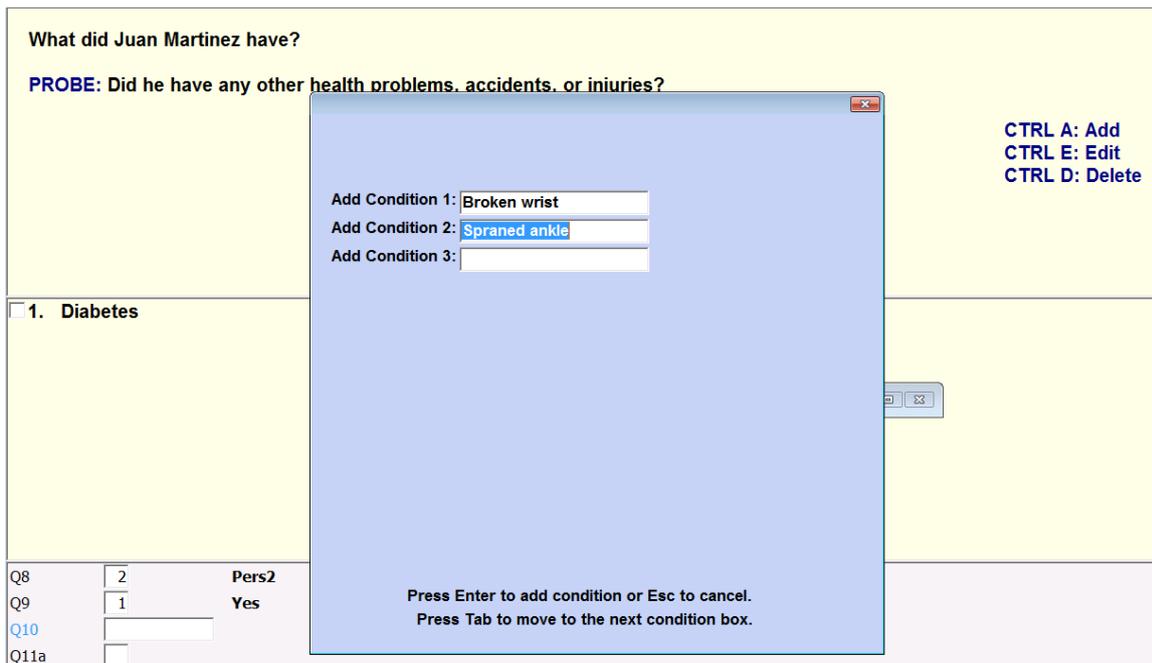
As we noted in our 2013 IBUC paper, one approach to solving complex routing or data access problems while the DEP is running is to shell out to Manipula. A Blaise datamodel can invoke Manipula at run-time in several ways, including menu selection, buttons programmed to appear (or not) on the task ribbon, or even as part of selected response using data types. Manipula programs can show dialog boxes, read and write data using Blaise datamodels, read and write data from non-Blaise sources, and even update the record in memory using the datamodel which called Manipula and is still active. There are new language keywords to help manage the return from Manipula to the active Blaise datamodel, including `GETACTIVEFIELD` (get the full name of the field from which the Manipula procedure was started), `SETACTIVEFIELD` (the full name of the field to return to after the procedure has been completed) and `SETALIENROUTERACTION` (to specify the desired action after the Manipula procedure has finished).

The basic approach is to invoke Manipula from the Blaise datamodel, while the DEP is running, using one of the methods above. We can then use Manipula to read, display, update and store data in and from a variety of data sources. Finally, the Manipula program can return to the DEP either at the place we departed or at some other point determined algorithmically. This relatively simple sequence can be used to solve some very complex problems. We discuss three of these challenges below - for each, we first describe the challenge and then provide a solution using Manipula during run-time.

3. Challenge 1: Managing Rosters

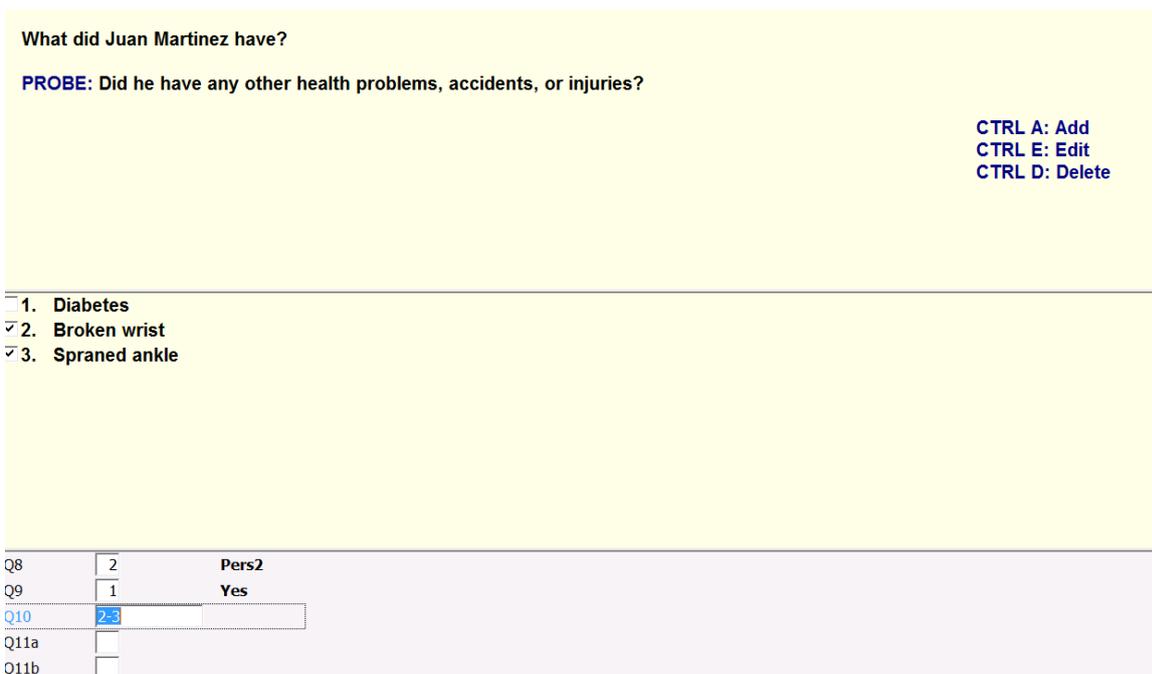
A roster is a list of items. Typical rosters include persons (such as a household enumeration) medical events or conditions, jobs held, etc. A nested roster is a list of items collected for each entry in a different list, such as the medical events for each person in the household. In some cases, entries in one list may be linked to another list, such as storing a medical provider for each medical event for

Figure 2. Manipula Add Dialog



We can add one or more health conditions as reported and either press <Esc> to cancel or <Enter> to add the new conditions and return to the roster question. After pressing <Enter> we'll see that the conditions have been added and selected at the original question:

Figure 3. Updated Conditions Roster



Manipula code added the new conditions and included them in the set question answer, after pressing <Enter> in the dialog and before returning control to the DEP screen.

Now note the typo in one of the added conditions, 'spraned' instead of 'sprained'. To correct that we press Ctrl-E for Edit:

Figure 4. Edit Dialog

What did Juan Martinez have?
PROBE: Did he have any other health problems, accidents, or injuries?

CTRL A: Add
CTRL E: Edit
CTRL D: Delete

Edit condition:
Edit Condition
Edit Condition2
Edit Condition3

Press Enter to Edit condition or Esc to cancel.
Press Tab to move to the next condition box.

1. Diabetes
 2. Broken wrist
 3. Sprained ankle

Q8	<input type="text" value="2"/>	Pers2
Q9	<input type="text" value="1"/>	Yes
Q10	<input type="text" value="2-3"/>	
Q11a	<input type="checkbox"/>	
Q11b	<input type="checkbox"/>	

And after pressing <Enter>:

Figure 5. Fixed Typo

What did Juan Martinez have?
PROBE: Did he have any other health problems, accidents, or injuries?

CTRL A: Add
CTRL E: Edit
CTRL D: Delete

1. Diabetes
 2. Broken wrist
 3. Sprained ankle

Q8	<input type="text" value="2"/>	Pers2
Q9	<input type="text" value="1"/>	Yes
Q10	<input type="text" value="2-3"/>	
Q11a	<input type="checkbox"/>	
Q11b	<input type="checkbox"/>	

The same principle applies for deletion.

Note that in the dialog box for editing the first condition, Diabetes, is grayed out. It shows that we can control per screen which roster elements can and which ones can no longer be edited. The rules for editing can be set as desired, but it is useful and often important to be able to have such rules.

4. Challenge 2: Managing off-path data

Respondents sometimes provide data at a time other than when it is asked. In simple cases, the interviewer can simply back up to the appropriate item and correct the previous response. However, there are situations where this may not be the ideal solution:

- It may be that the interviewer cannot back up because the data has been intentionally protected from corrections through the use of a wall or similar technique.
- Perhaps the information has been used to direct other main branches of the survey and the decision has been made not to allow changes. For instance, if the survey collects information about a school, then samples a few children, then collects information about each of those children, it may not be desirable to allow the interviewer to re-sample.
- In some cases the interviewer may require more flexibility, such as the need to pause a particular section, collect some other data possibly from a different respondent, and then return to the original point of departure. This can often occur in establishment surveys, where interviewers may need to speak with several respondents such as a principal, an administrator, and one or more teachers in a school.

Blaise does offer the use of parallel tabs, and these are useful in certain instances, but Manipula offers substantially more control. One recent project had a default path: first enumerate a household and then for each person, collect medical conditions, collect medical events and prescribed medicines, and finally collect jobs and insurance policies. However, interviewers required the flexibility to move to other sections as needed and collect that information, usually if one respondent needed to leave soon. Our solution was to use a menu button which led interviewers to the list of items that could be reviewed and updated. This single point of departure is organized and presented using Manipula.

This design has several advantages:

- We can collect data using dialog boxes, separate DEP datamodels, or external programs as appropriate.
- We can control where the interviewer goes. It may be appropriate not to allow the interviewers to enter sections, for instance, if the section has dependencies on another section that must be completed first.
- We can provide indications to help the interviewer. For instance, in the example below, the screens collecting detail for an entity added using off-path navigation have a green background so the interviewer expects to return to the original point of departure in the interview.

Figure 6. Status and 'Switch To' Popup

The screenshot shows a survey application interface. At the top, there is a menu bar with 'Forms', 'Answer', 'Navigate', 'Options', and 'Help'. Below the menu bar, there are two buttons: 'Review/Add (Ctrl-V)' and 'Switch To (Ctrl-Q)'. The main content area has a yellow background and contains the question: 'Is this medicine related to a health care event?'. Below the question, there are two radio button options: '1. Yes' and '2. No'. At the bottom of the main content area, there are four input fields labeled 'RAMedicine2', 'RAMedicine3', and 'RAMedicine4', with a 'Yes' label and a '1' in a box next to 'RAMedicine2'. A 'Switch To' popup window is open over the main content area. The popup window has a title bar with a close button and contains a table with the following data:

Prescribed Medicines	Status
Tylenol	In progress
Aspirin	In progress
Mothers Little Helper	In progress
Flexorall	In progress

At the bottom of the popup window, there are two buttons: 'Return' and 'Go'.

Here we have follow-up questions on every medication that has been collected earlier in the interview. Generally you have to go through these in order, but the above popup screen serves both as a status screen and as a way to deviate from the standard order. When we select Aspirin, the second medication in this list, and then press Go:

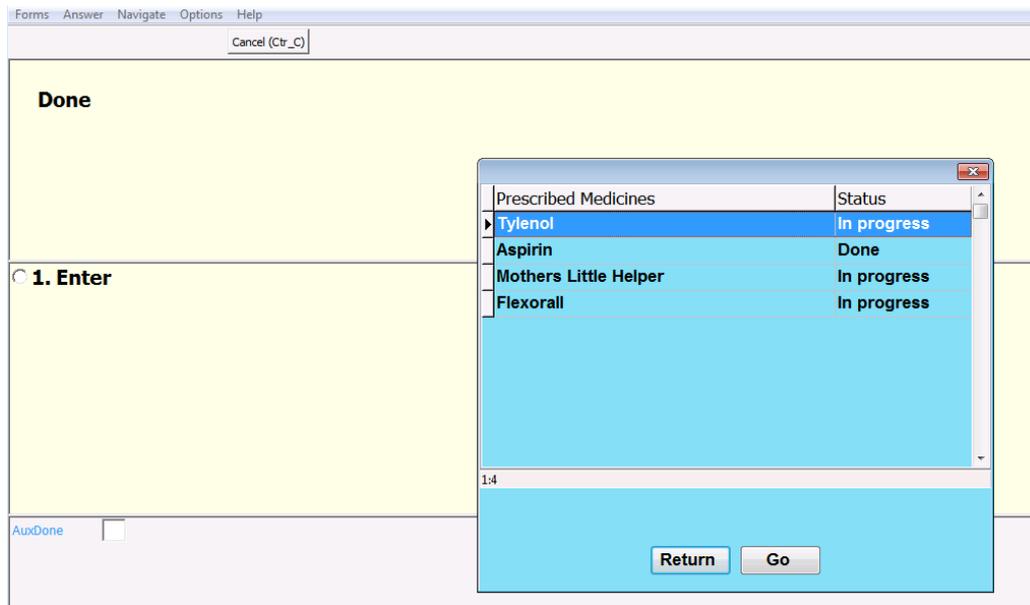
Figure 7. Off-Path Questions

The screenshot shows a software window with a menu bar containing 'Forms', 'Answer', 'Navigate', 'Options', and 'Help'. Below the menu bar is a 'Cancel (Ctrl_C)' button. The main content area has a green background and displays the text 'Medicines indicated this round for John Smith' and 'Aspirin'. Below this, there is a radio button next to '1. Continue'. At the bottom of the window, there are three input fields labeled 'RAMedicine2', 'RAMedicine3', and 'RAMedicine4'. The 'RAMedicine2' field is currently selected and has a dotted border.

The green background indicates that the interviewer has departed from the normal route, so it's hard to miss that we're off the beaten path here. Note also that the two previously available buttons are no longer showing and only a Cancel button is available. Pressing the Cancel button just gets you back to where you left off. The other buttons were removed temporarily, so as not to confuse the interviewer with 'options within options'.

After answering the questions for Aspirin we return, first to the overview screen to show the new status, and then back to where we left off:

Figure 8. Updated Status and 'Switch To' Popup



Note the 'Done' status for Aspirin. The options now are to choose another row, or simply Return to go back to the questions for the first medication, Tylenol:

After answering the questions for Tylenol we get, via the normal routing, to the questions for Aspirin, which have been answered already. We can change the answers or simply press <End> to go to the questions for the next medicine.

This is a very simple example, but this principle can be applied in many different ways: with much bigger sections, or even allowing to choose selections on multiple levels (e.g. allowing to select a person to talk about all his/her medications, but also allowing to select one specific medication for person 2, and then return to person 1).

Alternatively, once the interview has gone too far beyond the normal spot to ask a section, you could allow a subset of questions when a respondent volunteers that they forgot to mention a job, a medication, a doctor's visit or even a member of the household.

5. Challenge 3: Using External Programs to Collect and Manage Data

There are design requirements which sometimes call for input options other than Blaise data types. A common example is the use of a graphic element to collect data, such as a graphical date picker that lets the interviewer navigate to the correct date by selecting the correct day on a calendar rather than entering numeric date values. It is relatively straightforward to locate a third-party date picker and invoke it from Blaise using the ALIEN calls. However, we recently encountered a project with a more sophisticated set of requirements for collecting medical event dates:

- Bound the legitimate dates. In our example, we have a reference period and event dates must fall within that period.
- Allow begin and end dates for certain event types to contain an interval. For instance, a doctor visit occurs on a single date, but a hospital stay may begin and end on different dates.
- Select multiple event dates on the same screen. We did not want a time-consuming loop between the date picker and a question such as "Any more dates?". The requirement is that the interviewer collect all of the dates that respond to the question prompt at once.
- Allow the same date to be selected for more than one event. An individual may see more than one doctor on the same date.
- Allow events to be deselected if they were added by mistake on this screen.

- Enable interviewers to enter recurring events (such as dialysis) in the most efficient way possible.
- Store all events in an array in the current Blaise record in memory.

Our solution uses an external application written at Westat in C# using open source Microsoft .Net components. Interviewers may either select “Add an Event” in response to the appropriate question, or they may use the button described above to review and update events. Manipula computes all necessary parameters – reference dates, event type, previously-entered events, etc. – and sends them to the .Net application at call time. When the interviewer leaves the date picker, Manipula accesses all the new events and writes them into the medical event array in the Blaise record in memory.

Figure 9. Select Person

INTERVIEWER: SELECT CORRECT PERSON FOR THIS EVENT.

1. Jennifer Lynn Martinez
 2. Juan Martinez
 3. Corey Martinez
 4. Celia Martinez

Q2	<input type="text" value="1"/>	Pers1
Q3	<input type="text" value="1"/>	HS
Q4	<input type="text" value="1"/>	Prov1
Q5	<input type="text"/>	
Q6	<input type="text"/>	

Here is an initial screen, just to select the correct person for a medical event. Next we select the type of event –

Figure 10. Select Event Type

Where did you receive the care?

1. HOSPITAL STAY (HS)
 2. MEDICAL PROVIDER VISIT (MV)

Q2	<input type="text" value="1"/>	Pers1
Q3	<input type="text" value="1"/>	HS
Q4	<input type="text"/>	
Q5	<input type="text"/>	
Q6	<input type="text"/>	

In this small example there are only two options, as each has its own 'calendar behavior'.

Next we select the medical provider for the event:

Figure 11. Select Medical Provider

What is the name of the person or place that provided health care to you?

SELECT CORRECT PROVIDER ASSOCIATED WITH THE EVENT.

1. Dr. Holly Harris

2. Johns Hopkins Hospital

3. Mayo Clinic

4. MedOne Urgent Care

5. Righttime Medical

6. Dr. Albert Takem

Q2	<input type="text" value="1"/>	Pers1
Q3	<input type="text" value="1"/>	HS
Q4	<input type="text" value="2"/>	
Q5	<input type="text"/>	
Q6	<input type="text"/>	

Then we get to the screen that starts the external executable:

Figure 12. Screen to Launch External Executable

When you were admitted to and discharged from Johns Hopkins Hospital? Please tell me the dates of all stays between 1/1/2015 and 6/17/2015.

IF NECESSARY, PROBE: On what date did you enter Johns Hopkins Hospital? On what date did you leave Johns Hopkins Hospital?

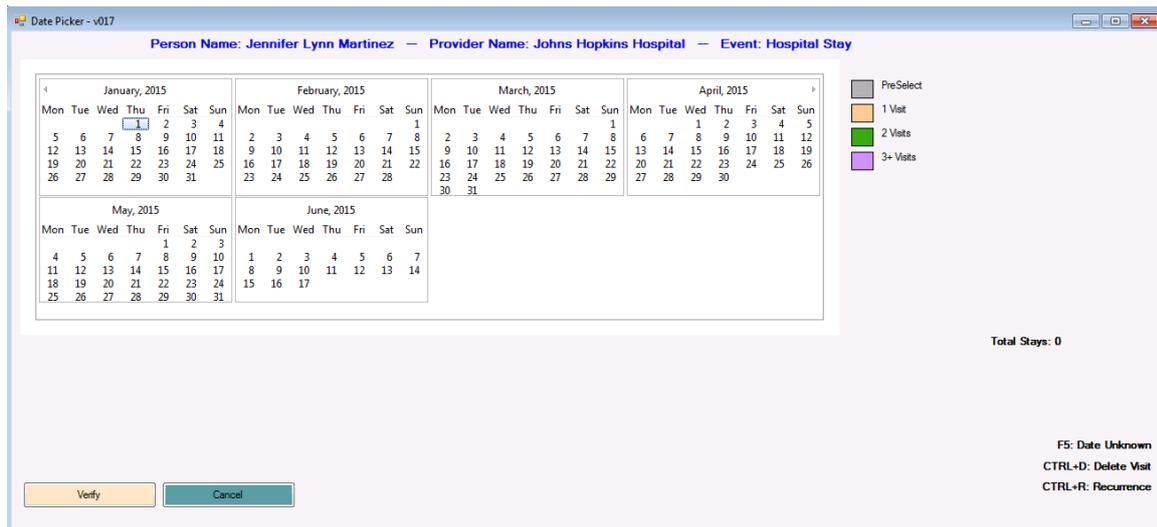
PROBE: Any other stays?

PRESS 1 AND ENTER TO LAUNCH THE CALENDAR

1. LAUNCH DATEPICKER

Q2	<input type="text" value="1"/>	Pers1
Q3	<input type="text" value="1"/>	HS
Q4	<input type="text" value="2"/>	Prov2
Q5	<input type="text" value="1"/>	
Q6	<input type="text"/>	

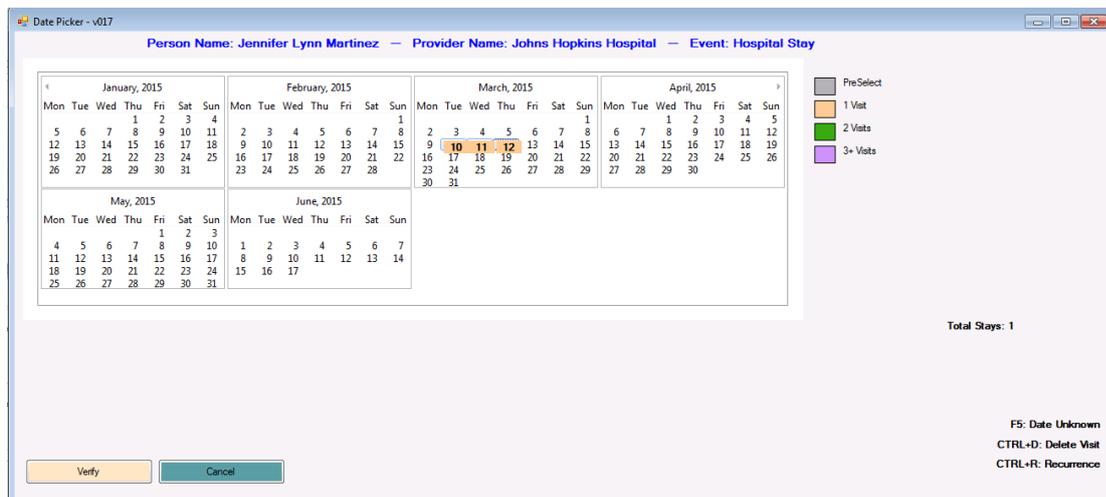
Figure 13. Date Picker



Note the ‘context header’ line at the top, and that the complete reference period for this person shows up at once. It makes it easy to select the correct dates, while automatically safeguarding against entry of dates that are outside of the reference period and therefore not allowed.

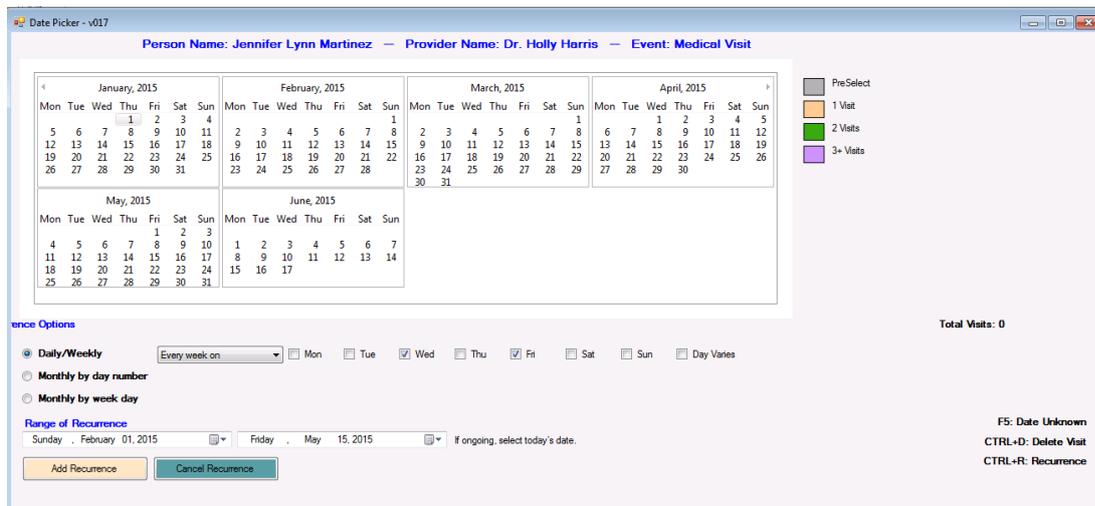
Both mouse and keyboard can be used for selecting dates. Here is an example of one hospital stay as selected in the calendar:

Figure 14. Sample Hospital Stay



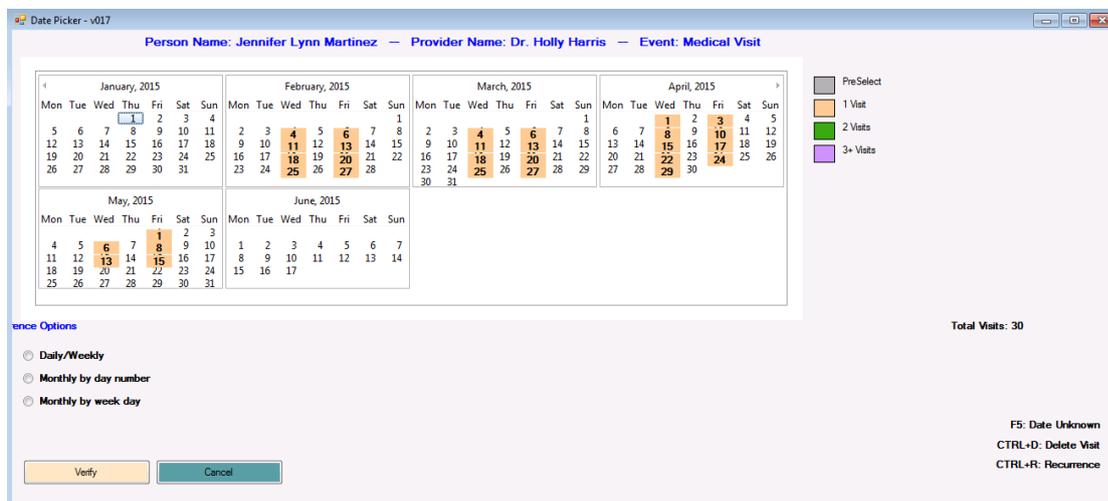
Here's an example of the recurrence feature for single date events, adding many events at once instead of one at a time:

Figure 15. Recurrence Feature for Single Date Events



After pressing the Add Recurrence button:

Figure 16. Add Recurrence

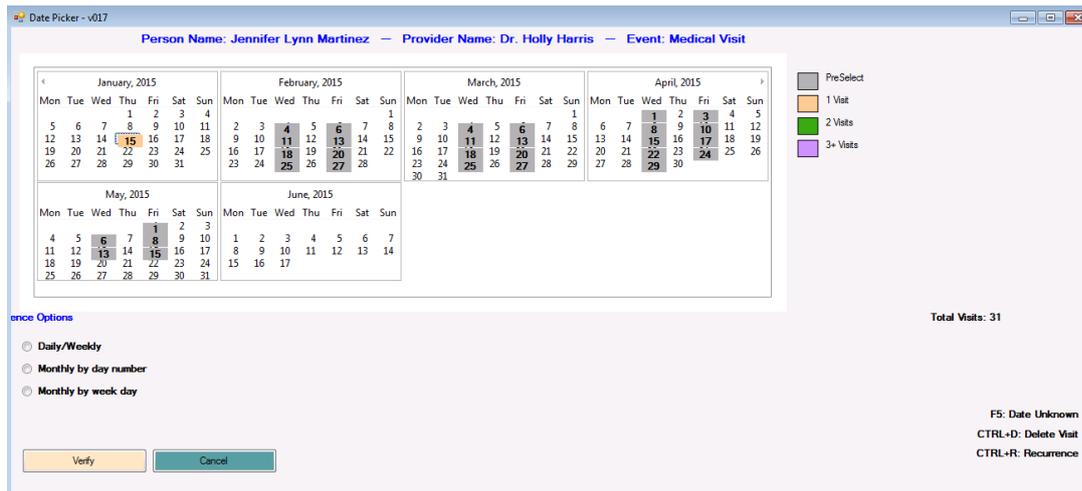


As is shown with the counter on the right side of the screen, this added 30 medical visits at once.

Just like with the example of roster popups earlier, Manipula procedures are doing the work behind the scenes, passing to the Calendar executable any info it needs and writing the selected dates to an array in the main datamodel.

To show that the dates are written correctly and that rules can be enforced for previously entered event dates, here's the Calendar screen for the same person-event type-provider combination, for a second time, showing the previously entered dates, now non-editable:

Figure 17. Non-Editable Dates



6. Conclusion

There are some considerations to be aware of when using these capabilities as described above. First, data collected or edited using Manipula procedures are not subject to the selective checking mechanism that governs fields collected directly in Blaise. This blend of techniques – selective checking for DEP and traditional batch processing for Manipula routines – requires the careful development of good design and programming practices. Second, to the extent that the Blaise datamodel is used to document or direct additional survey activities beyond data collection, the use of Manipula routines may have significant implications. Examples include the use of the datamodel to support data editing, metadata systems, or other downstream processes.

The examples we have shown are the tip of the iceberg – the ability to use Manipula to access the DEP record in memory along with any other data source puts solutions to many challenges within reach.

Blaise on Touch-Screen Tablets: the ELIPSS example

Alexandre Chevallier and Mathieu Olivier, CDSP – SciencesPo, France

1. Abstract

Starting in 2012, the ELIPSS project is a French probability-based web panel. All panel members are equipped with a touch-screen tablet to answer, each month, a self-administered survey programmed with Blaise 4, through a pre-install app.

This paper will describe, through some specific examples, how we had to overcome different challenges impacted by the specific technologies used, interaction between servers, and data collection.

Most of submitted surveys did not require specific developments, but some have instead required a lot of investment to adapt our solutions to Blaise.

We will present the solutions we found to address these problem, as for example for the development of drag & drop, clickable map, or the need to take pictures.

2. Introduction to the ELIPSS Panel

ELIPSS (Étude Longitudinale par Internet Pour les Sciences Sociales) is a probability based Internet panel, dedicated to Social Sciences, in similar way than the LISS Panel.

The pilot study is running since 2012 and consists in 1,039 panel members.

The target population is individuals living in private households in metropolitan France, aged from 18 to 75 at the entry in the panel, having sufficient command of the French language to answer self-administered questionnaires.

The surveys are conducted each month and it takes 30 minutes maximum to answer it. It could be on any topics. These surveys are designed by researchers and selected by the Scientific and Technical Board.

The covered topics are the ones of the projects selected by the Scientific and Technical Board. No topic is *a priori* excluded. In order to be eligible, surveys must have an exclusively scientific purpose. In other words, the purpose cannot serve the particular interests of public or private institutions.

One distinctive feature of ELIPSS is to equip all panel members with touch-screen tablets, in order to answer the self-administered surveys programmed with Blaise 4.8, through a pre-install app.

3. Tablets

Contrary to the other probability based Internet panels, Internet access is offered to all panel members, not only to those who do not have Internet access at home. Thus, each panel member receives a touch-screen tablet with a 3G Internet connection.

We chose tablets instead of other devices (as smartphones or PCs) because their interface is intuitive, providing simplified Internet access to people unfamiliar with new technologies, with a good agreement of the screen size (7 inches).

Due to, the design of surveys is exactly the same for all panel members equipped with the same device.

Moreover, mobile Web access gives more flexibility in completing surveys: panel members can choose their time and location to answer to questionnaires.

Besides the possibilities offered by the Internet (images, video...), the touch-screen tablet allows to administer innovating surveys, which is interesting for researchers and also motivating for panel members.

At last, in the Elipss project, there is no additional incentive each month. To equip all panel members with tablet which is considered as a material incentive. When we asked panel members to cite main reasons for which they agreed to participate in the Elipss panel, the touch-screen tablet was the primary motivation.

4. Blaise's integration

4.1 App

Figure 1. To the app to a questionnaire



The questionnaires are administered through a specific app which was internally developed and is designed for touch-screen device

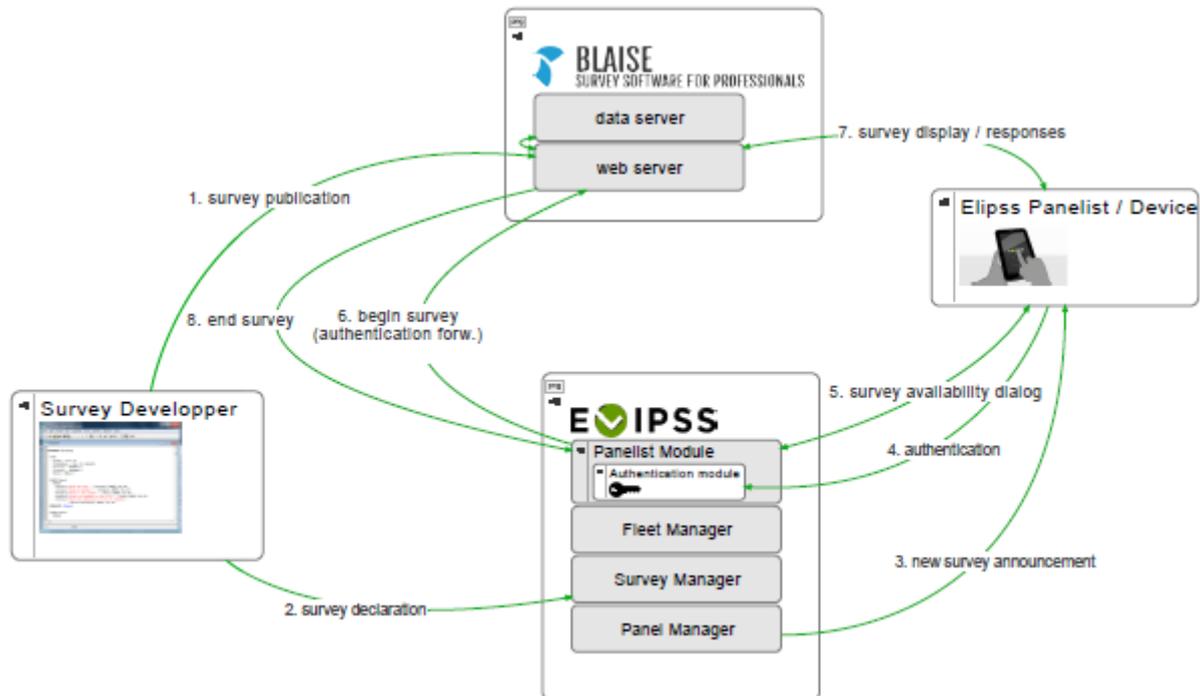
The screenshot displays the pre-installed (on each tablet) app on the home page of the tablet. The panelists have to log in to access the home page of the Elipss app (panelist module). The interface is simple. There are 4 icons : one to complete the questionnaires, another to send app messages to the panel managers, one to keep panel managers informed when panel members cannot answer the survey and the last one is to be used for any change in the panelist contact information (phone number, email address, etc.). FAQ, contacts and logout are at the bottom of the page.

When the tablet was delivered to the panel member's home, panelists were offered training by phone mainly on handling the tablet and exploring the Elipss applet. Two thirds of the panel members were trained, most of the others did not need to be trained. Trainers reported that 20% of panel members were not at ease with the tablet.

The first survey began with a tutorial part presenting different types of questions and the design of Elipss surveys.

4.2 Tools used

Figure 2. Technical Workflow



Through this diagram, we can distinguish different steps, from the survey's conception to retrieve data using Blaise software (version 4.8).

After programming, the survey is published on the Blaise server (1) and declared on the Elipss server (2). From Elipss server, survey is published on all our active panelists and they receive a notification on the Elipss application installed on the tablet (3). A message with a copy mail plus a SMS are sent to our panelists to inform them.

After the panelist's authentication (4) and the selection of a survey (5), Blaise server take over the communication (6). So panelists are able to answer the survey and the answers are saved in the database (workbdb) on the Blaise server after each page of the survey. Once the survey is finished, data are archived (so no more accessible) on the Elipss apps.

One of the difficulty was to create a new interface adapted with constraints of the tactile tablets. A stylesheet (.xsl) special for Elipss (first simplified version was kindly provided by LISS panel, thanks so much to them) is used in contrary of the default Blaise stylesheet. This allowed us to have a specific Elipss design (layout) for "classical" questions and questions adapted to the tablet (touch-screen keyboard, popup help, picker, etc.), but also some news types of question.

5. Adapting surveys with Blaise

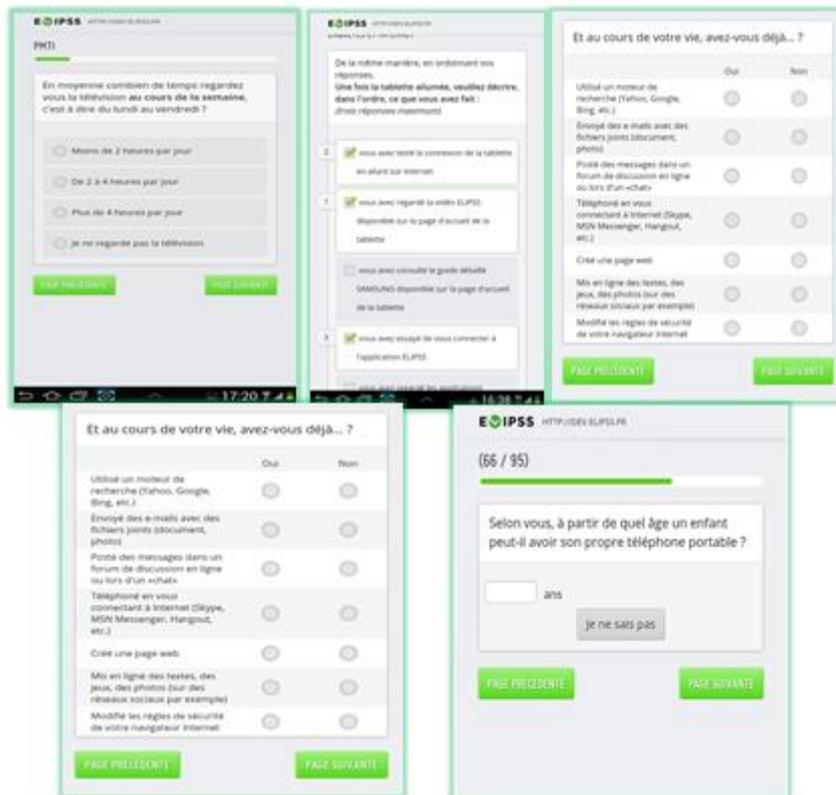
Contrary to Web surveys using different devices and different browsers, all the Elipss panel members have the same device and the same browser to answer questionnaires. This key design choice enables control over how questionnaires are displayed and aims for maximum measurement equivalence.

But we know that tablets also have limitations linked to connectivity problems and screen size.

The Liss panel team provided us their stylesheet to serve as a reference. So we were able to develop our own survey design, and to adapt field types as radio-button for enumeration type, checkbox for set

of type, sliders, group table, matrix, etc. We also designed field attributes like the non-response options refusal and dontknow as buttons when the option is allowed in the first time or after a reminder.

Figure 3. example of “classical” questions



Another difficulty is to adapt “unclassical” questions.

Some surveys that requires more interactivity have provided the opportunity to develop specific features. This was only made possible mainly because all the panel members have the same equipment.

Then we used previous surveys functionalities like auto-recording with the microphone, drag and drop for an adaptation of an experimental survey on social stratification originally played with cards, interactive map to define residential strategies, etc.

Figure 4. example of “unclassical” questions: drag&drop and interactive map



We will see more on details how we developed these features through the photo’s example.

6. Specific feature: Photo example

To use all the functionalities of the tablet we added a Javascript layer composed with Vanilla, JQuery and Cordova. Cordova is a framework which allows to use the functionalities like camera, microphone or GPS on the tablet.

First of all, we call the library (Jquery and Cordova) inside the XSL file. Secondly, in the Blaise code survey manager, we add a special XML tag (<photo></photo>) in the specific question who will be parsed by our javascript.

Figure 5. xml tag

```
d7 (invisible) "<photo><folder>SHAMA_201607</folder><variable>d7</variable><nb>5</nb><nb_max>6</nb_max><username>^uid</username></photo>
@BVeillez prendre en photo le ou les principaux espaces disponibles que vous utilisez pour cuisiner ou préparer des repas
@/@IMerci de ne prendre en compte que les surfaces qui sont dégagées de tout élément tel que plaques de cuisson, four, é
"Espaces préparation repas" : STRING, EMPTY
```

Indeed, our javascript layer parse the result from blaise and detect all the xml tag to initiate the different functionalities.

Then, the specific code linked with this tag will run and activate all the interface and code.

Figure 6. xml parser

```
ds.b.xml_question(
    "photo",
    [
        { 'name': "variable", 'expression': "photo > variable" },
        { 'name': "folder", 'expression': "photo > folder" },
        { 'name': "username", 'expression': "photo > username" },
        { 'name': "limit", 'expression': "photo > limit" },
        { 'name': "nb", 'expression': "photo > nb" },
        { 'name': "nb_max", 'expression': "photo > nb_max" },
    ],
    ds.b.utils.takephoto
);
```

Cordova camera function have two callback (success and fail), success callback give your image encoded in base64 in argument and allows you to manage easily the picture with vanilla javascript.

Figure 7. Cordova call

```
navigator.camera.getPicture(onPhotoDataSuccess, onFail, { quality: 50,  
  destinationType: Camera.DestinationType.DATA_URL });
```

Figure 8. JavaScript after Cordova

```
function onPhotoDataSuccess(imageData) {  
  // Uncomment to view the base64-encoded image data  
  
  // Get image handle  
  //  
  var photo = document.getElementById(photoFileName+'0');  
  var i=0;  
  
  while(photo != undefined){  
    i += 1;  
    photo = document.getElementById(photoFileName+i);  
  }  
  
  if(typeof(Storage) !== "undefined"){  
    var answer = document.getElementsByClassName('answer')[0];  
    localStorage.setItem(photoFileName+i, imageData);  
    var smallImage = document.createElement("img");  
    smallImage.id = photoFileName+i;  
    smallImage.className = "stored-photo";  
    smallImage.style.width = "400px";  
    smallImage.style.height = "600px";  
    smallImage.style.display = 'block';  
    smallImage.src = "data:image/jpeg;base64," + localStorage.getItem(photoFileName+i);  
  }  
}
```

Lastly to save the images, the panelist valid the question to go to the next one and we simply contact our server through an AJAX method with list of images to send them to the django server. At the time, Blaise server receive an answer “true” if the image was saved otherwise, it’s “false”.

7. Conclusion

Challenges are coming!

Since the beginning of 2016, 2,500 new panel members have been recruited to integrate the Elipss panel. This panel members are equipped with a new touch-screen tablet, with 4G connection.

The heterogeneity of equipments require a lot of work, in order to have surveys adapted to both devices and browsers.

Though “old” panel members recruited in 2012 will gradually receive the new tablet, we have to manage surveys on the 2 platforms for a while, as surveys continue to be published each month, waiting for the development of new features for future projects.

From CPI to CPIX

Gerrit de Bolster, Statistics Netherlands

1. Abstract

For the data collection of the Consumer Price Index (CPI) a stand-alone Maniplus 4.8 application was developed in 2013. Based on a list of shops it uses several dialogs with lookup's and functionality to edit the price and weight or size (quantity) of a pre-defined set of products that can be found in the selected shop. As an experiment this Maniplus 4.8 application was converted to a Blaise 5 questionnaire (CPIX) to be used with the data entry program (DEP.exe). The extended and powerful functionality of Blaise 5 made it possible to support more or less the same functions available in Maniplus 4.8 in a Blaise 5 questionnaire. However, the layout was adapted for Blaise 5. The paper will show how the Maniplus 4.8 functions were converted to Blaise 5 fields and rules.

2. History

In June 2013 the office of Statistics Netherlands in the Caribbean Netherlands (the islands Bonaire, Saba and St. Eustatius) started to use Windows 8 tablets for its data collection by CAPI. The first survey was the "Omnibus" survey built in Blaise 4.8 by the Blaise team. It was quite successful and this success got known throughout the office. As a result the statistical department responsible for the CPI contacted the Blaise team and asked them to create also a Blaise solution for the CPI to be used on the tablets as the PAPI solution they were using until then was costing a lot of effort and error prone. The Blaise team agreed as in this way more knowledge about using Blaise for tablets was gathered and, because of the way the CPI was designed, a Maniplus application was created. Although several updates have been issued since its introduction at the end of 2013 the application is still working fine.

3. The CPI

The Consumer Price Index (CPI) is a well-known survey performed by most of the National Statistical Institutes. In Caribbean Netherlands it is a monthly survey. A small group of interviewers gets every month a list of shops and for each shop a list of products with quantities and prices as was collected the previous month. They have to check if the products are still available and if so if the price or quantity (kilo, liter, pieces, etc.) has changed. If so they have to report the changes. If the product is not available anymore they have to report this too and search for alternative and add it to the list. They also have to report if the shop is closed (permanent or temporarily). Every interviewer receives its own unique list.

4. Three different implementations

The Maniplus 4.8 application built by the Blaise team (and which is still in production) is a stand-alone implementation. The application is encapsulated by a shell (also built in Maniplus 4.8) which takes care of the export of completed monthly results and of the import of a new list of shops and products for the next month. As in 2014 a CAPI system called CAPI Logistics System (CPLS, presented during the IBUC 2015 in Beijing) came available the Maniplus stand-alone application was adapted to run with the CPLS. This implementation was never taken into production because there was no need for it as the original stand-alone implementation is still running fine.

To learn more about Blaise 5 a third implementation was created: the CPIX. As Maniplus was not available in Blaise 5 the challenge was to build a Blaise 5 CAPI questionnaire with similar functionality as the Maniplus 4.8 applications.

5. The Maniplus 4.8 implementations

5.1 The stand-alone implementation

Starting the application from the shell a lookup appears with the list of shops and counters indicating the total number of products listed for that shop and how many products already had been completed (see fig 1). As the islands are relatively small the name of the shop is enough for the (experienced) interviewers to know which shop was meant, an address is not necessary. The application is bi-lingual (Dutch/English), the language can be switched by pressing a button on the bottom right hand corner.

Figure 1. The list of shops

Count	Done	R	Name of store
37	0		ANGELO BONAIRE NV
9	0		BARBERSHOP (was Antriol)
156	0		BONAIRE SUPERSTORE (CURAC.)
468	0		BONAIRE WAREHOUSE
62	0		BOOMERANG HARDWARE NV
165	0		BOTIKA KORONA NV
55	0		BOUTIQUE CELINE
49	0		CARCINERIA LATINO NV
20	0		CITY CAFE (COCKT & DREAMS)
67	0		CITY SHOP
12	0		DIERENARTSEN NIKIBOKO
20	0		EL TROPICAL RESTAURANT
34	0		FI AMINGO BOOKSTORF NV

Done: 0 of 2200

NED

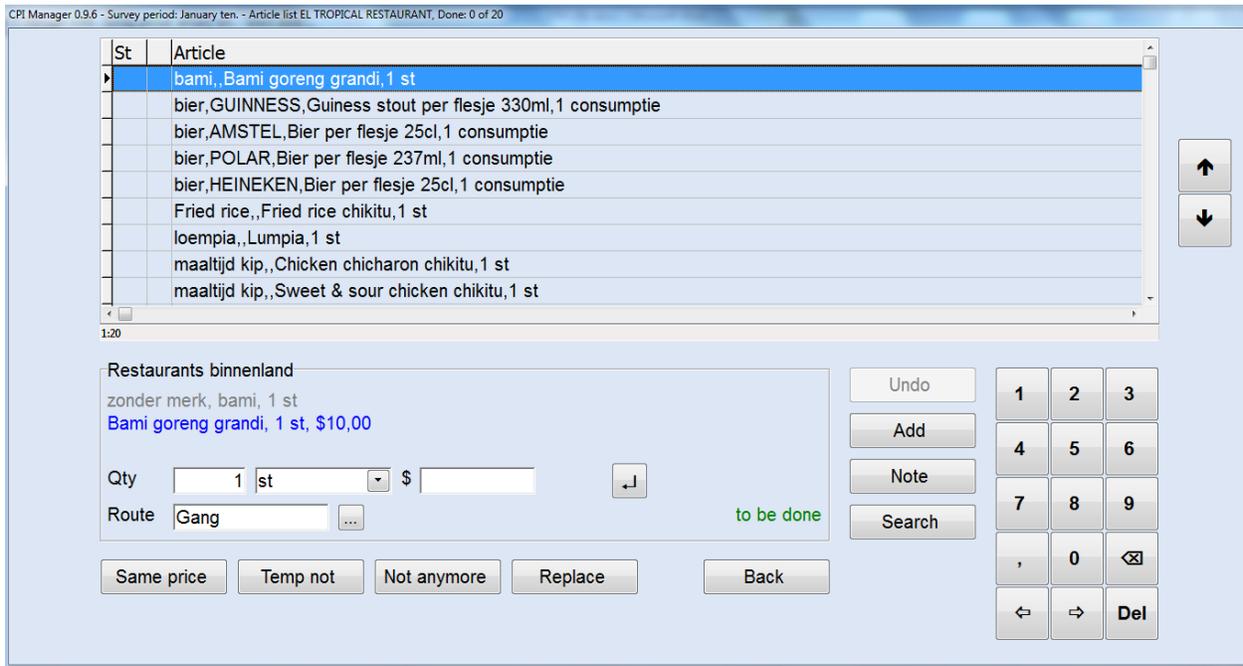
Select Close store Open store Exit

Using the buttons at the bottom of the screen the selected shop can be marked as “closed” or marked as “opened” again. To go to the list of the products of the focused shop the interviewer should click on the “Select” button.

The list of products (see fig.2) contains the type of product, the brand, the volume/weight and the price (in US \$ being the currency of the islands). The first three are shown in a lookup in the dialog, the price is shown with more information below the lookup. The first column (marked “St”) will contain a symbol indicating the change in one of the properties of the product. The interviewer can use input lines or the buttons at the bottom of the screen to report the changes. It is also possible to add a product or make some notes.

The product list is sorted according to the best route through the shop. The interviewer can edit the location of the product if the shop has changed it. As it is running on a tablet without a separate keyboard a small numeric keyboard was added to the dialog.

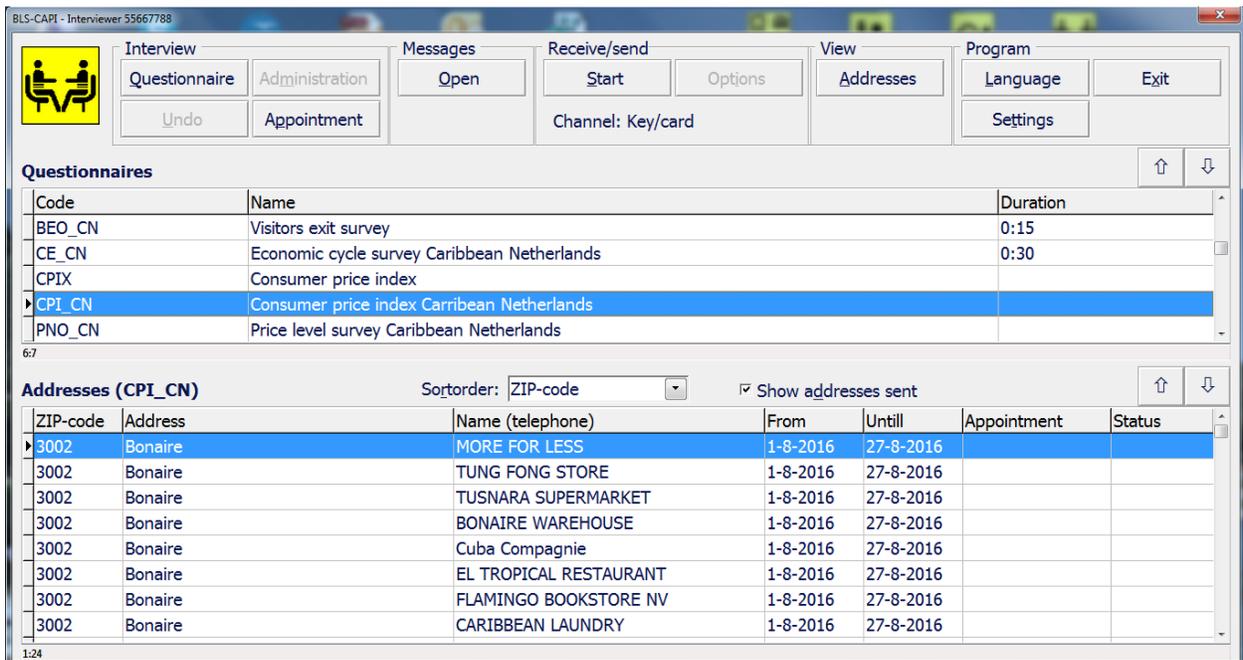
Figure 2. The list of products



5.2 The CPLS implementation

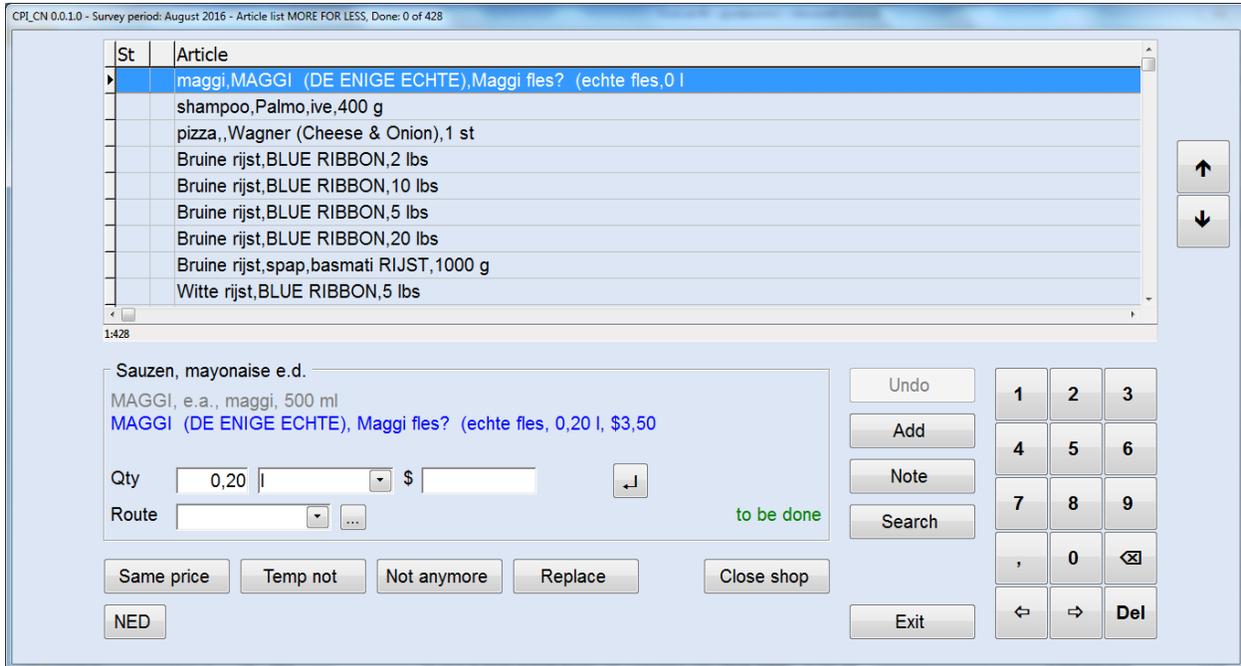
The main difference between the CPLS implementation is that the list of shops has moved to the list of survey cases (addresses) in the main window of the CPLS (see fig. 3).

Fig 3. The CPI in the main window of the CPLS



After opening the instrument with the button “Questionnaire” (in the top left of the window) the product list appears in a screen very similar to that of the stand-alone implementation (see fig. 4). Its functionality is also the same.

Fig 4. The list of products in the CPLS



Another difference between the Maniplus implementations is that the export of completed results of the previous month and the import of a new list for the next month is done automatically by the data communication process of the CPLS. In the stand-alone solution separate means are used to download and upload the files with data to the office in the Netherlands.

6. The Blaise 5 CPIX implementation

The CPIX implementation was also created for the CPLS. As a consequence the list of shops is now shown as the list of addresses in the main window. The real challenge was the product dialog including the product list and the sub-dialogs connected to it. To be able to offer similar functionality as in the Maniplus implementations an alternative had to be found for these dialogs working within a Blaise 5 instrument. The choice was made to use a table embedded in the text of a field with integrated images containing actions. This table as part of the field text (see fig. 5) could be generated on the fly making it very flexible. Images have been chosen instead of buttons because symbols could be used making these type of “buttons” very compact and language independent (see fig. 6). Furthermore the use of text fills in this so called inline table makes it even more flexible. Leaving these referred fields empty will result in empty space on the screen. This is e.g. used with the action of closing a shop (see fig 7).

With the buttons on the top of the screen you can navigate through the product list. The products are pre-filled in an array of blocks in the record. The trick behind it is that the OnClick events of these buttons (see fig. 8) contain actions that are filling fields with a start value. In the rules the products matching these values are stored in the fields connected to the text fills (“^{aVariant[x]}”) in the inline table. Clicking on the arrows “Up” or “Down” the list will focus on the previous or next product yet to be edited (status is empty). The navigation buttons on the right-hand side are defined in the resource library and are disabled

because the questionnaire does not consist of consecutive screens between which the interviewer can browse.

Figure 5. The inline table

```

679 aVariants
680 "<table><column width=525><column width=*><row><cell>^ShopName<br>- ^{aHeader}</cell>
681 <cell>^{aNavigation}</cell></row></table>
682 <table><column width=20><column width=*><column width=70><column width=225>
683 <row background=#FFECECEC><cell>^{aStatus[1]}</cell><cell>^{aVariant[1]}</cell><cell>^{aLocation[1]}</cell><cell>
684 ^{aImg1[1]} ^{aImg2[1]} ^{aImg3[1]} ^{aImg4[1]}</cell></row>
685 <row><cell>^{aStatus[2]}</cell><cell>^{aVariant[2]}</cell><cell>^{aLocation[2]}</cell><cell>
686 ^{aImg1[2]} ^{aImg2[2]} ^{aImg3[2]} ^{aImg4[2]}</cell></row>
687 <row background=#FFECECEC><cell>^{aStatus[3]}</cell><cell>^{aVariant[3]}</cell><cell>^{aLocation[3]}</cell><cell>
688 ^{aImg1[3]} ^{aImg2[3]} ^{aImg3[3]} ^{aImg4[3]}</cell></row>
689 <row><cell>^{aStatus[4]}</cell><cell>^{aVariant[4]}</cell><cell>^{aLocation[4]}</cell><cell>
690 ^{aImg1[4]} ^{aImg2[4]} ^{aImg3[4]} ^{aImg4[4]}</cell></row>
691 <row background=#FFECECEC><cell>^{aStatus[5]}</cell><cell>^{aVariant[5]}</cell><cell>^{aLocation[5]}</cell><cell>
692 ^{aImg1[5]} ^{aImg2[5]} ^{aImg3[5]} ^{aImg4[5]}</cell></row>
693 <row><cell>^{aStatus[6]}</cell><cell>^{aVariant[6]}</cell><cell>^{aLocation[6]}</cell><cell>
694 ^{aImg1[6]} ^{aImg2[6]} ^{aImg3[6]} ^{aImg4[6]}</cell></row>
695 <row background=#FFECECEC><cell>^{aStatus[7]}</cell><cell>^{aVariant[7]}</cell><cell>^{aLocation[7]}</cell><cell>
696 ^{aImg1[7]} ^{aImg2[7]} ^{aImg3[7]} ^{aImg4[7]}</cell></row>
697 <row><cell>^{aStatus[8]}</cell><cell>^{aVariant[8]}</cell><cell>^{aLocation[8]}</cell><cell>
698 ^{aImg1[8]} ^{aImg2[8]} ^{aImg3[8]} ^{aImg4[8]}</cell></row>
699 <row background=#FFECECEC><cell>^{aStatus[9]}</cell><cell>^{aVariant[9]}</cell><cell>^{aLocation[9]}</cell><cell>
700 ^{aImg1[9]} ^{aImg2[9]} ^{aImg3[9]} ^{aImg4[9]}</cell></row>
701 <row><cell>^{aStatus[10]}</cell><cell>^{aVariant[10]}</cell><cell>^{aLocation[10]}</cell><cell>
702 ^{aImg1[10]} ^{aImg2[10]} ^{aImg3[10]} ^{aImg4[10]}</cell></row>
703 </table>"
704 : SET OF ( yes),EMPTY,NODK,NORF

```

Figure 6. The list of products in the CPIX

The screenshot shows the 'Consumer price index' interface. At the top, there is a green header with the text 'Consumer price index'. Below this, a navigation bar contains several icons: a left arrow, a double left arrow, a left arrow with a vertical bar, a right arrow with a vertical bar, a right arrow, a double right arrow, a magnifying glass, and a lock icon. Below the navigation bar, the text 'MORE FOR LESS' is displayed, followed by '- commodity 1 - 10 of 428, done 0'. The main content area is a list of products, each with a description, a price, and a set of four icons (equals, minus, X, and three dots). The products listed are:

- maggi, MAGGI (DE ENIGE ECHTE), Maggi fles? (echte fles, 0,2 l, \$3,50)
- shampoo, Palmo, ive, 400 g
- pizza, , Wagner (Cheese & Onion), 1 st
- Bruine rijst, BLUE RIBBON, 2 lbs, \$1,61
- Bruine rijst, BLUE RIBBON, 10 lbs, \$8,80
- Bruine rijst, BLUE RIBBON, 5 lbs, \$4,40
- Bruine rijst, BLUE RIBBON, 20 lbs, \$17,05
- Bruine rijst, spap, basmati RIJST, 1000 g
- Witte rijst, BLUE RIBBON, 5 lbs, \$3,85
- Witte rijst, BLUE RIBBON, 2 lbs, \$1,53

On the right side of the screen, there are several icons: a red X icon, a left arrow, a right arrow, a keyboard icon, and a flag icon showing the Netherlands, the United Kingdom, and the Euro symbol.

At the bottom left, there is a logo for 'cb' and the text '© 2015 Centraal Bureau voor de Statistiek'.

The number of products shown on one page has been limited to 10 without any real reason, it could have been more or less too. The navigation line on the top of the screen also contains a button to search for a product or to close the shop. A closed shop can be re-opened again. The mechanism works with a field set

to “0” (opened) or “1” (closed). Beneath the top line there is a line indicating which set of the products (“commodity”) is shown and how many are already completed (“done”).

Figure 7. Closing the shop



Figure 8. The navigation field with OnClick events

```

899 IF ShopClosed <> 1 THEN
900   aNavigation := '<img source=FirstSmall OnClick="{Action AssignField(aNr, \'-6\')}">'+
901     '<img source=PreviousSetSmall OnClick="{Action AssignField(aNr, \'-5\')}">'+
902     '<img source=PreviousSmall OnClick="{Action AssignField(aNr, \'-4\')}">'+
903     '<img source=JumpUp OnClick="{Action AssignField(aJump, \'-1\')}">'+
904     '<img source=JumpDown OnClick="{Action AssignField(aJump, \'-1\')}">'+
905     '<img source=NextSmall OnClick="{Action AssignField(aNr, \'-3\')}">'+
906     '<img source=NextSetSmall OnClick="{Action AssignField(aNr, \'-2\')}">'+
907     '<img source=LastSmall OnClick="{Action AssignField(aNr, \'-1\')}">'+
908     '<space count=3><img source=Search OnClick="{Action GotoURI(\'+REPLACE(aKBopen,\'\\\',\'\\\\\')+
909     '\');AssignField(SearchVariant.aSearchNext,\'\');SetParallel(\'+SearchVariant\')}">'+
910     '<space count=4><img source=Lock OnClick="{Action AssignField(ShopClosed,\'-1\')}">'+
911 ELSE
912   aNavigation := '<img source=Unlock OnClick="{Action AssignField(ShopClosed,\'\');AssignField(aNrCur,\'\')}">'+
913   aNrCur := 0
914 ENDIF

```

On every line a product is shown. In the column behind it the location of the product in the shop is listed. This is the field “Route” in the Maniplus implementations. The OnClick event of the 4 buttons at the end of each line contain actions to edit the product as in the Maniplus implementations: [=] for [Same price], [-] for [Temp not] and [X] for [Not anymore]. The symbol on the button will appear in the first column of the line. With the fourth button [...] a new screen is opened to edit the product. This can lead to new symbols: [>] for more expensive, [>>] for much more expensive, [<] for less expensive, [<<] for much less expensive, [~] for changed but not possible to determine if the product is more or less expensive and [#] in case only the description has changed (see fig. 9).

The edit screen is created as a parallel. This parallel contains the array of blocks with the pre-filled products and fields to store the changes. When activated through the edit button [...] in the main window the parallel will only show the data of the related product. This is done by looping through the array and only showing the fields if the array number matches the product sequence number. If the numbers do not match the fields are opened with “KEEP” and therefore not visible:

Listing 1. Implementation Source Code

```
FOR x := 1 TO 999 DO
  IF x = aProdNr THEN
    Commodity[x].ASK(x)
  ELSEIF x <= NrMax THEN
    Commodity[x].KEEP(x)
  ENDIF
ENDDO
```

The field NrMax prevents that the loop goes beyond the maximum number of products. In the instrument a maximum of 999 products per shop is supported.

Figure 9. Status symbols



In the edit screen (see fig. 10) all kind of properties of the product can be changed. As several of these properties are descriptions or names or numeric values an on-screen keyboard was needed as the instrument should be able to run on a touch screen tablet without a keyboard. Currently it is not possible to create an on screen keyboard within a Blaise 5 questionnaire. This has to do with the different controls used for different type of fields. Instead of an integrated keyboard a C# program was developed based on the WOSK project using an XAML definition for the keyboard. Included in the OnClick event of the edit button [...] it opens automatically with the edit screen using a GotoUri action.

To get several fields on the same line the “grouping” function is used to create a table for that line. To organize the fields “Brand” and “Variant” being next to the field “Notes” a table within a table was created (see fig. 11). The fields in the edit screen are all critical. As soon the focus jumps to another field the rules are invoked so that a line with changes in red is shown. At the end of this line the status symbol that will be filled in the first column in the main screen is added. The green rectangle image with an arrow under the field “Notes” activates the action “Save”. It forces the instrument to make a round trip to the server so all the fields are updated.

Figure 10. The edit screen with the keyboard

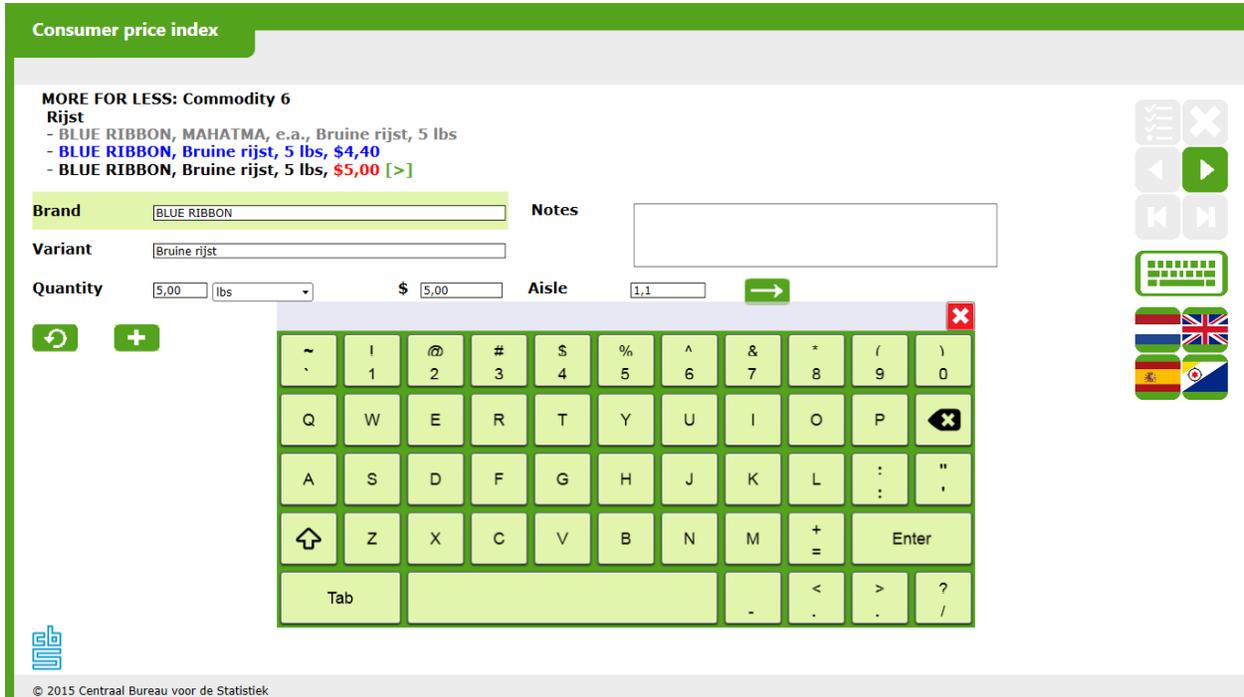


Figure 11. A table within a table



The unit of the quantity is store as a dropdown list. There are 7 groups of units of more or less the same type (e.g. weight, liquid, length, etc.). The texts are imported from an external file based on the number of the group:

Listing 2. Group Source Code

```
FOR x := 1 TO 7 DO
  FOR y := 1 TO 20 DO
    IF extUnits.SEARCH(STR(x)+STR(y,2)) THEN
      extUnits.READ
      IF SUBSTRING(Product.Var_Unit,1,1) = SUBSTRING(extUnits.TagStr,1,1)
THEN
        aUnit[y] := extUnits.ValStr
      ENDIF
    ENDIF
  ENDDO
ENDDO
```

The primary key of the external consists of a 3 digit number. The first digit is the group number. It is only possible to change the unit to one available in the group. It is e.g. not allowed to switch from kilograms to centimeters.

The curled arrow image contains an action to undo all changes. It sets the value of the auxiliary field “aResetStatus” to 1. This value causes in the rules to activate 3 fields “aConfirm”, “aOK” and “aCancel” in the place of the [+] field (called “aNew”) which is used to create a new product to replace the current one:

Listing 3. aResetStatus Source Code

```
aReset.ASK
IF aResetStatus = 1 THEN
  aConfirm.ASK
  aOK.SHOW
  aCancel.SHOW
ELSE
  aNew.SHOW
ENDIF
```

The field “Confirm” is a so-called QuestionTextOnly field displaying a text asking for confirmation. The two fields “aOK” (image [✓]) and “aCancel” (image [X]) are images with actions in the OnClick event changing the value of the field “aResetStatus”. Based on this value the original pre-filled values of the product are shown again by emptying the fields with the changes.

When the field “aNew” (image [+]) is clicked the value of a field called “aNewStatus” is changed to 1 using the AssignField action in the OnClick event. This value causes in the rules to increment the maximum number of products with 1 at the end and focus on this product which has still empty properties. After filling in the properties of the product and clicking on the field “aOK” (image [✓]) the value of the field is changed to 2 using the AssignField action in the OnClick event and in the rules the array of products (now with one extra product) is sorted on the location field using the EXCHANGE function of Blaise.

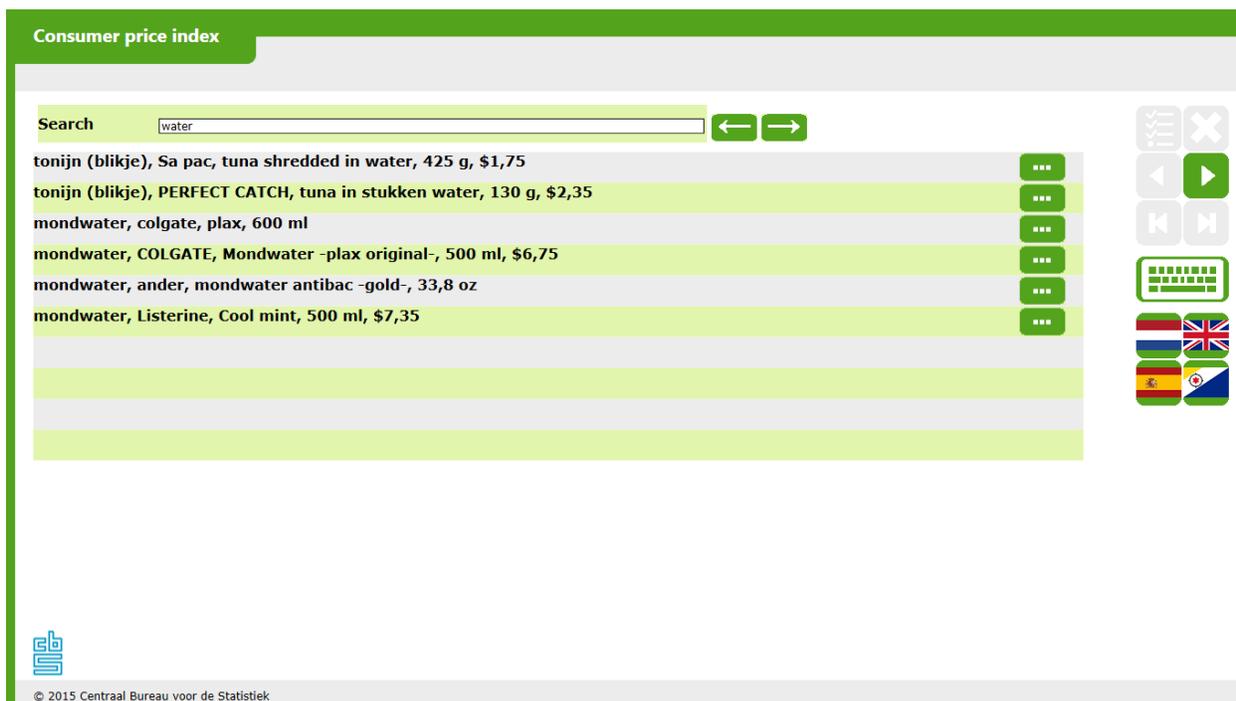
The location of the product in the edit screen can also be changed using the text field with the label “Aisle”. When changed in the rules the array with products is sorted on that text using the EXCHANGE function.

Clicking on the arrow button on the right hand side of the screen will make the main screen appear again using the NextPage action. It focuses on the product that has been relocated by assigning the new

sequence number of the product to the field which is used by the main screen to load the subset of products into the table. This (image) button is defined in resource library together with the buttons for the keyboard and the languages. Using different resource sets these buttons can also appear on the left hand side of the screen for left-handed interviewers. This is controlled by a setting in the CPLS. The CPIX is only designed to use in landscape position although it is possible to create a layout for portrait too.

It is also possible to search for the product based on a text in the description. Therefor a search screen was created that is also defined as a parallel (see fig. 12). Clicking on the magnifying glass in the main screen will jump to the search screen using a SetParallel action. After filling in the search text a click on one of the arrow buttons will cause the table to be filled with the products which description matches the search text. With the two arrow buttons you can browse (up and down) through the screens if there is more than one. To jump to a product it suffice to click on the [...] button next to the product. The OnClick event contains actions which will close the on screen keyboard with a GotoUri action that invokes a small C# program that kills the task (in case the keyboard was opened), assign the sequence number of the product to the field which is used by the main screen to load the subset of products into the table and a SetParallel action to jump to the main parallel.

Figure 12. The search screen



7. Conclusion

With Blaise 5 it is now possible to create very exotic instruments. Although not always easy the CPIX proves the richness and power of Blaise 5. The only things you need are a creative mind and a solid knowledge of the Blaise language and layout functions. The creative mind is given to you by birth ☺ (or not ☹), the solid knowledge is a matter of learning and practicing.

Converting a Blaise 4.8 Survey Instrument for Tablet Use

Caroline Donegan, Conor MacDomhnaill, and Michelle Keniry, Central Statistics Office, Ireland

1. Abstract

In 2015 our field interviewer's laptops were reaching end of life. The dilemma we faced was to buy more laptops or look at alternative devices that were more suitable for mobile working. The focus of this paper is our experiences adapting our Blaise 4.8 survey instruments and management user interfaces to make them suitable for tablet use. There are three main aspects we will cover, firstly the technical challenges we faced with development and design, these include the requirement of a keyboard for text input, coding and searching in 'Browse' view and the problems encountered when system keyboards interacted with some Blaise functionality. The options and solutions implemented to cater for both right and left hand users and decisions on font and button images. Secondly, we will look at the expectations, challenges and training issues encountered by the field interviewer and how their feedback and interactions were invaluable when determining the scale and priority of improvements required for future development. Thirdly, our request for some functionality change to CBS Blaise developers and how Blaise 4.8.5 addresses a number of the challenges we faced.

2. Introduction

In 2015 our field interviewer laptops would reach end of life. The dilemma the organisation faced was to purchase more laptops or look at alternative devices that were more suitable to mobile working.

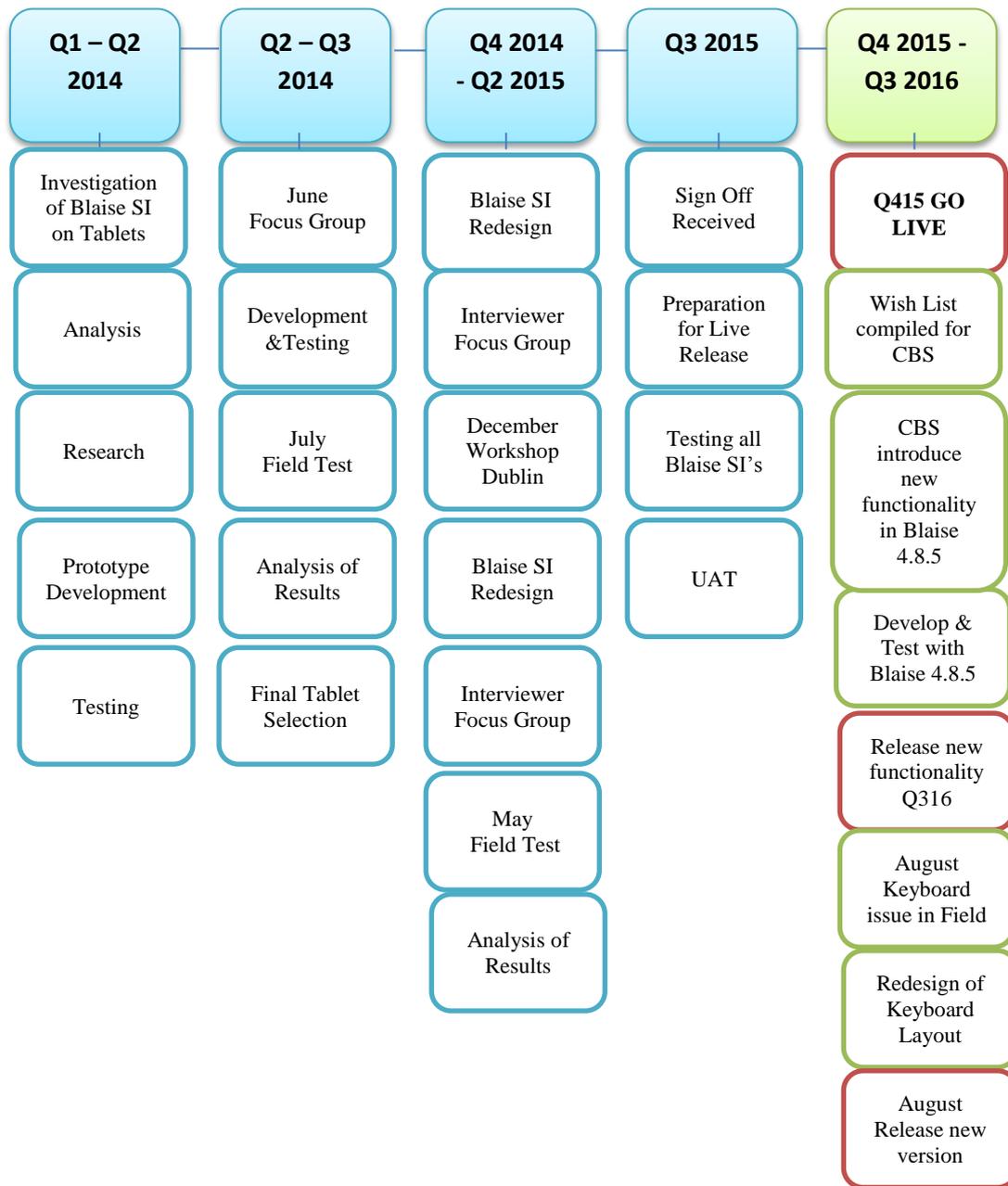
Our initial planning phase highlighted some of the risks and impacts that had to be considered and resolved before we could proceed:

- device specifications –windows device required for Blaise
- resource availability
- impact on data and interviewing techniques
- conversion of Java UI and Blaise survey instruments for touch screen device
- move from Windows XP and 7 operating systems to Windows 8.1 - its impact on our existing applications
- other organisations transitioning experiences

The CSO was also committed to the Labour Force Survey (LFS) transformation project. This is a project of significant scale involving a complete rewrite of the Labour Force Survey to introduce new mode CATI, improvements to survey design and a Case Management application. Due to the resources and timeframe required by the transformation project a stipulation that minimal changes to existing applications and questionnaire design was applied to the tablet redevelopment. Our focus for redevelopment work would include the Quarterly National Household Survey (QNHS), Survey on Income and Living Conditions (Silc) and Household Budget Survey (HBS).

Early discussions with the business areas raised a number of issues. Most notable were the availability of external keyboards, design of survey instrument including screen orientation, text versus images on buttons as well as how the User Interface (Java UI) should be redesigned. This paper will look at some of the planning, technical challenges, expectations and functionality changes required to convert Blaise SI's conducted on laptop to touch screen device for survey data collection.

3. Time Line for Development



4. Initial Planning

Investigations and development work began in early 2014. The Dell Latitude 10 Pro Tablet with operating system Windows 8.1 was configured for the purpose of test and review of all required applications. Universal standards in fonts, papers on tablet design and developments in the wider Blaise community were researched.

One of the existing CAPI Surveys (QNHS) was redeveloped using a template design from CBS Netherlands. A button panel menu file replaced the previous dropdown format of the laptop. Laptop

layouts were reconfigured for tablet display and question and information text referencing actions required using a laptop where updated to reference tablet actions.

The laptop Java UI consisted of a series of inter linked dropdown menus. They were difficult to navigate on a touch screen device and not suitable for use with a stylus (*Figure 1*). A tile format similar to the tablets own window tile display was applied. The position of the tiles were categorised to the previous dropdown order. Also a one tap access was applied to each function (*Figure 2*). A new Case Management System was being developed for LFS survey through IBM Notes (*Figure 3*)

Figure 1. Java UI Laptop for QNHS and Silc

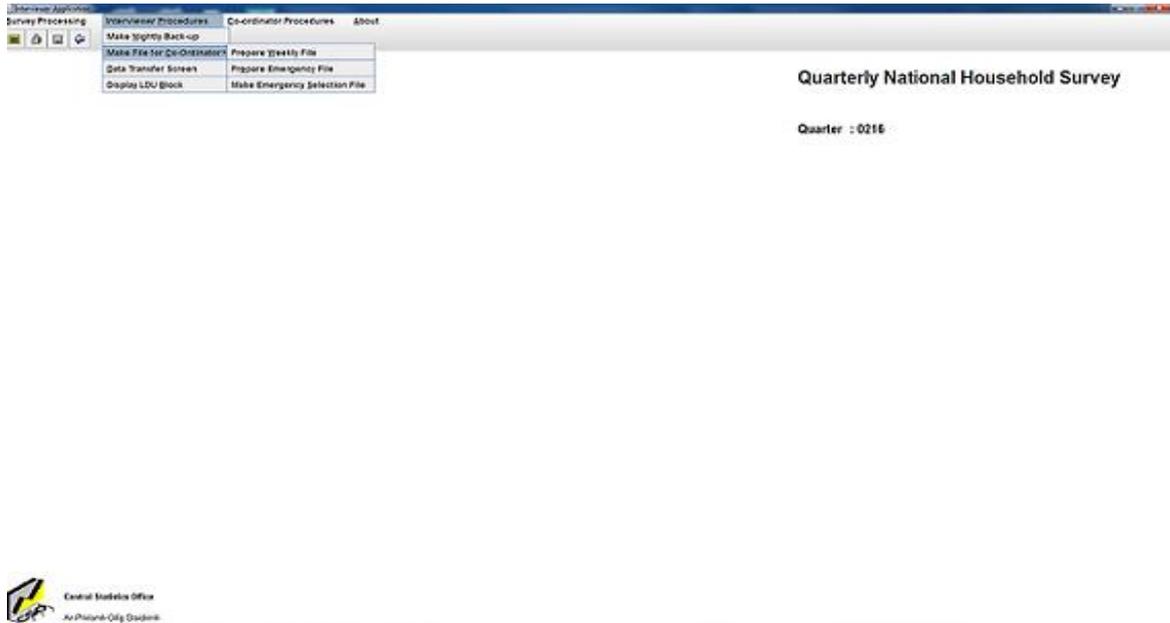
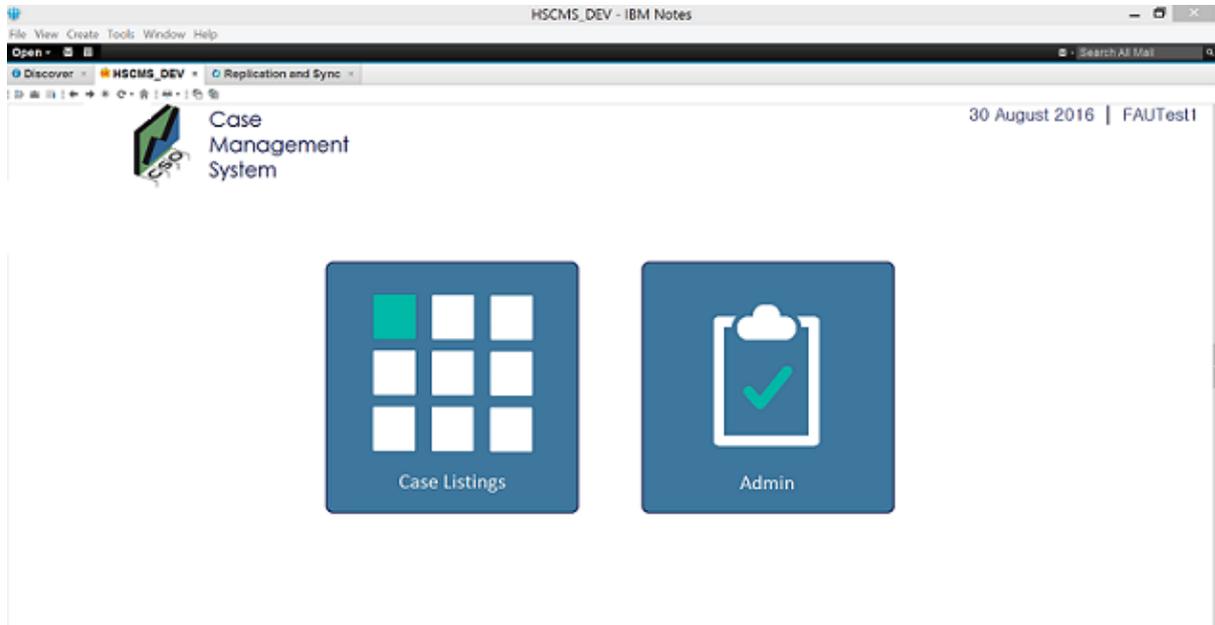


Figure 2. Java UI Tablet for QNHS and Silc



Figure 3. IBM Notes Case Management System Tablet for LFS



There were a lot of discussions on the overall design of the questionnaire for tablet however where consensus could not be reached was the layout and images of the buttons on the button panel menu file. For testing purposes we developed two separate button panel menu files, one consisting of text and numeric descriptions (*Figure 4*) while the other consisted of images using bitmaps, Blaise icons and Windings font (*Figure 5*).

Figure 4. Text Descriptions



Figure 5. Images

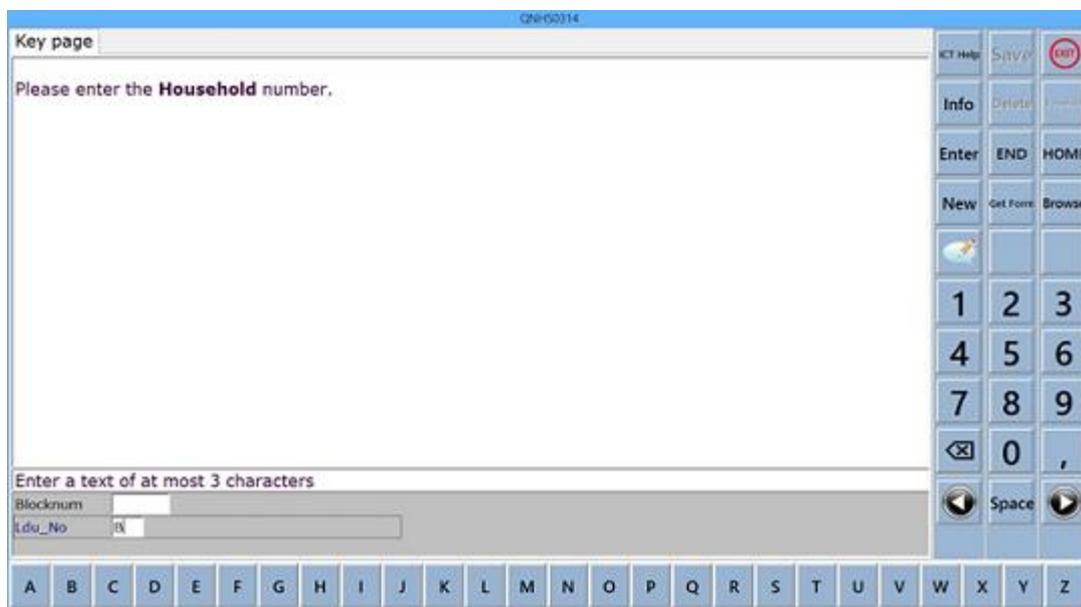


We encountered significant difficulties sourcing a keyboard which would interact with dialogues and not mask text input. Some of the difficulties included:

- When actioned dialogues would not interact with the tablet system keyboards.
- With the layout of our questionnaire, the tablet keyboard (Tabtip.exe) covered the Grid when inputting text.
- Accessing the keyboard for a search in browse view could not be automated.

To mitigate these issues, we created an additional button panel of Alpha A-Z (*Figure 6*). This activated at the bottom of the screen when text input was required. Lookup dialogues would not allow the input of more than one character for a search. The solution implemented was to recompile lookups to default to Alfa search for questions where descriptive text was unavailable for direct read. For the browse search the tablet keyboard (Tabtip.exe) was accessed through a number of steps from the tablet charms bar

Figure 6. Additional button panel of Alpha A-Z



5. Initial Focus Groups and Field Test

In the summer of 2014 a focus group of six interviewers was formed and a field test conducted. Interviewer's interactions with the applications, button bar, as well as general tablet use were evaluated and we found some aspects were not conducive to doorstep interviewing.

Feedback included:

- Current tablet screen too small for interviewing.
- Change of font style and size from the existing laptop settings Courier New Baltic (Form pane 10, Rich text 12) to the recommended Verdana font (Form pane 11, Rich text 16) was easier to read on screen.
- Scrolling was present in questions and answer types containing large volumes of text. Could lead to answer options being missed thus affecting data return. Layouts, font and information instructions would need review.
- Text descriptions on the button panel menu file were deemed too cluttered and illegible. A simple self- explanatory image would be more conducive to the nature of the work.

- Button panel menu files should have a consistent layout across all surveys. Position memory would be very important for hand-eye coordination preventing errors and enabling the interviewer to maintain eye contact with participants.
- Use of a stylus for interviewing was the preferred option. Buttons should be of sufficient width for a larger stylus.
- Portrait orientation allowed additional space for keyboards and large answer types but was cumbersome and not suitable as a hand held device for long interviews.
- Button panel menu files developed for right positioning only. We had not considered left hand persons interaction with right positioning. It was awkward and not user friendly.
- The increase in display text on the Browser from laptop setting Ms San Serif 8 to Verdana 16 was clear on screen and suited the width of the stylus for selection. Scrolling required when viewing blocks of work, meant being prone to select an incorrect record.
- Interviewers did not use the search facility in the Browser during the test because accessing the keyboard from the charms bar was frustrating and cumbersome.
- Coding in Alfa search was a big change to work practice. Interviewers found the process very time consuming and frustrating for both interviewers and participants as it was easier to lose position in the file.
- All Interviewers particularly touch typists disliked the Alpha keyboard (*Figure 6*). The position and format of a straight line at the bottom of the screen was not suitable for text input. All agreed that if the Alpha keyboard was the only solution to a keyboard within the questionnaire an external keyboard would have to be provided.
- Existing help application (RoboHelp with Java Interface) did not work well with touch screen devices. The application consisted of drop down functionality and did not fit correctly to screen.
- The Java UI tile structure was a good design however there was an increase in the instance of locked files with the Blaise SI. It was determined this was caused by multiple taps of the Open Questionnaire tile. Interviewers requested an egg timer or prevention measure to prevent locked files being generated.

Analysis of the data collected identified no inconsistencies between laptop and tablet devices. The results of the tests determined that using the tablet device and the applications in their current form were not suitable for CAPI interviewing. Due to the given timeframe a decision was made to exclude the Household Budget Survey from tablet redevelopment and remain with laptops.

6. Planning for Second User Acceptance Testing

IT teams began redeveloping aspects of the applications and tendering for tablets and cases suitable to mobile working. A number of tablets were sourced for investigation and a focus group again conducted with the Fujitsu Windows 8.1 Pro 32 Bit (10.1 screen) and Dell Venue 11 Pro Windows 8.1 32 Bit (10.8 screen).

Table 1. Sample of testing results

Fujitsu 10.1	Dell Venue 11 Pro 10.8
10.1 screen Resolution: 2560 x 1600 Due to increase in screen height all layouts and Menus would require redesign	10.8 screen Resolution : Screen size closer to the look and feel of the laptop view. Minimum redesign of layouts and menus required.

Active stylus not user friendly	Would require robust stylus with some form of attachment.
Tablet white in colour does not work well with current colour scheme of questionnaire or email. Hard on the eyes. Would require a darker screen surround or redesign of questionnaire colour scheme.	N/A
Ports water proof	Case with port protection required

After multiple rounds of testing The Dell Venue 11 Pro Windows 8.1 (10.8 screen) with larger screen was selected.

7. Questionnaire Redesign

The button panel menu files were redesigned using bitmaps and symbol images from specific open source fonts (Icon Works, Windings and Heydings). The cost of fonts had to be taken into account as 100+ tablets would be issued to the field. Buttons were categorised and those most frequently used placed in a position to limit wrist movement. The layout of panels was standardised across all surveys (Figure 7).

Left position button panel menu files were added for left hand users. A Maniplus menu was developed to input the selection to an external datamodel which could be read into all survey questionnaires. Conditions set in the button panel menu files then determine which position panels should invoke. Flexibility was required in the selection to allow tablet exchange in the field.

The Alfa coding system remained and a specific coding and comments button panel menu was developed in the style of the main button panel menu files (Figure 8). An access control button panel menu file panel was also required when the rules were not invoked. For example when Browse view is cancelled, the interviewer requires options to exit or return to the Blaise questionnaire (Figure 9).

Figure 7. Main Button panel



Figure 8. Coding Button panel



Figure 9. Access Control Button panel



Access to keyboard for text input was still a technical issue to overcome. We had hoped to develop a keyboard which would activate when required, work with all dialogues and allow control within the Blaise SI. As a solution we included a button on the button panel menu which used a manipula script to launch the tablets on-screen keyboard (osk.exe). While the keyboard had to be manually opened

and closed there were options to move the keyboard up and down the screen, fade to review question text and resize.

In December 2014 a workshop was held in Dublin Ireland. Those in attendance were Gerrit de Bolster and Peter Sinkiewicz (CBS Statistics Netherlands), Karl Dinkelmann (University of Michigan), Johnny Wallace (NISRA Northern Ireland), and Jacqueline Hunt, Conor MacDomhnaill, Edward Dwane, Garry Dunphy, Michelle Keniry (CSO Ireland). The group discussed their experience with mobile devices and looked at development work being carried out in each organisation.

The NISRA Northern Ireland had been using Blaise with tablet devices for 13years. The positive experience outlined by Johnny Wallace encouraged our move to tablets. The sharing of knowledge and code also enabled us to adapt our button functionality making the execution of the rules more efficient as well as incorporating help text into the existing questionnaires.

Karl Dinkelmann's (University of Michigan) keyboard functionality was incorporated into our button panel menu. Due to the structure of our existing questionnaires overall expressions using \$Basetype could not be implemented, instead Field tags were assigned to relevant String fields.

In January 2015 a focus group consisting of two interviewers was conducted. The result of this test was disappointing. The interviewers felt that as the keyboard did not auto adjust and covered the Formpane especially in a table displaying multiple lines; it would not work in the field environment. The option of the use of the on-screen keyboard would be more suitable to the field. The coding dialogues while not resolved could be released to the field with the solution provided.

8. Second Field Test for User Acceptance Testing

In May 2015 a second field test was conducted over a two week period using live interviews across six regions capturing rural, urban and apartment demographics. This test would not only evaluate the interviewer's interaction with the tablet but also differences highlighted in data returns from both the laptop and tablet. The test included interviewers with varying levels of computer and typing skills and for consistency included one interviewer involved in all previous tests.

A one day training program on tablet and application functionality was provided. Participating interviewers would retain their laptops for the duration of the test, however a tablet version number was included in the Blaise questionnaire to insure that all assigned work was completed on the tablet.

The test was successful and feedback from the majority of interviewers was very positive. There was however some resistance to change especially from experienced touch typists.

Positive Feedback Included:

- The tablet was light weight and easier to interview at the doorstep. The case was robust and the stylus attachment sufficient. The stand adaption of the case was useful for longer interviews. Some interviewers in urban areas felt there could be more vulnerability with the tablet being snatched while those in rural areas felt that the tablet was prone to damage from the elements.
- The font style's, colours and images of the buttons and panels were suitable for day and night time interviewing.
- The inclusion of help within the questionnaire was better
- The Coding Dialogues and on screen keyboard while not entirely suitable were workable.

Negative Feedback included:

- Eye contact and personal interaction with respondents was diminished.
- Blaise questionnaire prone to lock and freeze especially when reviewing work – interviewers tapping next or previous arrows too quickly.
- Interviewers found it difficult to adjust to touch reaction and this caused large number of miskeys. The reaction also caused navigation problems when screen was accidentally touched during an interview.
- Position of next and previous arrows required review to minimize wrist movement
- Browse view required review as Primary key not locked when swiping to review additional fields. Keyboard access cumbersome.
- Other applications posed problems especially access to email and pay.
- Some interviewers reported a loss of familiarity with the questionnaire.

Data analysis showed that the results between laptop and tablet devices were not skewed however the time taken to conduct interviews was increased slightly.

9. Release to the Field

The field test was deemed a success and the tablet in its current form was considered suitable for release to the entire field force. Due to compatibility issues this release was planned to coincide with the release of all quarterly and Bi annual surveys. As a contingency plan laptop versions of the Blaise SI's were also compiled.

The volume of testing involved required a large amount of time and resources. The planning for this testing required significant organisation, not only were routine quarterly tests required but also the testing of all layouts, menus, text changes, routing, field tags, user interfaces and Manipula applications.

A one day training course was delivered to each region before live interviewing began. This day was also used to set up tablets for each user and transfer some data required during the roll-out. A restricted survey properties file containing access to the new quarter of each survey was applied to the tablet. This allowed the interviewer to practice interviewing prior to live release. On the day of live interviewing the interviewers ran an AutoIt* executable to remove the practice data files from their tablets.

**AutoIt -AutoIt v3 is a freeware BASIC-like scripting language designed for automating the Windows GUI and general scripting. Reference AutoITScript.com*

10. Blaise 4.8.5

Once the tablets and new style questionnaire were established in the field we began reviewing the feedback from the 100+ interviewers in the field. Feedback was broken down into categories of what issues being reported frequently by the field force as a whole and those reported as individual cases. We looked at what was relevant to mobile working and questionnaire completion versus individual preferences.

Each item was prioritized and discussions were held on improvements that could be implemented. We also began communication with CBS Statistic Netherlands on the possibility of including some functionality for tablets in Blaise 4.8.4

A wish list was compiled and sent to CBS Statistics Netherlands, our requests included:

- Lookups, Remarks, Browser Shells
Keyboard built into Lookups, Coding, Browser and Make Remarks Dialogue boxes.

Dialogue buttons and scroll bars increased or a possible way of setting these elements through the lookup section in the Modelib info panes.

- Buttons and Panels
Increase size of parallel entry box and buttons on other dialogue boxes (e.g. browser, errors)

Increase and/or provide a way of setting these elements through the Modelib info panes.

Over a number of months Blaise CBS Statistics Netherlands developed a Beta version of Blaise 4.8.5 to include tablet functionality. We began testing and working with CBS to implement the changes while redeveloping our button panel menu file, Run Parameters, BCF files and Modelibs.

The improvements included

- Tablet mode and Scale factor options.
- Keyboard facility contained in the button panel menu file which interacts with the dialogues and browser and auto –adjusts in blocks and tables when the keyboard displays allowing the constant display in Formpane.
- Increase to warning and instruction dialogues making stylus use easier

We redesigned the button panel menu file using the sample style supplied by CBS Netherlands as the colour scheme and bitmap images were uncluttered and fresh. We again reviewed the positioning of each element trying to keep the flow and position of the buttons throughout all surveys (*Figure 10 & 11*).

Figure 10. Blaise 4.8.5 Main Button panel

Button Bar images : Fonts setting in Dep Menu.bmf			
	Help : Caption: i Font :Webdings	Settings Caption: j Font :Icon Works	Exit : Paint image converted to Bitmap
	New Form : Caption: A Font : Icon Works	Get Form: Caption: a Font : Icon Works	Browse Form : Caption : c Font : Icon Works
	Delete Form : Caption: o Font: Icon Works	Home : Caption: Ç Font: Icon Works	End : Bitmap
	Space	Don't Know : Paint image converted to Bitmap	Refusal : Paint image converted to Bitmap
	Space	Calculator Caption : U Font : Heydings Icon	Populate : Caption :1 Font : Icon Works
	1 : Bitmap	2: Bitmap	3 : Bitmap
	4: Bitmap	5: Bitmap	6 : Bitmap
	7 : Bitmap	8: Bitmap	9 : Bitmap
	Backspace : Bitmap	0: Bitmap	Period : Bitmap
	Q : Bitmap	Y: Bitmap	Space

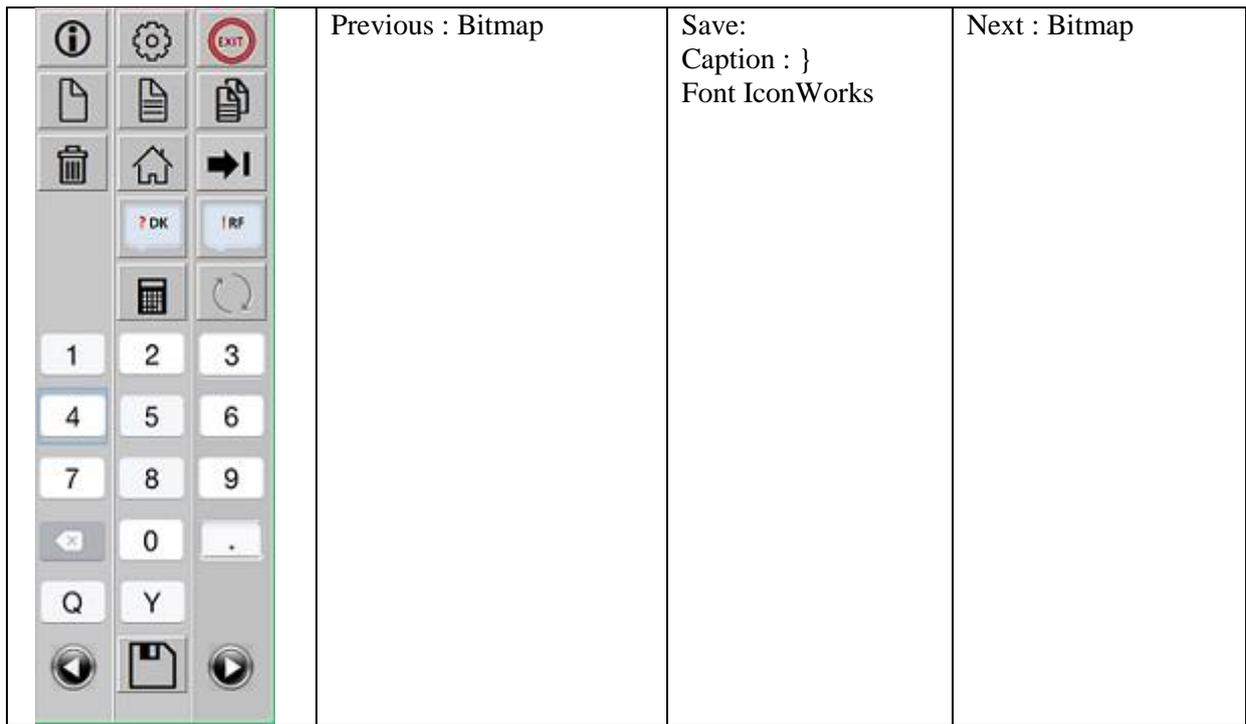
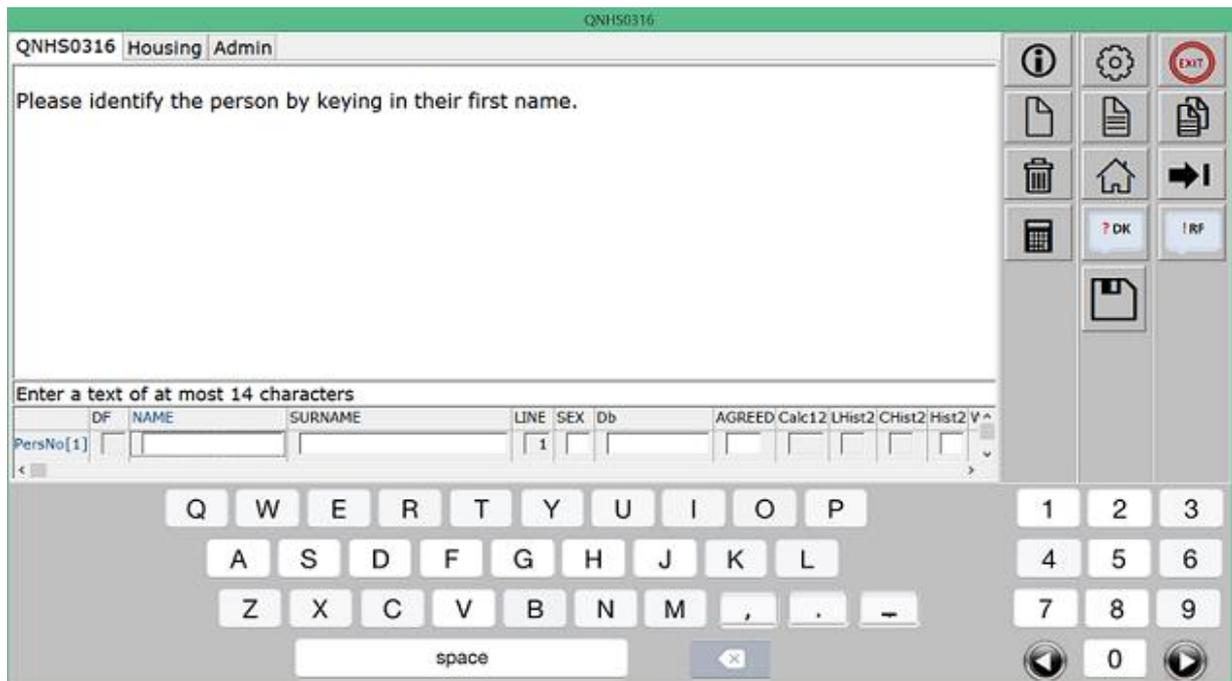


Figure 11. Blaise 4.8.5 with keyboard



The keyboard availability in the coding dialogue (*Figure 12*) allowed us to return to Trigram search and previous laptop work practices. This had been the highest level of complaint from the field force.

Figure 12. Blaise 4.8.5 coding with keyboard

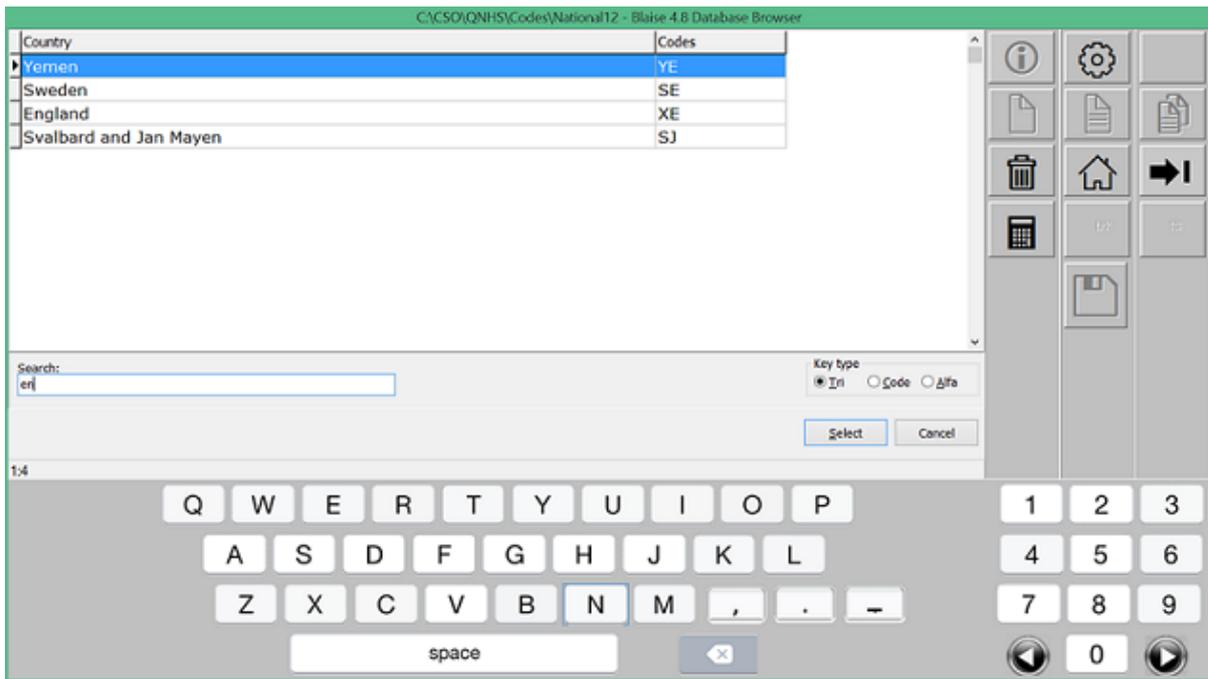
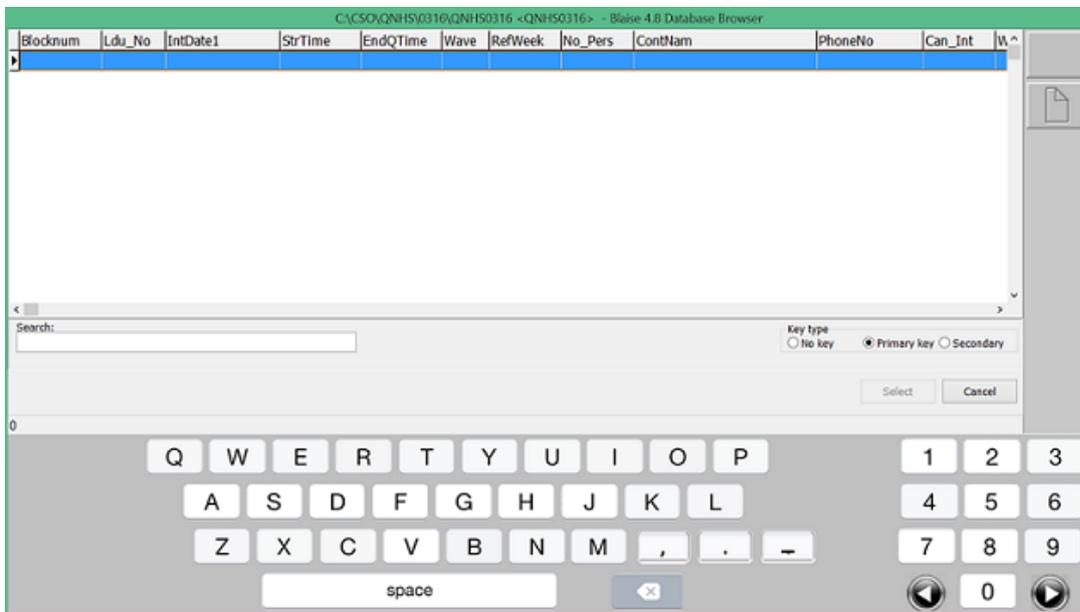


Figure 13. Blaise 4.8.5 browser with keyboard

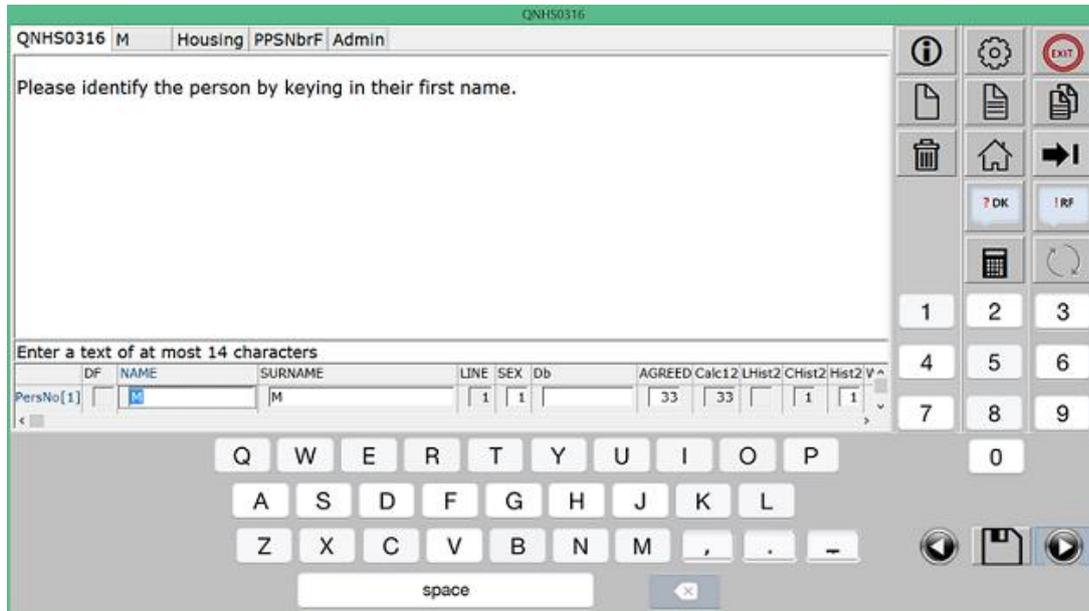


In June 2016 two interviewers tested and approved the changes. These changes went live to the field in July 2016. Interviewers noted an improvement to CAPI interviewing especially when coding. During week 4 – 6 however interviewers reported problems with the new keyboard. The position of the navigation arrows on the main button panel menu file were a fraction higher than the position on the keyboard. Interviewers were inadvertently tapping the 7 or 9 buttons when the keyboard launched

thus overwriting the original entry. They also reported that the questionnaire appeared sluggish when reviewing cases or completing longitudinal interviewers.

Once brought to our attention we understood the problem and a redesign was actioned. This again highlighted the importance of consistent positioning. Redesign entailed removal of numeric keys from the keyboard and placing them in the same position as the main button panel menu file. The navigation arrows were placed in the positions previously held by 7 and 9 (Figure 14).

Figure 14. Blaise 4.8.5 redesign of Keyboard



Regarding the sluggishness we felt that the processing of the questionnaire routing and rules for the button pane menu files were slowing the application and movement from one field to the next. The DepMenu.bmf files were split into survey specific menus and rules. The redesigned menu was released to the field a week later. We have yet to receive feedback.

11. Conclusion

- At the start of this project we hadn't anticipated the amount of resistance we would experience in introducing a new tool for interviewing. Interviewers had become very comfortable with their laptops and some felt that the value of their skill as typists was being diminished.
- While we in the Blaise team were focused on the instrument menus, appearance, layout, impact on routing or data collection, initially interviewers placed a higher emphasis on the equipment that came with the tablet, e.g. stylus, type of case or straps.
- In view of this resistance to change as well as a review of helpdesk requests from interviewers after live release, it became apparent that more interviewer training sessions would have been beneficial. Refresher training would also have been useful for some interviewers.
- The design and usability of the button panel menu is very important. Positioning of the buttons and ensuring consistency across menus prevents errors and redesign at a later date. There were many different opinions on requirements for the various fonts, images and position of buttons etc. We had a limited timeframe available and felt that more time spent at requirements gathering stage would have been useful.
- Again this limited timeframe for our transition to tablet meant that the Blaise Workshop held in Dublin was very beneficial and saved a lot of time when researching keyboards & menus in

particular. This would suggest that this style of meeting would be beneficial within the Blaise community as a whole.

- Likewise work with CBS Statistics Netherlands greatly improved usability and overall experience within Blaise SI's
- In adapting for tablet we were unable to continue using our previous method of calling Help, forcing us to include question help as a second language within the questionnaire. This was a welcome change for the interviewers and was a useful selling point to encourage reluctant interviewers to embrace the changes.
- Although initial issues with the tablet port did have some negative impact. Ultimately the majority of our interviewers are very happy with the change from laptop to tablet.
- Most importantly, after testing our business areas have assured us there was no effect on survey data.

Capturing Signatures Electronically from a Blaise Instrument

Todd Flannery, Ed Dolbow, Michael Kapombe, and Curtis Cooper, Westat, United States

1. Study Background and the Use of Electronic Signatures

For A Large Household Survey (ALHS) Westat interviewed several sample groups residing at the same address including adults, youths and parents. Each interview required a signed consent or assent form and led to an incentive and signed receipt at the completion of the instrument. Consenting adults provided specimens and received a second incentive. Depending on the household composition up to 8 different forms were required in a household and some households had more than one participant in each sample group. It became clear during the field test that the paper management required of the field interviewer and the paper tracking required of the home office was excessive and error prone. We devised an electronic signature approach and integrated it into the Blaise instruments to solve the problem.

We use Blaise 4.8.4 to collect data related to the consent or receipt; call a .NET C# application passing the relevant data; capture the e-signature image; and return control to the Blaise instrument. In addition, we store and encrypt a PDF file containing the statement and signature associated with the participant. Our paper will discuss the process of having Blaise 4.8 administer an electronic e-signature application for the purpose of collecting and documenting these household consents and incentive receipts, the limitations of using a shell-to-EXE application, as opposed to activation of a DLL as an alien router, and adapting the e-signature procedure to Blaise 5 for administering future surveys to this population.

2. The Electronic Signature Form

The e-signature application is a .NET console application written using the C# language to collect signatures. A Blaise instrument validates an array of strings in the main program and launches the e-signature application, passing the array of strings as arguments. The validation process checks that important parameters are not missed before the e-signature form is shown to the respondent. This .NET application was designed mainly using forms with the following controls:

- **Label control:** display text on the form
- **Panel control:** assign frame on the form
- **Textbox control:** capture user signature. This control required using Microsoft.Ink.DLL.
- **Checkbox control:** keep track of type of sample that will be collected.
- **Picture Box control:** save the entire form as GIF picture file on a secure location where it will be encrypted and electronically transmitted to WESTAT.
- **Resource file:** store text strings that are displayed on the form. This control helps customize the E-Signature app for different kinds of signatures for various consents and incentive receipts. In addition it can also be used for images, icons, audio, and video files.

We produced an executable application, external to Blaise, to display one of several forms depending on the Blaise instrument calling the application. The form accepts several parameters that determine text displays. We also display the respondent's selections made in the Blaise Data Entry Program (DEP) as check boxes on the form (see Figure 1).

Figure 1. The .NET E-Signature Form

CONSENT

English Spanish

Please scroll down to the bottom to view the consent form:

- The ALHS Study will use my sample(s) for a variety of tests.
- I will not get results back from the tests done on my sample(s).
- What the risks and benefits are if I give sample(s).
- I can ask more questions at any time.
- I'll get a copy of this consent form.

I agree to give:

A saliva sample.	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> No
A skin cell sample.	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> No
I agree to the use of my samples for genetic research.	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> No

By signing this form, you agree to give a saliva sample now and in the future. You have the right to stop giving samples at any time and may refuse to participate in this or any future sample collections.

I have read the information about giving samples or someone has read it to me. I have had a chance to discuss it and to ask questions. I will receive a copy of this permission form.

Please click the scroll bar to scroll to the bottom

 08/08/2016
Signature of Participant Month/Day/Year

ALHS RESPONDENT
Printed Name of Participant

 08/08/2016
Signature of Person Obtaining Consent Month/Day/Year

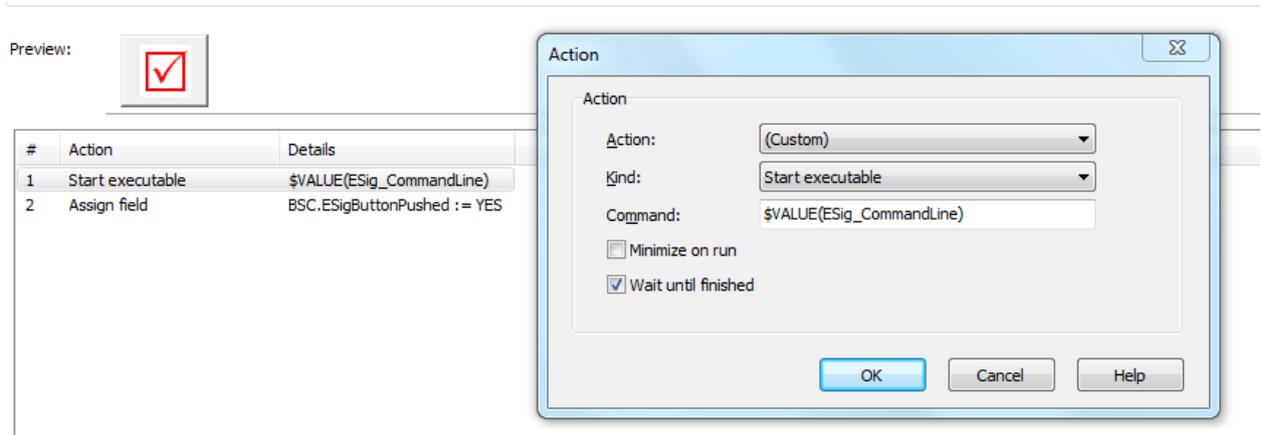
Clear All Signatures Save and Close Close w/o Saving

Using a stylus, touch screen, touch pad, or some other input method, the respondent chooses to accept the displayed selections, sign via the e-signature controls, and “Save and Close” or return to the DEP to alter the selections “Close without Saving”. If the form is saved, the data are validated to ensure that a signature was collected, and a GIF picture file of the electronically signed form is stored in an encrypted location.

3. Modifications in Blaise 4.8

In the DEP, we shell out to our external application, pass parameters, determine what took place, and ensure that we have a saved copy of the document. To accomplish these tasks, we opted to use a customized DEP menu (see Figure 2).

Figure 2. Settings in the DEP Menu



We define in our Panels a raised check button as a visual cue for collecting e-signatures, and we enable it when the respondent has provided the necessary consents (see Figure 3).

Pressing the button starts the external application, passing parameters as a string and setting the focus to the external application until the application is closed. If we are successful in obtaining the signed e-signature document, we disable the button, unless the respondent decides to change their consent in the DEP. We use a hard edit to prohibit the interviewer from moving past this screen without clicking the button.

Figure 3. Screen in DEP for Calling E-Signature Form

Answer Options Help

Consent Skin Cells Saliva Shipment

PRESS THE SIGNATURE BUTTON ABOVE TO HAVE THE SP SIGN THE ELECTRONIC CONSENT FORM.
PRESS 1 TO CONTINUE AFTER THE ELECTRONIC CONSENT FORM HAS BEEN SIGNED.

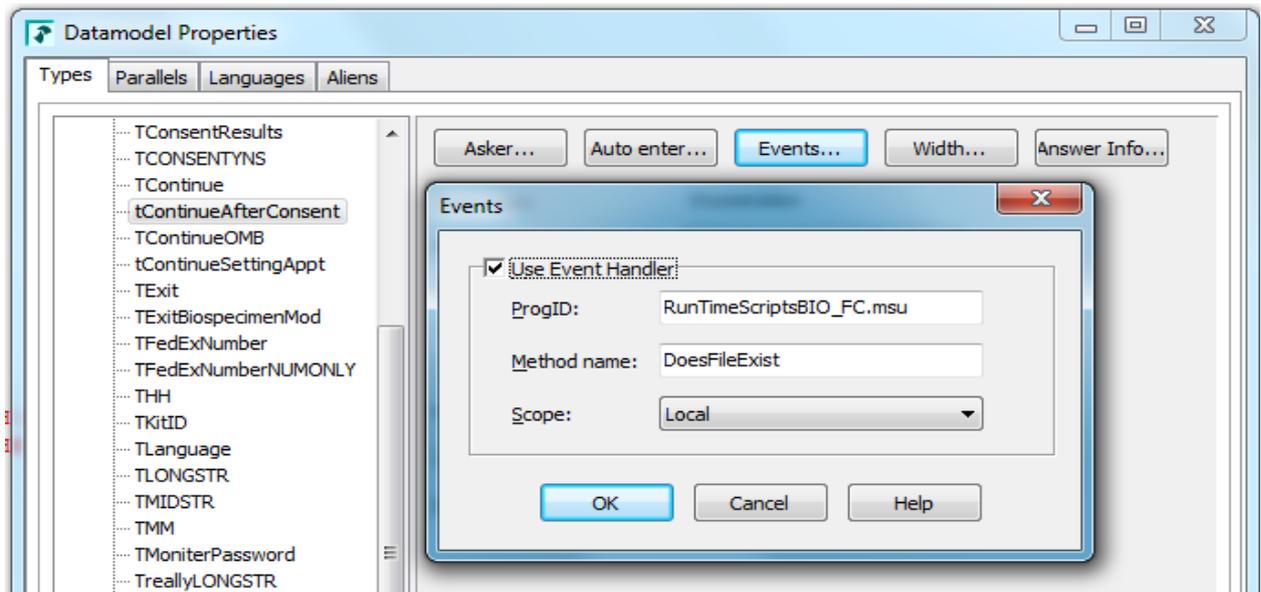
1. CONTINUE

BCT06

BCT06C

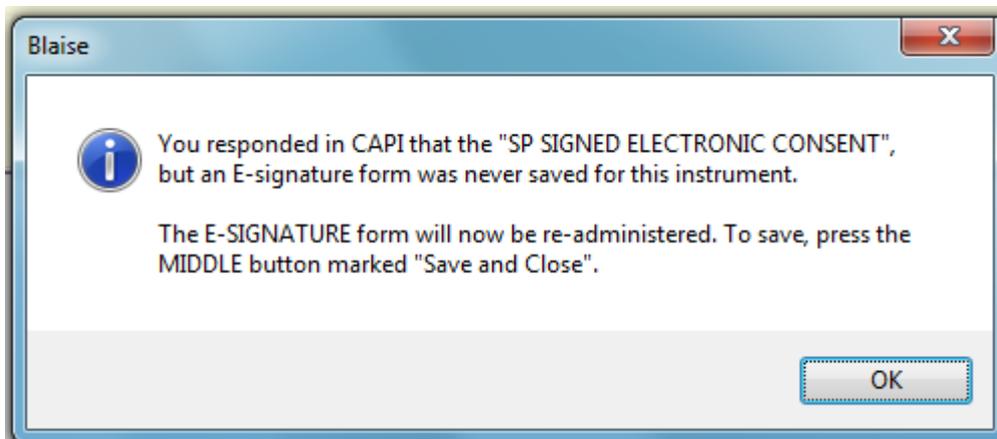
Since DEP is unaware of the result of the e-signature application, we are able to use some Maniplus scripting to determine if the result was successful by detecting that a GIF picture file exists for the captured signature. To accomplish this, we first associate a Manipula procedure as an event in the Datamodel Properties with a predefined TYPE (see Figure 4).

Figure 4. Datamodel Properties Settings



The interviewer is asked in DEP whether or not the respondent signed an e-signature form. A positive response initiates the Manipula procedure. The procedure uses the FILEEXISTS function to ensure that the image of the signed document exists. If there is a discrepancy, an error message is displayed for the interviewer via Maniplus, and the E-Signature module is re-initialized (see Figure 5).

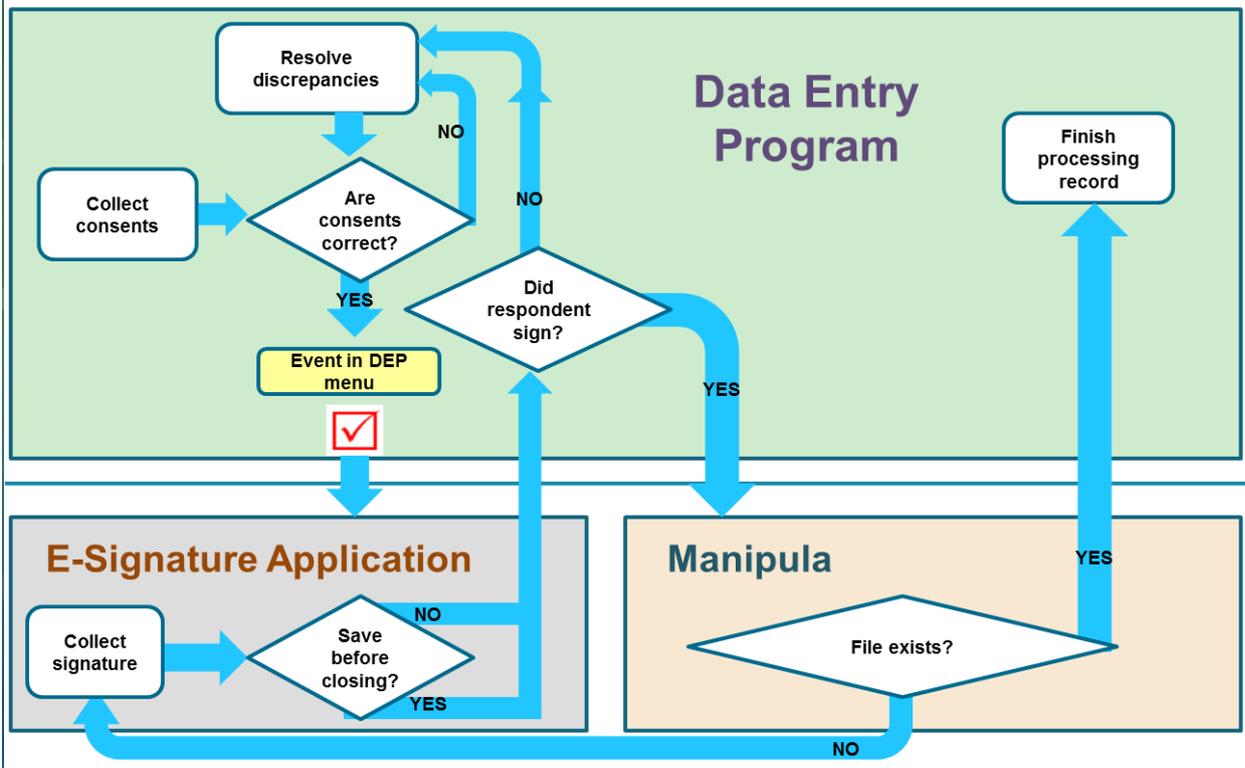
Figure 5. Error Message When a Signed Document is not Found



The process of using an external application to collect electronic signatures is detailed in the following flow chart (see Figure 6).

Figure 6. Processing Electronic Signatures as External Application

Processing Electronic Signatures



4. Limitation of Shell-to-EXE Approach in 4.8

Since the .NET application exists externally, DEP is unaware of what is taking place in the e-signature form until after the application is closed, DEP regains the focus, and the interviewer indicates a result (signed or not signed). We could avoid asking the interviewer for a result, and perhaps displaying an error message, if the buttons on the e-signature form sent values directly to the active record in DEP.

5. Using a DLL as an Alien Router in 4.8

As an alternative approach we could develop our .NET application as a DLL and reference an alien router in the code. This approach eliminates the need for input from the interviewer after the menu button is pressed, since the alien can be initiated through logic in the RULES, and the result can be passed directly to the active record.

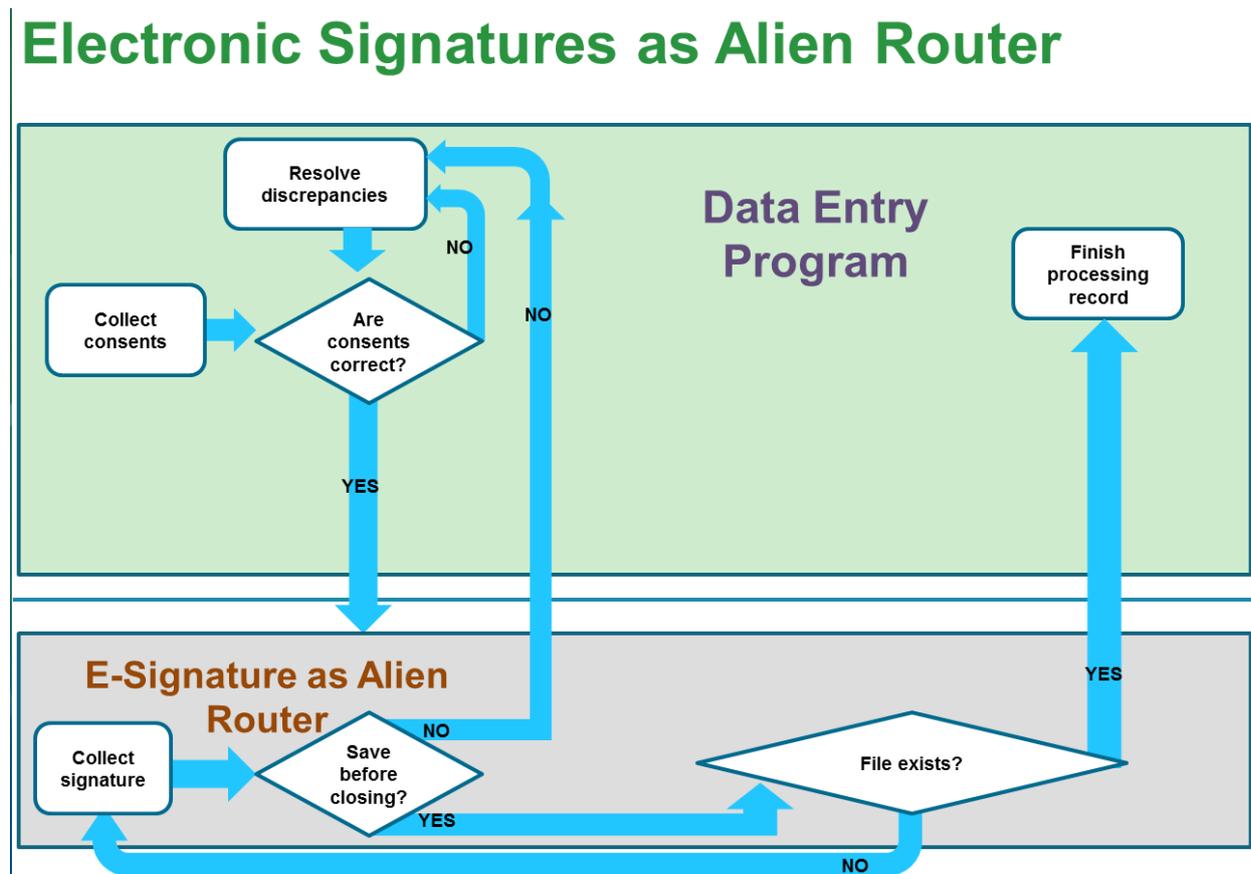
6. Converting the External Application to DLL

To convert this standalone program into a proper Blaise router, we convert it to a COM object, giving the class a GUID and a ProgId, and marking either the appropriate classes and methods or the assembly (as a whole) as COMVisible. This is the process for creating any COM object. As a standalone program, the e-signature application is called with the command line parameters via Maniplus or a button in the DEP menu file. Rewriting the program as a DLL requires that we change how the program is called. Since the look and behavior of the e-signature form as an external application depends on these command line parameters, this poses a significant problem. To replace this interface we would create a method for every e-signature form type that we need and create string auxfields in the router block for any dynamically-determined or optional arguments that the e-

signature program can then locate and read. Creating auxfields for all arguments is another valid approach, but this results in a negligible difference in programming effort for the DLL programmer (a big conditional statement instead of many methods) and a significant increase in programming effort for the Blaise programmer (declaring and setting extra auxfields). Likewise, the e-signature result (whether or not the e-signature is successfully recorded or requires edits) is also returned directly to the DEP, eliminating the requirements for user input from the interviewer and resolving the presence of an image file via Manipula.

The alien can return values from our process back to the active record in DEP, and we can use the result to determine if a form was saved and the image file exists in the secure location. We eliminate the step of resolving discrepancies due to user error related to an interviewer misreporting a result via data entry as shown in the following flowchart (see Figure 7).

Figure 7. Processing Electronic Signatures with Alien Router



7. External Applications in Blaise 5

For the administration of an electronic signature application via Blaise 5, it will be necessary to first include the .NET application as a Windows Service. In the Blaise datamodel code, we can create a procedure that calls the service as an ALIEN.

As per Blaise 5 Help, in the .NET code for the service we define a ServiceHost instance and service endpoints for data exchange.

In our .NET procedure for the e-signature form, we can validate the parameters passed to the form as well as any activity within the form, and then export our relevant result (saved, did not save) as a WCF (Windows Communication Foundation) “int” service type to the active record. In addition, we could allow editing of the form to pass back a WCF “string” for more complex form results.

8. Conclusions

The use of the electronic signature form as a means of obtaining, documenting and saving crucial consents data has proven extremely effective in our household survey. The process mitigates user error; allows our home office processing staff quick access to the electronic documentation; and has proven to be effective and reliable in a complex data collection. There has been a significant reduction in paper management with the security risks and other problems associated with paper data collections.

As a platform, Blaise 4.8 has provided Westat with several efficient approaches to coordinate parameterized input data from the DEP with the extended capabilities allowed via .NET applications and processing. As we transform our processing to utilize Blaise 5 capabilities, we feel that enhanced platform will further advance our efficiency in processing this key data.

A Web Compatible Dep

Maurice Martens, CentERdata, Netherlands

1. Abstract

The Survey of Health Ageing and Retirement in Europe (SHARE) is currently in its development phase for its seventh wave. This survey uses a centrally developed source questionnaire and using an online translation management environment this source version extended to 41 country/language versions.

In SHARE wave 7 the sample was divided into subsamples who are assigned two largely different questionnaires. The larger sample is assigned to a life history calendar questionnaire (like was done in wave3), the respondents who already did a life history calendar were assigned to do a regular interview. The SHARE wave 3 history calendar was developed in VB6. After some trails, it was decided that reusing this instrument was not wise. Some alternative approaches were reviewed, including using Blaise 5, but ultimately this redevelopment was done in Java, using a browser component providing the freedom to use web techniques combined with calls to the Blaise 4.8 API.

This Java/browser approach gives us the possibility to implement web techniques in CAPI. This allows for use of jQuery, and JavaScript code for CAPI. It allows for use of interface design using style sheets JavaScript solutions developed for CAWI mode can be reused in CAPI mode. Using this technique we were able to rapidly implement more advanced features like a history calendar, autosuggest lists, and word-recall list in a short time without major changes to the Blaise questionnaire.

This paper discusses the route that led us to this architecture and will showcase some exiting implementations, within the context of the SHARE multi-mode, multi lingual, multi-questionnaire panel study.

2. Introduction

CentERdata has been developing software for the Survey of Health Aging and Retirement in Europe (SHARE) since its first wave in 2004. The SHARE project is a multidisciplinary and cross-national panel database of micro data on health, socio-economic status and social and family networks of individuals aged 50 or over. Currently its seventh wave is under development, which will be fielded in 28 countries, in 28 languages, for which we need to generate 41 local versions. All these versions are derived from one source Blaise questionnaire which is developed in English. A compiled version of this questionnaire is uploaded to a central online database. From this, we identify and import translatable elements. These translatable items are presented in a web-environment for translation, TMT (Translation Management Tool). This tool and it successor LMU, Language Management Utility (See: Managing Translations for Blaise Questionnaires, Martens at al., 2009 and Blaise Translation Challenges: Versioning, Multimode and Exporting, Martens, 2012) allow for a quick flexible decentralized translation process of Blaise questionnaires, hiding programming code from translators.

SHARE interviews are done mainly via face to face interviewing. SHARE uses an ex ante harmonized model. All data which is collected will be collected using the same data model. Each wave fieldwork is done at the same time for all participating countries, the questionnaire is designed once, translations are done centrally via the web system, and country specific versions are generated centrally and distributed to our fieldwork agencies. On a bi-weekly schedule, data collected so far is transferred and collected at a central location. This centralized design has the advantage that the survey can be managed very well and has a quick turnaround, it however needs the support of many tailor-made software solutions.

To support the SHARE survey, Centerdata has developed a rich environment to support the flow of data, to help the interviewer identify respondents and to launch the questionnaires. In chapter 2 this architecture will be discussed.

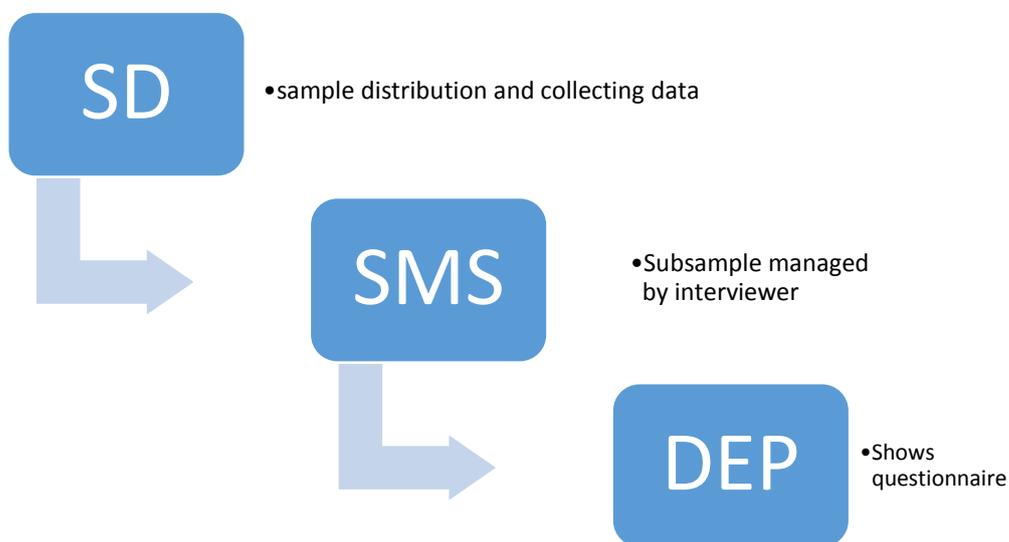
In this latest wave, we needed to present a Life History calendar to part of the sample. The Visual Basic 6 software that was developed in 2006 to support the Life History calendar in wave3 of SHARE was reviewed and deemed not suitable for reuse in the current wave. Since it still is not possible to generate such a tool natively in Blaise we had to develop an alternative solution. In chapter 3 various approaches will be discussed and we will explain why we chose our solution.

In chapter 4 the solution we developed will be presented. We added a browser object to the SMS system that communicates with the Blaise API. An html webpage is built to display the questionnaire. This allows for the use of (web-) techniques like cascading style sheets, jQuery, JSON objects, and html5 in a CAPI environment. It provides the freedom to develop your interface and questionnaire behavior to your own liking. With the web-compatible DEP we can reuse many of the tools we already developed for our websites. Since web techniques are well known, and quite easy to learn, we hope that Blaise will allow for a similar model natively, so it will be quicker and the full power of Blaise could better be used.

3. SHARE architecture

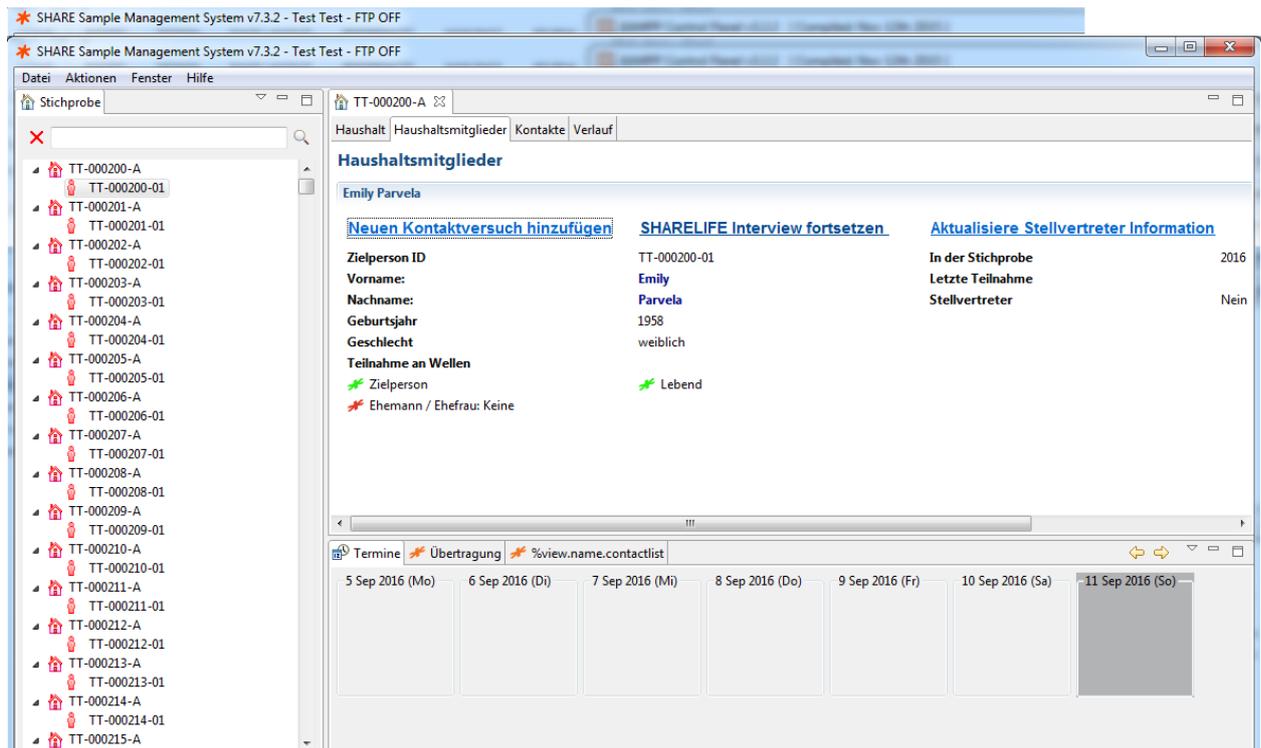
Various tools that need to interact were developed to support the SHARE survey. Since it is a panel study, in many cases the respondent has been visited before, therefore we often need to preload some of the information, to reduce interview length. When available, both a respondent and their partner are interviewed. To optimally identify changes in the household and manage the sample from a central location we developed a Sample Management System (SMS), which is installed on the interviewers' laptop. This program connects to a central server in each country at which subparts of the sample are assigned to interviewers.

Figure 1. Diagram of SHARE Architecture



The SMS uses a wizard style interview to determine the current composition of the household. Once done, it can launch the correct questionnaire for the respondent.

Figure 2. SMS Wizard



In wave 7, two big challenges were presented to us. It was decided that part of the sample, should get the wave 3 instrument again, which was a history calendar. We developed this software in 2010, using VB6 and the Blaise 4.8 API (See SHARELIFE Methodology wave3, Chapter 3). The respondents who already completed the history calendar in wave 3, will now get the regular (or rather updated wave 6) questionnaire. A second challenge was that at a very last moment, it was decided to add 8 new countries, which gives SHARE full EU-28 coverage. This put an extra burden on time and personnel.

4. Approaches

Given the time constraints and complexity of our task, our initial hope was to reuse the software we developed in wave 3. This was a VB6 application, which used the Blaise API, to show a history calendar. This software was developed in 2010, and we weren't really sure, to what extent it would still run on newer versions of windows, and how easy it would be to introduce right-to-left support for Hebrew and Arabic versions. It could also be problematic to integrate other tools we developed since then, for example, we need to support the display of movies, and allow for look-up tables to code, countries and occupations.

To examine the feasibility of reusing this software, we started further developing this version of the VB6 software, and tested it in a pretest in 25 of the SHARE countries. To a certain extent this approach actually seemed to work, it was very slow in some cases, and it was a big hassle to find the correct libraries to install, but it behaved above expected.

In our review we decided that we could not properly support this anymore; if problems occurred during fieldwork, we could not guarantee a timeframe in which problems could be solved. It was also reported by interviewers that the tool was very much outdated.

This presented a new challenge, how were we to display this calendar if we could not use the old tool anymore? Could we think of a trick to do it in Blaise 4.8 DEP? Would Blaise 5 present a solution? Should we develop something like the VB6 tool using another programming language? Should we

maybe strip functionality of the calendar down to only displaying images? We had some experiences with an online version of the history calendar (See Martens, 2013. Jumping around in Blaise IS), but this backend was programmed in PHP and used BlaiseIS, maybe that could help. We would need to install a server on every laptop, which seemed a bit overdone. Should we leave Blaise altogether, develop something ourselves, or look for another vendor that might support the features we need in their software? It was clearly time to make some sort of overview and properly discuss what path we should take to solve the challenge within the given time. In the following table we will present our thoughts and findings:

Table 1. Possible Solutions

Solution	Pro	Con
Reuse VB6 build on Blaise 4.8 API	<ul style="list-style-type: none"> • Proven • Works with current questionnaire 	<ul style="list-style-type: none"> • Outdated • Limited recent experience • Bad to support • No future solution
Blaise 4.8 DEP	<ul style="list-style-type: none"> • Stable 	<ul style="list-style-type: none"> • Can't develop a rich interface
Blaise 5 DEP	<ul style="list-style-type: none"> • New Blaise features seem to fit the regular questionnaire very well 	<ul style="list-style-type: none"> • Can't develop a rich interface • Limited experience • Complete redesign of all tools involved
Build something on Blaise 4 API	<ul style="list-style-type: none"> • Experience 	<ul style="list-style-type: none"> • What IDE?
Build something on Blaise 5 API	<ul style="list-style-type: none"> • Future proof • Build up experience 	<ul style="list-style-type: none"> • Complete redesign of all tools involved • What IDE?
Use the earlier developed web solution	<ul style="list-style-type: none"> • Proven • Easily adapted to work with current questionnaire 	<ul style="list-style-type: none"> • Questionnaire in browser • Install Blaise IS on every laptop
Other vendor	<ul style="list-style-type: none"> • Could try to find something that fully supports all our requests 	<ul style="list-style-type: none"> • No experience • Complete redesign of all tools involved • Can't reuse already developed code
Develop own survey system	<ul style="list-style-type: none"> • Full control 	<ul style="list-style-type: none"> • Can't reuse already developed code • Would take very long

Since we only had a few months to find a working solution, the directions that would cost too much time were ignored. For both Blaise native solutions we decided that it would not be possible to create the tools we needed. This left us with either develop something on top of the Blaise 4.8 API again, or find a way to get the web tool working on laptops.

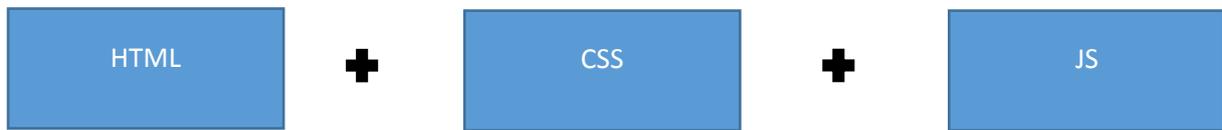
The code from the web tool we developed earlier was an html frontend, with a style sheet for the layout, which was called from a Blaise IS questionnaire. The calendar was generated by a PHP-script, driven by JavaScript calls hidden in the Blaise question texts.

We wanted to avoid installing servers on laptops, so the PHP part had to be replaced by some other part. The display of the questionnaire should not be done in a browser. The standard menus, shortcuts, and other behavior would need to be deactivated. Since our sample management system already launched the DEP to start up an interview, a logical step was to use a standard browser class to launch the browser window from within the SMS.

Once this idea was born, everything fell in place. The Java environment using a library called com4j needed to call the Blaise API, much like our VB6 solution had done before. The browser window could display the html-solution. In the next chapter we will discuss this new solution more in depth.

5. The webdep

Figure 3. Display



As described in chapter 3 we decided that we needed to integrate the functionality of the DEP in our SMS system. Java is able to show a browser component. We would need to show only one webpage, and all submitted actions done by this website will be detected by the system and translated into Blaise actions. The system will call the Blaise API, and transfer the information needed to be displayed back to the website.

A set of high level operations was defined:

- NextQuestion
- PreviousQuestion
- GotoLastQuestion
- GotoFirstQuestion
- GotoQuestion

When one of these actions needed to be done by the webpage, a JSON structure would be formulated like:

Listing 1. JSON Source Code

```
PreviousQuestion
  Key1
  Key2
  Answer
  Status
  RemarkText
  Suppress
```

This minimum information is transferred from the webpage to the browser and tunneled through to the Blaise API and generates a new state of the interview. The new state is detected by the Java environment and a JSON object is returned to back the website:

Listing 2. JSON Source Code

```

BlaiseObj
  Key1
  Key2
  Name
  Text
  questionText
  Remarked
  RemarkText
  Interface
  ErrorMessage
  HardErrorMessage
  HardErrorInvolved
  SoftErrorMessage
  SoftErrorInvolved
  DontKnowAllowed
  RefusalAllowed
  Required
  Value
  Status
  FieldDef
    MinValue
    MaxValue
    DataType
    Categories
    IsSet
    TextAsSetString

```

This set allows us to display the questionnaire and provide us with interaction with the questionnaire. JavaScript reads out the new values and adapts the webpage accordingly.

Figure 4. Questionnaire Display

Bitte benennen Sie das Land und wählen Sie dieses von der Auswahlliste aus.

Ascension Island
 Andorra
 Afghanistan
 Antigua und Barbuda
 Anguilla
 Albanien
 Niederländische Antillen
 Angola
 Antarktis
 Amerikanisch-Samoa
 Alandinseln
 Aserbaidshan
 Bangladesch
 Bhutan

<- Zurück | An | Weiter ->

Jahr:	'58	'59	'60	'61	'62	'63	'64	'65	'66	'67	'68	'69	'70	'71	'72	'73	'74	'75	'76	'77	'78	'79	'80	'81	'82	'83	'84	'85	'86	'87	'88	'89	'90	'91	'92	'93	'94	'95	'96	'97	'98	'99	'00	'01	'02								
Alter:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44								
1 Kinder																																																					
2 Partner																																																					
3 Unterkunft																																																					
4 Arbeit																																																					

This BlaiseObj object was extended to support some optional extra objects on top of this:

BlaiseObj['Movie']; a list of words with their duration is defined in the Blaise questionnaire. Using the setTimeout-function we can trigger when the words are shown. Before we used a movie to show this list, which we needed to generate for all translations, by hand.

BlaiseObj['Coder']; several lookup-tables were defined as JavaScript arrays, countries, job titles, languages, currencies. The searching in these tables can now be performed in JavaScript giving us full control of the behaviour of these tools.

BlaiseObj['Calendar']; the information that needs to be displayed in a calendar, is translated into a html structure, that displays the calendar.

The html page where we load the questionnaire in, is very basic, it is a set of divs. Javascript controls the content of these divs, and stylesheets control the layout.

Listing 3. HTML Source Code

```
<div id="completedep">
  <div id="questiontext-outer">
    <div id="questiontext">
    </div>
  </div>
  <div id="moviediv"></div>
  <div id="errordiv"></div>
  <div id="helptext"></div>
  <div id="answeroptions" class="answeroptions"></div>
  <div id="navigation" class="navigation">
    <div class="navigation-inner">
      <input name="previousbtn" value="⏪ Back" />
      <div style="display: inline-block;">
        <div id="remarked" class="remarked">&nbsp;</div>
        <div id="answerdiv" class="answerdiv">
          <input type="text" name="answerfield" />
        </div>
      </div>
      <input name="nextbtn" value="Next ⏩" />
    </div>
  </div>
  <div id="CalendarDiv"></div>
  <div id="remark">
    <span id="remarklbl"></span>
    <textarea id="remarkText"></textarea><br>
    <input type="button" name="savebtn" />
    <input type="button" name="cancelbtn" />
  </div>
  <div id="console"></div>
</div>
```

This approach gives us the possibility to implement web techniques jQuery, and JavaScript code for CAPI. It allows for use of interface design using style sheets. JavaScript solutions developed already for CAWI mode can be reused in CAPI mode. Using this design we were able to rapidly implement more advanced features like a history calendar, autosuggest lists, and word-recall list in a short time without major changes to the Blaise questionnaire.

An added bonus is that we can directly present the questionnaire over the internet, we could present the calendar to our respondents as feedback, or could attach the questionnaire to our translation environment and present questions in context to our translators.

While implementing, two problems were detected. There was an issue with speed. It took a long time before we got a response from the API. When tracking it down, we found that this was due to the storing of answers. To solve this, we save the answers after the response is returned to the browser.

A bigger problem presented the use of open answers. We wanted the instrument to be completely UTF-8 encoded. However, internally Blaise 4.8 stores strings in ANSI, but depending on your windows installation, it will use different code pages for this. When we want to use open answer responses as a fill in a following question, the open answers that were coded with this different code page, would show as gibberish. To solve this we needed to recode the open answers strings to something that could be formulated in the first 128 characters of the code pages, because these are all identical. Since we now presented the questions in a browser a logical candidate for this would be to encode special characters into htmlchars (Ӓ). For each non-latin character this however takes 7 characters to encode. Since we sometimes need to integrate these open answers in fills we now encounter that these fills can become too long easily. Since these composed fills break off when they exceed length 256, we could still encounter Gibberish texts when open answers are too long. If we could somehow enforce open answers to be stored in a specific code page or if the limitations on string length would disappear, we would appreciate it. As we understood these issues are not relevant for Blaise5 anymore, so these problems may solve themselves in time.

We hope the use of internet techniques in Blaise CAPI, integrating JavaScript, designing interfaces with Style sheets is something that could be supported natively in future versions of Blaise.

References

Martens at al., 2009. Managing Translations for Blaise Questionnaires

Schröder, M. (ed.), 2011. Retrospective Data Collection in the Survey of Health, Ageing and Retirement in Europe. SHARELIFE Methodology. MEA, Mannheim.

Martens, 2012. Blaise Translation Challenges: Versioning, Multimode and Exporting,

Martens, 2013. Jumping around in Blaise IS

Converting Paper Collection on the International Passenger Survey to Blaise 4 on a Tablet Computer

Mike Hart, Office for National Statistics, United Kingdom

1. Abstract

The International Passenger Survey (IPS) is a continuous survey that has been running since 1961 by the Office for National Statistics (ONS). This is the main source of information in the UK on international travel and tourism, as well as the associated expenditure.

Between 700,000 and 800,000 face-to-face interviews are conducted every year at the major air, sea and tunnel routes to and from the UK by a national team of 240 interviewers.

At the moment, the IPS data is collected on paper forms which are then manually typed into Blaise 4.8 and transferred electronically to ONS head office where data cleansing and validation take place.

ONS policy is to be digital by default and this coupled with the need to produce efficiencies has placed an emphasis on exploring more efficient of collecting and validating the data we collect.

This paper follows on from the work undertaken in 2014 and will demonstrate some of the challenges we have face and how we have overcome with them.

The vision remains the same:

- Speed up data collection
- Improve data quality
- Develop a collection instrument that is user friendly.

2. Introduction

All UK government departments have had a directive from central government to be digital by default and the International Passenger Survey (IPS) is the only Social Survey we still collect the data on paper. So naturally our attention has been focussed on removing paper from the data collection exercise. We have been looking at this for a couple of years now and initial investigations were carried out with Blaise 5, but we are now looking at Blaise 4 as the plan is to reuse as much of the current systems and adapting these systems to work with Blaise 5 was considered too onerous.

In this paper we report on our progress towards transferring the International Passenger Survey (IPS) from a paper form to an electronic questionnaire on a tablet. The paper will reflect on learning and concepts developed in Blaise 5 and how we have reflected this learning in the Blaise 4 instrument. Whilst the directive from central government is digital by default we have to maintain data quality and ensure the collection exercise is not a burden on the respondent.

3. What benefits for the survey?

The IPS is a continuous survey that has been conducted by ONS since 1961. Between 700,000 and 800,000 interviews are conducted every year at major airports, seaports and tunnel routes covering both departures and arrivals terminals. This is the main source of information used by the UK governing bodies for planning, monitoring and informing decisions on tourism and immigration policies.

At the moment, the IPS data is collected on paper forms which are then manually keyed into the CADI system in Blaise 4.8. The data is then transferred electronically to an ONS data centre where

editing and validation take place.

Figure 1. Conducting Survey



The vision for the project as previously stated in the ONS paper delivered by Lan Benedikt in Beijing 2015 [1] remains the same:

- Ensure data collection at point of collection is no slower than collecting the data on paper
- Take advantage of in-built validation and the routing capability in Blaise to cut down on human error and improve data quality
- Eliminate manual post data collection tasks

4. First prototypes

The project started in July 2014 and a questionnaire was developed in Blaise 5. A site visit was carried out at Bristol airport to gain an understanding of how the field work was conducted and how it could be adapted to collect the data on a Tablet.

Three main areas for improvement were identified.

1. Survey flow: the IPS questionnaire has complex routing with a Main Questionnaire and various trailers e.g. Immigration Trailer, Employee Trailer, Student Trailer short/long forms [2][3], making it sometimes difficult to quickly identify the next question to ask, especially for inexperienced interviewers.

2. Coding frames: IPS interviewers usually carry a folder with all the lookup tables that they need for filling out the survey. Examples include flight numbers, airport IATA codes, country codes, region codes, UK towns. Other documents they carry include the trailers mentioned above, as well as “show-cards” in 16 different languages. It is difficult to shuffle through the folder to quickly find the correct forms, even more so because IPS data is collected on the move.

3. Data keying (CADI): this process is unanimously perceived as very time consuming and cumbersome. So much so that most field managers have to take work home to complete their tasks on time and manual data keying also has the potential to introduce human error. The elimination of this step is where significant improvement can be achieved, both in terms of efficiency and data quality.

The good news was that most of the problems identified could be solved with routing, in-built checks and lookup features. Thus, a first prototype of the electronic questionnaire was developed and deployed on a tablet computer, Lenovo Miix2 8-inch Windows 8.1, in order to explore what could be done. Examples of screenshots are shown in Figure 2.

Figure 2. Look-and-feel of the First Prototype of the IPS Questionnaire in Blaise 5. Different background colours were used to differentiate various trailers and the main questionnaire.



At the end of August 2014, the IPS research team organised a focus group in Newport during which the prototype was demonstrated to a group of interviewers and field managers from Bristol, Manchester and Liverpool airports. The main aims of this meeting were to gauge users' first reactions to the new survey instrument and to gain feedback on what could be improved. The results were broadly positive and no major blockers to implementation identified (the results are available in the ONS paper delivered by Lan Benedikt in Beijing 2015 [1]).

5. Change of direction from Blaise 5 to Blaise 4

Since late 2015 there has been a change of direction within the project and we have decided to implement this survey with Blaise 4 rather than Blaise 5. The driver behind this decision is that ONS is undergoing a large transformation project and significant resource is being devoted to developing new systems and it was considered that moving to Blaise 5 would divert resource away from the transformation project.

ONS has recently moved to a new methodology when developing software, AGILE. There are 12 principles underpinning the Agile approach (The Agile Manifesto [4]) and the first states that 'our highest priority is to satisfy the customer through early and continuous delivery of valuable software'.

So with that principle in mind our first task was to identify the customer. For a statistical office you would think the customer would be the end user of the data but in this instance the customer is the interviewer collecting the data. This is because they are the user of the software and without acceptance of the software from them then the final product, the statistical outputs, may be flawed because the interviewer is not able to perform the function required of them.

Another key concept of AGILE software development is 'fail fast, fail often' [5], whilst failing may sound undesirable it is a fact of life. The point to be understood about 'fail fast' is the second word, it is about reducing delay. If a failure is going to take place you want to reduce the time lag in detecting the failure and relaying the detection back to the developer(s). Failing fast and often also gives you an opportunity to terminate a project without expending lots of resource and money to a project that is never going to deliver should an insurmountable problem emerge.

To complete the new development framework ONS is moving to DevOps teams where teams are made up of multi-skilled individuals to help remove silos and where the people understand they are all on the same side, working together to build and develop software fit for purpose.

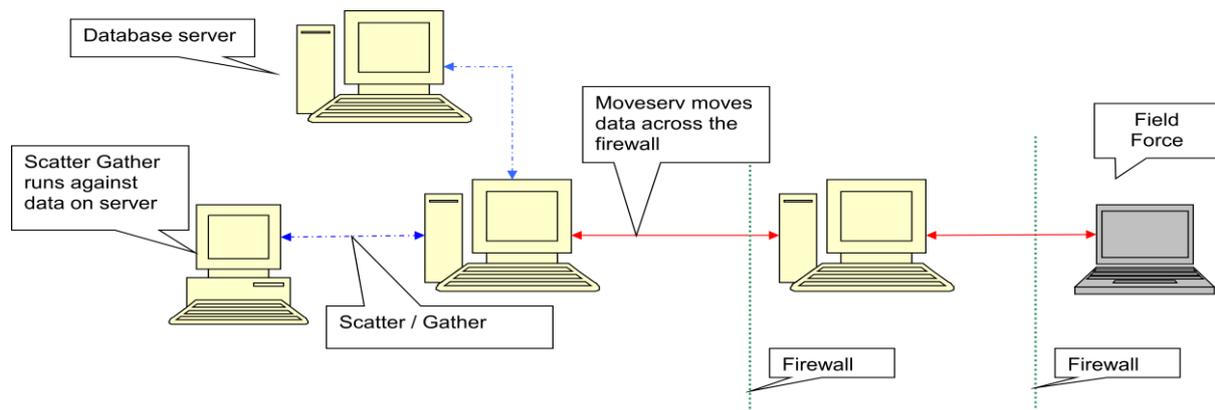
6. Testing

With the AGILE concepts fresh in our minds and looking to build on the previous work done in Blaise 5 we adopted a new approach to delivering the project. A series of Sprints were defined, each being designed to build on the previous one and add incremental value to the project.

Sprint 1 – Using Legacy systems can we transmit and receive data from the Tablet

This sprint focussed on plugging the Tablet into the existing transmission systems (see Figure 3). As we were reusing existing systems we were not expecting any issues and none were encountered. The test was still valuable as this is a key part of the system and if data could not be transmitted to and from the Tablet then this needs addressing before progressing to the next stage.

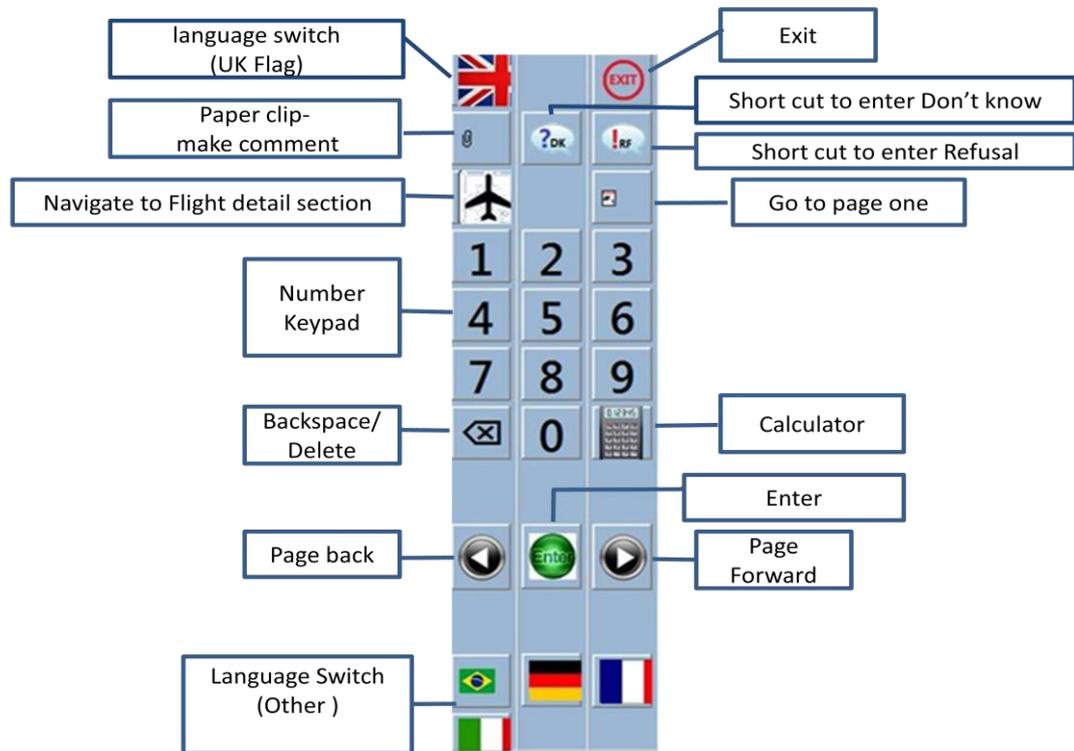
Figure 3. Schematic Representation of Transmission Architecture



Sprint 2 – Can we design a user interface to enable data collection?

Using the interface developed for Blaise 5 was not an option as the features available in Blaise 5 are not available in Blaise 4. After some head scratching and asking other NSI's how they use Tablets we came up with the following design (see figure 4), a big thanks to CSO Ireland who sent us some example code. This navigation panel coupled with the on screen keyboard native to the tablet means the interviewers do not have to switch between numbers and alpha inputs during data entry and gives them the complete range of functions that they need to have to perform an interview. We have been able to make use of some of the design principles identified when we were designing the survey for Blaise 5 making use of Don Normans principles "The Design of Every Day Things" [6].

Figure 4. Navigation Panel

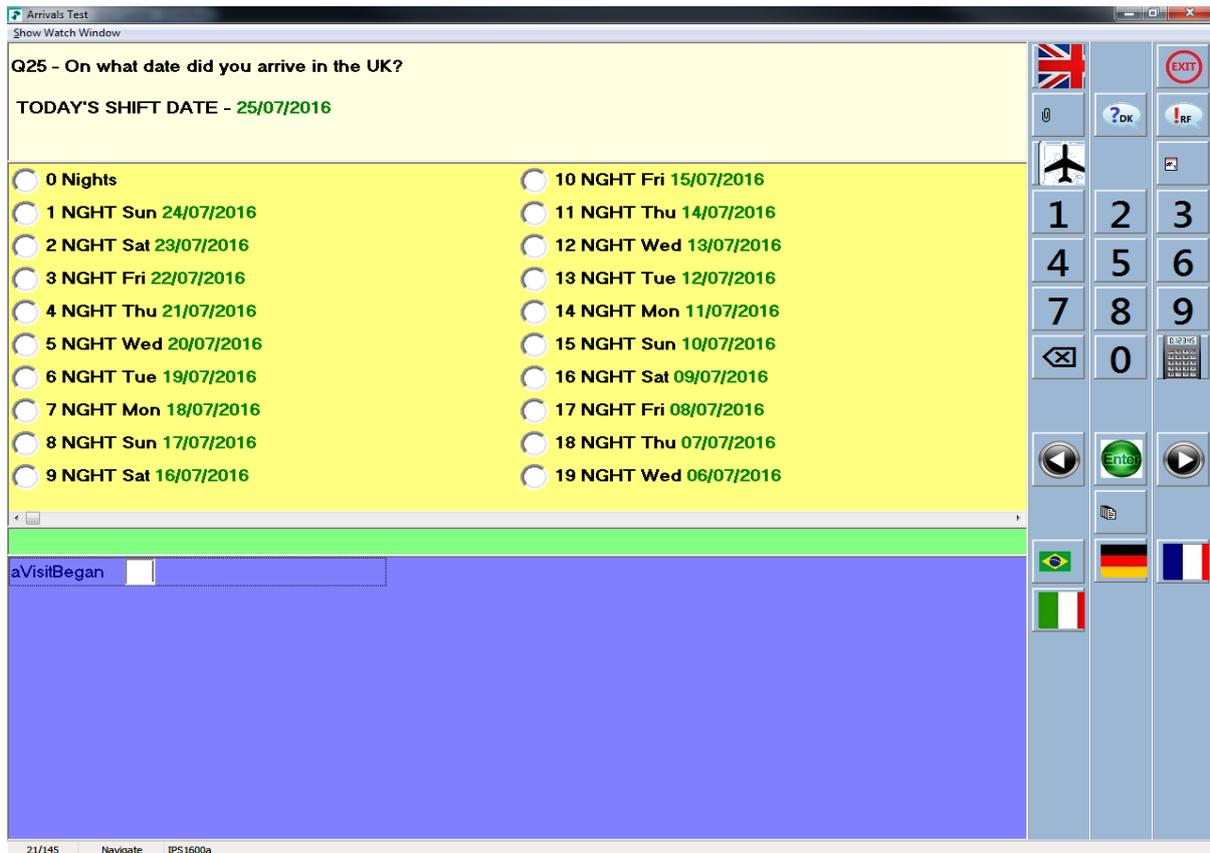


Sprint 3 – Demonstrate user interface to interviewers

During this sprint we gathered a small group of interviewers together to review some of the challenges facing us during data collection: how to collect dates, how to collect information displayed on maps and how to best use lookups.

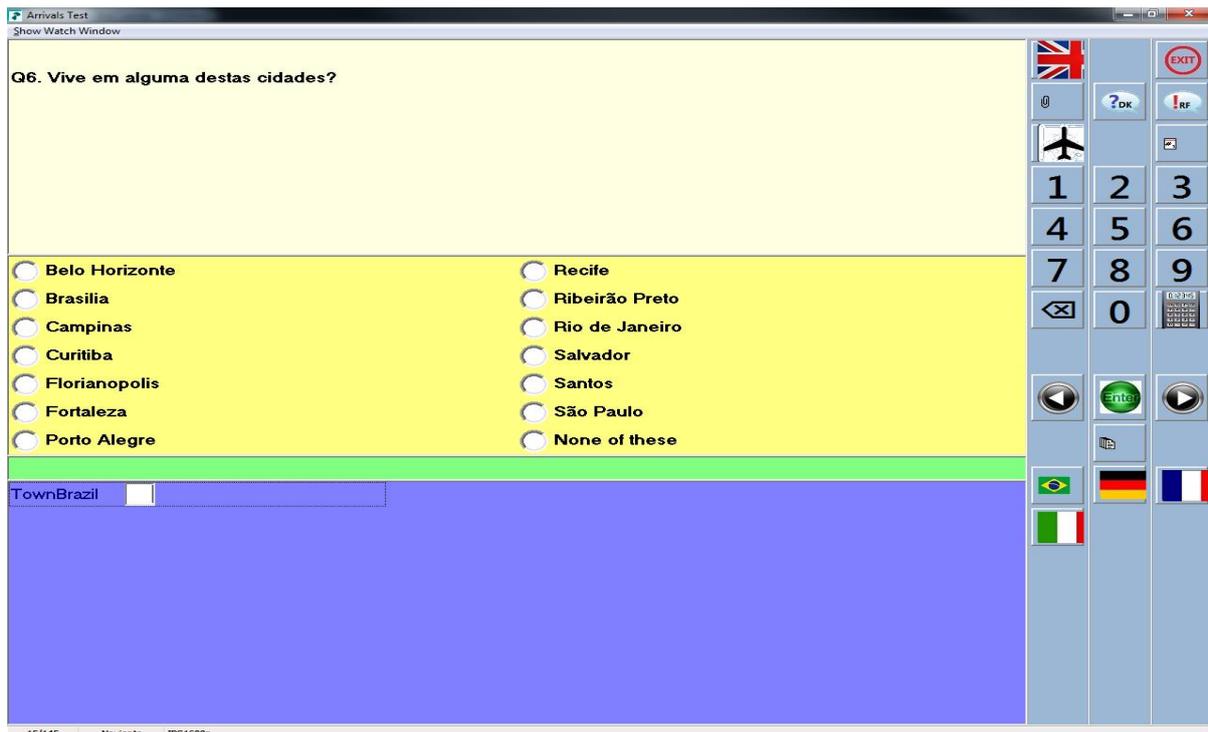
Collecting dates raised an interesting problem, we looked at implementing a date picker in Blaise 5 but this feature in Blaise 4 does not display large enough on a Tablet device to collect the information speedily and accurately. So what to do? We looked to increase the size but even with an upgrade of the software from CBS size remained a problem. We therefore looked at the problem from a different perspective. What data are we trying to collect and how to collect it in the most efficient way? So whilst a date picker is an industry standard, based on the data we collect we know most trips last less than 22 days, why use a date picker when you can visually present this data with radial buttons for easy selection (see figure 5). We also added in additional information to aid collection, such as the number of nights and the day of the week of the end date displayed in full to aid people’s memories. If the date is out of range then we have the ability to collect this at another question where we collect the date in its most basic form of DD/MM/YYYY.

Figure 5. Date Information displayed as a List of Valid Dates with Associated Aid Memoir Information alongside the Data Items



The next issue to tackle was how to collect information currently displayed on a paper map about which city or region a respondent lives in. In Blaise 5 we looked to develop interactive maps with a complex network of grids with each cell designed as a selectable button. Whilst this was effective this functionality is not available in Blaise 4 and it would also carry a maintenance burden when new maps need to be added. We looked at displaying the maps in the screen size available but doing this made them unreadable for countries with lots of individual regions, for example Switzerland. Again we looked at what data it was that we were looking to collect from the respondents, city of residence or region of country. Given that respondents already know this information presenting a simple list with the answers and question in the native language solves the problem without the need for anything more complex (see figure 6). Sometimes the simplest idea is the best idea.

Figure 6. City Information displayed as a List, Question Text displayed in Portuguese



Finally we looked at the use of lookups. We use them extensively within the questionnaire to collect data on destinations, nationality, airlines and towns within the UK. It might seem obvious to display a list of the most commonly used codes to collect the desired information; however the survey collection environment is all about speed and accuracy. This method can lead to more pages having to be displayed to collect the information, so currently we are experimenting with both methods of collection one being a list (see Figure 7) followed by a lookup and the other being a straight lookup (see Figure 8).

Figure 7. Radial Buttons to Show List of Countries

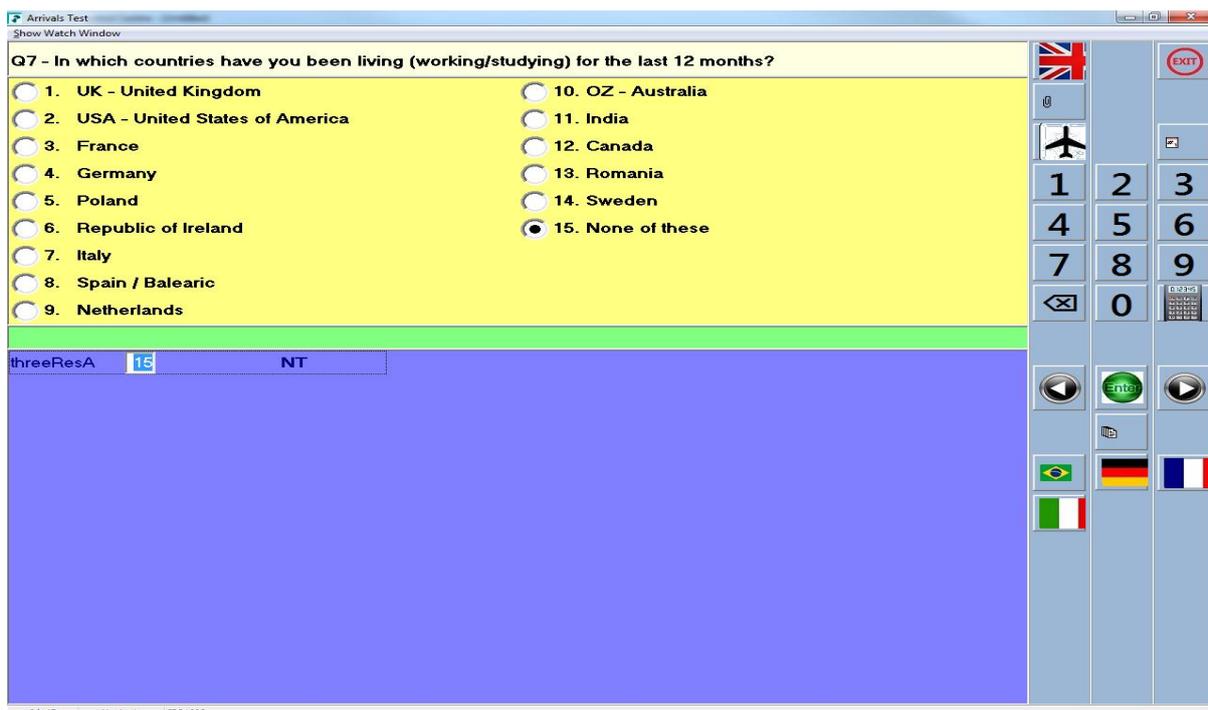
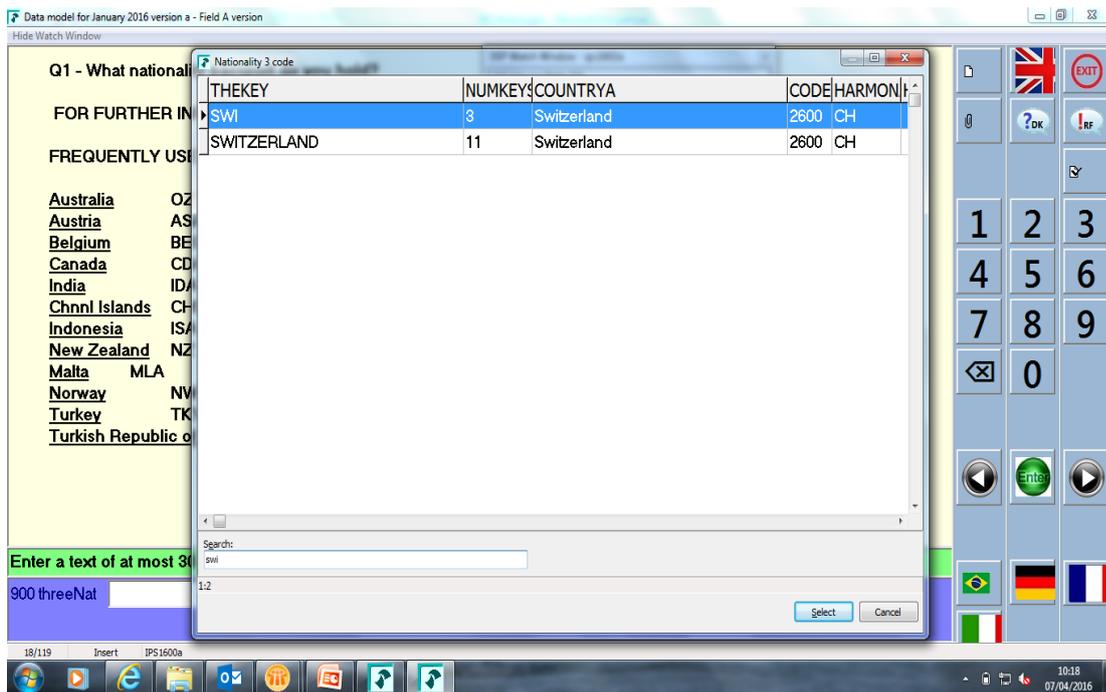
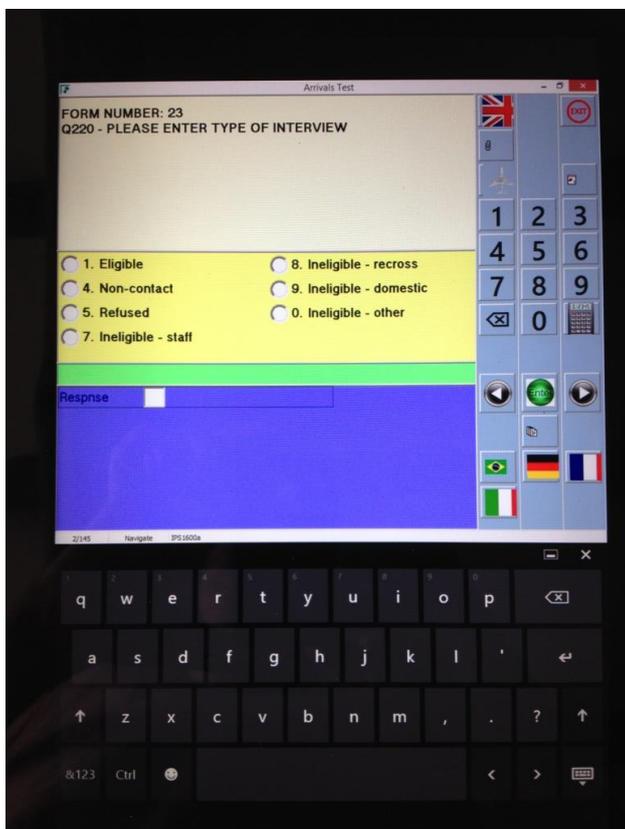


Figure 8. Entry to Lookup Coding Frame



Time will tell which is the best solution and it maybe that we develop lists for certain collection locations but we would like to develop a generic approach to reduce maintenance burden in the future.

Figure 9. Screen Shot of Navigation Panel, Native Keyboard, and Questionnaire



Having developed the solution to a point where we felt able to conduct an end to end interview now was the time to test our designs on the public and introduce a larger pool of interviewers to the solution. We conducted tests at the following sites:

Sprint 4 – Conduct Trial at Bristol to assess usability on a departure shift

Sprint 5 – Conduct Trial at Heathrow to assess flow on a departure shift

Sprint 6 – Conduct Trial at Manchester to assess flow on an arrivals shift

Sprint 7 – Conduct Trial at Portsmouth to see if location has any special requirements

Between each test we conducted reviews/retrospectives of what went well and what did not and made changes as and when required. The interface had some functions added, others removed and the interviewers made lots of suggestions about the questions themselves and certain sections were moved to aid navigation. The results from the tests have been very encouraging; no show stoppers to implementation have so far been identified. The key findings are summarised below:

Positive

Ease of use - Tablet is easy to use and the experience was enjoyable

Speed of use - Time taken to complete comparable or quicker than paper collection

Professional - Use of electronic data collection device makes us look professional

Interaction - Increased engagement from the respondents

Needs improvement

Navigation - Difficult to go back to certain sections

Weight - Potential strain on the hand and arm

Use of radio buttons - Buttons too close together

Questionnaire - Needs stream lining and adapting for new collection method

Other suggestions / requirements

Peripherals - Need stylus pens to aid answer selection

Peripherals - Need some type of carrying device to reduce strain

6.1 Measuring Flow

A key element of the IPS Survey is how we select respondents; this is by means of an imaginary counting line, every time a respondent crosses the line (as determined by collection location) a click is recorded on a clicker and after so many clicks (changes by location e.g. 1 in 20) an interviewer is tasked to interview the selected respondent. To prove that the use of the tablet would not affect the sampling interval i.e. could we keep up the pace of respondent selection in different environments we planned two tests, one at Heathrow on a departures shift and another at Manchester on an arrivals shift. Different shift types were selected for the test as the flow rate is different, on departures the flow of respondents tends to be steady whilst on arrivals then tends to be peaks and troughs.

The test at Heathrow produced mixed results in that we could easily keep up with the flow but this was in part down to the fact that the shift was unusually quiet. It did however throw up an issue with the navigation panel, in that the buttons changed places an error that was traced to having different versions of Blaise installed on the hardware (Dep.exe versus compiled Blaise code).

The test at Manchester was a lot more successful in measuring flow but we did take steps to avoid a repetition of the Heathrow test. Firstly we started the test when we knew we would have 4 or more planes arriving within 20 minutes of each other and we doubled the sampling rate from 1 in 20 to 1 in

10 to stress the test. The test was a great success and accounting for interviewers still getting familiar with the hardware, the new way of working and the increased sampling interval, results were comparable with normal shift conditions.

6.2 Increased Respondent Engagement

One of the more interesting findings from the three live tests is how respondents are now engaging with the interviewers. Currently the interviewers have a clipboard on which to rest the paper questionnaire but due to the small typeface and the fact that the paper form is cluttered the respondents take little notice of the questions on the form and the interview is often conducted facing each other. On the Tablet however the respondents are engaged in the interview process from the start as they are able to read the questions themselves, often quicker than the interviewers. In some cases the respondents are almost self interviewing either by providing answers before the interviewer has finished asking the question or selecting the answers on the Tablet themselves. This has highlighted the redesign element of this project as what we display to the interviewer must also be clear to the respondent.

7. What next?

We have developed a Maturity Roadmap which shows what functions/services we will be delivering out until September 2017, when we hope to have the full service delivered. It is based on AGILE principles where you deliver incrementally, at each stage there is an element of 'walk away ability', so if for some reason the project is stopped then you are able to demonstrate that some value has been delivered.

Figure 10. IPS Tablets Roadmap

The key activity we need to deliver by the end of this year is the methodological work that needs to be undertaken. This is required to gain an understanding of any discontinuity in the time series which

may occur as a result of switching collection modes and to understand if switching modes introduces any bias.

Other activities to be concluded by the end of year include:

Review of the IPS Questionnaire to ensure the questions and layout are enhanced for Tablet collection
Identification of hardware – currently we are using Surface 3 Windows Tablets

Review of the coding, validation and edits – we are aiming to get the data as clean as possible at point of collection

8. References

[1]. Lan Benedkit. Developing the UK Passenger Survey in Blaise 5 on Tablet Computer, Office for National Statistics, UK, Statistics. IBUC 2015, 16th International Blaise Users Conference

[2]. User Guide (Volume 1) Background and Methodology. International Passenger Survey Overseas Travel and Tourism http://www.ons.gov.uk/ons/guide-method/method-quality/speci_c/travel-andtransport-methodology/international-passenger-survey/index.html, 2014

[3]. User Guide (Volume 2) Background and Methodology. International Passenger Survey Overseas Travel and Tourism <http://www.ons.gov.uk/ons/guide-method/method-quality/specific/travel-andtransport-methodology/international-passenger-survey/index.html>, 2014

[4] The Agile Manifesto <http://www.agilemanifesto.org/principles.html>

[5] Fail Fast, Fail Often: How losing can help you win, Ryan Babineaux & John Krumboltz

[6] The Design of Every Day Things, Don Norman

Using Survey Paradata

Gina Cheung, Andrew Piskorowski, Lisa Wood, and Hueichun Peng, University of Michigan Survey Research Center, United States

1. Abstract

Paradata that are captured during the survey process are a valuable source of information in helping us understand and improve the data collection process.

One of the advantages of using Blaise for survey data collection is the rich capture of paradata that is native to the application. Paradata which are linked directly to the administration of a survey instrument are collected automatically through the Blaise software (i.e., audit trail). The ADT file from Blaise 4 has been very valuable in understanding interviewer behavior. With the advent of BlaiseIS and Blaise 5, we now are able to increase the richness of our paradata collection for understanding respondent behavior on web-SAQ (self-administered questionnaires) and/or mixed mode projects (i.e., interviewer and web-SAQ combined).

The main focus of this paper will be to provide some practical examples to illustrate how the survey paradata can be used to inform decision making throughout the data collection process and assist in fieldwork monitoring.

Specific practical uses include:

- Examining question timings and survey routing (i.e., which questions were asked but not answered and which questions were not seen due to survey logic?)
- Improving questionnaire design (including mobile survey design)
- Recovering lost survey data
- Identifying issues related to quality control
- Understanding respondent and interviewer behavior within a survey

These examples will illustrate the value of the native Blaise paradata and the supplemental survey paradata captured at the University of Michigan. Also some of the challenges/limitations of the native Blaise 5 paradata will be considered.

2. Introduction

Paradata that are captured during the survey process are a valuable source of information in helping us understand and improve the data collection process.

“Traditionally” paradata includes:

- Interviewer (experience, training grades, historical performance)
- Sample segments (PSU, Stratum, observations)
- Address (probability of selection, observations, # contacts, status)
- Screener contacts (call #, interviewer, time, date, informant behavior, outcome)
- Household (composition, informant behavior, sample respondent characteristics)
- Main interview contacts (call #, interviewer, time, date, informant behavior, outcome)

“Now” paradata also includes...

- Audit trails (keystrokes, timings, functions, consistency checks, suspensions)
- Sample management system (log and timing of actions)
- Digital photos
- Fingerprints
- GPS (Global Positioning System)
- Digital recordings
- Collection of various anthropometric data using digital devices

One of the advantages of using Blaise for survey data collection is the rich capture of paradata that is native to the application. Paradata which are linked directly to the administration of a survey instrument are collected automatically through the Blaise software (i.e., audit trail, called ADT file). The ADT file from Blaise 4 has been very valuable in understanding interviewer behavior. With the advent of Blaise 5, we now are able to increase the richness of our paradata collection for understanding respondent behavior on web-SAQ (self-administered questionnaires) and/or mixed mode projects (i.e., interviewer administrated and web-SAQ combined).

The main focus of this paper will be to provide some practical examples to illustrate how the survey paradata can be used to inform decision making throughout the data collection process and assist in fieldwork monitoring.

Specific practical uses include:

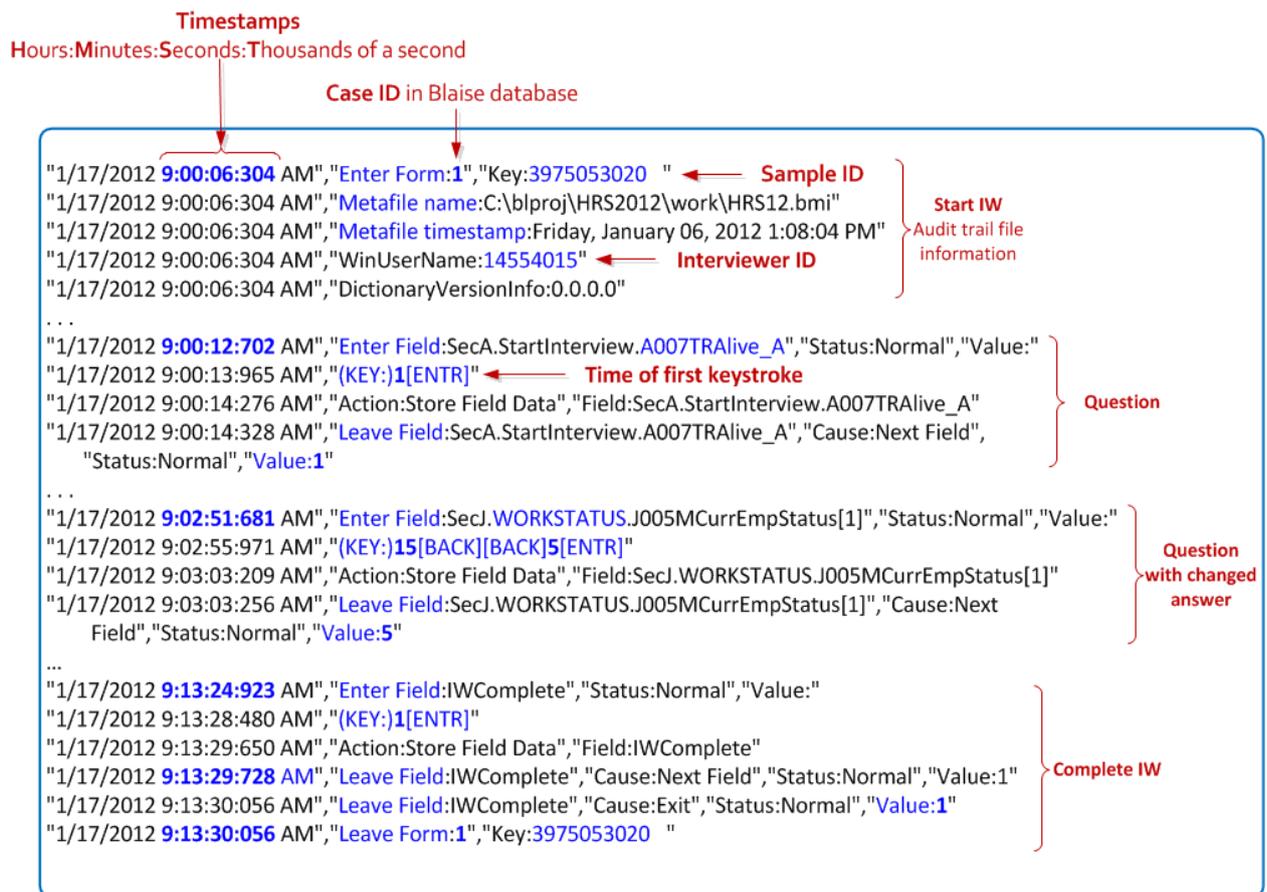
- Examining question timings and survey routing (i.e., which questions were asked but not answered and which questions were not seen due to survey logic)
- Understanding respondent and interviewer behavior within a survey
- Identifying issues related to quality control
- Improving Web questionnaire design

We discuss each of these examples in detail to illustrate the value of the native Blaise paradata and the supplemental survey paradata captured at the University of Michigan.

3. Blaise Audit Trail File

Blaise Statistics Netherlands provides a default program that permits users to record audit trails for all Blaise cases automatically. Every time a field in the instrument is entered or exited, or a specific action is performed, a record containing a time stamp and the current name, value, and state of the field is created. In the past 15 years, the UM Survey Research Center has created a few systems to parse out the ADT file as well as systems designed to utilize the ADT data for reporting purposes. Figure1 is an illustration of a typical Blaise ADT file layout.

Figure 1. Typical Blaise ADT file layout



Here is a list of IBUC papers related to use Blaise Audit Trail files presented by UM SRC staff

- IBUC 2001: Reporting on Item Times and Keystrokes from Blaise Audit Trails by Sue Ellen Hansen and Theresa Marvin, University of Michigan
- IBUC 2004: Replaying Blaise Audit Trail Files for Data Verification by Jason Langfahl
- IBUC 2004: Replaying ADK Files for Testing Blaise Applications by Jason Ostergren and Rhonda Ash
- IBUC 2004: Blaise AT Report System by Youhong Liu, Eduardo Galvan, and Gina Cheung
- IBUC 2006: Blaise PlayBack And Recovery System by Youhong Liu and Gina-Qian Cheung
- IBUC 2012: Blaise Audit Trail Data in Relational Database by Joel Devonshire, Youhong Liu, and Gina Cheung
- IBUC 2013: Blaise 5 Paradata Requirements by Rebecca Gatward, Lisa Wood, Patty Maher and Gina Cheung
- IBUC 2013: Adding Business Intelligence to Paradata: The Blaise Audit Trail by Joel Devonshire and Gina Cheung

Review of the ADT data can reveal potential problems in question wording or question comprehension, for example, if a disproportionate amount of time is being spent on a single question. It might also reveal problems with the survey logic, for example, if a question is erroneously being skipped or displayed for respondents.

4. Using paradata to understand respondent and interviewer behaviors

4.1 Examine web SAQ (self-administrated questionnaire by the respondent) routing:

To reduce the burden on respondents, we have designed the web SAQ to allow a respondent to skip an answer and move onto the next question. But for analysis purposes, we need to know which questions are intentionally skipped as a function of instrument logic -NASK, and which questions are being skipped by respondents - NANS, either intentionally or unintentionally. In conjunction with the survey data and audit trail, we are able to identify fields within the survey that a respondent skipped without answering and fields that a respondent did not see – off route. The audit trail contains an entry for every field a respondent has entered or “focused” on. Comparing this list of fields from the audit trail to a list of survey data fields with responses, we are able to determine which fields were presented to the respondent [EnterField in the audit trail] but were not answered [no value in survey data] as well as which fields were never seen due to logic [no value in survey data] and [no value in audit trail]. In Figure 2, we can see some of this process.

Figure 2. NANS and NASK

		Audit Trail Data						
		Present	Missing					
Survey Data	Answered	ANSWERED	ERROR					
	Blank	NANS	NASK					
SampleId	Sex	Age	Status_F	OverallFeel	ETS	ThinkEnlist	K001	K002
11111	1	22	1	1	8	NASK	1	2
22222	2	23	1	2	4	NASK	1	3
33333	2	24	1	3	3	NASK	1	2
44444	1	25	NANS	4	NANS	NANS	NANS	NANS
55555	1	26	2	1	NASK	2	1	1
66666	1	27	2	5	NASK	2	1	2
77777	1	28	2	2	NASK	NANS		

4.2 Trace interview routing to understand how the interviewer administered the interview:

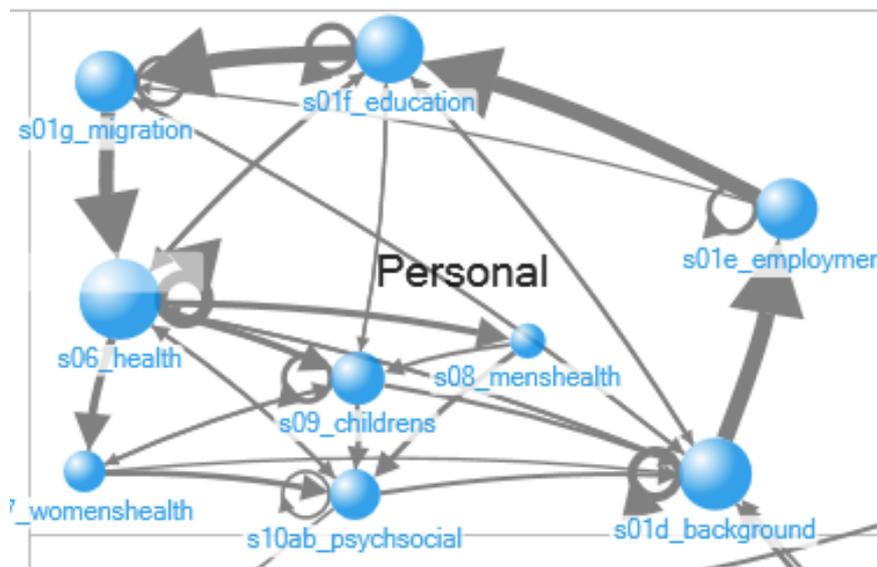
The Blaise instrument can be designed with parallel blocks so that multiple sessions (topics) with multiple respondents in one household can be administered easily. For example, as shown in Figure 3, the interviewer can “jump” to any “cell” depending on the respondent’s answers. This design provides a high level of flexibility for a large, complex household interview. However, it is important to know whether this kind of design flexibility has any impact on interview length.

Figure 3. Blaise instrument with parallel blocks

Name	Background	Employment	Education	Migration	Health	Womens Health	Mens Health	Children	Pysch/Social
ADAM K (AK)	Started	Done	Done	Done	Started	---n/a---	Not Started	---n/a---	Not Started
AMINA A (MINA)	Not Started	---n/a---	---n/a---	Not Started					
ABDUL A	Not Started	---n/a---	Not Started	---n/a---	Not Started				
TANLIDOW A	Not Started	---n/a---	---n/a---	Not Started	---n/a---				
YUSSIF A	Done	---n/a---	Not Started	---n/a---	Not Started	---n/a---	---n/a---	Not Started	---n/a---
LAILATU A	Done	---n/a---	---n/a---	---n/a---	Not Started	---n/a---	---n/a---	Not Started	---n/a---

Figure 4 illustrates the multitude of paths an interviewer can take when administering an interview with parallel blocks. Most interviewers will follow the traditional route and ask one respondent to answer each item, in order, before moving onto the next respondent in the household; in Figure 4, this is depicted by the largest arrows. Another interviewing technique that an interviewer may use is to ask all respondents in a household to answer one item before moving onto the next item; in Figure 4, this is illustrated by the grey circle next to each item name. The remaining arrows in Figure 4 demonstrate alternate paths that may be taken throughout the interview based on how the respondent chooses to answer—or not answer—survey items. When comparing the various routes that an interviewer may take when administering an interview, the traditional route (asking one respondent to answer all survey items before moving onto the next respondent) proves to take less time than the alternative of asking all respondents to answer one survey item before moving onto the next survey item.

Figure 4. Routes taken by an interviewer when administering a survey with parallel blocks



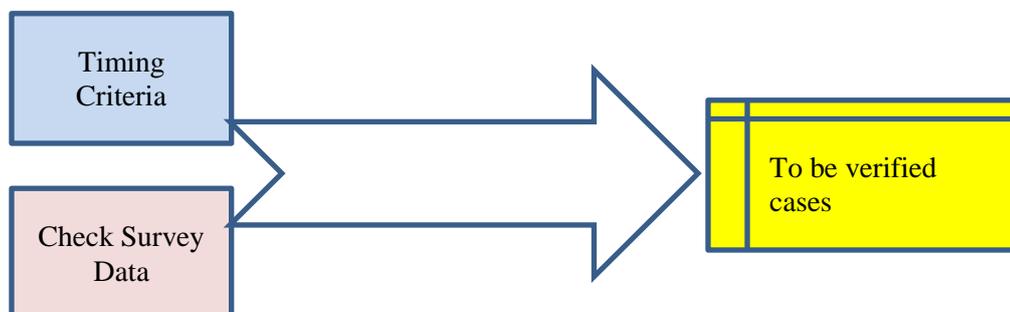
5. Using paradata for quality control

Traditionally, quality control procedures for some survey research organizations are categorized into three steps:

1. Verification by quality control team or verifiers;
2. Evaluation or live-monitoring of interviews;
3. Real-time data-driven assessment by project managers.

To reduce the cost of evaluations or verifications, and to increase the efficiency of the quality control process, many projects have been using Blaise ADT files to identify cases for evaluation or verification. Figure 5 provides a simple illustration of how paradata such as timing and review of the survey data might be used to efficiently select cases for verification.

Figure 5. Using Blaise ADT files to select “to-be verified” cases



The following examples are types of timing criteria that might be included in a paradata review:

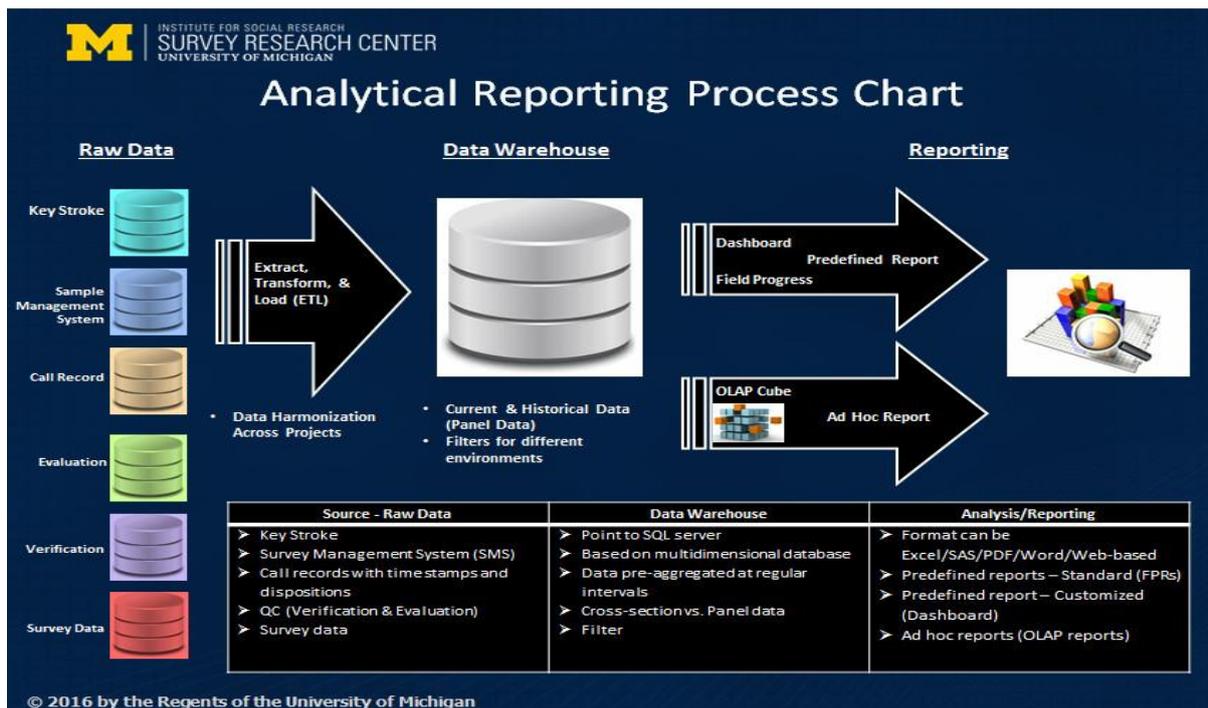
- Deviations from a pre-determine minimum time required to complete each module
- Interviewers who do not ask questions in a standard manner (such as any question read < 1 second)
- Long pauses within the questionnaire

In checking the survey data, review of paradata might include high rates of missing data or “No” responses to the following types of questions:

- Key Questions (most important questions)
- Sensitive Questions (%missing values)
- Branching Questions (%taking shortcut)
- Subjective Questions
- Vignettes (min time)

By using rich paradata, SRC at Michigan has created a QC reporting process. Figure 6 shows data warehouse extracts including not only Blaise (keystroke) data but also rich paradata including call records, sample management information, interview verification, and interviewer evaluation data. The automated system transforms the raw data, creates a relational database, and parses each file into SQL server columns. Eventually, the data are securely stored on the centralized server and can be more easily joined to other data sets and analyzed. Paradata are aggregated to create predefined field progress reports and the static quality control reports. Using the Online Analytical Processing (OLAP) Cube to access the data warehouse allows for fast and flexible queries.

Figure 6. UM SRC Analytical Reporting process Chart



6. Using paradata for improving web questionnaire design

With web surveys, research shows that there is a strong relationship between the presentation of the questionnaire and how respondents answer the questions. Even minor changes in the visual layout of the survey question can affect the way respondents answer. Therefore, it will be helpful for questionnaire design purposes to know exactly how the layout (and the display on the device) potentially impacts the answer or the way the response is provided.

Paradata provides information about the respondents' behavior on the question and web browser, and collects information about the respondents' environment, such as their browsers, OS, etc. This information helps analyze the respondents' behavior on the page and better understand the way respondents construct their answers. Recently, with the increasing number of respondents taking web surveys on smartphones and tablets of all different sizes, the way a questionnaire displays or behaves on mobile devices has become its own field of research. Even within the same brand of smartphone, there might be differences in screen layout due to screen size, specific version of operating system, and the browser used. This introduces new dimensions of research and complicates how to identify the relationship between questionnaire design and the respondents' behavior.

The following is a list of some key research items in regards to questionnaire design and paradata can help with research purpose.

- Break-off rates

Respondents using mobile devices seem to have higher break-off rates. They generally spend a longer time on the survey, and they exhibit lower response rates. Plausible explanations for these behaviors include:

- Frustration with answering the question like scrolling, both vertical and horizontal
- Font is too small and need pinch-in
- Too much text to read on the small screen
- Respondents' internet speed on the phone is too slow

- Page-load time is slow on mobile
- Respondents may be in a distracting environment when taking survey on a mobile device

- **Screen Size**

With the variety of device and screen size on mobile platforms (smartphone and tablets), research communities wish to know if the answers given on a small screen device (e.g. smartphone) differ from answers given on a large screen device (e.g. laptop or desktop computer). Although recently most web surveys will use mobile style-sheets for mobile platforms like responsive design, the screen layout may still vary within a single platform (like iPhone) due to different versions of the phone, aspect ratio and sizes.

- **Grid question**

The difficulties with presenting grid/table questions on a smaller mobile device like a smartphone or small tablet are a major concern with questionnaire design. The scrolling problems in either vertical or horizontal modes create challenges for respondents and are expected predictors of higher non-response or unreliable data. A common approach is to change a grid question to item-by-item questions on the same page. Either judging by the mobile OS or using media query, the page will automatically change from a grid/table layout to question level layout on a mobile device or viewport. However, this raises the concern that the change of display may affect the data or the respondents' judgement. (Revilla, Toninelli and Ochoa, 2015) The vertical scrolling required to answer item-by-item questions on the mobile device could result in increased non-response or less reliable data.

- **Scrolling and long text (like consent statement)**

The need for scrolling can potentially impact other aspects of the survey instrument as well. It is common to have information presented in a web survey that has long text/paragraphs, like consent forms or instructional paragraphs. Will the scrolling mean require a longer amount of time to download on a mobile device? Could this, in turn, be perceived by respondents to be more burdensome? Will the scrolling result in a higher rate of early break-offs? On a mobile device, what can we use to replace scrolling? Some surveys use an accordion format for long scrolling text or grid question. Will this format produce the same data quality?

Paradata is useful information to assist further analysis on all the above issues. Especially by collecting the User-Agent-String and other client-side paradata, we can identify the respondents' operating system, browser type and name of the device. This information, with other attributes captured via paradata functions and more detailed tracking of respondents' behavior, are likely to inform methodological issues in instrument design.

7. Final observation

- Rich paradata collection is becoming the norm
- Paradata can be used across the lifecycle for design issues as well as quality control
- Using paradata for data quality control monitoring is highly effective
- Paradata analysis should be specified throughout the data collection lifecycle but should also have a dynamic component for problem exploration
- Analyzing rich paradata can require a great deal of effort; well-designed systems can make a considerable difference

8. References

Couper, Mick P. (1998). Measuring survey quality in a CASIC environment. *In Proceedings of the Section on Survey Research Methods of the American Statistical Association.*

Couper, Mick P. and Peterson, Gregg J. (2016). 'Why Do Web Surveys Take Longer on Smartphones?' *Social Science Computer Review*, ssc.sagepub.com, p 1-21.

Peng, Hueichun and Ostergren, Jason. (2016). 'Capturing Survey Client-side Paradata'. *Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam*

Wood, Lisa, Piskorowski, Andrew and Williams, Jennie. (2016) 'Transforming Survey Paradata.' *Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam*

Piskorowski, A (2016). 'Data In and Data Out'. *Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam*

The Uses of Blaise Audit Trail Files at Statistics Canada

Éric Joyal, Statistics Canada

1. Abstract

The increasing use of computer-assisted data collection methods has provided a wide scope of ongoing and timely process data called “paradata”. The “call transaction history” file created by the Blaise system contains a record for each call made by interviewers to contact the selected units and the “audit trail” which collects each interviewer’s action during the interview process are two examples. Both files have become a tremendous source of information. Until recently, “paradata” were essentially used to monitor, evaluate and report survey progress using only a subset of available information. Over the last few years, many survey researchers have demonstrated the usefulness of paradata in the data collection context and have greatly improved the understanding of this complex and evolving process. In addition, paradata are used (and continue to be used) to identify strategic opportunities for data collection improvements for which active management and responsive design initiatives are good examples. At the same time, it is also well recognized by many survey researchers that paradata can also be of great methodological use in many other survey steps prior and after data collection, for example, questionnaire design, non-response adjustment or estimation.

The main objective of this paper is to provide an overview of the historical use of the audit trail file produced by the Blaise system at Statistics Canada. It covers activities such as monitoring, support, questionnaire design, improving and assessing the survey instrument as well as the survey process which includes both operational and methodological aspects.

2. Introduction

The Blaise audit trail data records an interviewer’s interaction with the questionnaire. It shows us what fields the interview entered, what the interviewer did on the field, and how long each event took. Many special actions are also recorded, such as edits, making remarks, and changing languages. The audit trail file is produced by a dll that is included in Blaise. The raw file, called an adt file (because its file name extension is .adt), is textual, and does not directly support analysis.

Audit trail data was first activated in 2001 for CAPI social surveys applications at Statistics Canada through the insistence of the Blaise development group. There was some resistance from the operation area mainly due to the enormous amount of collected data that have to be archived and managed. The Blaise development section agreed to take the responsibility of managing this paradata source.

At the start, the main developers’ motivation was to remove internal block-level timer logic and fields from subject matter content blocks and use the audit trail files to derive timing information.

3. Timing Information

There have been several attempts trying to get timing information for our Blaise instrument. Usually, the requirement was to collect time data to provide block-level timing information, but the code could be extremely difficult to write, the requirements were not detailed in the block specifications, and the code could not be verified at any level of testing. The most frustrating thing about this awkward and un-testable code was that it would produce inaccurate data even if coded perfectly, because time spent when returning to a block which had already been “finished” would be charged to another block.

The developers knew that it would be far easier to extract block level timing data from audit trail data. The timing data would be accurate, and one time-extraction process could be used on all surveys.

3.1 Audit Trail Logs Analysis System (ATLAS)

Getting timing information from the ADT files was fairly easy and extremely efficient. It also didn't take too long to figure out that there was a lot more than accurate timing data that could be derived from the audit trail files. All this initial research and prototyping gave birth to ATLAS, our first audit trail processing system.

ATLAS was developed using Maniplus for the interface and Manipula to create the output. The first goal of ATLAS was to create timing reports at the field and block level. As reports were created, it generated some new ideas in terms of other information that we could extract and also about different ways to present the output. A prototyping approach was used, and we were very accommodating to any new feature and report requests we received. This flexibility came at a certain cost as we had to implement several layers of parameterisation to configure any new analysis.

ATLAS could be described as a three steps application. The first step is to set the survey specific parameters, which are entered through the Maniplus interface. Customization such as sub-level reporting and exclusion of blocks for the analysis was performed at that stage. At the next step, ATLAS generates and prepares the "survey specific" report-generating Manipula scripts. Finally, it runs the scripts to produce the reports. The outputs were mainly timing reports with a breakdown at the fields, the block and case level. ATLAS was also tracking edits and Non-responses.

During the early stage, ATLAS was able to extract a lot of significant and useful information in a "minimal" amount of time and effort. On average ATLAS was able to process 20 interviewing minutes per second, which we assume was satisfying. But, as we expand our usage from pilot tests to surveys with larger sample, we realised that processing time increased considerably. Request from our internal clients also increased exponentially, which resulted in several new iterations of the software and also a constant degradation in performance. We were producing more information for sure, but we started noticing that the content of certain reports were maybe more trivial than useful.

3.2 AtCetera

We took a step back and asked ourselves what really needed to be produced by an Audit trail files analysis. Basically, we needed to cut out extra and useless output, and create a solid application which could be use by everyone without any assistance. We came up with AtCetera, which is a Visual Basic interface with standard Manipula scripts used to produce the output.

The principal goals of AtCetera were to simply and quickly produce useful descriptive information from audit trail data, for any questionnaire of any survey. We simplified our approach by getting rid of the survey specific aspect that was predominant in ATLAS. The core of AtCetera is three Manipula scripts that organize the raw audit trail data, prepare that data for aggregation, and then perform the aggregation. The scripts do not need to be recompiled; they work for all surveys, for all questionnaires. In terms of performance, AtCetera processes audit trail data at the speed of 50 interviewing minutes per second, which was significantly faster than its predecessor.

For the output we kept field and block timing information as it is globally requested. All clients want this information, especially from field tests, as the timing data shows how long their questionnaire really takes. This is quantitative data. There are some special events that are tracked by AtCetera. We keep track of when edits are triggered, where answers are changed, where the language is changed, and where remarks are written. This data is more qualitative than quantitative, as it looks into areas that could indicate problems with the design of the questionnaire.

In AtCetera, there are four ways of visiting (entering and exiting) a field.

- **Entry:** A field was EMPTY when entered, and it had a value or a status when it was exited. An empty field has a normal status and no value.

- **No Change:** A field was empty when entered, and remained empty when it was exited. This either means that the field is allowed to remain empty, or that the interviewer backed out of a field without entering any data.
- **Update:** A field has a value or status when entered, and had a different value or status when exited. Normally, a previously entered answer is being changed.
- **Review:** A field has a value or status when entered, and has the same value or status when exited. Normally, the interviewer is going through this question to get somewhere else.

There are several special events that can happen during an interview which are tracked by AtCetera. This list of special events is not exhaustive; it's just the events that are tracked by AtCetera.

- **Question Level Interviewer Remarks:** By pressing F4, interviewers can record remarks for the current question.
- **Changing Language of Interview:** When an interviewer changes the language of interviewing, it is recorded.
- **Edits:** When an interviewer encounters an edit (SIGNAL or CHECK), it is recorded.

The standard output of AtCetera is a single Excel file which has three sheets. One sheet has data at the unique field level, a second sheet has data at the (data model) block level, and the third sheet explains the column headers used in the other two sheets. Using Excel lets output recipients look at the data in many different ways, using standard software.

The following example of the field level output shows all of the fields in a block called AD.

Table 1. AtCetera field level report

CapTime	DropT	Hits	LngH	RpdE	LngE	RpdN	LngN	RpdU	LngU	RpdR	LngR	Edit	Rmrk	F2	Block-ID
3079	0	310	0	5	305	8	3	4	4	68	9	0	0	0	AD.AD_Q1
705	0	80	0	5	72	1	9	0	1	20	0	0	0	0	AD.AD_Q2
3545	0	308	0	13	295	7	2	0	0	58	3	0	0	0	AD.AD_Q3
3000	22	308	1	22	286	3	1	2	0	54	3	0	0	0	AD.AD_Q4
2627	0	308	0	36	272	4	2	0	1	50	3	0	0	0	AD.AD_Q5
2404	0	308	0	39	269	1	0	1	0	50	6	0	0	0	AD.AD_Q6
2658	0	308	0	38	270	3	2	7	5	57	11	0	0	0	AD.AD_Q7
670	0	68	0	4	68	6	9	1	0	6	0	0	0	0	AD.AD_Q8
3063	0	308	0	40	268	5	8	2	1	56	4	0	0	2	AD.AD_Q9
3340	0	310	0	2	0	134	265	0	0	4	0	0	0	0	AD.AD_QINT

The next example is of the block level sheet. The AD line shows what happened in all of its fields.

Table 2. AtCetera block level report

CapTime	DropT	Hits	LngH	RpdE	LngE	RpdN	LngN	RpdU	LngU	RpdR	LngR	Edit	Rmrk	F2	Block-ID
24091	22	2616	1	204	2100	172	301	17	12	418	39	0	0	2	AD
159292	305	21900	6	4969	14430	1805	1697	234	202	3627	246	43	7	4	AL
154455	6393	13265	5	441	12822	202	195	124	133	2002	199	2	1	17	ALS

4. Questionnaire evaluation and design

The qualitative aspect of the audit trail data quickly gained some interest in the question design, operation and methodology fields. Some studies were performed to learn on how questionnaires were used by the interviewers, while others were focusing on how to improve and/or correct the design of a questionnaire.

4.1 Adapting questionnaire based on Interviewer Interaction

You can learn a lot about your questionnaire based on how the interviewers use it. By analysing the audit trail files and looking at numerous actions, you can derive information that could help you improve your design.

4.1.1 Remarks

In the course of an interview, the interviewer has access to a Remark window, in which he can capture “parallel” information provided by the respondent that is not part of the questionnaire itself, but that can help survey analysts in understanding the content of the data. The capture of remarks by the interviewer can be a good source of information about the questionnaire design, the data quality, or characteristics of the respondent. The fields that often necessitate the addition of a remark can indicate that a question was not clear or that the choices of response categories provided to the respondent did not cover some potential responses. Also, having numerous remarks for a given respondent might provide information on the respondent. For example, a woman that has just recently delivered could feel the need to justify her responses to a set of questions about her level of physical energy by her “temporary situation”.

4.1.2 Help

Fields where the Help key is frequently used could indicate that a question is not clear. Most studies are showing that very few problematic fields can be identified using this concept. In fact, we observed that the Help key is very rarely used. This may indicate that the interviewers have a good knowledge of the survey content and procedures, and hence rarely need assistance, but it could also mean that the interviewers do not make use of the information available in those screens for other reasons. Perhaps they are not aware that Help screens are available, or perhaps they don't found them to be useful.

4.1.3 Language change

Changing the language in which the interview is conducted during the course of an interview between French and English can indicate that the respondent or the interviewer was seeking a clarification on the question wording, perhaps not fully understanding what was asked. Fields that often necessitated a change of language can indicate a problem with question wording.

4.1.4 Flow

Analyzing the flow of the questionnaire can shed some light on potential problems with the question wording or questionnaire design. A “perfect” interview that goes from beginning to end without any backward navigation in the questionnaire should consist only of visits of the “Enter” type. That is, each field would be empty when visited for the first time, filled on the first visit, and never visited again. By exploring fields that do not seem to follow such a pattern, or blocks with many such fields, one could highlight problematic areas of the questionnaire. However, it can be difficult to define what should be considered problematic, and to target the source of the problem. Fields or blocks that seem to have “outlying patterns” could be investigated. These outputs could also be used to assess specific issues that might have been raised up by interviewer during collection.

4.1.5 Multiple visits

A high proportion of visits that are not the first visit to the field indicates interviewers often had to revisit the field, either for updating it, or to move to previous or following fields.

4.1.6 Proportion of time spent on Enters

Fields with a low proportion of interview time spent on visits of the type “Enter” can give an idea of the magnitude of “wasted interview time”. Note that the proportion of visits that are not of the type “Enter” should be highly correlated with the proportion of multiple visits, since in theory, each respondent to a field should not have more than one visit of the type “Enter”.

4.1.7 Updates

Fields that often necessitated to be updated could indicate a problem with the question wording, the definition of a concept, or an edit rule that would be too strict.

4.1.8 Rare visits

Fields visited by very few respondents could indicate a problem with the skip patterns defined in the question, if the question was not designed to be addressed to a rare population. Note that fields that are never visited cannot be identified solely with the Audit Trail data. To identify these, a list of all existing survey fields would have to be available.

4.1.9 Long visits

Visits that last more than 180 seconds should be rare. A field with a high proportion of Long Visits should be investigated.

5. Monitoring

As we spent time looking at how interviewers were using the Blaise questionnaire in order to improve our design, we realised that some level of monitoring could be done on the interviewing task.

5.1 POInt System

Interviewers are trained to maintain a moderate pace while interviewing, to read all questions at the same pace, and to wait for the respondent to answer. The POInt (Pace Of Interviewing) system uses audit trail data to determine if interviews are, in fact, being conducted in that manner.

Blaise audit trail data contains a wealth of information on the interviewing process. For each field that is traversed, the time and state of the field is recorded, as it is entered and as it is exited. That data can be used to determine what happened to the field, and how long it was active. By evaluating the field level data for the subject matter portion of a data collection application, and comparing it against criteria for that application, it is possible to identify calls that are irregular (where the fields were traversed too rapidly).

CATI (Computer Assisted Telephone Interviewing) interviewers at Statistics Canada have been subject to unobtrusive monitoring for many years. CATI monitoring is performed on a fraction of calls, and only a fraction of a call is monitored. CATI monitoring evaluates an interviewer subjectively, in ways that cannot be done by computers. The monitor determines if the interviewer is being polite, is speaking clearly, is asking the question as worded, and is entering the answer as given.

POInt evaluates all calls, but its evaluations are objective, rather than subjective. POInt is not a replacement for CATI monitoring, as it cannot do the subjective evaluations. It is a complement to CATI monitoring.

5.1.1 POInt Genesis and development

In the summer of 2006, we looked into a small number of interviews which seemed to have been done at a rapid pace. A manual review of the audit trails of several interviews showed that some of them had been completed quickly. A more thorough investigation was clearly needed, as there might have been other irregular interviews. It was not feasible to review the 8,000 audit trail files for this survey by hand, so a computer program was developed to automate the process. This was the infancy of the POInt system.

This original program determined how many subject matter fields had been changed, how much time had been spent in those fields, and how many of the fields were non-response. Pace and item non-response rate were determined for each call. It was noticed that this base code could be used on any of our social surveys, with only minor changes (thanks to our Blaise coding standards). We soon had generic code that could be used to evaluate calls from any social survey application.

We were very fortunate in some of our existing practices. Audit trail data was already being returned from regional offices to head office on a nightly basis. There was an expandable report interface that was used by regional office staff (Data Integration and Production Planning system - DIPP). We had the ability to process data daily, and an existing portal through which POInt reports could be delivered. Having determined that full scale production was viable, development for a generic POInt evaluation and reporting system began in late spring 2007. POInt started being used in production in December 2007.

At first, a single “speed limit” for all collection was thought to be appropriate. The study of historic audit trail data quickly disproved that notion: surveys are collected at surprisingly different paces. Clearly, each survey needed its own speed limit. Our early thoughts on item non-response also needed to be updated. Initially, the boundaries for item non-response were too low. Non-response is almost always a reflection of the respondent, rather than the interviewer. Only extremely high item non-response should cause a call to be flagged.

At the start of development, evaluation criteria were hard-coded. Individual survey level scripts were considered, but it was quickly noticed that the only lines of code that would change would be those that defined the criteria. The most obvious mechanism to record the evaluation criteria for a survey was initialization files; the use of survey-specific initialization files removed the need for survey-specific process control scripts.

Each survey (application) in POINT has an initialization file, which tells the standard programs how to identify content fields, which field is used to determine whether the case is a full-complete, and what the evaluation criteria are. An example follows.

Listing 1. POINT Example Source Code

```
[PointMode]
Parms=Known
CallsToSet=600

[Dates]
StartDate=20160101
EndDate=20201231

[ContentDef]
ContentField=Content.
```

```
LastField=Content.SO_N01
```

```
[Criteria]
```

```
AvePace=6.7
```

```
MaxPace=11.7
```

```
MaxINR=25.0
```

```
MinFCH=20
```

```
[ProcessControl]
```

```
MakeUFICReport=Yes
```

```
CopyReportsToArchive=Yes
```

```
CopyMATToArchive=Yes
```

In the *PointMode* section, we can see that the parameters have been established (*Parms=Known*). The *Dates* section can be used to restrict processing to files from certain dates. This feature has only been used during testing, and is not a universal restrictor: some programs do not have any *StartDate* or *EndDate* logic.

In the *ContentDef* section, the content superblock is *Content* (*ContentField=Content.*), and the field used for auto-coding full-complete outcomes is *Content.SO_N01*.

In the *Criteria* section, we can see that the early average pace of content collection was 6.7 field changes per minute (fcpm), which is another way of saying “questions per minute”. The maximum pace has been established at 11.7 fcpm, and the maximum item non-response rate (INR) is 25.0% (this is a default value). We can also see that at least 20 content fields have to be changed for a call to be evaluated (MinFCH=20).

In the *ProcessControl* section, we can control whether UFIC (*unusual fields for irregular calls*) reports are generated, and whether reports and audit trail files are copied to the archive. During POINT testing, setting these controls to “No” will allow for faster and simpler repetitions.

Before being able to generate its reports, POInt needs to create a processable data file from the Blaise audit trail file. The first script will produce a field by field (FxF) file which contains one record for each visit to each field. The following is an example of the FxF data file.

```
20080123,15400702171,1,intrvwr,YITS.U.U_Q01[1],178,2,,N,1,1,2,1,N,7,1300,21:40,  
20080123,15400702171,1,intrvwr,YITS.U.U_Q03[1],179,2,,N,1,1,2,1,N,5,1305,21:45,  
20080123,15400702171,1,intrvwr,YITS.U.U_Q04,181,2,,N,1,1,2,4,N,8,1320,22:00,  
20080123,15400702171,1,intrvwr,YITS.U.U_Q04a,182,2,,N,1,1,2,2,N,2,1322,22:02,  
20080123,15400702171,1,intrvwr,YITS.U.U_Q07,183,2,,N,1,1,2,2,N,7,1329,22:09,  
20080123,15400702171,1,intrvwr,YITS.U.U_Q09,184,2,,N,1,1,2,1,N,8,1337,22:17,  
20080123,15400702171,1,intrvwr,YITS.U.U_Q37B,185,2,,N,1,1,2,2,N,9,1346,22:26,  
20080123,15400702171,1,intrvwr,YITS.U.U_Q59,186,2,,N,1,1,2,2,N,7,1353,22:33,  
20080123,15400702171,1,intrvwr,YITS.U.U_Q60[1],187,2,,N,1,1,2,6-4-2,N,21,1374,22:54,  
20080123,15400702171,1,intrvwr,YITS.U.U_R01,177,2,,N,1,2,2,,N,2,1293,21:33,  
20080123,15400702171,1,intrvwr,YITS.U.U_R04,180,2,,N,1,2,2,,N,7,1312,21:52,
```

Next step, using the FxF as input, is to create a call by call (CxC) file. This is an example of the content of a CxC data file. This represents one call made by one interviewer. It is for the same interview that was used to provide the previous example.

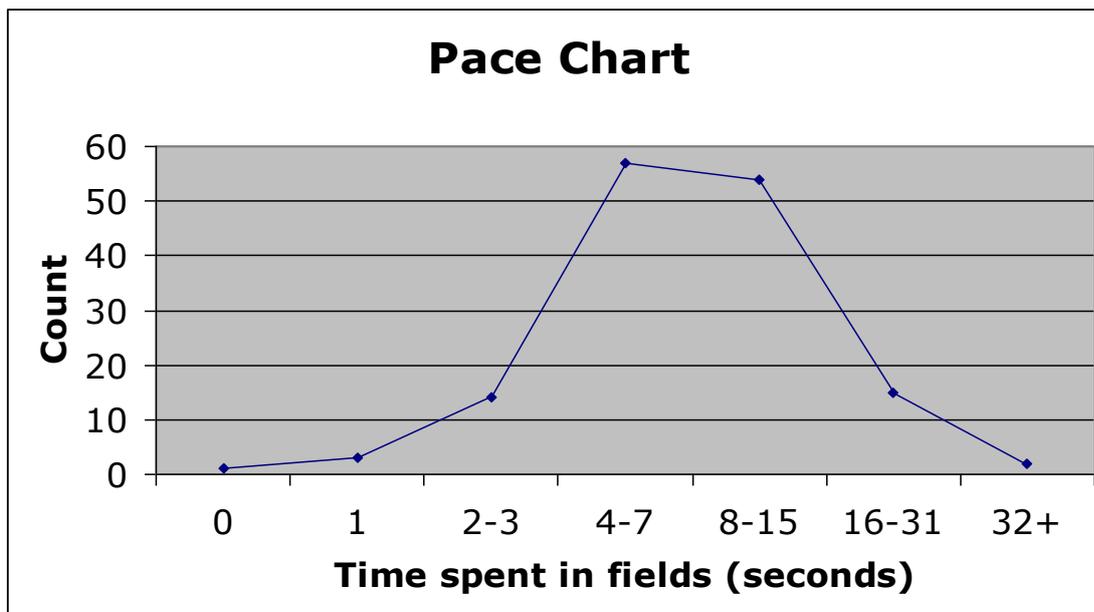
```
20080123,15400702171,1,intrvwr,159,6,146,0,0,0,0,1273,1210.4,6.9,7.2,0,0,0,0,1,3,14,57,54,15,2,1
```

To the experienced eye, this is a very readable set of data. It starts with the *Date*, *CaseID*, *CallID* and *InterviewerID*. 159 fields were visited. 6 field-visits were considered unusual, and 146 fields were changed. There was no item non-response: 0 DK, 0 RF, 0.0% item-non-response rate (0,0,0.0).

A total of 1,273 seconds were spent in content fields that changed. The amended time for these fields is 1,210.4 seconds. The actual pace of collection was 6.9 field changes per minute (fcpm), while the amended pace was 7.2 fcpm. A call is evaluated on amended pace.

There were no extra-long-visit fields. A field-visit must last at least 180 seconds to qualify as an “extra-long-visit field”. The next few numbers (1,3,14,57,54,15,2) are frequencies of how many fields were active for certain time periods. In the following chart, you can see that most of the questions were active for between 4 and 15 seconds.

Figure 1. Pace Chart



The last data item shows that the last question was answered. This was a full-complete interview.

The last step is to run the scripts that create the POInt reports.

5.1.2 POInt Reports

POInt produces report data in the form of comma delimited text files. These files are sent to the DIPP system, where they are converted into a series of inter-related Microsoft® Excel spreadsheets. This allows report users to manipulate report data using familiar software. They are able to drill into more detailed reports through hyperlinks in the spreadsheets.

DIPP reports are role sensitive. Regional office collection managers can see details about individual interviewers, while head office staffs get no interviewer-level detail.

Summary of Calls by Survey is the highest level report. It shows the number of evaluated calls by survey and collection period, with counts of regular and irregular calls.

Survey Calls by Interviewer summarizes the performance of all interviewers who worked on a specific survey. It shows counts of regular and irregular calls, amended pace, and field-level non-response rate. A somewhat related report is *Interviewer Calls*, which has the same report content for all the surveys of one interviewer.

The *Irregular Calls* report has information for each irregular call in a given survey and collection period. Interviewer-id, pace and field-level non-response rate are the major items of interest.

Finally, the *Unusual Fields of Irregular Calls* report shows, as its title implies, all of the “unusual fields” that were encountered during an irregular call. This report allows managers to better understand why a call was flagged as irregular, and provides a solid set of information that can be used during a discussion with the interviewer. An “unusual field” is one that: was changed in less than two seconds; or was changed to Don’t Know (DK)/Refusal (RF); or had an interviewer comment; or was active for at least 60 seconds.

Collection managers should use POInt reports to coach interviewers towards better performance, and to provide positive feedback for good performance. A fact that leaps out from looking at these reports is that most interviewers do an excellent job, every day.

5.1.3 POInt Status

POInt has been in use since December 2007. Almost all CATI social surveys now use POInt. To date, about 1% of calls have been evaluated as irregular. There have been instances where offices had more than 1,000 calls for a survey, with all calls evaluated as regular. In 2009, POInt was expanded to evaluate very short interviews. Rather than evaluating individual units of work (calls or questionnaires), a body of work (an interviewer’s shift) is evaluated.

6. Centralisation and standardization

With audit trail data being used extensively throughout Statistics Canada, a need to centralise and standardise the audit trail process was identified.

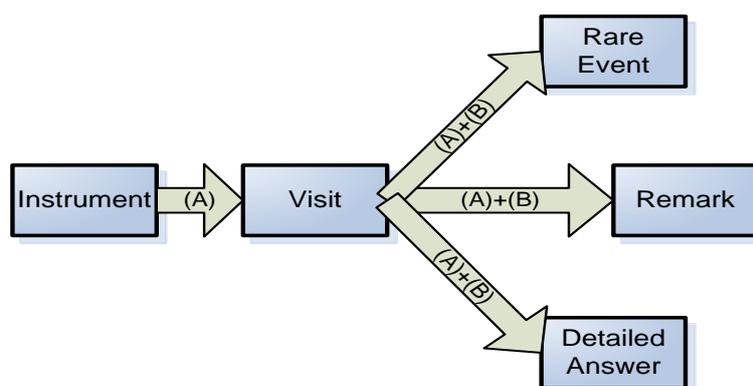
6.1 Standardized Audit Trail data

In practice, each visit to a survey question generates about 180 characters on the audit trail file. This format does not allow for direct and friendly use and analysis of the data. In order to fully benefit from this enormous and detailed amount of data, the raw audit trail files were transformed into a more structured and coherent format using a standardised process that can be applied to all surveys.

The standard storage structure for processed audit trail data for all CATI and CAPI surveys was influenced by concerns for ease of use, concerns about storage burdens and data base normalization theories. The standard storage structure is an inter-related set of five (.txt flat file) outputs.

In the early discussions that led to this proposal, it was agreed that the structure should hold only that data which would be required by more than one group. Unique needs would be handled by giving the group which had the unique requirements access to the raw audit trail files.

Figure 2. Standardized output



The Output Tables can be linked together using (A) “Uniqueinst” and (B) “VisitSeq” variable.

The **Instrument Table** contains summary information about the use of the collection instrument. Each combination of *Enter Form* and *Leave Form* corresponds to one record on the Instrument Table. This table records what instrument was used, who used it, when it was used, which sample record was involved and the file name from which the data are coming from. The table is sorted by SampleID and in chronological order. The table can be linked to the Visit Table using the variable *UniqueInst*.

The **Visit Table** contains field-visit level data. There is one record for each visit to each field. A visit to a field represents the sequence of actions that are performed between the moment a field is entered (keyword *Enter Field*) and the moment the following field is entered (through another *Enter Field*). The Visit Table contains a visit sequence identifier, the field name, the value when entering the field and the value when leaving the field, the type of visit and the duration of the visit. Very long field visits in terms of duration are capped to 18000 seconds (5 hours), to be consistent with BTH outlier treatment.

The Visit Table can be linked to the Instrument Table using the variable *UniqueInst*, and to the Rare Event Table, the Remark Table and the Detailed Answer Table using the combination of the variables *UniqueInst* (A) and *VisitSeq* (B).

Each visit can be categorized into one of 5 “types of visit” (variable *Visit_Type*), based on the value when entering the field (either blank or not) and the value when leaving the field (either value modified or unchanged during the visit).

The main aim of this variable is to help to quickly have an idea of how the interviewer is navigating in the application, that is:

Table 3. Type of visit

Type of visit	Value when entering field	Value when leaving field	Interpretation
1: Entry	Empty	Value	Initial capture of a value for the field. An interview that would go from start to finish without any back and forth in the questionnaire should consist of visits that are all of this type (except if the survey has pre-filled fields).
2: Update	Value	Different value	Modification to previously reported value
3: StillBlank	Empty	Empty	Leaving a field without having entered a value. Usually because the interviewer goes back into earlier fields.
4: Review	Value	Same Value	Moving upward or downward in the application, through fields that were already responded to
5: Truncated record	In some rare instances, problems with the survey application (e.g. "frozen application") can result in incomplete Audit Trail data, which makes it impossible to determine the value at entry and at exit with certainty.		

To minimize the size of the Visit Table, the two variables containing the answers to the survey questions (*EntryValue* and *ExitValue*) are only 8-digits long. When a value is either a numeric value of more than 8 bytes or a character value, only the length of the value is stored in the Visit Table (if needed, the initial complete values can be retrieved from the Detailed Answer Table).

The values in the Visit Table are enclosed in one of three types of brackets, depending on the type of data it contains. That is:

- **Square brackets []**: Value reported by the respondent. This can be either a numerical value with length <= 8, or *Don't Know* and *Refusal* (stored as [??] and [!]), or blank values (stored as [])
- **Curly brackets { }**: Length of numerical values when length is greater than 8

- **Parenthesis ():** Length of character value

Below is an example of how different values found in the raw data would be stored in the Visit Table. It also shows what the value of *Visit_Type* would be for the different values at entry and at exit. Note that *Visit_Type* is derived from the raw values, not from the values found in the Visit Table (for example, a change from 1-4-9-13-17 to 1-5-9-14-18 results in *Visit_Type* = 2: Update, even though both values appear as {11} in the Visit Table).

Table 4. Types of brackets used to store different types of values in the Visit Table

Values in the raw ADT files		Values in the Visit Table		
Entry	Exit	EntryValue	ExitValue	Visit_type
	1	[]	[1]	1: Entry
Don't Know	Refusal	[??]	[!]	2: Update
		[]	[]	3: StillBlank
Ottawa Region	Ottawa Region	(13)	(13)	4: Review
1-2-3	1-2-3-6-9-13	[1-2-3]	{12}	2: Update
	K1A 0T6		(7)	2: Update
2-8-9	2-6-8	[2-8-9]	[2-6-8]	2: Update
1-4-9-13-17	1-5-9-14-18	{11}	{11}	2: Update
1-2-3-6-9-13	Don't Know	{12}	[??]	2: Update
9 St-Denis	209 Saint-Denis	(10)	(16)	2: Update
1-2	One or Two	[1-2]	(10)	2: Update

The **Rare Event Table** contains information about rare events that happen during collection. It is understood that storing these rare data in a separate table should significantly reduce the size of this data base. Rare events are things like use of the Help key, changing the language of the CAI application, entering field-specific remarks and interviewer's action after an edit failure.

The interviewers' reactions to edit failures are also recorded on the file. The interviewer can react in three different ways. The interviewer can press the Escape key or 'Close' button, which will make the application go back to the field that was just left, allowing the interviewer and respondent to make the necessary modifications to previously reported fields in order to correct the inconsistency (*Action: Error Escape*). The interviewer can also press the Suppress button that appears on the pop-up window, which will make the application move on to the next field (*Action: Error Suppress*). The Suppress button is only available for soft edits, since hard edits detect impossible values and are not allowed to be ignored. Finally, the interviewer can press the 'Goto' button (*Action: Error Jump*). For *Error Jump*, the application will return to the field highlighted in the edit pop up window.

The **Remark Text Table** contains the remark text entered by the interviewer about specific field. There should be one record for each record of the Rare Event Table that has *RmrkChng* = 1.

The **Detailed Answer table** contains the complete values reported by the respondent when only the length of the value was kept in the Visit Table. This table aims at preserving as much as possible the information found in the raw Audit Trail files.

The Detailed Answer Table contains the complete reported values for such cases, with one record for each visit of the Visit Table that has the length of the value instead of the value itself (at entry and/or at exit). All values in the Detailed Answer Table are stored in square brackets. This is consistent with the convention used for the Visit Table, since all values in the Detailed Answer Table are values reported by the respondent.

Below is an example of how different values found in the raw data would be stored in the Detailed Answer Table. The records of the Visit File that have both *EntryValue* and *ExitValue* enclosed in square brackets (i.e., both are values reported by the respondent) do not have a corresponding record on the Detailed Answer Table.

Table 5. Detailed Answer Table

Values in the raw ADT files		Values in the Visit Table		Values in the Detailed Answer Table	
Entry	Exit	EntryValue	ExitValue	Detailed EntryValue	Detailed ExitValue
	1	[]	[1]		
Don't Know	Refusal	[??]	[!]		
		[]	[]		
Ottawa Region	Ottawa Region	(13)	(13)	[Ottawa Region]	[Ottawa Region]
1-2-3	1-2-3-6-9-13	[1-2-3]	{12}	[1-2-3]	[1-2-3-6-9-13]
	K1A 0T6		(7)		[K1A 0T6]
2-8-9	2-6-8	[2-8-9]	[2-6-8]		
1-4-9-13-17	1-5-9-14-18	{11}	{11}	[1-4-9-13-17]	[1-5-9-14-18]
1-2-3-6-9-13	Don't Know	{12}	[??]	[1-2-3-6-9-13]	[??]
9 St-Denis	209 Saint-Denis	(10)	(16)	[9 St-Denis]	[209 Saint-Denis]
1-2	One or Two	[1-2]	(10)	[1-2]	[One or Two]

7. Support

As a developer, the primary use of the audit trail files is predominantly for support purposes. It helps in the testing and the debugging of the Blaise code. Audit trail data is usually the first source of information we will look at when issues are reported during testing and collection. It will generally provide more information than what will be sent to us in the issue description. The adt files allow us to troubleshoot the issues more efficiently and effectively. And this is just the tip of the iceberg, this last section will briefly highlight other technical uses of the audit trail data throughout the last decade.

7.1 Automated Testing

Testing can be a repetitive and tedious task. Finding ways to improve and simplifying it will always be at the centre of development practices discussions. Automated testing is an area where we feel we can make some gain in our quest to streamline the testing process. Different initiatives were put in place, and one of them used audit trail files.

Some recurrent surveys utilise the same pre-defined testing scenarios rounds after rounds to perform some level of regression testing. This set-up allowed us to develop a custom made “replayer” which would receive audit trail files from a previous round as input and then “replay” the scenario to perform regression testing. This was saving a lot of keying time for the tester and we were making sure that the same testing situations were applied between rounds.

However, this strategy had its limitations; for instance, a change in flows, randomization in the questionnaire, or the addition or the deletion of a question resulted in unmanageable situations for our replayer program. More development time would have been required to make this process more robust, but it has been put on hold for resourcing issues.

7.2 Data recovery

As documented, the audit trail file also offers a means for data recovery in case of technical glitches, power shutdowns/failures or improper Blaise coding. We did make use of this feature occasionally. It has been a good emergency plan when for a short period of time we experienced some databases corruption issues throughout several applications. During that time, several cases were dropped during our overnight process from the databases after the execution of the Hospital recovery program. While we were troubleshooting the cause of this issue, we used audit trail files to repopulate the dropped cases in the databases applying an adapted version of the strategy supplied with the Blaise software (AuditSummary.man and Repopulate.man). The plan in place allowed us to automate the data recovery process and concentrate our effort on identifying the source of the problem (which was an issue with the Hospital program).

Audit trail files saved several interviewing hours for another of our CAPI project. The Longitudinal and International Study of Adults (LISA) consists of a very complex datamodel. For the wave conducted in 2014, we realised once in production that in certain circumstances part of the data wasn't kept when the interview required multiple visits. A KEEP method was missing in our source code and a patch was required and sent out in the field. Unfortunately, it took a while before realising this problem, and the damage was already done. Over 1,000 cases were affected by this programming error and our internal client initial thought that this data was gone and irrecoverable. Using the adt files generated by Blaise, we were able to recuperate the missing values by developing a process to recreate the output required by our internal client.

7.3 Performance metrics

It's not always easy to assess the performance of a system. To be able to determine the impact of a change on the infrastructure and/or inside the Blaise application we often need to trust the eye test. In the last years, we went through several infrastructure changes (Vista, Windows 7, VDI, Blaise services, etc.) and some resulted in perceived performance degradation based on interviewers' feedback. We always felt that we needed a way to baseline the performance and have metrics to properly evaluate the effect of a change on the system.

The audit trail files can, to a certain extent, provide valuable metrics to assess performance. For example, we are using the question to question delays recorded on the audit trail files (calculate to the thousandth of a second as we are using the PreciseTiming option in the AIF file) to calculate system responsiveness. The assumption is that the faster the system responds, the better the performance. Using years of audit trail data from our Labour Force Survey, we've been able to come up with good indicators and determine more accurately the performance impact of a change in the collection system and/or on the infrastructure.

8. Summary

Even before implementing the audit trail files functionality into our projects we knew how useful it would be to technically support our applications. Being able to derive precise timing information using the ATLAS and AtCetera system was already a big plus of having this feature enabled, but it turns out to be only the beginning of an interesting journey. We improved our data quality by enhancing our questionnaires design practices and by monitoring objectively our interviewers using the POInt system. We now have a centralized and standardized approach set up for audit trail file analysis, which should translate in new ways to utilise this vast and rich source of information.

9. References

Ali, Jennifer, “Data Quality Monitoring Using the Blaise Audit Trail”, 2003, SSC Annual Meeting, June 2003, Proceedings of the Survey Methods Section.

Egan, Michael D.A., “POINT – A Method for Evaluating the Quality of Interviewing”, 2010, Statistics Canada internal documentation.

Hamel, Sylvain, “Audit Trail Files (ADT) – Structured Output”, 2014, Statistics Canada internal documentation.

Lapierre, Bruno and Meyer, Scott, “Using the Audit Trail data to evaluate the quality of collection of the Canadian National Longitudinal Survey of Children and Youth”, 2006, American Statistical Association 2006 Proceedings of the Section on Survey Research Methods.

Capturing Survey Client-side Paradata

Hueichun Peng and Jason Ostergren, University of Michigan Survey Research Center, United States

1. Abstract

Paradata that are captured during the survey process are a valuable source of information in helping us understand and improve the data collection process.

One of the advantages of using Blaise for survey data collection is the rich capture of paradata that is native to the application. Paradata which are linked directly to the administration of a survey instrument are collected automatically through the Blaise software (i.e., audit trail). The ADT file from Blaise 4 has been very valuable in understanding interviewer behavior. With the advent of Blaise IS and Blaise 5, we now are able to increase the richness of our paradata collection for understanding respondent behavior on web-SAQ (self-administered questionnaires) and/or mixed mode projects (i.e., interviewer and web-SAQ combined).

The team at the University of Michigan has implemented a process to capture paradata from the client-side using JavaScript. This client-side paradata (CSP) can capture movement within a page on the respondent's or interviewer's device like scrolling and mouse-clicks. As research interest has been strong in using these paradata to analyze respondents' behavior, we are working to capture more events like "loss of focus" and "geolocation". Furthermore, as more and more respondents start to use mobile device to take web surveys, we added a new library to capture the events specific to mobile device such as tapping, orientation change, pinch-in, etc. As the University of Michigan has been investigating using tablets (mobile) for CAPI interview due to cost and efficiency consideration, the capture of mobile paradata will be crucial for analyzing interviewer's behavior as well.

We will share what we have learned and the challenges we have considered, with the intent that other Blaise 5 users can also supplement their paradata capture.

2. Introduction

Paradata that are captured during the survey process are a valuable source of information. They help researchers better understand the respondents' behavior and their device. They also have the potential to improve the data collection process.

There are two main types of paradata collected in web surveys: server-side paradata and client-side paradata. The terms "server-side" and "client-side" are based on the technical concepts in web programming that the scripts or functions are primarily triggered and executed at the web server (server-side) or the web browser (client-side). Server-side paradata usually tracks the respondent's *visit* to the web page, capturing timestamps, the respondent's unique identifier, question items on the page, question answer provided when R moves to next page, browser type, user agent string (UAS), etc. Client-side paradata records the respondent's actions within a specific page by means of JavaScript. Generally it records the *timestamp* (timespan) together with the actions: for example, when the respondent clicks a radio button, enters some texts, unclicks the radio button, scrolls the browser window, etc.

These paradata are collected from web surveys deployed either as a web-SAQ (self-administered questionnaires) or an interviewer-administered SAQ.

Currently, the primary web survey platforms the Survey Research Center has been using include Blaise and Illume. For each, we have added extra scripts and functions to capture the server-side and client-side paradata.

With the ever increasing number of respondents taking the web surveys on mobile device and newer browsers equipped with HTML5 features, we decide to enhance our client-side paradata to include

events specific to mobile platforms and other events that could provide more information about respondents' behavior and their device.

This paper focuses on the client-side paradata (CSP) enhancement we have worked on, mostly related to mobile platform and enhancements on various areas. It describes the proof of concept we tested experimentally, the implementation with Blaise-5 instruments, and other enhancements we have been working on. Detailed implementation of the last version of our CSP JavaScript functions in BlaiseIS can be found in the paper presented at the 2010 Blaise conference (Ostergren and Liu, 2010).

3. Events on Mobile device – a Proof of Concept

In 2015, we added a new JavaScript library of functions in a cross-campus web survey (on the Illume Web Survey platform) in addition to the original CSP JavaScript library we used to collect CSP. The new library of functions aims to capture new events mostly related to mobile platform, including the screen size at page-loading time, tapping, pinching (in and out) and pinch distance, and orientation change. The JavaScript functions are primarily based on the following JavaScript events:

- Window Screen size in pixels when page is loaded: Record the `window.screen.height` and `window.screen.width`.
- Tap: Add listener to capture the Touch (OnTouchStart) event. We identify Tap event when the `touches.length = 1`
- Pinch: Add listener to capture the Touch (OnTouchStart and OnTouchEnd) event, and calculate the distance between the start point and end point
- Orientation change: The original orientation can be inferred from the starting screen size. Add listener to capture the `window.orientation` properly to identify the new orientation.

The new library **only** recorded specific events that mostly happen on mobile platform and did not record the basic events logged by our original library. Our plan was to compare the data from the two sources and merge them into one library after testing and validation. We decided to incorporate this testing into this project with cross-campus surveys for which the majority of the respondents are college students.

Below are examples of the raw data captured in comparison with the CSP captured from the original library of function.

- **Example-1 from iPhone:**

- CSP from the new library is as below.

414,736;landscape,portrait,tapped,tapped,19.977213593219687,pinch_together,tapped,tapped,tapped,tapped,tapped,tapped,tapped,tapped,tapped,tapped,

Data Notes:

- *414,736 -> screen size in pixels.*
- *landscape, portrait -> change of orientation*
- *99999 pinch_together -> pinch together by the distance of 9999*
- *tapped -> any touching event recorded*

- The corresponding CSP from the original library is as below.

^t=40290:endScrollForCSP(x=false,y=true)^t=123116:Q39=4^t=956:endScrollForCSP(x=false,y=true)^t=1005:Q41=3^t=723:endScrollForCSP(x=false,y=true)^t=23

68:Q42=4^t=988:endScrollForCSP(x=false,y=true)^t=3685:Q68=1^t=23480:DATSTAT.NEXT=[CLICK]

Data Notes:

- ^t: milliseconds after page-load
- endScrollForDCP (x=false,y=true) -> scrolling action at either x or y direction
- Q39=4 -> Users enters response 4 for Question ID Q39
- DatStat.Next = [Click] -> Users click the Next button

- **Example 2 from Android (HTC One Max):**

- CSP from the new library is as below.

432,768;landscape,tapped,tapped,14.09574790846645, pinch_together,portrait,tapped,tapped,

Data Notes: explained as the above

- The corresponding CSP from the original library is as below.

^t=5218:endScrollForCSP(x=true,y=true)^t=1154:STUDQUES10A_2015=4^t=708:DATSTAT.NEXT=[CLICK]

Data Notes: explained as the above

This experiment suggests that the new tracking data provide valuable information about our users' behavior on a Mobile OS. We decide to consolidate the 2 libraries by merging the mobile event listener into the original CSP JavaScript functions. We implement the new library to a production project that uses Blaise-5 Web Surveys.

4. Implementation on Blaise-5 Web Survey

The following is a discussion of the details of our Blaise 5 client-side paradata (CSP) implementation, from the use of the Data Entry API to the JavaScript we use. Generally speaking, there are three necessary parts to developing a scheme to capture CSP. These are:

1. Developing a JavaScript to capture the CSP on the client browser.
2. Developing a mechanism to save the captured CSP upon server contact during page change.
3. Developing a mechanism to update the captured CSP string with information about the current state that cannot be reproduced later (namely replacing temporary and possibly reused element identifiers from the html with related fieldnames, taking advantage of the server-side ability to do that translation).

The JavaScript we are using to collect Blaise 5 CSP is a further refinement of the script described in the IBUC 2010 paper by Jason Ostergren and Youhong Liu entitled "BlaiseIS Paradata." The main purpose of our CSP JavaScript is to capture keystrokes and mouse clicks on question elements such as text boxes and radio buttons, and to capture other actions that may take place in the intervening time, such as scrolling and pinch/ zoom. Capturing keystrokes and clicks may show, for example, that the user changed answers many times before leaving the page, possibly indicating a problem with the clarity of the question or task. Incidentally, that information is also available to some extent in the keystroke-level paradata provided with Blaise 5 out-of-the-box. Capturing scroll activity and zooming gives us an idea of how often the content exceeded the viewable area and how much effort

was required for the user to absorb the content of the full page, alerting us to whether it was formatted for efficient viewing. This type of information is not provided in the out-of-the-box paradata. The inclusion of both types in our CSP script (keystroke-and-click-type CSP and scroll-and-zoom-type CSP), gives us more information about the order of events on a page, since one can determine which typing or clicking events took place before a scroll/zoom event and which took place after.

In terms of technical details, we looked at third-party libraries to capture touch paradata like pinch-zoom (for now we have settled on “hammer.js”), because detecting such activity requires handling of spatial and temporal tolerances for what constitutes such an event that are beyond what we currently want to invest in the effort. We discovered some technical limitations along the way, such as the fact that Firefox touch events are presently disabled by default in Windows (discovered because the script is being developed on a Surface Book). Because Windows-based touch devices have not been commonly used to complete surveys, we are not attempting to address this problem at the present time, but this points to the fact that unexpected platform and browser considerations can still cause unexpected results (and potentially hinder development).

It may also be worth noting that, while it might have proved easier to write this script with the aid of the JQuery library, we have run into a few problems that convinced us to stick with standard JavaScript. This is because Blaise 5 is currently using a somewhat older version of JQuery, and we have run into a few conflicts in other scripts when we do not match that version of JQuery exactly. We have, however, integrated our script into the Require.js framework that Blaise 5 currently uses, which is the reason for the define statement at the beginning of the script.

The description of the custom Data Entry API code (hereafter: router) implementation provided here is based on current (as of August 2016) Data Entry API architecture for ASP, but we know that the upcoming release will include a new architecture meant to supersede it. While the principle will remain the same, the details will no doubt differ.

The router implementation begins with attaching some code to events raised by the [IDataEntryControllerAsp](#) implementation. These are:

`ControlFactory.OnCreateDataFieldInputControl`, `ControlFactory.OnCreateCategoryInputControl`, `BeforeExecuteActions`. Other code needs to be added to provide a hidden field on the page in which to store the captured CSP, but that is not specific to Blaise 5.

An important part of the process is to update the captured CSP with identifiers that are meaningful for later analysis. The element ids connected with the client-side events that the JavaScript is capturing are no longer meaningful once the page changes (the next page may have identical ids referring to different question elements). Therefore, in the server-side code, which has access to the Blaise API, there must be code which translates the temporary identifiers used on the page into something like fieldnames. This can be done by cataloguing the fieldnames associated with the temporary identifiers while those html elements are being generated. The `ControlFactory.OnCreateCategoryInputControl` event provides the opportunity to do this for controls like radio buttons associated with the categories in codeframes. Here is a code snippet illustrating how we handle this:

Listing 1. `ControlFactory.OnCreateCategoryInputControl` Source Code

```
ICategory category = e.DataObject;
string fieldName = category.Field.Name;
if (category.Field.ValueType.DataType ==
StatNeth.Blaise.API.DataRecord.DataType.Set)
{
    fieldName += "-" + category.Code;
}
string clientID = definition.ID;
StoreClientIds(clientID, fieldName);
```

For `ControlFactory.OnCreateDataFieldInputControl` the code is slightly different to account for textbox naming conventions (note that the “_mask” naming is for custom controls developed by SRC, which shows how the system can be extended if needed).

Listing 2. `ControlFactory.OnCreateDataFieldInputControl` Source Code

```

string fieldName = field.Name;
string clientID = "";
string definitionID = definition.ID;
if (!string.IsNullOrEmpty(rawMask))
{
    clientID = definitionID + "_mask";
}
else
{
    clientID = definitionID + "_tb";
}
StoreClientIds(clientID, fieldName);

```

The `StoreClientIds` function called at the end of the above two snippets saves these pairs until the captured CSP is returned to be stored in the instrument database. We use the `BeforeExecuteActions` event to handle updating these ids and saving the CSP, like so:

Listing 3. `BeforeExecuteActions` Source Code

```

if (_hiddenField != null)
{
    IRouteItem paraDataField =
    _depcontroller.Page.RouteItems.GetItem("ClientSideParadata");
    if (paraDataField != null)
    {
        string csp = _hiddenField.Value;
        foreach (KeyValuePair<string, string> clientIdentifier in
        _clientIdentifiers)
        {
            csp = csp.Replace(":" + clientIdentifier.Key, ":" +
            clientIdentifier.Value);
        }
        paraDataField.Value.Assign(csp);
        _clientIdentifiers = new Dictionary<string, string>();
    }
}

```

Note that the field accessed using `Page.RouteItems.GetItem("ClientSideParadata")` is one that we set as a Field Reference in the Resource Database. The above function takes the captured CSP (accessible via the `_hiddenField` variable we add to the page) and replaces the temporary identifiers with fieldnames from the collection constructed as described above. This modified string is then assigned to the Field Reference. After it is assigned back, the Blaise 5 source code handles concatenating this CSP to that collected from previous pages, stored permanently in a field called `ClientSideParadataStore` as seen in this snippet of Blaise 5 source code we have placed near the beginning of our datamodel:

Listing 4. `ClientSideParadataStore` Source Code

```

ClientSideParadata.keep
ClientSideParadataStore.keep
IF ClientSideParadata <> EMPTY THEN
    ClientSideParadataStore := ClientSideParadataStore +
ClientSideParadata
    ClientSideParadata := EMPTY
ENDIF

```

To be clear, we add an Open field named ClientSideParadataStore to each of our instruments and this one field captures all the CSP for each case (which can be lengthy). It is extracted normally with the data like any other Open field we use. Here is an example of the data contained in this field after a few pages:

Listing 4. CSP Data

```
^t=5:onload-SecA.StartInterview.A006_=screen-1500x1000,client-1374x712,dt=2016-08-30T18:13:16.609Z
^t=2396:SecA.StartInterview.A006_=1[ENTR]
^t=0:onload-SecA.StartInterview.A007TRALive_A=screen-1500x1000,client-1374x712,dt=2016-08-30T18:13:20.363Z
^t=3293:PinchApart(Scale=3.7391,Duration=1003)
^t=73:Scroll(x=true,y=true)
^t=2934:SecA.StartInterview.A007TRALive_A=1
^t=1601:PinchTogether(Scale=0.2272,Duration=474)
^t=1238:Scroll(x=true,y=true)
^t=-124:SecA.StartInterview.A007TRALive_A=[ENTR]
^t=0:onload-SecA.StartInterview.A002_IwBegin=screen-1500x1000,client-1374x712,dt=2016-08-30T18:13:30.211Z
^t=2884:SecA.StartInterview.A002_IwBegin=1[ENTR]
^t=2:onload-SecA.StartInterview.A155_SelfPrxy=screen-1500x1000,client-1374x712,dt=2016-08-30T18:13:34.123Z
^t=6062:Help=[CLICK]
^t=2581:SecA.StartInterview.A155_SelfPrxy=1[ENTR]
```

Each line begins with the delimiter “^t=” and the number of milliseconds since the last event, or since the script loaded if the page just changed. Each page change is indicated by the presence of the text “onload-”. In this example, you can see the respondent typing numeric answers and pressing enter to advance the survey which has one question per page. There are also examples of “pinch zoom” with information on the scale and duration of the pinch event. Incidentally, pinch zoom is the only touch event that we capture which is fully working for us right now. Note that the pinch event by definition causes normal scroll activity to register in our script as well, and this appears in the CSP output even though the user would not think of the pinch as scrolling. We may remove this possibly redundant information later, because we generally try to avoid capturing things we don’t need in order to keep the size of the CSP string transmitted to the server as short as possible, not to mention limiting the size of the fully concatenated CSP string stored in the ClientSideParadataStore field. Finally, there is an example of a click on a help button near the end of the above CSP.

5. More to listen and more to capture

CSP has been providing important and interesting information about respondents’ behavior on the screen (like the events on the browser) as well as some environment information about the respondents’ machine (like screen dimension and browser type). Researchers are becoming increasingly interested in new areas, properties or events that provide information about the respondents’ behavior or their environment. Although we see an increasing number of respondents using mobile devices to take web surveys, we also find a higher percentage of break-offs with mobile platforms. (Couper and Peterson, 2016) Although we implemented efforts to make the display mobile friendly, like changing the font size, color and/or changing table questions to item-by-item questions vertically stacked, it seems that respondents still have higher break-off rates when using a mobile device. There is considerable literature addressing questionnaire design issues related to this point. To help understand the causes of this issue, the CSP captured on mobile platforms might help us learn more about respondents’ behavior and their challenges on a mobile browser, Thereby informing both questionnaire design and usability on mobile platform.

Besides logging events specific to mobile platforms, we also recognize that other attributes or behaviors might be requested because they are of interest to research communities, so we have been

continuously working on the enhancement of our CSP library. We also recognize that some information from CSP has the potential to improve our operational efficiency and efficacy as well (like in a CATI Web Mode).

- **GPS location:** Record navigator.geolocation.getCurrentPosition (which is only available with the html5 compliant browsers) to know the latitude and longitude of the device.
- For security reasons, when a web page tries to access location information on the user's machine, the user is notified and asked to grant permission. Each browser has its own policies and methods for requesting this permission. For example, some will work only if the device turns on the location service and explicitly allow for user GPS satellites. If the device does not turn on the location service or the user does not *agree* with the service, no warning will display and no data will be recorded.

For implementation purposes, the pop-up confirmation/agreement window to collect GPS information from the device has raised concern that this might discourage respondents from taking the web survey. On the other hand, if the web survey is collected by field interviewers as a data entry mechanism, this feature might be helpful for operational and QC purposes because it helps to identify the physical location where the survey is conducted.

- **Coordinates of the object that being clicked on or touched on:** event.clientX or event.touches.screenX. Record coordinates of the mouse pointer when the mouse button is clicked on an element or the coordinate of the touch point relative to the screen, not including any scroll offset.

Recording the coordinates of the objects might provide information about the exact display on the various platforms. Although we generally use responsive design to achieve optimal display on different browsers, there may be still differences among different mobile devices. Also, recording this might help us create a click-map and provide info about usability issues.

Also, the coordinates of the clicked spots unveil some interesting aspects of users' behavior. For a simple radio button question, users could be clicking on the radio button itself, or the users might be clicking *around* the control (like the label of a radio button).

There is a variety of coordinates that can be tracked with either JavaScript or JavaScript library like JQuery. For example, the coordinates of the monitor screen, the fully rendered content area in the browser, or the content area (the viewport) of the browser window. These attributes might become more complex and have different implications on mobile platforms. For example, when we zoom in on a mobile device, what do we wish to capture -- the position of the current browser screen, namely, the visual viewport (the part of the page that's currently shown on-screen) or the corresponding location coordinates relative to the original page layout, namely, the layout viewport (synonym for a full page rendered on a desktop browser)? All the different coordinates might have different *meanings* or might address different research purposes.

- **Window Focus or Blurred:** Add Event Listener to the event of "focus" and "blur" to track if respondents stay focused on the web survey pages.
- **Double-Tap:** On some mobile devices, double tapping triggers zooming effect, or hover-click event. We use the tapping events to capture this if the tapping occurs within 3 milliseconds.

CSP from our new testing SCP library is as below.

- **Example 1 from iPhone:**
 - dimensions:414,736;orientation:portrait; ^t:243629,coord:(42.27419841289512,-83.74521005579236);^t:277411,tap:(134,411);^t:277916,direction:together,distance:1;

^t:278765,tap:(546,860); ^t:279355,tap:(566,1537);^t:279926,tap:(513,2014);
^t:281373,tap:(299,1978);^t:281486,direction:apart,distance:-26;
^t:282104,tap:(267,2196);

- dimensions:414,736;orientation:portrait;^t:387666,coord:(-27.487006101743173,152.99031924516342); ^t:390988,orientation:landscape; ^t:392899,orientation:portrait; ^t:394585,orientation:landscape; ^t:395964,orientation:portrait; ^t:402344,orientation:landscape; ^t:404388,orientation:portrait; ^t:407339>window:blurred; ^t:414152>window:focused;
- *Data Notes: please see below explanations in Example 2 from Android*
- **Example 2 from Android:**
 - dimensions:432,768;orientation:portrait;^t:20871,coord:(42.2743974,-83.7451881); ^t:23494,orientation:portrait; ^t:25507,tap:(497,364);
 - dimensions:432,768;orientation:portrait;^t:263992,coord:(42.2743974,-83.7451881); ^t:274153,tap:(329,616);
 - dimensions:768,432;orientation:landscape; ^t:478316,coord:(42.2679279,83.7419255); ^t:482056,orientation:landscape;^t:488314,tap:(293,757);^t:488896,doubletap:(310,669); ^t:489925,tap:(148,654);
 - *Data Notes:*
 - *dimensions: x, y -> the screen size in pixels*
 - *orientation: landscape or portrait -> orientation and orientation change*
 - *coord: (x,y) -> the GPS location*
 - *^t: x -> action occurs at x milliseconds after page load*
 - *window: blurred -> window focus leaves the current page*
 - *window: focused -> window focus returns to the current page*
 - *tap: (x,y) -> touch the screen at the coordinates of (x,y)*
 - *direction: together, distance x -> pinch together by the distance of x*
 - *direction: apart, distance: y -> pinch apart by distance of y*
 - *doubletap: (x,y) -> tap on the coordinates of (x,y) twice within 3 milliseconds*
- **Items to work on**
 - **Swipe and Scroll:** we are currently working on capture the swipe-scroll events on mobile platform. We plan to use the touch event with the direction of movement for this.
 - **Zoom in or out after double tapping**

6. Implementation and Deployment

We have been continuously working to enhance our CSP JavaScript function library. However, there is still significant work and decisions to make regarding the specific items to include and record in a production environment.

We need perform load testing to make sure the **listening and recording** will not slow down the page or interfere with any other JavaScript function on the page. For example, in a few tests we conducted, we found the time to record GPS location seems higher on a mobile device than that on a regular PC. As we do this GPS recording as the 1st step in our JavaScript function, it is not clear if the delay is caused by the device, by the slow response time from the browser get the GPS location, or the

respondent's response time to the warning message with the GPS recording. These questions need further investigation before we can implement the functions in a production setting.

Technically, we are trying to expand our horizon and sharpen our *ears* to listen to more events that happen on the page, browser or the respondents' environment. These events will be expected to further inform research objectives and improve operational efficiency.

7. References

Ostergren, Jason, Liu, Youhong. 2010. BlaiseIS Paradata. *Presented at the 13th International Blaise Users Group Conference, Baltimore, USA*

Couper, Mick P., Peterson, Gregg. 2016. Why Do Web Surveys Take Longer on Smartphones? *Social Science Computer Review 2016*.

Transforming Survey Paradata

Lisa Wood, Andrew Piskorowski, and Jennie Williams, University of Michigan Survey Research Center, United States

1. Abstract

Paradata that are captured during the survey process are a valuable source of information in helping us understand and improve the data collection process.

One of the advantages of using Blaise for survey data collection is the rich capture of paradata that is native to the application. Paradata which are linked directly to the administration of a survey instrument are collected automatically through the Blaise software (i.e., audit trail). The ADT file from Blaise 4 has been very valuable in understanding interviewer behavior. With the advent of Blaise IS and Blaise 5, we now are able to increase the richness of our paradata collection for understanding respondent behavior on web-SAQ (self-administered questionnaires) and/or mixed mode projects (i.e., interviewer and web-SAQ combined).

The main focus of this paper will be to share a process to make these sources of Blaise 5 paradata more usable for analysis and reporting. The native Blaise 5 paradata, the user agent string, and the University of Michigan's client-side paradata (CSP) are unstructured data and very cumbersome to "untangle". To make these data more useful for analysis, various data management techniques are applied with the following goals.

- Parse this unstructured data (e.g., SAS, Python, SQL)
- Calculate new measures (e.g., time on/time between page, last question seen/answered)
- Aggregate data at various levels (e.g., page-level, session-level, respondent-level)

In addition, reporting tools such as OLAP cubes and SSRS (SQL Server Reporting Services) are used to distribute the data to various user groups (e.g., PIs, production managers, statisticians, etc.). The resulting output is also available in a SQL database and can be accessed using other reporting and analysis tools. The transformation techniques and standard paradata reports can be implemented by any user of Blaise 5 paradata to enhance the use of this data.

2. Introduction

Paradata that are captured during the survey process are a valuable source of information in helping us understand, monitor and improve the data collection process (Couper, 1998; Couper and Peterson, 2016; Cheung, Piskorowski, Wood and Peng, 2016). With the advent of BlaiseIS and Blaise 5, we now are able to increase the richness of our paradata collection for understanding respondent behavior on web-SAQ (self-administered questionnaires) and/or mixed mode projects (i.e., interviewer and web-SAQ combined).

There are many possible sources of paradata surrounding the survey data collection process. In this paper we focus on the two key types of survey paradata:

1. *Native Blaise 5 paradata (Server-side)*
 - a. This is the paradata which are linked directly to the administration of a survey instrument, and that are collected automatically through the Blaise software (i.e., audit trail).
 - b. One of the advantages of using Blaise for survey data collection is the rich capture of paradata that is native to the application.
 - c. We have seen that the ADT files from Blaise 4 have been very valuable in understanding interviewer behavior (Devonshire, Liu and Cheung, 2013).
2. *Supplemental Client-Side Paradata (CSP)*

- a. The team at the University of Michigan has implemented a process to capture paradata from the client-side using JavaScript (Peng and Ostergren, 2016).
- b. This client-side paradata (CSP) can capture movement within a page on the respondent's or interviewer's device, like scrolling and mouse-clicks.
- c. As the use of mobile devices has increased, the CSP can now capture events specific to the mobile device like tapping, orientation change, pinch in/out, etc.

The usefulness of the paradata is dependent on the ease with which it can be accessed, manipulated, and analyzed by end users. This stream of data can quickly grow exponentially, and in its raw data state, it is nearly impossible to use for analysis.

The main focus of this paper is to share how we are working to make these sources of Blaise 5 paradata more usable for analysis and reporting. These sources of survey paradata are unstructured data and very cumbersome to untangle. To make these data more useful for analysis, various data management techniques are applied with the following goals:

- Parse this unstructured data (e.g., SAS, Python, SQL)
- Calculate new attributes and measures (e.g., time on/time between page, last question seen)
- Aggregate data at various levels (e.g., page-level, session-level, respondent-level)

In addition, reporting tools such as OLAP (online analytical processing) cubes and SSRS (SQL Server Reporting Services) can be used to distribute the data to various stakeholders (e.g., Principal Investigators, production managers, statisticians, etc.). The resulting output is available in a SQL database and can be accessed using other reporting and analysis tools. The transformation techniques can be implemented by any user of Blaise 5 paradata to enhance the use of this data.

3. Where the Paradata Starts (Native Blaise 5 Audit)

From the Blaise 5 Help documentation, *“Audit trail or audit log is a chronological sequence of audit records, each of which contains evidence directly pertaining to and resulting from the execution of a Data Entry session. **Audit Trail is a feature of data entry which comes disabled by default in Blaise.** With this feature enabled you can show detailed information on visited pages, keys pressed, mouse clicks etc.”* (2015 Statistics Netherlands)

When the audit trail is enabled, there are various levels of information that can be captured (see the Blaise 5 help documentation for more details). The team at the University of Michigan uses the most detailed level of logging called “Keyboard”.

We begin by looking at the native Blaise 5 paradata in its initial (raw) form. This data lives in the AuditTrailData.db (note: this data may be stored in a SQLite or a SQL db, but the structure/contents are the same).

The Blaise 5 native paradata has one row for each “event” and the attributes include:

1. Keyvalue - the unique key for the specific case (e.g., SampleId, LoginId)
2. InstrumentId - this identifies the specific Blaise 5 instrument
3. SessionId - this identifies all events within a Blaise session
4. Timestamp - the datetime that the server records the event
5. Content - all the data about the event; the structure/contents of the paradata in this attribute depends on the type of event.

These are the types of events that we have uncovered (note: this list may not be exhaustive):

1. StartSession - this should be the first event at the start of a survey session; it includes information about device used to access the instrument including browser and the user agent string

2. Action - e.g., nextPage, PreviousPage
3. UpdatePage - capturing the movement between pages
4. EnterField - shows when the page is actually rendered. However, multiple enter fields events can happen on a page, and shows when a field is in focus (at beginning of a page load and navigation within a page)
5. Category and Keyboard - shows the response selection on radial buttons (Category) and the recording of keystrokes (Keyboard) on a page
6. LeaveField - capturing the movement from the page, and like the Enter field events, multiple leave field events can happen on a page
7. EndQuestionnaire - shows the completion of the Blaise instrument (note: there is no end session event if a survey is started and abandoned before completion)
8. InterruptSession - this is when the interviewer clicks the “quit” (interviewer administered)

Table 1. An Example of the Data in Its Raw Form

End of Session - CATI (Suspended)				
KeyValue	InstrumentId	SessionId	TimeStamp	Content
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:24:08	<LeaveFieldEvent FieldName="Section_a.Sec_S.S_SC_AcadRank_b" Value="5" AnswerStatus="Response" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:24:08	<ActionEvent Action="NextField()" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:24:08	<UpdatePageEvent LayoutSetName="Cati" PageIndex="109" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:24:08	<EnterFieldEvent FieldName="Section_a.Sec_S.S_SC_CertsDeg" AnswerStatus="Empty" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:27:08	<KeyboardEvent KeyStrokes="5" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:27:08	<LeaveFieldEvent FieldName="Section_a.Sec_S.S_SC_CertsDeg" Value="5" AnswerStatus="Response" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:27:08	<ActionEvent Action="NextField()" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:28:08	<UpdatePageEvent LayoutSetName="Cati" PageIndex="149" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:28:08	<EnterFieldEvent FieldName="Section_b.HE_GenHlth" AnswerStatus="Empty" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:31:08	<LeaveFieldEvent FieldName="Section_b.HE_GenHlth" AnswerStatus="Empty" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:31:08	<ActionEvent Action="Save()" />
L19178	{9b255695-0a9c-40}	{2d6e47c6-d489}	8/29/2016 11:00:31:08	<InterruptSessionEvent />
Start of Session - Mobile				
KeyValue	InstrumentId	SessionId	TimeStamp	Content
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:53:37:08	<UpdatePageEvent LayoutSetName="MobM" PageIndex="2" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:53:37:08	<EnterFieldEvent FieldName="Welcome" AnswerStatus="Empty" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:53:37:08	<StartSessionEvent Width="980" Height="1461" Device="Browser" Browser="Safari" Language="Mobile" KeyValue="L19330" Platform="HTML" OS="Mozilla/5.0 (iPhone; CPU iPhone OS 9_3_4 like Mac OS X) AppleWebKit/601.1.46 (KHTML, like Gecko) Version/9.0 Mobile/13G35 Safari/601.1" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:53:46:08	<ActionEvent Action="NextPage()" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:53:46:08	<UpdatePageEvent LayoutSetName="MobM" PageIndex="5" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:53:46:08	<EnterFieldEvent FieldName="INTRO_ConsentScreen_01" AnswerStatus="Empty" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:54:56:08	<ActionEvent Action="NextPage()" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:54:56:08	<UpdatePageEvent LayoutSetName="MobM" PageIndex="9" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:54:56:08	<EnterFieldEvent FieldName="xxxxdummy4" AnswerStatus="Empty" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:55:06:08	<LeaveFieldEvent FieldName="xxxxdummy4" AnswerStatus="Empty" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:55:07:08	<ActionEvent Action="NextPage()" />
L19330	{9b255695-0a9c-40}	{bba30c98-470f}	8/24/2016 13:55:07:08	<UpdatePageEvent LayoutSetName="MobM" PageIndex="13" />

4. Processing the Native Blaise 5 Paradata

The data management team at the University of Michigan has developed SAS routines to parse the native Blaise 5 audit data. These routines interrogate the attribute called “Content”. They have been developed to be transferable across any Blaise 5 project with just the definition of some key parameters. A sample of this code is provided in Appendix A.

When the native Blaise 5 paradata is processed for analysis, the resulting output is a data set that has the same number of records as the input file. However, new attributes or measures have been added. A data dictionary for the output data set is provided in Appendix B.

Here is an overview of the types of attributes/measures that are added to the Blaise 5 native paradata.

- **Event Type Flags:** Flags were added that indicate the event type (0/1 values). These flags are called SessionStart, SessionEnd, IWComplete, EnterField, PageUpdate, LeaveField.
- **Sequence/Counters:** The order in which events occur are very important. There are sequence or counters to indicate (a) the number of unique sessions for the case (SessionNum), (b) the sequence of events within the session (SessionSeq), (c) the sequence of events for a case across all sessions (EventSeq), and (d) the sequence of fields for a case (FieldSeq)
- **User Agent String:** Various elements from the user agent string (found in the StartSession event) were parsed out. These include Width, Height, Browser, and Platform. Additional fields can be parsed or inferred (e.g., type and name of device – e.g., desktop, tablet, phone).
- **Field Information:** Each row or event in the audit data can be associated with a field. This is provided in the new attributes called FieldName and Block. As the associated field/block are not always provided in the Content (native Blaise 5) for all events, some decisions are made on how to attribute an event/row to a Blaise field. In addition, if the response to a Blaise field is included in the Content, it is stored in the new attribute called Answer.
- **Timings:** Now we can start to determine timing between each event (EventTimeSec), time in each field (FieldTimeSec) and the time it took for a page to update (PageLoadSec).

Table 2a: An Example of the Processed Data – Start Sessions

KeyValue	Content	Width	Height	Browser	Platform	AgentString	Language
L19178	<StartSessionEvent Width="1936" Height="1056" Device="WindowsDesktop" Language="_CATI" KeyValue="L19178" Platform="Windows" />	1936	1056		Windows		_CATI
L23925	<StartSessionEvent Width="1920" Height="997" Device="Browser" Browser="Chrome" Language="WEB" KeyValue="L23925" Platform="HTML" OS="Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36" />	1920	997	Chrome	HTML	Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36	WEB
L23925	<StartSessionEvent Width="980" Height="1546" Device="Browser" Browser="Chrome" Language="Mobile" KeyValue="L23925" Platform="HTML" OS="Mozilla/5.0 (Linux; Android 6.0.1; SAMSUNG-SM-G935A Build/MMB29M; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/52.0.2743.98 Mobile Safari/537.36" />	980	1546	Chrome	HTML	Mozilla/5.0 (Linux; Android 6.0.1; SAMSUNG-SM- G935A Build/MMB29M; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/52.0.2743.98 Mobile Safari/537.36	Mobile

Table 2b: An Example of the Processed Data

End of Session - CATI (Suspended)										
KeyValue	TimeStamp	Session Start	Session End	SessionType	FieldName	SessionSeq	FieldSeq	FieldTime Sec	EventTime Sec	PageLoad Sec
L19178	8/29/2016 11:00:24:08	0	0	<LeaveFieldEvent	Section_a.Sec_S.S_Si	235			0.00	
L19178	8/29/2016 11:00:24:08	0	0	<ActionEvent	Section_a.Sec_S.S_Si	236			0.28	
L19178	8/29/2016 11:00:24:08	0	0	<UpdatePageEvent	Section_a.Sec_S.S_Si	237			0.00	0.28
L19178	8/29/2016 11:00:24:08	0	0	<EnterFieldEvent	Section_a.Sec_S.S_Si	238	62	3.51	2.99	
L19178	8/29/2016 11:00:27:08	0	0	<KeyboardEvent	Section_a.Sec_S.S_Si	239			0.20	
L19178	8/29/2016 11:00:27:08	0	0	<LeaveFieldEvent	Section_a.Sec_S.S_Si	240			0.00	
L19178	8/29/2016 11:00:27:08	0	0	<ActionEvent	Section_a.Sec_S.S_Si	241			0.32	
L19178	8/29/2016 11:00:28:08	0	0	<UpdatePageEvent	Section_b.HE_GenHit	242			0.00	0.32
L19178	8/29/2016 11:00:28:08	0	0	<EnterFieldEvent	Section_b.HE_GenHit	243	63	3.65	3.57	
L19178	8/29/2016 11:00:31:08	0	0	<LeaveFieldEvent	Section_b.HE_GenHit	244			0.00	
L19178	8/29/2016 11:00:31:08	0	0	<ActionEvent	Section_b.HE_GenHit	245			0.08	
L19178	8/29/2016 11:00:31:08	0	1	<InterruptSessionEvent	InterruptSession	246	64			
Start of Session - Mobile										
KeyValue	TimeStamp	Session Start	Session End	SessionType	FieldName	SessionSeq	FieldSeq	FieldTime Sec	EventTime Sec	PageLoad Sec
L19330	8/24/2016 13:53:37:08	1	0	<StartSessionEvent	StartSession	1	1	8.90	8.84	
L19330	8/24/2016 13:53:46:08	0	0	<ActionEvent	Welcome	2			0.06	
L19330	8/24/2016 13:53:46:08	0	0	<UpdatePageEvent	INTRO_ConsentScree	3			0.00	0.06
L19330	8/24/2016 13:53:46:08	0	0	<EnterFieldEvent	INTRO_ConsentScree	4	2	70.07	70.01	
L19330	8/24/2016 13:54:56:08	0	0	<ActionEvent	INTRO_ConsentScree	5			0.06	
L19330	8/24/2016 13:54:56:08	0	0	<UpdatePageEvent	xxxxdummy4	6			0.00	0.06
L19330	8/24/2016 13:54:56:08	0	0	<EnterFieldEvent	xxxxdummy4	7	3	10.79	10.27	
L19330	8/24/2016 13:55:06:08	0	0	<LeaveFieldEvent	xxxxdummy4	8			0.43	
L19330	8/24/2016 13:55:07:08	0	0	<ActionEvent	xxxxdummy4	9			0.08	
L19330	8/24/2016 13:55:07:08	0	0	<UpdatePageEvent	Section_a.AA_MilStatu	10			0.00	0.08

5. Client Side Paradata Processing

In addition to the events captured from the server side, the actions/events that occur on the client are also very useful paradata for understanding the user behavior as they interact with the Blaise 5 instrument. In particular, the analysis of the behaviors on a mobile device can provide insight into questionnaire design and user experience (Cheung et. al, 2016). This is important to understand as we move from interviewer-administered, to self-administered mode of data collection (Couper and Peterson, 2016).

Of course, the Blaise 5 native paradata captures some client-side paradata. The University of Michigan team is working on routines to supplement, capture and parse this complex data related to actions on the client-side (Peng and Ostergren, 2016). The data transformation processes for enhanced client-side paradata continue to be developed and are beyond the scope of this discussion. However, as development continues, it is our wish that the client-side paradata capture is integrated into a native solution within the Blaise software (e.g., providing a paradata API for us to write custom code against, or capturing the supplemental CSP data). The value of both the server and client-side paradata will be enhanced if these data can be linked, and analysis can more easily integrate both of these rich sources of survey paradata.

6. Aggregation and Reporting

Many different types of aggregation and reports can now be generated from this processed Blaise 5 audit data and client-side paradata (Piskorowski, 2016). To facilitate the access to this processed paradata, we have developed routines to push the data into SQL database(s). By increasing access to the paradata for analysis, there is almost no limit to the questions that we answer. Some examples include:

- Timings: Total Interview Length, Interview Length by Respondent, by Mode, by Field, by Block, by Interviewer
- Surveys Abandoned: Average Number of Sessions, Session Length, Number of Sessions (by Mode) for each Respondent, Last Question Seen (Answered), Last Mode Accessed.

More details and examples can be found in Piskorowski, 2016.

7. Some Challenges, Lessons Learned and Recommendations

We have been working with the native Blaise 5 audit data for over one year now and for many beta releases. Some of the kinks are being worked out (with dedicated response and collaboration from the Statistics Netherlands Blaise 5 team at CBS). We want to share some of the challenges and lessons learned, and highlight a few of the more important enhancements that we would like to recommend.

1. The consistent sequence of events in the audit data is important!
 - When milliseconds were added to the date/time stamps, we got very valuable information for sequencing events.
 - **Challenge:** Although not occurring very often, we are still finding that some events are not sequenced properly. For example, the StartSession event is sometimes not always the first event in a new session.
2. Some other data messiness has been detected and audit data processors should be on the lookout.
 - We continue to see multiple rows with the same enter field event (with the same timestamp). Our processes have been modified to ignore these duplicate events.
 - We have also seen events without the date/timestamp information.
3. Are we getting leave fields? We really need them!
 - Although more stable now, we have found that the evolving versions of the Blaise 5 software produce different results in terms of the inclusion of leave field events when expected.
 - In order to capture leave fields, the Blaise programmer needs to set the audit trail level of capture to “Field” (or more extensive level of logging).
 - It’s tricky to figure which fields should have a leave field event. Some fields that involve display of information (or click yes to continue) do not have leave field events by design.
 - At the University of Michigan, the data manager reviews the audit data with the Blaise programmer to determine which fields should not have a leave field event (**Hint:** generally the “should not have” is a much shorter list than the “should have” list).
 - **Lesson Learned:** We have established quality checks to make sure expected fields have a leave field event and to make sure that leave field events are sequenced correctly.
4. What about page-level analysis?
 - **Challenge:** The presentation of multiple fields on one page can definitely be a challenge when trying to analyze the audit data at a page-level. Mostly this presents us with an interpretation challenge (i.e., how do we identify the visit to the same page across respondents - or even for multiple visits to the same page by the same respondent?!?!). Of course, page-level analysis is much easier when there is only one field on each page.
 - Our approach has been to identify a page using the first field presented, but this does not always work.
 - The Blaise programmer controls where the cursor goes on the initial page load. So the programmer is very powerful - and should be thoughtful when setting up!
 - However, grids are one object on a page, and the question order within the grid may be randomized (so now the “first field” solution won’t work).
 - The Blaise 5 audit data captures a PageIndex. However, we are still exploring how best to use. Of course, the same page number could be displayed differently to different Respondents (based on logic and/or randomization of questions). Pages may be very different across modes too!

- **Lesson learned:** We try to get everything down to the field level. Recommendations or collaborations for other approaches to page-level analyses are welcomed!
- 5. Combining the server-side (native Blaise 5) and the supplemental client-side paradata
 - **Challenge:** We are still working on processes for combining these two sources of paradata for analysis
 - **Recommendation:** As indicated in section 4, it is our wish that the supplemental client-side paradata capture is integrated into a native solution within the Blaise software!
- 6. Determining the end of a Blaise session using the paradata
 - There are various ways that a Blaise session might end including completed, aborted, expired or interrupted (and removed?)
 - In the audit data, we do get EndQuestionnaire (completed) and InterruptSession events, but we can only make inferences about expired sessions.
 - The StatNeth.Blaise.API.ServerEvents Namespace provides components for listening to server-side events (Statistics Netherlands, 2015 Help documentation), and includes expired session events.
 - **Recommendation:** It is our wish to also have other session-level events (like Expired) included in the Blaise native audit data.

8. References

- Cheung, Gina, Piskorowski, Andrew, Wood, Lisa and Peng, Hueichun. (2016). ‘Paradata Uses’. Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam
- Couper, Mick P. (1998). Measuring survey quality in a CASIC environment. *In Proceedings of the Section on Survey Research Methods of the American Statistical Association.*
- Couper, Mick P. and Peterson, Gregg J. (2016). ‘Why Do Web Surveys Take Longer on Smartphones?’ *Social Science Computer Review*, ssc.sagepub.com, p 1-21.
- Peng, Hueichun and Ostergren, Jason. (2016). ‘Paradata Capture’. Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam
- Piskorowski, A (2016). ‘Data In and Data Out’. Presented at the 17th International Blaise Users Group Conference, The Hague, Amsterdam
- Statistics Netherlands. 2015. *Blaise 5 Help Documentation. Paradata - Audit Trail.*
<http://help.blaise.com/>

9. Appendix A – SAS Processing Code

Listing 1. SAS Processing Source Code

```

/*****Developed by the Survey Research Center at the*****/
/***** University of Michigan for use with SAS v9.4*****/
/*****Last updated: September 2, 2016 *****/
/*****Questions contact tsdataops@umich.edu *****/

/* SAS options to set - replace [] where indicated */
options symbolgen;
options noxwait; /*External command prompts will auto-close*/
options xsync; /*SAS will wait for applications to finish before resuming
processing*/
options nodate nonumber compress=yes;

/* Date parameters */
proc format;
  picture Processdt other='%0d%0b%0y' (datatype=date);

```

```

run;

%let datetimenow=%sysfunc(date(), Processdt);
%put &datetimenow;

/*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
/*!!!!!!!!!!!! Project-specific parameters to set !!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
%let servertimeout=600; /*This represents the Blaise 5 server timeout value
(in seconds - e.g., 10 minutes=600);
This value is used as a cutoff for session breakoff timings calculations */

%let Instrumentid={[INSTRUMENTID]}; /* This is for Audit data - replace
[INSTRUMENTID] and make sure to keep the {} */
%let InstrumentName=[INSTRUMENTID]; /* This is for Session data - replace
[INSTRUMENTID], but no {} needed */

%let ServerPath=[SERVERPATH] /* This is the location of the native Blaise5
Audit data - replace [SERVERPATH] */
%let YourPath=[YOURPATH] /* This is the location of the output - replace
[YOURPATH] */

%let keep=[YOURCRITERIA1]; /* Used for filtering out records - used for
Blaise audit data - replace [YOURCRITERIA1]*/
%let keep3=[YOURCRITERIA2]; /* Used for filtering out records - used for
Blaise session data - replace [YOURCRITERIA1] */
/* NOTE the keep criteria may be different based on the unique identifier -
e.g., audit uses SampleId and session uses PrimaryKeyValue */

/*!!!!!!!!!!!! End setting project-specific parameters !!!!!!!!!!!!!!!!!!!!!!!*/
/*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/

/* USE IF YOU ARE STORING AUDIT/SESSION data in a SQLite db*/
/* Make a copy of the audit and session data (SQLite) before doing
anything else */
%sysExec copy "&ServerPath.\AuditTrailData.db"
"&YourPath.\AuditTrailData_&datetimenow..db" ;
%sysExec copy "&ServerPath.\RuntimeSessionData2.db"
"&YourPath.\RuntimeSessionData2_&datetimenow..db" ;

/*****
* ACCESS SQLITE DATABASE FROM SAS
* REQUIREMENT: HAVE SQLITE ODBC DRIVER INSTALLED
*****/

/* Connection to audit db (B5 native), Session DB (B5 Native) - ODBC
connections to the SQLite DBs need to be set-up on the user's computer */
libname Audit odbc complete="dsn=SQLite3 Datasource;
Driver=SQLITE3 ODBC Driver;

Database=&YourPath.\AuditTrailData.db";

libname session odbc complete="dsn=SQLite3 Datasource;
Driver=SQLITE3 ODBC Driver;

Database=&YourPath.\RuntimeSessionData2.db";

/* USE IF YOU ARE STORING AUDIT/SESSION data in a SQL db */
/*Blaise Audit Data*/
%let db=[YOURDATABASE]; /* !! ENTER DATABASE HERE */

LIBNAME b5AUDIT OLEDB

```

```

INIT_STRING = "PROVIDER=SQLNCLI11.1;
INTEGRATED SECURITY=SSPI;
PERSIST SECURITY INFO=TRUE;
INITIAL CATALOG=&db;
DATA SOURCE=&ServerPath;"
SCHEMA=&schema4 PROMPT=NO
PRESERVE_TAB_NAMES=YES PRESERVE_COL_NAMES=YES AUTOCOMMIT=NO DBMAX_TEXT =
32767;

/*!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!*/
/*   NOW LET'S START THE PROCESSING !!!!                                   */
/* STEP 1: First step for RAW ADT is to merge Session and Event data      */
/* Let's also rename KeyValue as SampleId */
proc sql;
create table RawADT1 as
    select a.KeyValue as SampleId,
           b.instrumentId,
           b.SessionID,
           b.TimeStamp,
           b.Content
    from audit.AuditSessionData a join audit.EventData b
    on a.SessionID = b.SessionID and a.InstrumentID = b.InstrumentID
       where b.instrumentid = "&instrumentID"
    order by a.KeyValue,b.timestamp,b.sessionid;
quit;

/* Use if you want to filter out any cases */
data RawADT1a;
    set RawADT1a;
    if &keep.;
run;

/*Get list of only Start Session events*/
data RawADT2;
    set RawADT1;
    where Content like ('%StartSessionEvent%');
run;

/*Get list of EnterField events*/
data RawADT3;
    set RawADT1a;
    where Content like ('%EnterFieldEvent%');
run;

/* STEP2: Let's start with some parsing
NOTE: THIS STEP NEEDS SOME UPDATES FOR ADDITION OF CATI */
data work.RawADT4 ;
    set RawADT1a;
    Seq = _N_ ; /* Create a "overall" sequence for this Instrument(s)
data - as found in the DB */
/*Data is already sorted by SampleId then timestamp - see STEP 1*/
/*This is crucial to the paradata tables - not adding explicit sort in the
assumption */

/* Flag the Session Start/End Events and parse out session level data */
    format SessionStart SessionEnd IWComplete 8.;
    format EventType $50.;
    format Width Height $8.;
    format Browser Language LoginId Platform $20.;
    format AgentString $200.;
    SessionStart = index(content, "<StartSessionEvent");

```

```

/* Flag the Start Session Event */
    SessionEnd = index(content, "<InterruptSessionEvent");

/* Flag the Interrupt Session Event */
    IWComplete = index(content, "<EndQuestionnaire");
/* Flag the End Questionnaire Event*/
    EventType = strip(scan(content,1,' '));
    if IWComplete ne 0 then SessionEnd=1;
/* Also flag the end of the Q as the session end */
    if SessionStart ne 0 then Width = strip(scan(content,2, ''));
/* Width is the first double quote value in start session */
    if SessionStart ne 0 then Height = strip(scan(content,4, ''));
/* Height is the second double quote value in start session*/
    if SessionStart ne 0 then Browser = strip(scan(content,8, ''));
/* Browser is the fourth double quote value in start session
(ignore third, which says device, but it isn't) */
    if SessionStart ne 0 then Language = strip(scan(content,10, ''));
/* Language is the fifth double quote value in start session */
    if SessionStart ne 0 then LoginId = strip(scan(content,12, ''));
/* LoginId is the sixth double quote value in start session*/
    if SessionStart ne 0 then Platform = strip(scan(content,14, ''));
/* Platform is the seventh double quote value in start session*/
    if SessionStart ne 0 then AgentString = strip(scan(content,16, ''));
/* AgentString is the eighth double quote value in start session*/

/* Flag the Enter Field Events and parse out field level data - note:
Start/End session and IwComplete are considered field events */
    format EnterField PageUpdate LeaveField 8.;
    format FieldName Block $100.;
    format AnswerStatus $10.;
    format Answer $100.;

    /* Add more flags for each type of event */
    EnterField = index(Content, "<EnterField" );
    PageUpdate = index(Content, "<UpdatePageEvent" );
    LeaveField = index(Content, "<LeaveFieldEvent" );
    PageAction = index(Content, "<ActionEvent" );

if SessionStart ne 0 or IWComplete ne 0 or SessionEnd ne 0 or LoginPassed
ne 0 then EnterField=1; /* Also flag these as Enter Field events */

    /* Now let's get the field name */
    if EnterField ne 0 then FieldName = strip(scan(content,2, ''));
    if LeaveField ne 0 then FieldName = strip(scan(content,2, ''));
    if SessionStart ne 0 then FieldName = "StartSession";
    if IWComplete ne 0 then FieldName = "EndQuestionnaire";
    if IWComplete =0 and SessionEnd ne 0 then FieldName =
InterruptSession";
    *if PageUpdate ne 0 then FieldName="PageUpdate";
    Block = strip(scan(FieldName,1, '.'));

    /* Let's capture if response was given at leave field */
    if LeaveField ne 0 then AnswerStatus = strip(scan(content,4, ''));
    if LeaveField ne 0 and AnswerStatus ="Response" then
Answer=strip(scan(content,6, ''));
    WHERE timestamp ^= .; /* This removes any records without the
timestamp*/
run;

```

```

/* This is to remove the "extra" enter field events - i.e., one that is
right after another */
/* If it is a EnterField event and the row preceding is also enter field
event, then keep the 1st occurrence of the Enter */
/* We have to index for EnterField again because the code above also
assigns some events (e.g., StartSession) as EnterField */
/* Sort to correct for the fact that StartSession events don't always go to
the top of each session because of duplicate timestamps, missing
milliseconds*/
proc sort data=RawADT4;
    by SampleId TimeStamp descending SessionStart ;
run;

/* Now taking care of some duplicate events */
data rawADT4a;
    set rawADT4;
    EnterOnly = index(Content,"<EnterField" );
    EnterDupe=0;
    if EnterOnly=1 and lag(EnterOnly)=1 and lag(seq)=Seq-1 then EnterDupe=1;
run;

data RawADT4_noEnterDup (drop=EnterDupe EnterOnly);
    set rawADT4a;
    if EnterDupe=0;
run;

/* Data with and without Category events */
data RawADT4_noCategoryEvent;
    set RawADT4_noEnterDup;
    if EventType ne '<CategoryEvent';
run;

data RawADT4_CategoryEvent;
    set RawADT4_noEnterDup;
    if EventType='<CategoryEvent';
run;

/* Now we are populating FieldName and Block for all records - use the row
above if missing */
data RawADT4b;
    set RawADT4_noCategoryEvent;
    retain _FieldName;
    if not missing(FieldName) then _FieldName=FieldName;
    else FieldName=_FieldName;
    drop _FieldName;
    Block = strip(scan(FieldName,1, '.'));
run;

proc freq data=rawADT4b;
    table fieldname /out = adt_fieldnames noprint;
run;

/*Here, we are sorting descending, because the previous code assigns update
page(first row of page) to last page, so Im now doing it in reverse
to assign the update page to the row after - a reverse lag of sorts -
Piskorowski*/
proc sort data= rawADT4b;
    by sampleid descending Seq ;
run;

data rawadt4c;

```

```

set rawadt4b;
by sampleid;
FieldN = lag1(FieldName);
if index(content, 'UpdatePageEvent') > 0 then FieldName = FieldN;
Block = strip(scan(FieldName,1, '.'));
drop FieldN;
*SID = lag(sampleid);
*seqn = lag(Seq);
*end;
run;

/*Sort it back to normal*/
proc sort data=rawadt4c;
    by SampleID Seq;
run;

/*Now let's add Event/Session Sequence and Session Counters */
data RawADT5;
    set RawADT4c;
    by SampleId;
        EventSeq+1;
        If First.SampleId = 1 then do;
            SessionNum = 1;
            EventSeq = 1;
            End;
        If First.SampleId ne 1 and SessionStart=1 then SessionNum+1;
        if SessionNum < 10 then
NewCount=Compress(SampleId||"00"||SessionNum); /* We are adding these
leading zeros so that Rs with 10+ 100+ sessions are ordered correctly*/
        if 10 <= SessionNum < 100 then
NewCount=Compress(SampleId||"0"||SessionNum);
        if SessionNum >= 100 then
NewCount=Compress(SampleId||SessionNum);
run;

/* REVIEW WITH AP/JENNIE this code that adds mode and AttemptNumber */
data RawADT5b (drop=NewCount);
    set RawADT5;
    by NewCount;
        If First.NewCount = 1 then do;
            SessionSeq = 1;
            End;
        If First.NewCount ne 1 then SessionSeq+1;
        if last.NewCount then SessionEnd=1; /* We also need to flag the
SessionEnd - i.e., last Record for the SID/Session is the SessionEnd */
        if last.NewCount then SuspendField=FieldName;
        AttemptNumber = strip(SampleId) || '_' || put(SessionNum,z2.);
        /*EDIT these per Survey - Programmers use different laynames for various
versions

/*****
Format Mode $10.;
if index(content, 'LayoutSetName="Interviewing1"') > 0 then Mode = 'Web';
if index(content, 'LayoutSetName="Web"') > 0 then Mode = 'Web';
if index(content, 'LayoutSetName="Cati"') > 0 then Mode = 'CATI';
if FieldName = 'StartSession' then Mode = 'Web';
run;

/*Getting a better mode variable here, putting all rows as Login, CATI or
Web*/

```

```

data RawADT5c;
  set RawADT5b;
  retain _Mode;
  if not missing(Mode) then _Mode=Mode;
  else Mode=_Mode;
  drop _Mode;
  /*EDIT these per Survey - Programmers use different laynames for various
versions

/*****
  if index(content, 'UpdatePageEvent') > 0 then Page = strip(scan(content,
4, ''));
  if index(content, 'StartSession') > 0 then Page = 'StartSession';
  if index(content, 'SwitchSession') > 0 then Page = 'PassedintoSurvey';

  retain _Page;
  if not missing(Page) then _Page=Page;
  else Page=_Page;
  drop _Page;

  run;
proc sql;
create table pagefields as
select distinct page, fieldname from
rawadt5c
where page is not null; quit;

/*Here we're going to use the events 'UpdatePageEvent' Action Event
NextPage and Previous page for the timings on a page.
We'll also identify each page using the page index. We'll link this later
to a crosswalk*/
data rawadt6a;
  set RawADT5c;
  if index(content, 'UpdatePageEvent') > 0 or index(content, 'ActionEvent') >
0 then output rawadt6a;
run;

/* Now we are going to separate the EnterField Events from all other events
so that we can add Field Counters and Time between fields */
data RawADT6a RawADT6b ;
  set RawADT5c;
  if EnterField=1 then output RawADT6a;
  else output RawADT6b;
run;

proc sort data= RawADT6a;
  by SampleId SessionNum descending SessionStart;
run;

data RawADT6a2;
  set RawADT6a;
  By SampleId SessionNum;
  FieldSeq+1;
  If First.SampleId = 1 then do;
    FieldSeq = 1;
  End;
run;

proc sort data=RawADT6a2;
  by SampleId descending FieldSeq;

```

```

run;

data RawADT6a3 ;
  set RawADT6a2;
  format FieldTimeSec 20.4;
  FieldTimeSec = Lag(TimeStamp) - TimeStamp;
  if FieldTimeSec > &servertimeout then FieldTimeSec=0;
  if SessionEnd=1 then FieldTimeSec=.;
run;

proc sort data=RawADT6a3;
  by SampleId FieldSeq;
run;

/* Combine together again and sort so that we can add duration calculation
(EventTime) */
data RawADT6;
  set RawADT6a3 RawADT6b;
run;

proc sort data=RawADT6;
  by SampleId descending EventSeq;
run;

data RawADT7;
  set RawADT6;
  format EventTimeSec 20.4 ;
  EventTimeSec = Lag(TimeStamp) - TimeStamp;
  if EventTimeSec > &servertimeout then EventTimeSec=0;
  if SessionEnd=1 then EventTimeSec=.;
run;

data RawADT7b;
  set RawADT7;
  PreviousField = lag(FieldName);
run;

proc sort data=RawADT7b;
  by SampleId EventSeq;
run;

data RawADT_final;
  set RawADT7b;
  format PageLoadSec 20.4 ;
  PageLoadSec = Lag(EventTimeSec) ;
  if PageUpdate=0 then PageLoadSec=.;
run;

```

10. Appendix B: Data Dictionary of Processed Data

Figure 1. Data Dictionary of Processed Data

FieldName	Type	Length	VarNum	Field Description	Source	Notes
SampleId	VarChar	255	1	KeyValue	Blaise Native	Rename of KeyValue
InstrumentId	VarChar	40	2	InstrumentId	Blaise Native	
SessionId	VarChar	40	3	SessionId	Blaise Native	Each SampleId has unique SessionId, but all sessions for a SampleId use the same SessionId
TimeStamp	Numeric	8	4	TimeStamp	Blaise Native	We are getting all the milliseconds with more recent build!
Content	VarChar	1024	5	Content	Blaise Native	
Seq	Numeric	8	7	Overall Sequence	Computed	Basically the same as the record # for the data set (to preserve the sequence as written by the Blaise native)
SessionStart	Numeric	8	8	Flag to show that this record indicates the START of a session	Computed	0/1
SessionEnd	Numeric	8	9	Flag to show that this record indicates the END of a session	Computed	0/1
IWComplete	Numeric	8	10	Flag to show that this record indicates the IW was Completed	Computed	0/1
EventType	VarChar	50	11	Text describing the type of Event	Computed	Rename of SessionType (in SRC original code)
Width	VarChar	8	12	Width (at Session Start=1)	Computed	This information is populated only on Session Start events
Height	VarChar	8	13	Height (at Session Start=1)	Computed	This information is populated only on Session Start events
Browser	VarChar	20	14	Browser (at Session Start=1)	Computed	This information is populated only on Session Start events
LoginId	VarChar	20	15	LoginId (at Session Start=1)	Computed	This information is populated only on Session Start events
Platform	VarChar	20	16	Platform (at Session Start=1)	Computed	This information is populated only on Session Start events
AgentString	VarChar	120	17	AgentString (at Session Start=1)	Computed	This information is populated only on Session Start events
Language	VarChar	20	18	Language (at Session Start=1)	Computed	Looks like a proxy for mode
EnterField	Numeric	8	19	Flag to show that this record is considered an EnterField event	Computed	0/1 (note: sometimes we get 2 enter field events in a row)
PageUpdate	Numeric	8	20	Flag to show that this record is considered an PageUpdate event	Computed	0/1 (note: the FieldName on this row indicates the field that is being exited)
LeaveField	Numeric	8	21	Flag to show that this record is considered an LeaveField event	Computed	0/1 (note: we don't always get leave field for each Q, but we can start to examine)
FieldName	VarChar	100	22	Field Name - populated for all records	Computed	If this is not provided in Contents, populated using the FieldName from the Row above
Block	VarChar	100	23	Block	Computed	Based off of Field Name
AnswerStatus	VarChar	10	24	Answer Status (if LeaveField=1)	Computed	Values of Empty/Response
Answer	VarChar	100	25	Answer (if LeaveField=1 and AnswerStatus=Response)	Computed	
PageAction	Numeric	8	26	Flag to show that this record is considered an Action event	Computed	0/1
EventSeq	Numeric	8	27	Counter of events for the SampleId (across sessions)	Computed	
SessionNum	Numeric	8	28	Number of Sessions for the SampleId	Computed	
SessionSeq	Numeric	8	29	Counter of events within a Session for each SampleId	Computed	
SuspendField	VarChar	100	30	Last Question accessed in that Session (if EndSession=1)	Computed	
Mode	VarChar	10	31	Using "LayoutSetName" to determine mode	Computed	Most useful if a survey is using multiple modes. Some custom SAS editing may be required.
Page	VarChar	20	32	PageIndex (as provided by Blaise native)	Computed	
FieldSeq	Numeric	8	33	Counter of Fields accessed for the SampleId (across sessions)	Computed	
FieldTimeSec	Numeric	8	34	Total Time from Field to Field (i.e., between each "EnterField")	Computed	
EventTimeSec	Numeric	8	35	Time between each event	Computed	If there is a session break-off, there is no EventTimeSec for the last event in the session
PreviousField	VarChar	100	36	Simply a "LAG" of the field - which returns the field from the row prior	Computed	See SAS LAG function for more details
PageLoadSec	Numeric	8	37	Time took for page to update to the next page	Computed	FieldName indicated is the last field before the Page Update (i.e., the field being EXITED not ENTERED)

Using Audit Trail data to move from a Black Box to a Transparent Data Collection Process

Jacqueline Hunt, Central Statistics Office, Ireland

1. Abstract

CSO is currently transforming the management and administration of all its household surveys. Part of this transformation is the introduction of CATI for the LFS waves 2-5. The telephone interviewing is being outsourced and all CATI interviews are being conducted by a third party service provider. The contract negotiated by the office for the service is based on the duration of calls. Early testing with the call centre revealed using timestamps embedded in the questionnaire as a method of generating interview durations did not provide the accuracy required for contract purposes. This led us to look at an alternative method to calculate the interview durations. Our previous use of Audit Trails was limited to using the data to recreate corrupted interviews and for ad-hoc analysis to investigate incidents of unusual interviewing behaviour. The third party contract requirements led us to investigate how we could use the audit trail data to extract more accurate interview durations. As part of the design process we examined other potential uses of the Audit Trail data, such as creating key process indicators that can be used to monitor and improve the data collection process. This paper will look at our design approach and the solution that has been implemented as part of the Household Transformation project.

2. Introduction

Paradata is defined as data about the data collection process that can be used to manage and improve data collection processes. A particular type of paradata, Audit Trail data, is generated as part of the computerised interviewing process. Audit trail files are key logging files that record every keystroke made by an interviewer during the course of an interview. The files created are large, semi-structured text files that are difficult and time consuming to process, however, the data they contain can be very valuable. Finding a way to convert the audit trail data into a useable format that can be searched and queried to provide insights into the data collection process is desirable for many organisations. As part of the CSO Household Survey's Transformation Project a new datamodel structure was developed and implemented for the audit trail data that makes it easier for users to extract information that can be used to measure and monitor the data collection processes and interviewer performance. Access to this data, moves the office from a 'black box' data collection process to one with complete transparency that can be measured and improved.

This paper describes a solution that allows the integration of the audit trail data with other household paradata to provide full transparency across CSO's household surveys field and call centre operations. Finding a solution that converts all the data available within the audit trail files and storing it where it can be used with other paradata maximises the business value that can be achieved from the data. This project resulted in the creation of new database tables for the audit trail data, with an accompanying ETL (Extract, Transform and Load) process, to parse the data into the new format. Thereby, providing a new structure for the audit trail data; making it easier to work with the data, and easier to extract information that can be used to inform many aspects of the data collection processes.

3. Paradata Benefits

Audit trail files are automatically generated key stroke logging files that are created as part of a computer assisted data collection process, such as face to face interviewing using an electronic questionnaire, telephone interviewing or self-interviewing via the web. The Audit trail data can provide information about how interviewers complete and navigate through the electronic survey

instrument. The data is collected at the time of the interview with the survey data and without interfering with the interview process (Snijkers & Morren, 2010). The audit trail data can be used to provide transparency of the interviewing processes; making it possible to improve the management and quality of the data collection processes. Unfortunately, the audit trail files that are generated are large semi-structured text files that are difficult to process. It is believed that survey paradata can be used to provide a wealth of information about the interview process but it is only as useful as the ease with which the data can be accessed, manipulated, and analysed (Devonshire, et al., 2012). The audit trail files in their natural state are difficult to work with and need to be converted into a different format to make them more usable. There is a consensus in available literature on the subject that the audit trail files contain a wealth of information that can provide valuable insights into the data collection processes for instrument diagnostics, survey methodologists and interviewer managers.

Table 1. Documented Uses of Audit Trail Data

Use of Audit Trail Data
<ul style="list-style-type: none"> • Better understanding of the data collection processes
<ul style="list-style-type: none"> • Monitor data collection and identify problems in a timely fashion
<ul style="list-style-type: none"> • Produce quality control measures
<ul style="list-style-type: none"> • Inform survey design decisions; use to evaluate survey instruments, alleviate response burden and look for potential difficult areas in the questionnaire
<ul style="list-style-type: none"> • Minimise survey error and improve the data collection process
<ul style="list-style-type: none"> • Monitor non response bias and correlation of likelihood of participation
<ul style="list-style-type: none"> • Review interviewer effect on measurement error using question timings, keystrokes, common suspension points, interviewer response to an error check.
<ul style="list-style-type: none"> • Look for patterns of interviewer actions associated with fatigue, use the results to improve recruitment, training and supervision of interviewers.
<ul style="list-style-type: none"> • Manage interviewer performance – monitor date/time stamps are in chronological order
<ul style="list-style-type: none"> • Examine keystrokes to determine fields with comments, fields associated with backup actions, fields with error checks

4. Problem Evolution

The Household Surveys Development (HSD) project objective is consolidation and automation of routine tasks across all household surveys to deliver efficiencies and improve the processes associated with the data collection operations. A major component of this project is the introduction of a third party contractor to conduct telephone interviews for the repeat interviews of one of the office’s flagship household surveys, the Labour Force Survey (LFS). Traditionally, CSO household surveys have all been conducted as Computer Assisted Personal Interviews (CAPI), also known as ‘face to face’ interviews. Telephone interviewing is being introduced for repeat waves of the LFS to free up capacity among the permanent field staff to make it easier to conduct new surveys. Until now to introduce a new survey, a new survey administration section was required to manage the survey and a new field force had to be recruited, trained and managed for the duration of the survey. Introducing telephone interviews for repeat waves of the LFS will increase the capacity among the permanent interviewers to undertake new surveys as they arise.

The contract payments negotiated with the call centre service provider is based on the duration of complete interviews. Call centre pricing models can vary; examples include costs per call, per agent hour and all inclusive rates per campaign. Usually the trade-off based on the pricing model is

between the quantity and quality of calls (Bucher, 2013). The model chosen should reflect the call outcome objectives that are desired. For CSO, the objective is good quality interviews and respondents' commitment to continue to participate in the longitudinal aspect of the survey. Building a rapport with the respondents to encourage future participation in the survey and the likelihood of a positive response the next time they are called is more important than conducting a large quantity of calls within a specific timeframe. Negotiating a pricing model based on interview durations was chosen to focus the call centre and their agents on the interview quality rather than the volume of calls per hour.

The original business problem emerged during the initial test phases with the call centre when it proved difficult to extract accurate interview durations from the questionnaire. Until now the duration of interviews was not an essential requirement for the field interview operations as it was not a factor in the field interviewer remuneration. The interview duration was previously calculated using time stamps that have been inserted into the electronic survey instruments at predetermined questions. It was accepted that the time stamps provided an indication of the interview duration only as the timestamps built into the questionnaire can prove unreliable for a number of reasons. For example, the multiple routes and exit options which can result in no end-time being recorded; in addition, an interviewer may open and close the case a number of times, this may be to preview the case before a call or correct text values after a call, a high number of sessions per case can result in mis-matched start and end times. The contract with the call centre highlighted the deficiencies using the timestamps to calculate accurate interview durations. The time stamps did not provide the accuracy required to calculate contract payments. The test case results included incidents of cases that did not match the third party recorded times and incidents where it was not possible to calculate interview durations due to missing end-times.

An alternative paradata source is available that records all interviewer activity with the electronic questionnaire during the interview; including the start and end times of every interview session. The audit trail file is a key logging file that is generated automatically during the interview process. The audit trail data captures every key stroke activated during an interview session. As a result a large volume of data is generated for each interview session however, the volume of data generated leads to a more accurate representation of the interviews. This data is in a semi-structured format which means it cannot be queried or searched in its original form. Current uses of the audit trail data had been limited to recreating interviews if they've been corrupted or investigating interviewer performance if there is a suspicion of inappropriate behaviour. Both of these tasks were manual, laborious, complex ad-hoc processes. However, the options available to resolve the call centre contract problem were either to develop a more robust way of recording time stamps in the questionnaire or finding a solution to use the audit trail data. The difficulties associated with the timestamps focused attention on the possibility of finding a more efficient way to use the audit trail data to extract the duration information required.

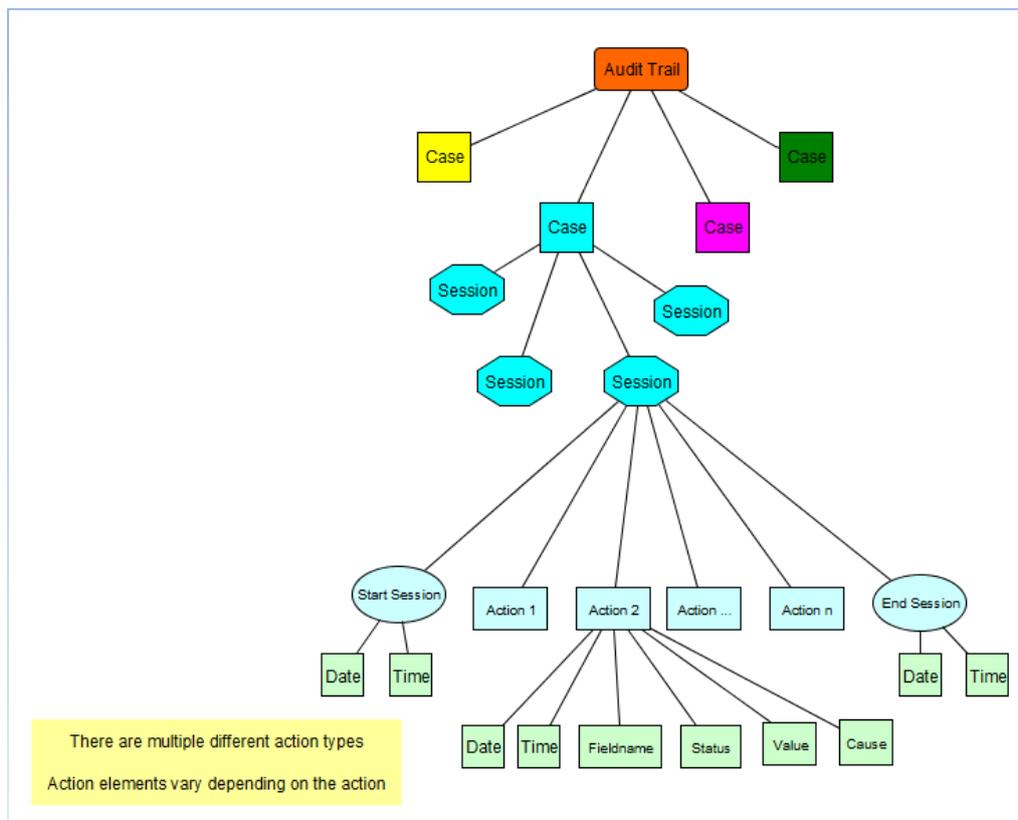
5. Solution

The primary objective was to build a set of database tables and a parsing tool that could be used to restructure the audit trail data to enable easy extraction of the interview durations. Semi-structured data is characterised by a lack of a fixed schema and record layout, although typically the data has some implicit structure. The audit trail data is further complicated because the number of actions taken by the interviewers are not fixed in advance resulting in a variable number of observations per interview session (Olson & Parkhurst, 2013). In addition, each record can have a different structure and length. The goal of working with this data, presenting and querying it is impaired by its original structure. The critical problem to resolve is the discovery of the structure implicit in the raw data and, subsequently, recasting the data into a structure that is easier to work with (Nestorov, et al., 1998).

Patterns in the data were used to map and identify data elements, see Figure 1. Depending on how the audit trail is generated it can contain many case files. Each case will contain one or more interview, browsing and/or editing sessions. Each session will have a start and end action with an associated date and time. Each session usually has a number of actions; the number of actions is a variable

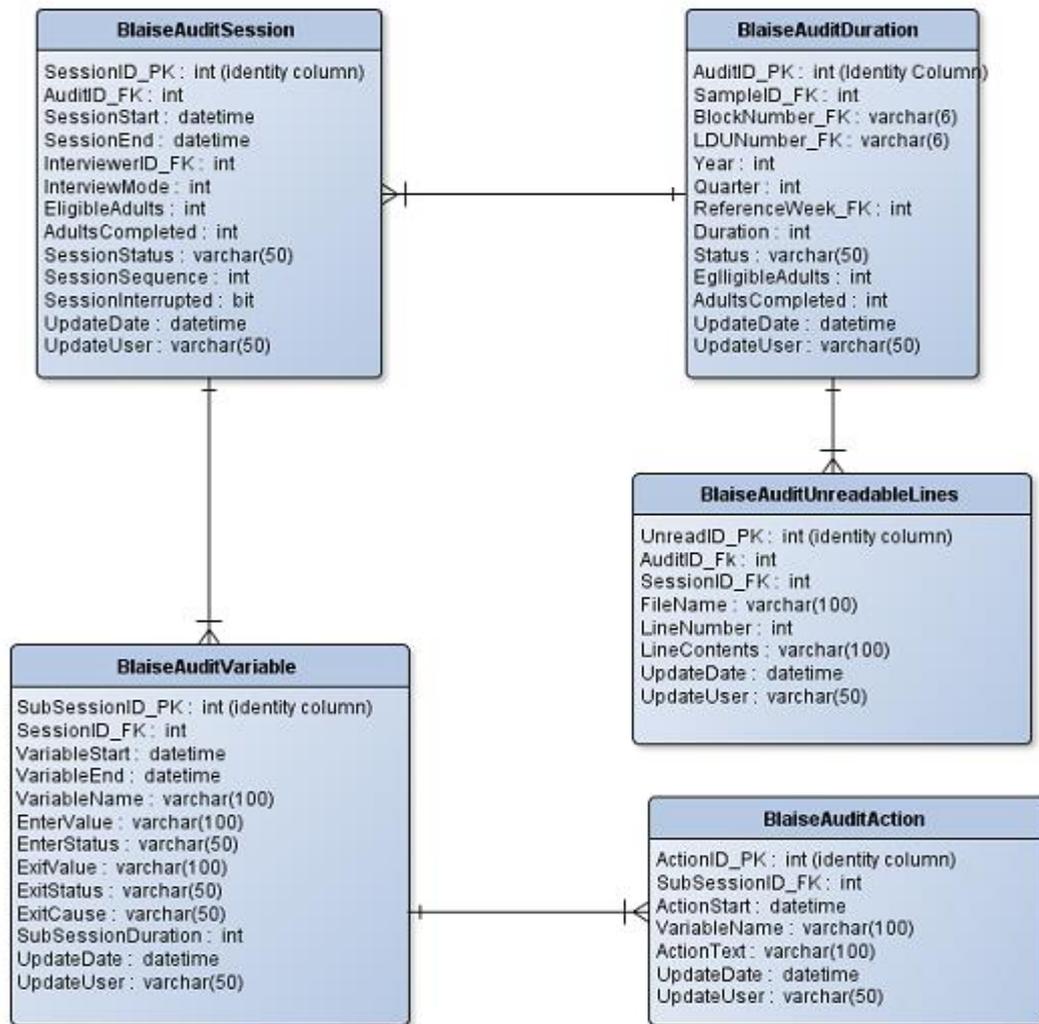
number completely dependent on what happens during an interview or editing session. There are a number of configuration options available that control how the files are generated. The current on-going household surveys generate an audit trail file by interviewer spanning a survey period. In this scenario there are multiple cases in each file, with corresponding lower level details. Every time a case is accessed by an interviewer the details are appended to the audit trail file. To facilitate transferring the data to the office the files are purged after a successful data sync with the office, to help manage the size of the files over the survey period. For our new developments the survey case files are being structured differently, each audit trail file will hold the details of one case only but there can still be multiple sessions and corresponding lower level details. The added complexity will also be that there may be different interviewers as the audit trail file is now part of the total case object package and will be kept with the case details while the case is active in the survey.

Figure 1. Audit Trail Data Graph



A set of hierarchical database tables were designed around the data elements to include a session table, containing details of each session; a duration table, to keep track of the overall time that the case was accessed; a variable table, to record each variable that was accessed and an action table to hold all actions associated with the variables. Some additional aggregate fields have been added to allow for easy access to the duration of an interview. A misc. table was also created to hold details of lines that could not be processed. An analysis of the records been added to this table identified them as being caused by an interrupted session where no end time was being recorded. Consulting with the users it was decided to manage these records by identifying them as interrupted records and adding the last time recorded to the record ensuring a total duration could still be calculated for all sessions and that interrupted sessions could be identified.

Figure 2. Audit Trail Database Implementation



An ETL process written in Java was used to populate the data tables. Each audit trail record is searched to find an identifying record attribute and then written to the corresponding table depending on the attribute found e.g. ‘Open Session’, ‘Enter Field’ etc. Each record added to the table is given a unique identifier that maintains the sequence of records from the original audit trail data. The final datamodel is not completely normalised but this was not the goal. The data is not required to support a transactional process rather the goal is to make it easy for users to query and search the data. Fully normalised data can be easier to import and update but it can be harder to get the data out as multiple tables and join paths can make queries less efficient and harder to code correctly (Corr & Stagnitto, 2012).

This project has resulted in a new data model created for the audit trail data that can be used to extract accurate interview durations for CATI and CAPI interviews. A new ETL process to handle the data import has been developed and tested. A new CATI Interview Duration report has been developed. The audit trail data parsing process has been added to the overnight case processing job ensuring the latest audit trail data is available to the business as it is received. The data quality and accuracy of the audit trail data has been verified and we can be confident that we are producing accurate interview durations for the contract. This is a good resolution of the original problem that meets all the requirements of the call centre contract manager; who can be confident in their ability to handle the call centre contracts based on accurate interview durations.

The audit trail data can now be used with the other para data such as the Call History data to provide complete transparency across the household survey data collection operations. This moves the office from a scenario where there were a lot of unknowns about the data collection in the field to having complete transparency across the data collection process for both modes; ‘face to face’ and telephone. The data provides a wealth of information to fully understand how the data collection process is operating. Data from the audit trails can be used to track days and times of interviews, monitor interview durations and review interviewer performance. The solution can be used to deliver automated reports to the business users on the field and telephone interviewing operations. The overall result should be an improvement in the management capabilities for household survey data collection through having access to detailed information on what actually happens in the field.

6. Potential Use of the Data

The audit trail data can be used to improve a number of the household survey operational and design processes ensuring they conform to the highest standards and best practices. The project identified opportunities to use the audit trail data to support process improvements for data collection processes via managing and monitoring key performance indicators. The audit trail data can be used to provide performance indicators to monitor and improve questionnaire design and testing processes, management of field and call centre operations and interviewer performance thus, leading to improvements in the overall data quality. Working with business stakeholders we were able to identify the following types of reports that could be created from the audit trail data to assist with managing the data collection operations.

Table 2. Reports Based on Audit Trail Data

Report Name	Description
Interview Duration	Interview durations viewed by session, mode, interviewer
No. of Sessions	Number of session to complete an interview - Analysis by mode, interview, duration of session
Interview Activity	Interview activity by day of the week and time of the day
Module Duration	Module duration – specify start and end question to calculate duration for a section of the interview
Interviewer Performance	Compare interview durations by interviewer to identify outliers (possible indication of performance issue)
Aggregate Performance	View durations, active days and time across multiple surveys by interviewer - report that integrates audit trails from all four surveys so that interviewer workload and working patterns across the days of the week could be observed
SI Activity	Monitor activity through the questionnaire to identify design issues - Route taken through questionnaire - No. of error generated - Use of Help functionality - Frequency of returning to specific questions

A key organisational strategic goal is the improvement of the quality of all our statistical processes. The Eurostat report on quality (Eurostat, 2002) distinguishes between different types of quality. They contend that in the case of a statistical organisation the quality focus should be on the data produced and the services provided. To achieve the best quality product outcomes improving process quality is a key aim. The report explains that process quality can be improved by identifying key process variables, measuring these variables, adjusting the process based on these measurements and checking what happens to the product quality. This is an example of the Plan, Do, Check, Act cycle advocated by Demming (1991).

Eurostat created a handbook to identify key process variables, their measurement and measurement analysis (Aitken, et al., 2004). Key process variables are those judged to have the largest effect on pre-defined critical product characteristics. The best indicators of quality are process variables that can be observed conveniently and continuously during the survey process and that are highly

correlated with the components of error that need to be controlled (Biemer & Lyberg, 2003). Table 3 is a list of key process variables for the data collection processes identified in the Eurostat handbook. Call History data is being collected as part of the Household Survey Transformation Project. Providing a solution for the Audit Trail data means it should also be possible to extract the key performance variables from that data source. Access to this data puts the CSO in an excellent position to monitor and improve the data collection processes based on the Eurostat defined key performance indicators. The Data Source column in Table 3 indicates the potential data source to access the variable from the CSO systems and data sources.

Table 3. Data Collection Key Process Variables

Process	Process Variable	Measurement	Data Source
Data Collection – survey instrument performance	Ability of respondents to answer a problem question	Analysis of responses and comments relating to the question	Survey data
	Number of editing errors by question	Number of errors activated	Audit Trail data – check number of errors activated by survey instance
	Route through interview	Number of deviations from automatic route	Audit Trail data – check next action after Exit from field
Data Collection – Interviewing Activity	Number of non-responses	Count of unanswered questions	Survey data
		Frequency of unanswered questions	Survey data
	Use of Help function	Count of Help activation	Audit Trail data – check number of time HELP is activated by survey instance/interviewer
		Frequency of Help by questions	Audit Trail data– check number of time HELP is activated by question
Data Collection – Interviewer Performance	Interview time	Duration of interview	Audit Trail data
	Travel time of interviewer	Duration of travel time to/from interview	Call History data
	Working hours by survey	Duration of total time including travel, admin and interviewing	Combined Call History, Admin and Audit Trail data
	Contacts by time period	Aggregate of case contacts by day and time	Call History data

Duration at all levels such as total interview time, module length and question time are considered good quality indicators that can be used to assess the quality of the survey instrument and monitor interviewer performance to assess that the questions are being asked correctly. Previously, it was not possible to access this level of detail; now this data will be easily available and can be compared across interviewers and modes. The data from the audit trails can be used to identify problem areas in the survey instruments and identify if they are design or interviewer training issues. The information can be used to target interviewer training sessions to address problem questions thus ensuring the data collection is more efficient and effective. Alternatively, insights from analysis of the data may indicate a questionnaire design issue that can be reviewed to improve the questionnaire.

The data can be used as part of a continuous improvement process, reviewing and monitoring the key performance indicators to improve the effectiveness of business processes. For example, innovation at the business process level can be achieved if the results from new questionnaire testing phases are used to inform and improve future design and development phases. The data can be used to evaluate new questionnaire designs, to identify potential problems with questions, routing or answer types. The test results can be used to improve new questionnaires ensuring the best possible questionnaires are being used for data collection, avoiding expensive data collection difficulties. The audit trail data provides the richest source of data into what is going wrong in a survey instrument and understanding the usability of questionnaires for interviewers (Olson & Parkhurst, 2013).

In addition to improving the quality of processes there are a number of known errors that can manifest during the data collection process that need to be managed (EuroStat, 2004), Table 4. Interviewers can have an effect on what is recorded in a computerised interview; in most household surveys the interviewer directly inputs respondents' answers into a computer. Variability across interviewers can lead to variation in what is recorded. In addition interviewers can affect what respondents report and respondents can affect interviewer's behaviour, for example, some interviewers may probe more and some respondents may be more likely to seek clarification (Olson & Parkhurst, 2013).

Table 4. Known Data Collection Errors

Error	Description	Measure and Corrective Action
Interviewer Error	Interviewer errors are associated with effects on respondents' answers stemming from the different ways that interviewers administer the same survey. Examples of these errors include the failure to read the question correctly (leading to response errors by the respondent), delivery of the question with an intonation that influences the respondent's choice of answer, and failure to record the respondent's answer correctly.	Analysis of question durations can be an indicator of questions not being asked as specified. If a problem is identified training session can be used to address the issue. Call recordings will be available from the call centre when the telephone interviewing starts. This data can be used to compare both interviewing modes and identify baseline durations for questions.
Measurement Error	Measurement error refers to errors in survey responses arising from the method of data collection, the respondent, or the questionnaire (or other instruments). It includes the error in a survey response as a result of respondent confusion, ignorance, carelessness, or dishonesty; the error attributable to the interviewer, perhaps as a consequence of poor or inadequate training, prior expectations regarding respondents' responses, or deliberate errors; and error attributable to the wording of the questions in the questionnaire, the order or context in which the questions are presented, and the method used to obtain the responses.	This may be due to poor survey design. Analysis of question durations that might indicate long pauses, high error counts or frequency of accessing Help can be indicators of difficulties for respondents understanding questions that can be improved with a better questionnaire design.
Non-response Error	Non-response errors, occur when the survey fails to get a response to one, or possibly all, of the questions. Non-response causes both an increase in variance, due to the decrease in the effective sample size and/or due to the use of imputation, and may cause a bias if the non-respondents and respondents differ with respect to the characteristic of interest.	Count of Don't Know and/or Refusal answers to identify problems with specific questions and/or issues with interviewer performance.

The new structure for the audit trail data allows different business users to access the data in a way that suits them and their needs, via reports or in a self-service way that allows exploration of the data. There is potential for considerable value to be realised as a result of this implementation however, it will take some time before the full impact of using the data in this way can be assessed.

7. Conclusion

This project has used a previously underused data source to solve the original business problem with the call centre contracts and provide complete transparency of CSO's household survey data collection operations. Business users have new opportunities to use the audit trail paradata to manage, monitor and improve the data collection processes. Using the audit trail data to provide insights to manage the data collection process is superior to the alternatives such as relying on subjective interviewer feedback or anecdotal data; the audit trail data is quantifiable and can be used to identify trends or problems (Duncan, 2015).

Easy access to this data provides the potential to develop a more mature approach to information management that can transform decision making at both strategic and tactical levels (Duncan &

Buytendijk, 2015). This project comes at a time when CSO is in transition and focused on improving the quality and precision of all its processes and statistical outputs. The audit trail data is an important data source that can be used to monitor new data collection process indicators and in Peter Drucker's¹ words 'What gets measured gets managed'. The data can be used to manage key process indicators that can be used to improve the quality of the data collection processes and consequently the quality and precision of the statistical outputs. Data analytics, such as this, can provide a stimulus for business areas to take transformational action steps and enable operations to be more data driven, using evidence based decision making to drive process improvements.

8. References

- Aitken, A. et al., 2004. *Handbook on improving quality analysis of process variables*, Brussels: European Commission Eurostat.
- Biemer, P. P. & Lyberg, L. E., 2003. *Introduction to Survey Quality*, s.l.: Wiley.
- Bucher, A., 2013. *5 top trends for call centres and the pricing model dilemma*. [Online] Available at: <http://www.trinityp3.com/2013/09/call-centres-pricing-model/> [Accessed 24 Feb 2016].
- Corr, L. & Stagnitto, J., 2012. *Agile Data Warehouse Design*. 1st ed. Leeds: DecisionOne Press.
- Devonshire, J., Liu, Y. & Cheung, G.-Q., 2012. *Blaise Audit Trail Data in a Relational Database*. s.l., Survey Research Center, University of Michigan.
- Duncan, A. D., 2015. *Take an Emotional Approach to Business Analytics to Develop a Data-Driven Culture*, s.l.: Gartner.
- Duncan, A. D. & Buytendijk, F., 2015. *How to Establish a Data-Driven Culture in the Digital Workplace*, s.l.: Gartner.
- Eurostat, 2002. *Quality in the European statistical system - The way forward*, Luxembourg: Office for the Official Publications of the European Communities.
- EuroStat, 2004. *The European Self Assessment Checklist for Survey Managers*, Luxembourg: Office for the Official Publications of the European Communities.
- Nestorov, S., Abiteboul, S. & Motwani, R., 1998. *Extracting schema from semistructured data*. New York, ACM, pp. 295-306.
- Olson, K. & Parkhurst, B., 2013. *Collecting Paradata for Measurement Error Evaluations*. Nebraska: University of Nebraska - Lincoln.
- Snijkers, G. & Morren, M., 2010. *Improving Web and Electronic Questionnaires: The Case of Audit Trails*. Heerlen: Statistics Netherlands, Department of Methodology and Quality.
- West, B. T., 2011. Paradata in Survey Research. *Survey Practice*, 4(4).

¹ Peter Drucker (1909-2005) American management consultant, professor and writer.

Blaise 5 Development at University of Michigan

Max Malhotra and Youhong Liu, University of Michigan Survey Research Center, United States

1. Abstract

We have used Blaise 5 for two survey studies at the Survey Research Center, University of Michigan since two years ago. Blaise 5 and Blaise 4.8 are similar in some ways, but are also different in many other aspects. The two studies we have developed are complex in many areas. One of them is converting a large Blaise 4.8 survey into Blaise 5. The second project is a Mixed Mode Survey. In order to make the surveys flexible, and to present users with friendly interfaces and experiences, we have designed and implemented many different features. We will present how the mixed mode functions are achieved, and how are the challenge requirements programmed. In addition, Blaise source programming, Blaise layout setting, and Resource database development will also be discussed.

Using Blaise 5 for self-administrated surveys posed unique technical challenges due to the complexity of survey design and stringent requirements pertaining to response collection of self-administered web respondents. These requirements mandated the design and implementation of unique back-end APIs that needed to be concise in their delivery so that they could be quickly integrated into a survey using role text to pass parameters from the Blaise 5 data model to the API. This paper looks at how Blaise 5 capabilities and features work with the creation of the aforementioned APIs and we shall show some practical examples of how some of these functions can be best utilized. A technical discussion of issues encountered, what worked well and lessons learned.

In particular, we will discuss:

- Select All, Clear All (Checkbox)
- Mutually Exclusive Response (MER) with Clear All (Checkbox)
- Select Mutually Exclusive Response (MER)
- Other Specify
- Other Specify to clear out textbox
- Single text box with single radio button
- Multiple textboxes with single radio button
- Single textbox with multiple radio buttons
- Multiple dropdown menus with single radio button
- Multiple dropdown menus with multiple radio buttons
- Multiple dropdown menus with multiple radio buttons and with textbox
- Disable Radio Buttons If Mask Textbox Is Zero or Empty
- Disallowing browsers based on browser type and/or browser version

We shall provide a demo of some of the above mentioned custom user interface functionality at the conference.

2. Introduction

The University of Michigan Survey Research Operations began using Blaise 5 at the end of 2014. Over the last two years, we have programmed and implemented two studies using Blaise 5. The first study is a longitudinal study that has been on-going at the University of Michigan since 1968. The Blaise 5 development for this study was implemented in two stages. The first stage was a pilot survey that was conducted using opt-in samples. The second stage was a pilot survey that was conducted among selected study sample.

The second study is also longitudinal, however it began more recently. This study also had two phases. The first phase was a pretest that was conducted at the end of last year. The second phase, which is currently in the field, is the production phase of the study.

The first study is a large survey. It has many fields, loops and preloaded data fields, so the datamodel programming was more challenging. The second study is complicated as well because it is a mixed mode project and the study project staff had more intricate requests with regard to layout and other aspects of the survey.

The Blaise out of box tools, for example, Blaise language, Blaise resource database can help us to achieve most of the functionalities. However, using Blaise 5 for self-administered surveys posed unique technical challenges due to the complexity of survey design and stringent requirements pertaining to response collection of self-administered web respondents. These requirements mandated the design and implementation of unique back-end APIs that needed to be concise in their delivery so that they could be quickly integrated into a survey using role text to pass parameters from the Blaise 5 data model to the API.

As Blaise 4.8 developers with some .net programming experience, there was a big learning curve. We faced many challenges and learned a tremendous amount about programming in Blaise 5. In this paper, we will present some major functionalities that were developed with Blaise tools, as well as API functionality.

3. Mixed Mode Programming

Data collection for one of the studies has two modes: the Self-Administered Questionnaire (SAQ) mode and the Computer-assisted Telephone Interview (CATI) mode conducted by our centralized telephone interviewers. We had several meetings to discuss the best approach to implementing a robust survey that would meet the client's requests, while keeping the process simple. In the end, we decided to use one instrument for both modes and came up with the following implementations:

1. Used a variable called *MODE* to route surveys differently if SAQ and CATI had different questions. One was for SAQ and two was for CATI. The field value was set by using a parameter that was passed into the survey. We had to pay special attention to the *AssignMode* parameter. By default, *AssignMode* was *New*, meaning values would be assigned to the fields in case a new form was created. *Existing* could be used to assign values to fields in existing forms. In our case we used *Always* to assign values to fields in new and existing forms.
2. Used two *MODES*: SAQ and CATI. *MODES* in this case is a Blaise 5 key word that is different from the *MODE* variable created in the survey. It is a new attribute in later releases of Blaise 5. With this new feature added, we were able to specify different attributes between SAQ and CATI easily. In our implementation, *Empty* was allowed in SAQ so Respondents could skip any questions in the survey, But *NoEmpty* is defined in CATI so interviewers could not skip any questions.
3. In the study pretest, Web Browsers were used for both SAQ and CATI interviews. In production, we moved CATI to DEP. It was a smooth transition and we found it was more stable and faster in terms of conducting the survey in DEP and integrating it with other components of the Blaise 5 service.
4. There were four layout sets defined in the survey: Desktop Web, CATI, Medium Mobile Device Web and Small Mobile Device Web. The Desktop Web layout was developed first. It took some effort since we were new to programming in Blaise 5. CATI also took longer to develop because it was significantly different from the Desktop Web layout. When the Desktop Web and CATI layouts were stable, we moved to Mobile Device layout development. The Desktop Web layout

had question table grids and other question elements that would not be displayed properly on small devices. After some investigation, a developer started to work on the mobile device resource set and applied the resource set to a sample survey. After the sample survey was tested by our client, we applied the resource set to our production survey. We found that implementing the survey on different devices in Blaise 5 was very easy by using resource and layout sets.

5. Define two data entry settings, one for each mode. For security reasons, we wanted a short Session Timeout: 10 minutes. This is great approach for the web SAQ mode. But during the pretest in CATI mode, we found a problem on one screen. On that screen, some interviewers spent more than 10 minutes entering data into an open text field. When they moved to the next screen, the “Session Timeout” message came up. At that point, the answers in the open text field were lost. In production, the solution for this problem was to add a special Data Entry Setting that is used for CATI mode only. In that setting, the Timeout is 30 minutes. At same time, the Save function was added on the screen for interviewers to use.

4. Integrating Datamodel and Resource Database Programming

Blaise 5 and Blaise 4.8 are similar, but they are also very different in some aspects. On some of our Blaise 4.8 Window and IS projects, we used expression programming to achieve requests from our clients. For instance, in one of the Blaise IS projects, expression programming in the project’s menu editor was utilized. Based on the field name, an external web page was opened on certain screens, so the Respondent could go to that site to take a set of cognitive tests.

But we could not find ways to communicate between Datamodel, Menu, Modlib or other parts of Blaise components. Therefore, there are many functionalities that could not be achieved in the Blaise 4.8 projects. For example, two instruments had to be developed for a SAQ and CATI mixed mode project. This resulted in dual programming - two sets of source code had to be programmed and maintained. With the two instruments approach, the data manipulations were also very challenge.

In Blaise 5, Datamodel and Resource Database programming communications are greatly extended. With careful implementation between these two, we believe we can fulfill almost anything requested by our clients. We elaborate a few examples below.

1. When a case was resumed, it jumped to the beginning of the previously suspended section. Further, if there is a mode switch, the section data should be deleted. One project made this request because some of the data were time related, meaning the follow up questions were related to their root questions and the data needed to be gathered in timely fashion. The deletion after a mode switch was because different set of questions could be asked between SAQ and CATI mode.

After many tests, below were the sequences we developed:

- First, two variables in the Datamodel were created: ActiveFieldName and SuspendFieldName. ActiveFieldName was then mapped in the Resource Database.
- ActiveFieldName was updated in the Master Page’s page loading event and then updated on every survey screen. This was done in the Resource database. Since it is a mapped field, the Datamodel recognized the value of the field.
- SuspendFieldName is updated only when a case is resumed. This was done in the Datamodel.
- Then in the Datamodel, the resume page was created. It was the first page of the survey. The page will show if SuspendFieldName contains block names, e.g., “Section_A”, Section_B”, etc. On the page, the resume button was created and actions were attached so that the survey

would jump to the beginning of the section when the button was clicked. This was done in the Datamodel.

- If a mode switch is detected, the section data was deleted.
- At first, the resume buttons were created on the master page of the Resource Database. It was not easy to show and hide, and difficult to add the GotoField event to the buttons. In production, only one button was used; the button itself and its event were moved into the Datamodel by using dynamic enhanced text fills in a text role. This was a significant improvement in terms accuracy and code maintenance.

The list above is the simple description of the function. The actual programming was much more complex. Careful and skillful Datamodel programming is essential. Some intermediate fields were needed. Also .Keeps statements are used to ensure that data are properly saved.

2. Determine if a survey is complete. In Blaise 4.8 CATI, we used the following code:

Listing 1. Determining Completion Source Code

```
Completed.Keep
IF Thankyou = Response and Completed=empty THEN
    Completed := Done
ENDIF
```

The above code could also be used in Blaise 5, but there was a problem in the SAQ interview. In the SAQ, we really could not require a Respondent to answer any questions. Therefore, the *Thankyou = Response* condition could not be used and extra steps were required. Our approach was to create a special page just for *ThankYou* question. The *Thankyou* field was mapped in the Resource Database. Then on the master page’s “Next” button, Assign *ThankYou = ‘1’*.

This approach was efficient but had some drawbacks. When the mobile *Thankyou* Master page was defined, we had to add the same code in the page’s “Next” button. This could be easily forgotten if we have other *Thankyou* pages that are used for different devices. We may be able to use the *ActiveFieldName* described above to program this function as well. The code might be cleaner since everything related is in the Datamodel.

3. Amount/Per Screen with a version button. (See the screen below.)

Figure 1. Amount/Per Screen with Version Button

The screenshot shows a survey interface for the Family Economics Study (FES) at the University of Michigan. The main question is "About how much rent do you pay a month?". Below the question is a blue button that says "To enter an amount per year instead, click here." Below that is a text input field with a dollar sign and ".00 per month". There are "Next" and "Previous" buttons. At the bottom, there is a status bar with the following information: "VDate: 6/29/2016 VTime: 10:10:00 FESCTI" and "SampleID: A\$\$\$\$1000001AS, Active Field: Section_A.A31Version". There is also a "First Page (Select Section)" button.

On this screen, the project staff requested that:

- 1) The amount per month field is presented to the Respondent by default;
- 2) A version button is also presented;
- 3) If Respondent clicks the button, the amount per year is presented;
- 4) If the button is clicked again, the amount per month is presented again.

We had never done anything like this before, neither in Blaise 4.8 IS nor in any other survey systems. Again, Blaise 5 provided new options. With careful Datamodel and Resource database programming, we were able to create the function.

- First the AmountPer Field Pane template was created. In the template, buttons were created based on the enumeration type of the version button defined in the Datamodel. The number of buttons, the show/hide, text property and the event behind the buttons were also programmed by expression based on the enumeration type.
- Then, on the Datamodel side, the following code was used to make the final show and hide the month and year fields mutually excluded.

Figure 2. Datamodel Source Code

```
GROUP GROUPA31 "About how much rent do you pay ^xA31_1? "  
  FIELDS  
    A31Version: TVersionButton8  
  
    A31          (A31)  
                EntryPre "$"  
                EntryDescr ".00 per month"  
                / "Rent Amt per month" : TDollar5  
  
    A31Yr        (A31Yr)  
                EntryPre "$"  
                EntryDescr ".00 per year" /  
                "Rent Amt per month" : TDollar5  
  
  RULES  
    A31Version.keep  
    IF A31Version = EMPTY THEN  
      A31Version:=Month_  
    ENDIF  
    A31Version  
    xA31_1:='a month'  
    IF A31Version = Year_ THEN  
      xA31_1:='a year'  
    ENDIF  
    IF A31Version = Month_ THEN  
      A31  
    ELSEIF A31Version = Year_ THEN  
      A31YR  
    ENDIF  
  ENDGROUP
```

In this study, there are many Amount/Per screens. Some have more than two versions and others have different texts to be displayed on the buttons. With careful implementation, we were able to use the same template for these similar screens.

5. Designing and Implementing APIs in Blaise 5

We will now look at how Blaise 5 capabilities and features work with the creation of the aforementioned APIs and we highlight some practical examples of how some of these functions can be best utilized, including discussion of challenges encountered.

In particular, we will discuss:

- Select All / Clear All (Checkbox)
- Mutually Exclusive Response (MER) with Clear All (Checkbox)
- Select Mutually Exclusive Response (MER)
- Other Specify
- Other Specify to clear out textbox
- Single text box with single radio button
- Multiple textboxes with single radio button
- Single textbox with multiple radio buttons
- Multiple dropdown menus with single radio button
- Multiple dropdown menus with multiple radio buttons
- Multiple dropdown menus with multiple radio buttons and with textbox
- Disable Radio Buttons If Mask Textbox Is Zero or Empty
- Masks
- Disallowing browsers based on browser type and/or browser version

We shall provide a demo of some of the above mentioned custom user interface functionality at the conference.

5.1 Challenges

Some challenges that we faced were that the current Blaise info-structure did not contain certain project-specific control capabilities that our projects demanded. One example of such a control would be *Select All / Clear All* functionality.

For example if we have a question that asks “In which months did you pay the utility bill last year?” it would be safe to say that a large majority of respondents may pick all twelve months, however we would not want a respondent to have to click on all twelve month checkboxes thus we developed an select all clear all API that allows the respondent to click one button to select or deselect all twelve months and if the respondent manually selects all twelve months the select all checkbox will automatically get selected as well.

Conversely, the respondent can choose to click on the select all button and deselect specific months and once a single month is deselected the API will deselect the select all checkbox as well saving valuable respondent time and providing for a better user experience.

5.2 Implementation

The initial design and construction of each API had to be carefully considered and architected to provide future expandability. The API’s construction utilized a variety of languages such as Blaise, C#, javaScript, and jQuery. However, our goal was to make it very easy to implement for any Blaise programmer and thus from a Blaise programmer perspective -- provided they have our more complex code installed for their specific survey -- the Blaise programmer will simply call the specific API (through a role text) and provide it the variables that are specific for that particular API.

For Example:

- Other Specify

Figure 3. Other Specify Source Code

```
Group GroupA_A07_Q
Fields
A_AA_ResnsNoOpt (A_A07_Q)
" ^xA_A07 ^xArmy military won't give you the option to reenlist? <I1>(Check all
that apply) </I1>"
/"Army - Reasons they will not let you reenlist"
RTMod "OtherSpecifyRoleText = Other"
: Set OF (_1 "You have a health, disciplinary, or legal problem",
_2 "You have an adverse personnel flag, such as failing to meet physical fitness or weight standards",
_3 "The ^xArmy military is reducing the number of servicemembers in your MOS or eliminating the MOS ",
_4 "You have a low supervisor recommendation or performance rating",
_5 "You have reached a Retention Control Point (up-or-out promotion policy)",
_6 "You are barred from reenlistment",
Other "Some other reason <I1>(Please briefly describe.)</I1>")

A_AA_ResnsNoOptSpec (A_A07_Q) : TStringOther
ENDGROUP
```

- Select All/Clear All

Figure 4. Select All/Clear All Source Code

```
F7MO (F7MO)
" For which months in ^PYear was that?
<br><br><Instruct>Please select all that apply.</Instruct>"

"¿Por cuáles meses fue eso en ^PYear?

<br count=2><tab><Instruct>• ENTER all that apply, For multiple response, use space bar or dash to
separate responses</Instruct>
<br count=2><tab><Instruct>• PROBE: </Instruct> ¿Algo más?"
/ "Months in ^P2Year Paid Child Care"
RTMod "SELECTALLCLEARALL = All"
:

TMOStringSet
```

5.3 Examples

- Select All, Clear All (Checkbox)
 - SPEC: 'Select All' (e.g., All 12 months) as a check box at the top
 - When checked, all check box items are checked
 - When unchecked, all check box items are unchecked
 - When all Items are checked the 'Select All' gets automatically checked
 - When all Items are checked, including 'Select All' and one item is unchecked, then the 'Select All' checkbox is automatically unchecked

Screen Capture:

Figure 5. Select All, Clear All Checkbox

During which months was that?

Please select all that apply.

All 12 months

January

February

March

April

May

June

July

August

September

October

November

December

- Mutually Exclusive Response (MER) with Clear All (Checkbox)

SPEC: 'Select None' (e.g., Nothing) as a check box at the bottom

When checked all check box items are uncheck

When checked if 'Other Specify' textbox exists, the textbox is cleared out.

When ANY checkbox item other than 'None' is checked, then the 'None' item is unchecked.

Screen Capture:

Figure 6. MER with Clear All Checkbox

What have you been doing the last four weeks to find work?

Please select all that apply.

Checking with public employment agency

Checking with private employment agency

Checking with current employer directly

Checked with other employer directly

Checking with friends or relatives

Placing or answering ads

Contacting school or university employment centers

Checking union or professional registers

Sending out resumes or filling out applications

Attending job training programs or courses

Going on job interviews

Looking at ads or employers without applying

Other - Please specify:

Nothing

- Select Mutually Exclusive Response (MER)

Depending on how this API is used, the API can function as a separate API that is capable of making individual checkbox items mutually exclusive as well.

SPEC (EXAMPLE):

Response options one, two and three are mutually exclusive and all other items are independent of this API.

This means if response option one is selected and then response option two is selected, then response option one will be deselected. Only one of the three allowed response options can be selected at a time. The other two cannot be selected at the same time. All the following response options are not impacted by this API.

Screen Capture:

Figure 7. MER

Now some questions about what you do. Are you...?

Please select all that apply.

- Working now
- Only temporarily laid off, or on sick or maternity leave
- Looking for work, unemployed
- Retired
- Permanently or temporarily disabled
- Keeping house
- A student
- Other - *Please specify:*

- Other Specify
 - a. CHECKBOX: When the respondent clicks on or starts to type inside the ‘Other Specify’ textbox a checkbox shall automatically be assigned to that item.

Screen Capture:

Figure 8. Other Specify

How is your home heated?

Please select all that apply.

- Gas
- Electricity
- Oil
- Wood
- Coal
- Solar
- Bottled gas or propane
- Kerosene
- Other - *Please specify:*

- b. RADIO BUTTON: When respondent clicks on or starts to type inside the ‘Other Specify’ textbox a selected radio button shall automatically be assigned to that item.

Figure 9. Radio Button

Do you live in a...?

- One-family house or condo
- Two-family house, duplex, or condo
- Apartment or condo in a multi-unit building, or project
- Mobile home or trailer
- Row or town house, or attached condo
- Other - Please specify:

- Other Specify to clear out textbox

When checkbox is unchecked any entry inside the textbox is cleared.

Figure 10. Other Specify to Clear Out Textbox

How is your home heated?

Please select all that apply.

- Gas
- Electricity
- Oil
- Wood
- Coal
- Solar
- Bottled gas or propane
- Kerosene
- Other - Please specify:

- c. If 'Select All' (checkbox) option is present then and 'Select All' is deselected, then the 'Other Specify' textbox shall be cleared out as well.

Figure 11. Select All/Other Specify

How is your home heated?

- Select All
- Gas
- Electricity
- Oil
- Wood
- Coal
- Solar
- Bottled gas or propane
- Kerosene
- Other

- d. RADIO BUTTON: If respondents type inside the 'Other Specify' textbox and then decide to switch to a different radio button, the text inside the 'Other Specify' is cleared out.

Screen Capture:

Figure 12. Radio Button/Other Specify

Do you live in a...?

- One-family house or condo
- Two-family house, duplex, or condo
- Apartment or condo in a multi-unit building, or project
- Mobile home or trailer
- Row or town house, or attached condo
- Other - Please specify:

- e. The following screen shows same process for the "Other – Please specify:" function.

Figure 13. Radio Button/Other Specify

About how much rent do you pay?

\$.00 per Week

- Two weeks
- Month
- Year
- Other - Please specify:

- Single text box with single radio button

SPEC: If the respondent types inside the textbox and then decides to click on the radio button, then the text box shall be cleared out. If respondent selects the radio button and then decides to click or start typing inside the textbox, then the radio button shall be cleared out.

Screen Capture:

Figure 14. Single Text Box, Single Radio Button

About how many more years will you have to pay on it?

Number of years

- Less than one year

- Multiple textboxes with single radio button

SPEC: If the respondent types inside any textbox and then decides to click on the radio button, then ALL the text boxes shall be cleared out. If the respondent selects the radio button and then decides to click or start typing inside any textbox, then the radio button shall be cleared out.

Screen Capture:

Figure 15. Multiple Text Boxes, Single Radio Button

What is the EARLIEST age at which you would be ELIGIBLE to receive "full" or "normal" pension or retirement benefits? If you are already eligible, what was the earliest age at which you could have become eligible?

Please enter age in years and months. For example, for 59½ years old, enter 59 years and 6 months.

Year

Months

No age requirement

- Single textbox with multiple radio buttons

SPEC: If the respondent types inside the textbox and then decides to click on ANY of the radio buttons, then the text box shall be cleared out.

If the respondent selects ANY of the radio buttons and then decides to click or start typing inside the textbox, then the radio button shall be cleared out.

Screen Capture:

Figure 16. Single Text Box, Multiple Radio Buttons

On a typical day, how many minutes is your round trip commute to and from Max?

Minutes

I use temporary lodging near work.

My commute time varies.

I have no commute time.

- Multiple dropdown menus with single radio button

SPEC: If the respondent selects ANY dropdown menu and then decides to click on the radio button, then ALL the dropdown menus shall be cleared out. If the respondent selects ANY dropdown menu then the radio button shall be cleared out.

Screen Capture:

Figure 17. Multiple Dropdown Menus, Single Radio Button

In what month and year did you last attend college between January 1, 2013 and now?

If you do not know the month, please select the season.

Month or Season Year

Still in school

- Multiple dropdown menus with multiple radio buttons

SPEC: If the respondent selects ANY dropdown menu and then decides to click on the radio button, then ALL the dropdown menus shall be cleared out.

If the respondent selects ANY dropdown menu, then ANY selected radio button shall be cleared out.

Screen Capture:

Figure 18. Multiple Dropdown Menus, Multiple Radio Buttons

In what month and year did you get married?

If you do not know the month, please select the season.

Select Month or Season Select Year

2013–2015 but don't know which year

Before 2013 but don't know exact year

- Multiple dropdown menus with multiple radio buttons and with textbox

SPEC: If the respondent selects ANY dropdown menu or textbox and then decides to click on the radio button, then ALL the dropdown menus and textboxes shall be cleared out.

If the respondent selects ANY dropdown menu or textbox, then ANY selected radio button shall be cleared out.

Screen Capture:

Figure 19. Multiple Dropdown Menus, Multiple Radio Buttons with Text Box

What is his full birthdate?

If you do not know the month, please select the season.

Select Month or Season Day Select Year

2013–2015 but don't know which year

Before 2013 but don't know exact year

- Disable Radio Buttons If Mask Textbox Is Zero or Empty

SPEC: Radio Button: If the respondent attempts to click on the radio button when the masked textbox is equal to zero or empty, the radio button shall automatically be unchecked and disabled.

Other Specify: If the respondent attempts to click on the 'Other Specify' textbox, the radio button shall be cleared out if mask textbox is zero or empty.

If the respondent attempts to type inside the 'Other Specify', the text field shall be cleared out and grayed out.

Number Text Box (Numeric Mask Text Box): Once the respondent clicks on the box or starts typing or makes a change, the radio buttons and the 'Other Specify' textbox shall be disabled and grayed out if the value is empty or outside of the bounds for the range set for that field. If the value is inside the range set for that field, then the radio buttons and 'Other Specify' textbox shall be enabled and the box shall change from gray to white.

Screen Capture: On page load:

Figure 20. Empty Mask Text Box, Disabled Radio Buttons

The screenshot shows a form titled "About how much rent do you pay?". On the left, there is a mouse cursor icon. To its right is a masked text box containing "\$" followed by a white box, ".00", and "per". To the right of the text box is a group of radio buttons with the following options: "Week", "Two weeks", "Month", "Year", and "Other - Please specify:". The "Other" option is followed by a greyed-out text box. A green rectangular box highlights the radio button group and the "Other" text box.

After the value (inside correct range) is entered into mask textbox:

Figure 21. Mask Text Box with Entered Value and Radio Buttons

The screenshot shows the same form as Figure 20, but now the masked text box contains the value "12,345". The radio buttons are now enabled. The "Other" option is followed by a white text box. A green rectangular box highlights the radio button group and the "Other" text box.

- Masks

Masks can be used in multiple ways:

1. Inserts appropriate group separators every 3 digits on a field (e.g. , a comma)
2. Ability to display (in the entry box) the prefix to a number, such as the currency symbol
3. Limits input between the maximum, minimum, and number of digits to the right of the decimal, according to the field definition
4. Empty values are allowed
5. "0" may/may not be allowed, depending upon the range of the field
6. Decimal places are not allowed if not specified in the field
7. The resulting custom control's width is automatically adjusted to the expected width of the entry field.
8. Values less than the minimum or more than the maximum are not allowed
9. Single key presses that make an invalid entry are deleted.
10. The mask action will truncate input that exceeds the width of entry (e.g., hold a key to allow it to repeat, and then release it: "9999999" may truncate to "999")
11. Selecting all the input and overwriting it with invalid data will cause the old input to reappear. For example, "230" is selected and overwritten with "999999999" (a value larger than the field can hold) the result will be 230.

Screen Capture:

Figure 22. Masks

About how much is the remaining principal on this loan?

\$ 6,626,266 .00

About how much rent do you pay?

\$ 12,345 .00 per Week
 Two weeks
 Month
 Year
 Other - Please specify:

6. Conclusion

In this paper, we only described a few functionalities. There were many other functions developed by using Blaise 5's new and updated features including APIs. We had a steep learning curve in developing some of the functions. However, once they were properly designed and implemented they have become an integral part of our project design workflow. These provide numerous benefits in terms of more accurate data collection, and likely less respondent attrition due to preventing frustration, and providing a more cohesive respondent experience. In the future we intend to expand our library and continue to grow as project needs warrant additional functionality.

In conclusion, we have gained significant knowledge in Blaise 5 in the past two years. We believe that Blaise 5 is a full-fledged survey software system that helped us to build robust surveys.

7. References

1. Blaise 5 Help Manual
2. Stackoverflow.com/ and other Internet resources

Running the DEP as Standalone

Roberto Picha and Mecene Desormice, U.S. Census Bureau

1. Abstract

For the last two decades, the U.S. Census Bureau has used Blaise 4 software to collect CATI and CAPI survey data in a “Standalone” environment, where the Blaise instruments interface with the Laptop and CATI case management systems. Our “standalone” (self-contained) environment involves the case management systems calling a Manipula transaction script, which in turn makes a call to the DEP via the Edit function. This script binds the metafile and local database containing a record or a single case. This process is highly integrated into the current survey systems that are currently being used at the Census Bureau. It is not an easy task to make drastic changes to this process.

With new software and hardware technologies emerging in the data collection world, a new paradigm is presented which involves more questions than answers. What is the ideal Blaise 5 environment for interviewing with different operating systems and on different devices? It is most likely a different environment than what is currently used for our CATI/CAPI data collection. Until we can figure out the best approach for this paradigm shift and while we are still running Blaise 4 production instruments, we decided to research what it will take to implement a Blaise 5 production survey in our existing environments (i.e., standalone). Can we continue to run our legacy control systems with a Blaise 5 instrument?

This paper covers our experiences in attempting to run a Blaise 5 instrument in our existing control systems environment (i.e., running the DEP as standalone). We began our research using Blaise 5.03 Build 810 and completed with Blaise 5.05 Build 975. We will share our lessons learned and experiences from this research, how the DEP can act as a service, and how to automate the process for installing of the minimum software required to deploy packages to this environment. We realize that this may not be the best approach for running Blaise 5 instruments, but it was important for us to determine what options we have for running Blaise 5 in production.

2. The use of Blaise for Data Collection

The U.S. Census Bureau CATI and CAPI data collection has been entirely built using Blaise 4 Enterprise; the software meets all the requirements to make it flexible and suitable with our systems. The Blaise 4 software is "distributed" to our Field Representative (FR) laptops and the phone center PCs. The software “distribution” primarily consists of the basic executables needed to bind the metafile and database running on the FR's laptop to collect data (Manipula.exe, Dataview.exe, and Hospital.exe).

It is important to mention that there are no special configurations or settings on the laptops in order to run Blaise 4. The data collection process is self-contained step and the instrument runs "as standalone" from within our case management system.

How Blaise 4 data collection is used by the Census Bureau most likely differs from how other agencies use Blaise 4 to collect data. Our current process was designed to be integrated with existing case management and control systems that could handle multiple data collection software applications. The Census Bureau is only using Blaise 4 to collect the data; it is not using Blaise to manage the assignments or case data. The “individual” case bdb concept was implemented so that cases could be easily re-assigned to other FR laptops or interviewer work stations. Tracking and re-assigning cases is handled by the control systems.

The goal of this research was to determine if it was possible to implement a production Blaise 5 survey with minimal changes to our control systems as it could take significant time, resources, and money to redesign the existing systems. It is essential that we keep it simple and manageable.

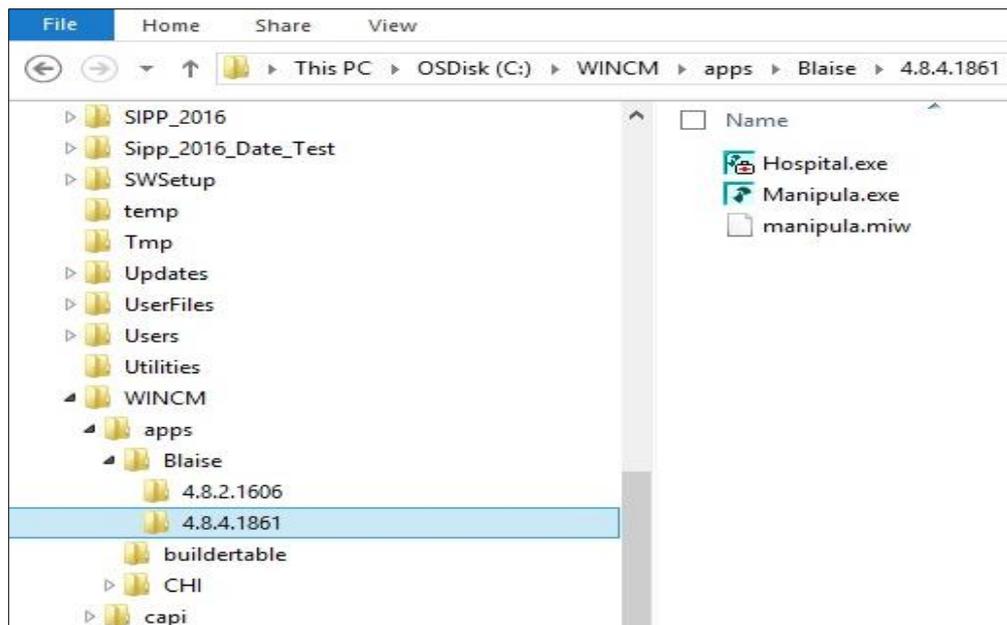
2.1 Current Process and Blaise software

Currently, the Census Bureau pushes down the minimum Blaise 4 software necessary to run our CAPI and CATI instruments. We are also required to run multiple version of the Blaise software as not all surveys are using the same version of Blaise 4.

The software distributed to the FR's laptops consists of the basic executables for data collection along with a few tools that are used for assisting the FR and/or Technical Assistant Center (TAC) for production related issues.

- a) **Manipula** – Case Management will use Manipula to launch the instrument scripts that control the actions taken. These scripts make updates to the Blaise data, execute the Blaise interview using DEP, and output updates to the control systems.
- b) **Data browser** – Used for analyzing the contents of the database when issues arise. Only the latest version of this executable is pushed to the laptops, therefore it is not parked in the same location as Manipula.
- c) **Hospital** – Used in the event that something goes wrong with the case or between the instrument and case management.

Figure 1. Example of the folder structure where Blaise 4 software resides



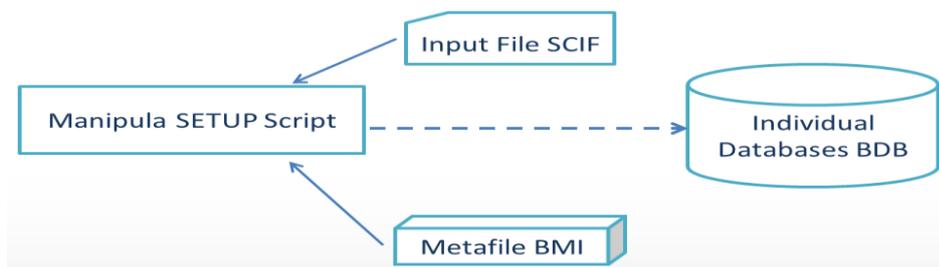
The Blaise 4 software is distributed to the laptops via AFARIA that uses a channel communication port. The executables are then parked to a fixed location on the laptop, the location is part of the OS path and therefore, no other configuration is needed. This action is only performed once for each version of Blaise approved for production.

Since our survey instruments are coded in different versions of Blaise, it is essential that the laptop environment is compartmentalized by the software version allowing us to support and run several versions of Blaise in the same OS.

2.1.2 Survey Instruments in the current process

The survey instrument is packaged as a compressed file and then delivered to our Master Control System (MCS). MCS runs a Manipula Setup script binding the instrument metafile and the SCIF input file (ASCII file) creating single Blaise databases for each case. These single databases are then placed inside a blob and provided to ROSCO, where individual case assignments are made to the FRs. MCS also delivers the instrument compressed file (package) to a server from where software testing performs testing and then copies the survey instrument to the production server. From there, the instrument is transmitted to the FRs once they connect to the server. There is a communication channel process that will push down the instrument to FR's laptops and unzip the contents in the appropriate survey interview period folder. The instrument package contains the metafiles, i.e., instrument and any external files used by the main metafile.

Figure 2. MCS runs the setup script to create individual case BDB files



Our Case Management System (LCM) and Instrument talk to each other via ASCII files to share & update information when the FR opens the case.

LCM makes the call to our Manipula transaction script as follows:

```
C:\Wincm\apps\Blaise\4.8.4.1681\Manipula.exe C:\Wincm\database\studies\<surevid>\e-inst\capi_trans.msu
```

Then the capi_trans internally makes the call to the DEP as in figure 3:

Figure 3. Manipula calls the DEP as EDIT function

```
Result2:= EDIT (THEINSTRUMENT + ' /M..\config\dews.bmf' +  
                ' /H..\config;..\externals /E..\config;..\externals' +  
                ' /F' + thecase +  
                ' /K' + thecase +  
                ' /G /X')
```

Overall, the interaction between the instrument and LCM must be flawless. The interaction not only calls the instrument, it makes updates to the case data via another Manipula script that is called from within our capi_trans.msu. Upon completing the case, updates (and possibly spawned cases) are made available to LCM to update, spawn and writing back information to LCM at times, other process are expected. The flow is described in figure (4).

Since Blaise 5 is not a simple copy of software process as in its predecessor Blaise 4, some initial steps must be conducted on the FR laptop prior to launching a survey. The Blaise services and server manager must be available one way or another. This imposes a challenge in identifying all dependencies. More importantly, would this new process allow laptops to run different version of the software without issues?

3.1 What do we required?

a) Blaise server

Blaise 5 services are required to launch an instrument, whether this survey is launched locally (laptop) or over the network or internet. The server will handle all the transactions and databases for various surveys.

b) Server Manager

The server manager is very useful and required to accomplish important tasks. With our early research of Blaise 5, we used the server manager to install/uninstall surveys. We learned that in order to use this tool, we pretty much had to install the entire enterprise to the laptops. This was not the most ideal situation, yet it was doable at the small scale test. We also found some early issues with the server manager that were later corrected by Statistics Netherlands.

While the server manager proves to be of use, we found that it has extremely large dependencies that needed to be pushed to the FR's laptops, which is impractical for our purpose.

3.2 What do we needed?

During the IBUC conference in China (2014), Statistics Netherlands indicated that the DEP could literally run as service, that is, the DEP contains an embedded service component. This service allows us to remove the need of the server manager and any extra dependencies.

The following syntax was all that was needed to launch a survey: **DEP.exe inst.bpkg**

At glance, this seemed simple and straight forward. However, we learned that there are a lot of things going on internally with the DEP and those findings are discussed below.

3.3 DEP to maintain our current process

Since the DEP can perform as a service, we decided to explore the mechanisms that trigger the events plus identify our needs and apply them accordingly. In the process, we learned that the DEP can do what is needed with minimum requirements.

Step 1: DEP Sets up the Environment

From using the syntax shared above, the DEP looks at the environment in which it is going to be run. It uses the information indicated by the **dep.exe.config** file to:

- Identify the location of the surveys
- Identify the location of the database configuration
- Identify the location where the package should reside

If the DEP does not find the environment properly configured, it will create one on the fly with default settings. Figure 5 shows an example of the DEP configuration file.

Figure 5. Example of DEP Configuration File

```
k?xml version="1.0"?>
<configuration>
  <appSettings>
    <add key="ExternalCommunicationAddress" value="http://localhost:8033"/>
    <add key="DeployFolder" value="C:\temp\Standalone\"/>
    <add key="ConfigurationDatabase" value="RuntimeConfiguration.db"/>
    <add key="ManagementCommunicationPort" value="8031"/>
    <add key="InternalCommunicationPort" value="8032"/>
    <add key="ExternalCommunicationPort" value="8033"/>
    <add key="ServerManagerDatabase" value="ServerManagerDatabase.db"/>
    <!--<add key="CertificateStoreName" value="My" />
  <add key="CertificateStoreLocation" value="LocalMachine" />
  <add key="CertificateThumbprint" value="7F94601C4FC2EE71C8AAC02947607F2E2CEEE4D2" />-->
  </appSettings>
  <!--<startup><supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.1"/></startup>-->
</configuration>
```

We can also create our own configuration file, specifying the appropriate parameters as seen above.

Step 2: DEP Installs the Instrument

Once the environment is configured, the DEP will un-package and install the instrument the first time the syntax is invoked. The DEP runs the following steps:

- Places a copy of the package under the “Upload” folder
- Unzips the package and places it under the “Survey” folder
- Reads the main metafile and extracts The following two items about the instrument:
 - **Instrument ID** - unique identifier. This items can also be obtained from the Control Center
 - **Checksum**. This is important for data integrity. After compilation, in the output window log, the checksum is available and apparently this contained in the main metafile

Step 3: DEP populates Databases and Tables

The DEP then proceeds to populate (at minimum) two databases and some tables:

- The configuration database and table, see figure 7.
- The server manager database and tables, see figures 9, 10 & 11

Once these three steps are completed, the DEP renders the instrument - binding the Metafile with the database - just as it is done currently with Blaise 4

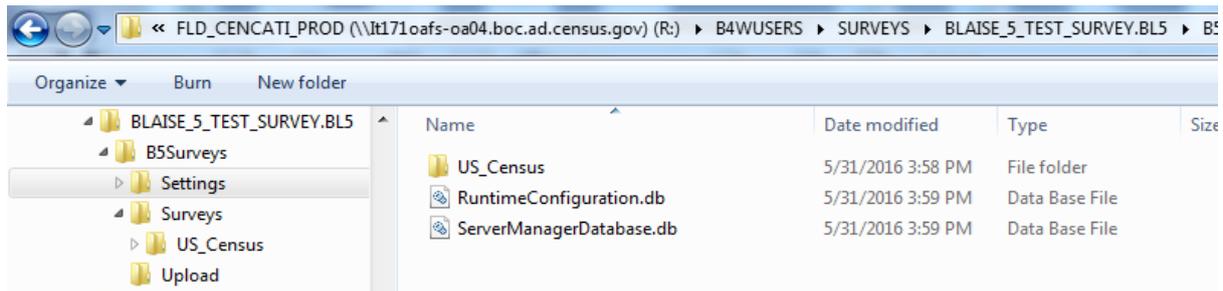
Note: If the instrument is called for subsequent interviews of the same case or any other cases for that survey instrument, the DEP acknowledges the environment and just proceeds to read the tables from the databases.

3.4 Database Configuration and Management

While the convenience of the DEP doing all steps automatically was great for us, we found that some management and configuration could enhance our capabilities for a quick turnaround. We decided to automate the process and manage it properly for our purposes. Note that we currently do not use the DEP to launch our instruments; rather we use Manipula that in turn calls the DEP as an EDIT function.

Figure 6 shows an example of the folder structure and the survey folder from where the instrument is launched as a standalone environment over the network.

Figure 6 – Example of Blaise 5 Folder with 2 basic databases to run standalone



3.4.1 Runtime Configuration Database

This database has only one table - the Configuration table. This table stores data such as Instrument ID, Instrument Name, Survey Root (locations used to find the metafile and database), SeverPark Name, and Content. Figure 7 shows an example of the information stored in this table.

Figure 7. Example of Configuration Database Information

InstrumentId	InstrumentName	SurveyRoot	StartPageName	ServerParkName	Content
049a6b38-56ff-4484-a86b-5776e6e065b6	US_Census	US_Census	US_Census	StandAlone	<InstrumentConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema..."

The “Content” is a string of XML text that contains the configuration of the instrument. Figure 8 shows an example of the data listed in the content column of the Configuration Table.

Figure 8. Example of Data contained in the “Content Column” of the Configuration Table

```
<InstrumentConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Version>1</Version><InstrumentName>inst</InstrumentName><InstrumentId>e0701098-4772-4d91-b2dc-281be9c22130</InstrumentId><OldMetaFileName /><MetaFileName>C:\wincm\B5Surveys\Surveys\inst\inst.bmix</MetaFileName><ResourceFileName>C:\wincm\B5Surveys\Surveys\inst\inst.blrd</ResourceFileName><DataFileName>C:\wincm\B5Surveys\Surveys\inst\inst.bdx</DataFileName><InitialLayoutSetGroupName>CASI</InitialLayoutSetGroupName><InitialLayoutSetName /><InitialDataEntrySettingsName>StrictInterviewing</InitialDataEntrySettingsName><SurveyRoot>inst</SurveyRoot><StartPageFileName>C:\wincm\B5Surveys\Surveys\inst\inst.aspx</StartPageFileName><ServerParkName>StandAlone</ServerParkName><Status>Active</Status><InstallDate>2016-08-10T14:01:54.000202</InstallDate><DataChecksum>4278984200.3387305133.650760389</DataChecksum><CatRole /><CatSpecificationFileName /></InstrumentConfiguration>
```

3.4.2 Server Manager Database

The Server Manager Database contains seven tables that store critical information that is required to run the Survey. The tables store data related to:

- Deployment location
- Server name
- ServerPark name
- Port Number
- Master address
- Run Mode
- IP address
- Server Roles

When deploying a package through the Server Manager tool, the tool automatically fills the database tables with the required data. However, due to the challenge we encountered in running the Blaise 5 package outside the Control Center and the Server Manager application, we have deployed a utility to

perform the task of server Manager and alleviate the burden of moving several files to several devices. The utility or tool helps us to save time and resources during our test. Some of the important tables are discussed below.

- **LogicalRoot Table** - The critical “LogicalRoot” table contains the **location path** as well as the **instrument ID** that is used to map the instrument and location (See Figure 9).

Figure 9. Example of the Logical Root Table

Table: LogicalRoot				
Id	Name	WebsiteId	Location	
Filter	Filter	Filter	Filter	
1	1	default	1	R:\B4WUSERS\SURVEYS\BLAISE_5_TEST_SURVEY.BL5\B5Surveys\Surveys

- **Park Table** - The Park table is used to set some individual settings for the instrument. The **ID** is used to map to the right instrument and server name (See figure 10).

Figure 10. Example of the Park Table

Table: Park										
Id	Name	Masteraddress	Loadbalancer	RunMode	IsPublic	SessionMode	AuditTrailMode	DeleteDataAfterUpload	SyncDataWhenConnected	rcSurveysWhenConnect
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	StandAlone	localhost:8031	http://localhost:8033	3	1	1	1	0	0

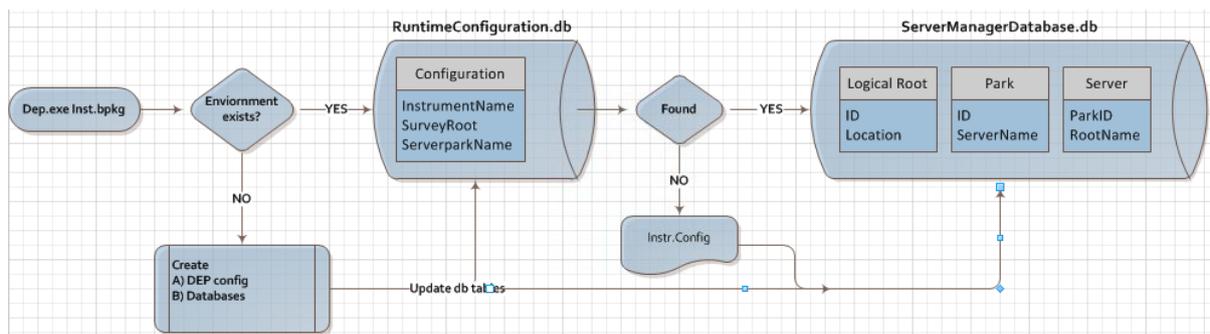
- **Server Table** - The Server table maps the instrument via the **ParkID** with specific roles (See figure 11)

Figure 11. Example of the Server Table

Table: Server										
Id	ParkId	Name	PortNumber	IPaddressV4	IPaddressV6	Roles	RootName	ExternalName	Binding	
Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	
1	1	1	localhost	8033	127.0.0.1	fe80::39c8:5796:9393:8dea%11	7,4,5	default		http
2	2	1	localhost	8032	127.0.0.1	fe80::39c8:5796:9393:8dea%11	3,6	default		http
3	3	1	localhost	8031	127.0.0.1	fe80::39c8:5796:9393:8dea%11	1	default		http

The flow of this process as we see it happening is depicted in Figure 12. We may be off in the links across tables; however, this is pretty close to what we have observed when calling the DEP as standalone.

Figure 12. Example of the DEP flow to configure the environment as standalone



4. Putting all together

At the time of the writing of this paper, Manipula in Blaise 5 is still unable to call the DEP as a function. So, was decided that we should work with a hybrid approach of Manipula and DEP.

- a) DEP is used to install the instrument (initial stage)
- b) Manipula is used to synchronize data between our case management and the Blaise 5 case (as we currently do in Blaise 4)
- c) DEP is used to launch the case once the update is complete

As a testing platform, this process worked. We used one single database with all the records (cases) loaded. The only inconvenience was to launch the DEP as initial stage in order to install the instrument. Subsequent calls were made properly by passing parameters such as the primary Key.

Introducing an external tool was necessary; the - “QLite_DbCreator” – its main mission is to create (QLite) databases needed by the DEP to run an instrument. The tool is responsible for creating the tables and filling them with data. This tools only dependency is the “System.Data.SQLite” library, which is the same library used by the DEP as well as the server manager.

4.1 Server Manager Tool

This section focuses on deploying a Blaise package as Standalone on a device with or without Blaise 5 installed. By default, the Server Manager is the primary tool used to deploy and administer Blaise 5 surveys. The Server Manager performs three major functions:

- Configure and manage the Server Park Servers
- Deploy surveys to Server Park
- Create and manage access to surveys

Like all the Blaise 5 components, the Server Manager is part of the Blaise installation. It is accessible through the Blaise 5 Control Center and can be accessed independently by opening the “ServerManager.exe” application from the Blaise 5 Bin folder.

To run Server Manager outside the Control Center or Blaise 5 Bin folder, all the reference libraries must be copied to the new location. Because the Server Manager depends on a range of libraries, that creates a challenge when copying or moving them into a new device or location. The challenge was to identify and copy the specific libraries to run the application. Identifying the reference library is time consuming. Copying all the libraries invoke resource and permission issues. The amount of files to copy range between 300 to 400 MB in size.

Not being able to identify the minimum dependencies to run the SeverManager, we decided to develop our own “ServerManager tool” to overcome this challenge.

4.2 QLite_DBCreator Tool

QLite_DBCreator.exe is a small C#.Net application that has the task of creating the required Blaise 5 SQLite databases and creating, updating and storing data in the database tables. Unlike the SeverManager tool, this application references only SQLite.dll as a dependent library and receives a list of parameters as input to produce the databases. The list of parameters includes a number following the task you want to achieve.

- To create a database,
 - a [1] <space> [database name] is entered as parameter.
- To create a table in selected database,
 - a [2] <space> [database name] <space> [Table name] is entered.

The application has the capability to create multiple db databases or tables simultaneously. QLite_DBCreator can be used manually and at the DOS prompt, to perform these tasks but this method was cumbersome. The ideal approach was to automate the process by including the application in the standalone installation package, and then call it through the VBScript.

Beside configuring and setting the environment, the VBScript would open a file called “instr.config” and extract information such as InstrumentName, InstrumentId, CheckSum, ServerParkName, and ParkId. The information is used to generate external files and update in the db database tables.

The VBScript calls the C# application and provides the required parameter using some internal procedure calls. The C# application proceeds in creating the databases and the tables, reads the external files, and loads data to the designated database tables.

Figure 13. QLite_DBCreator Tool code Example

```

createBlaiseDB db = new createBlaiseDB();
createTable tb = new createTable();
if (st[0] == "1" )
{
    /*Create one or more than database:
    * enter [1] <space>[database name]<space>[database name] <space>[...
    */

    for (int i = 2; i < st.Length; i++)
    {
        databaseName = getDatabaseName(st[i].ToLower());
        if (databaseName != "")
        {
            databaseName = strPath + @"\"+ databaseName + ".db";
            db.CreateDb(databaseName);
        }
    }
}

//CreateDb gets one parameter<database name and fullpath>
//Checks if path and db exist. If path and db don't exist, it will it
public void CreateDb(string db)
{
    string directory = Path.GetDirectoryName(db);
    try
    {
        if (!File.Exists(db))
        {
            // Creates directory if it doesn't already exist:
            Directory.CreateDirectory(directory);

            // Creates file if it doesn't already exist:
            SQLiteConnection.CreateFile(db);
        }
        else
        {
            Console.WriteLine(db + " already exists!");
        }
    }
    catch (FileNotFoundException e)
    {
        if (e.Source != null)
            Console.WriteLine("IOException source: {0}", e.Source);
    }
}

```

4.3 The New Process

What we learned from researching the Server Manager and the DEP, enhanced our view and allowed us to refine our installation instrument package and software distribution via a vbs installation script. The installation package we use for the deployment of Blaise 5 instruments contains the following:

- **Blaise 5 Software** (Containing the distribution software in a zip file):
 - Manipula.exe
 - Dep.exe
 - Dep.Resources.dll
 - System.Data.SQLite.dll
 - Two empty Blaise 5 databases:
 - RuntimeConfiguration.db
 - ServerManagerDatabase.db
 - Qlite_DbCreator.exe – C# software use for creating/managing databases.
- **InstallBlaise5StandAlone.vbs** - VBScript that prepares the environment; configuring, extracting software from zip, and installing Blaise 5 survey instruments.
- **Inst.Config** - Configuration file for the survey that is being installed.
- **Inst.bpkg** - Survey package (instrument) .

Figure 14. Example of the Blaise 5 Instrument Installation Package files

Name	Date modified	Type	Size
 B_5.0.5.950_StandAlone.zip	5/23/2016 3:04 PM	WinZip File	29,034 KB
 inst.bpkg	1/13/2016 12:04 PM	BPKG File	36,863 KB
 InstallBlaise5StandAlone.vbs	6/2/2016 11:25 AM	VBScript Script File	26 KB
 Instr.config	8/8/2015 10:16 AM	XML Configuratio...	61 KB

4.3.1 The Standalone Blaise 5 Installation VBScript

The Standalone Blaise 5 installation VBScript is the core element to setup the environment and Blaise instrument package. Its main responsibility is to create and configure the survey deployment location to install the package.

Listed below is the deployment folder structure created by the installation package. The VBScript unzips the file that contains the “Settings”, “Surveys”, and “Upload” folders and then copies all the required files (to run the survey instrument as standalone) and places them at the root level of the deployment folder

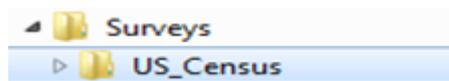
Figure 15. Example of Blaise 5 Standalone Folder Structure

Name	Date modified	Type	Size
Settings	5/31/2016 3:59 PM	File folder	
Surveys	5/31/2016 3:58 PM	File folder	
Upload	5/31/2016 3:58 PM	File folder	
Dep.exe	3/31/2016 5:00 AM	Application	19,968 KB
Dep.exe.config	5/31/2016 3:58 PM	XML Configuratio...	1 KB
Dep.Resources.dll	3/31/2016 5:00 AM	Application extens...	8,357 KB
Manipula.exe	3/31/2016 5:00 AM	Application	33,612 KB
QLite_DbCreator.exe	5/23/2016 3:03 PM	Application	19 KB
System.Data.SQLite.dll	11/29/2013 10:59 ...	Application extens...	1,209 KB

This folder structure contains:

- **Settings** Folder - contains a list of databases
- **Surveys** Folder - holds subfolders for individual surveys. The subfolders can follow our current naming convention (i.e., use survey ID) to align with our current process (Fig 16).

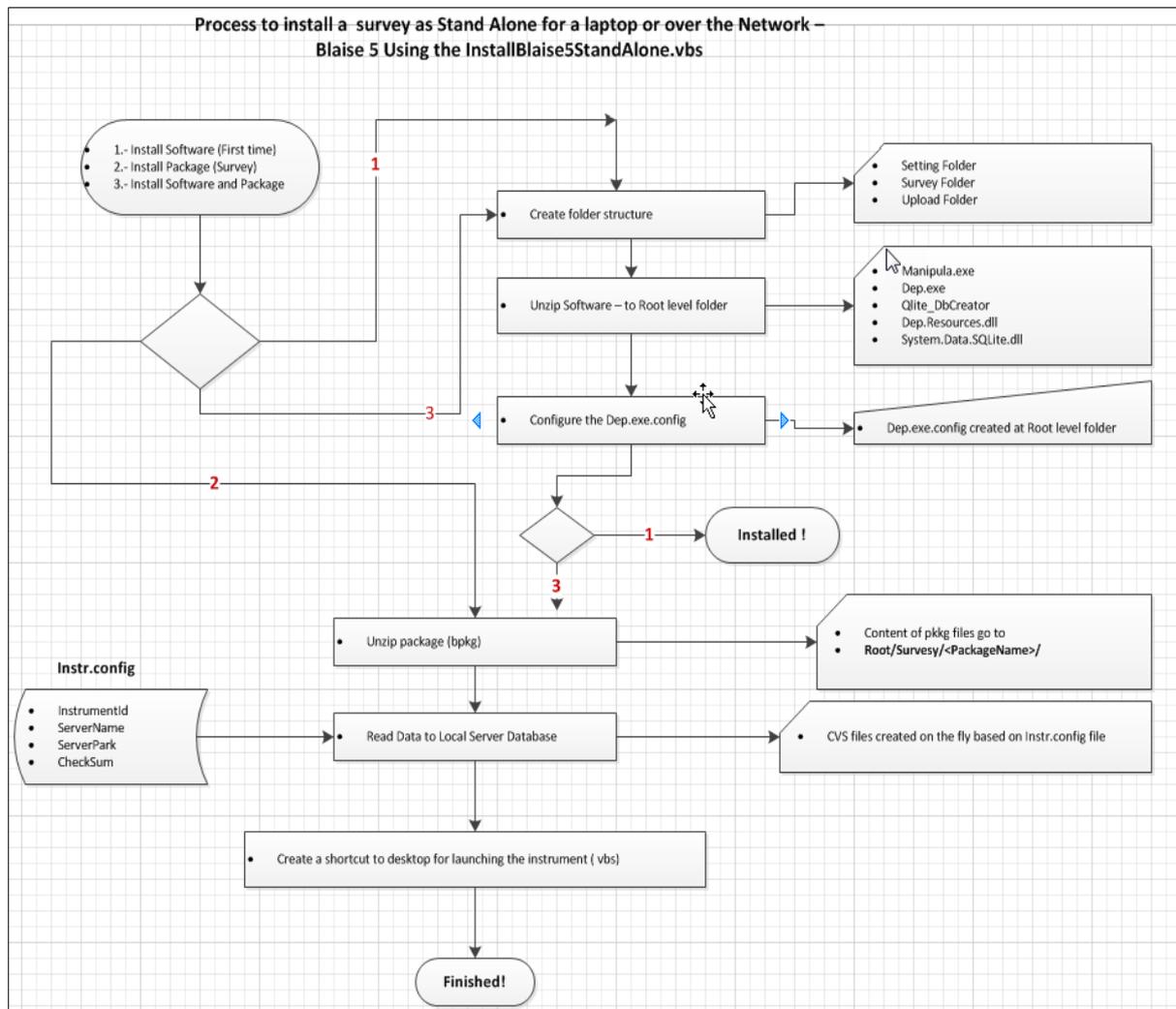
Figure 16. Example of the US_Census survey under the folder Surveys



- **Upload** Folder - stores a copy the package (.bpkg file) This is the folder where the packages would be placed.

The installation creates the “**inst.bpkg.vbs**” script. This script runs the survey as standalone when it is invoked. See Figure 17 for the flow of the VBScript instrument installation process.

Figure 17. Process describing how the VBScript installs and runs a standalone instrument



4.3.1 The Settings Folder

By default, Blaise 5 uses SQLite as the Database management System to store its database services. The SQLite Databases are part of the Blaise 5 installation and reside in the Settings folder.

There are a total of six databases in the Settings folder:

- RuntimeConfiguration
- ServerManagerDatabase
- RuntimeSessionData2
- Credentials
- ReportingCacheDatabase
- AuditTrailData

In our test, we use only two of six databases:

- RuntimeConfiguration
- ServerManagerDatabase

The two databases are critical to run the Blaise survey package. They store information related to the survey configuration, InstrumentID, InstrumentName, deployment location, Machine or Server name,

IP address, serverPark, and so on. They contain all that is necessary for the DEP to launch a Blaise 5 survey effectively.

5. Summary

We were able to successfully install and run a Blaise 5 instrument as a standalone application that could possibly run under our current interviewing environment. If for some reason we must move a survey to Blaise 5 in our existing interviewing environment, we feel that this is something we may be able to handle. This process would also allow us to continue to support multiple versions of the software on the FR's laptop.

However, there are still a few remaining challenges that we hope to resolve with future Blaise 5 releases. We are currently unable to call a Blaise 5 instrument running an individual case database, as we do for Blaise 4. Also, we are not able to run a Blaise 5 instrument from Manipula.

Recommendations for future enhancements:

- a) We would like the ability to extract the instrument ID and Checksum from the metafile by Manipula. This is currently done manually via the control center
 - The instrument ID is part of the project configuration setting
 - The Checksum is calculated when compiling/building the instrument
- b) We think it would be beneficial to include any msux file (Manipula script) as part of the Blaise 5 package.
- c) We would like to be able to bind the main metafile (bmix) with the database (bdbx) that is not the defaulting name. For instance, the F toggle in Blaise 5 means input file path indicating the input folder location. Whereas in Blaise 4, it indicates the name of the Blaise database. Currently there is no corresponding toggle to indicate a database (bdbx) name in the command line.
- d) Perhaps reducing the number of dependencies of the ServerManager can help. The ServerManager is a great tool that can be run at the command line prompt, allowing us to automate our process.

6. References

Peter Segel, Mangal Subramarian, Richard Frey, Ray Snowden (Westat) "Blaise 5 Server configuration for Web Surveys", Proceedings of the 16th International Blaise User Conference, April 2015.

Blaise 5 Help (version 5.0.5.950). (2016).

7. Acknowledgements

The author would like to acknowledge the work and knowledge of Mecen Desormice for his assistance with research project.

The views expressed in this paper are those of the authors and not necessarily those of the U.S. Census Bureau.

Experience with Blaise 5 Layouts and Templates

Sharon Johnson, Richard Squires, and Michael Johnson, U.S. Census Bureau

1. Abstract

Moving a current Blaise 4 survey instrument to Blaise 5 involves much more than simply converting existing code to run in Blaise 5. With the emergence of touch laptops, tablets, and phones as viable interviewing devices, one must consider these different form factors and how they will impact CAPI survey data collection. The move to Blaise 5, with all of the new features, appears to be the appropriate time to redesign our CAPI data collection instruments so that they take advantage of, and work well with, touch screen devices.

The Blaise Authoring team has been researching, learning, and testing many of the new Blaise 5 layout and template features in order to determine a reasonable approach for moving forward into Blaise 5. This paper will discuss some of the findings, issues, and challenges we have discovered in this effort. The paper will also review the latest settings we have defined for our Master Template, some of the custom templates that we have identified for our surveys, and how we implemented some of the Blaise 4 features into Blaise 5.

2. Overview

The Blaise Authoring team has been researching, learning, and testing many of the new Blaise 5 layout and template features at the Census Bureau in order to determine a reasonable approach for moving forward with Blaise 5. This paper will discuss some of the findings, issues, and challenges we discovered along the way. The paper will also review the latest settings we have defined for our Master Template, some of the custom templates that we have identified for our surveys, and how we implemented some of the Blaise 4 features into Blaise 5.

During the research effort, the Blaise Authoring team took the opportunity to develop the 2010 Census short form from scratch using the Blaise5 software. This allotted the research team to familiarize themselves with the additional enhancements and bug fixes that were provided with the more recent releases of Blaise 5. We discovered that using the Blaise Resource Editor to modify the layout presented a lot of flexibility when designing the look and feel of an instrument. This document identifies some of the capabilities that we discovered in the implementation process, and documents the experience based on this implementation.

2.1 Development Approach

Initially, we considered using the current Blaise 4 layout presentation which incorporates the Infopane and the Formpane as the Census Bureau standard. After further consideration and research of industry standards, we decided to use the Blaise 5 design concept. The design approach focused on the use of a touch screen device and the use of a laptop keyboard. With this approach in mind, we considered the use of buttons, dropdown lists, and hyperlinks for a successful execution of the questionnaire.

Once an initial prototype was developed, a user testing session was held with some developers to collect feedback. Changes were implemented, and then the process was iterated. At the conclusion of three of these sessions, the process was repeated for three iterations with the sponsors/end users.

Our comparison is with two different methods of collecting the same information. We developed both, a person-based and a topic-based solution. Each version incorporates different types of features made available by Blaise 5.

2.2 Design Approaches Implemented

The following suggestions were presented and implemented into the prototype.

- Allow 2-3 questions per screen.
- Highlight the active field when in a table.
- Auto advance on questions with radio buttons to speed up flow of navigation when either the radio button or the text is selected.
- Display a status bar with minimum information (must include the respondent's name). Additional case information is made available through the "Info tab".
- Provide the ability to use the arrow keys to move between questions if using a keyboard.
- Implement Signals involving multiple questions on different pages.

3. Blaise 5 Software Features Researched and Incorporated

3.1 Toggle Visibility

This feature is useful in the display of certain graphics depending on if it is to be used by the person administering the questionnaire. For example, the Help icon is displayed only if the field has help available. Another example for use of this feature is if a question allows for Don't Know or Refusal as an answer option.

3.2 Tooltip

This feature provides the capability of hovering over a field to display additional help instructions. Since this feature is not available on mobile operating systems (no cursor), we have not determined if this will be a common feature offered for all Census surveys.

Figure 1. Example of Toggle Visibility and Tooltip

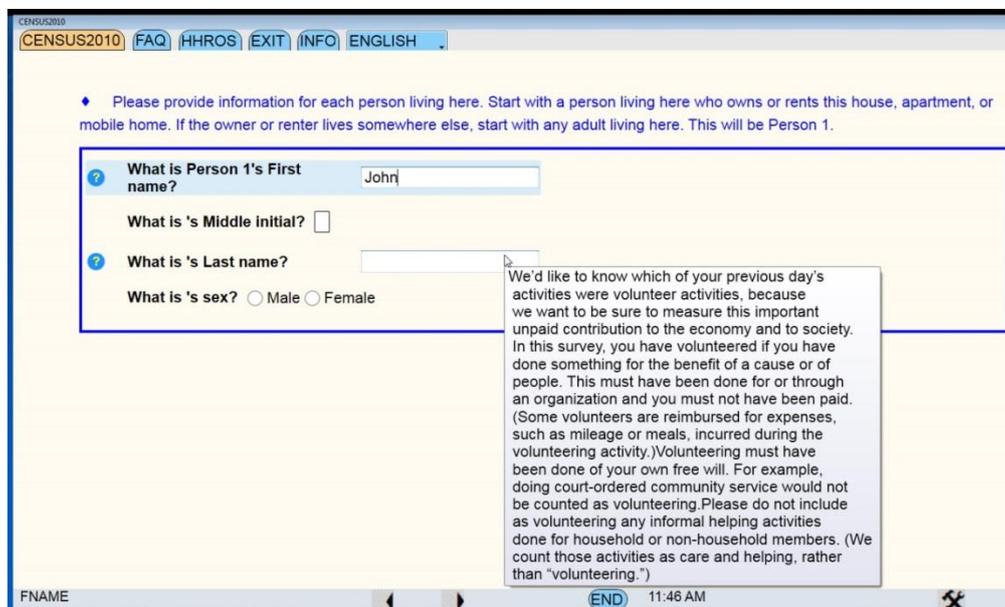


Figure 1 - This screen demonstrates both the Toggle visibility with the use of the Question Mark to represent help, and the Tooltip that displays a large amount of text that can be used for the help feature.

To incorporate the Tooltip functionality, first add Tooltip to ROLES. Then add tooltip to the field that will use the feature. See sample source code below.

Listing 1, Sample Tooltip Source Code

```

ROLES          = HELP "example of help",
                Tooltip "The ToolTip role provides a popup hint when the
                        user hovers over a control."

PHONE ENG "<newline><B>What is your telephone number? We may call if
           we don't understand an answer."
           ESP "<newline><B>TELÉFONO DEL HOGAR." /"Phone number:"
           Tooltip "Enter 10 digit phone number (no dashes)": TPHONE
    
```

3.3 Templates

Several templates are customized to design the layout that is currently being used. These templates have been implemented and tested using Blaise 5.04 b875.

In summary, we customized 21 templates in order to accommodate the design specifications received from the Sponsor and Developer inputs that include:

Table 1. Templates

Master Templates:

Template Name	Description
CensusStandardGrid	To accommodate Grid Style Structure
CensusStandard	To accommodate 2-3 questions per screen

Custom Templates:

Template Name	Description
InfoStatusPage1	Called when the INFO tab is selected
EndPage (EXIT)	Allows option to exit or return to survey
TOOLS	Accommodates tools used during the interview
RosterPage	Presents household roster information

Intro Page Template:

Template Name	Description
CensusDefault	Welcome screen with logo (picture)

Receipt Page Template:

Template Name	Description
CensusPrint1	Presented after the interview is completed. Thank you screen also provides option to PRINT. Show Household members interviewed, case status and Length of Interview.

Field Pane Templates:

Template Name	Description
Vertical1, Quantity1, Horizontal1, QuestionTextOnly	These were modified, so that the error message does not show directly under the field (since we are now showing standard error message location at the top.)

Data Value Templates:

Template Name	Description
DropDownList_age	Setup specifically for age drop down, made smaller and shows watermark "Enter age 0-120".
AnswerList	Answer list Area
DropDownList	Drop Down List Area

Category Templates:

Template Name	Description
Set	Set Category Code and Text
Enumeration	Enumeration Category Code and Text
RadioButton	Category Radio button
Enumeration FAQ	AutoEnter set to True for easy Navigation

Language Templates:

Template Name	Description
Default1	Specialized look for Language Button

Parallel Templates:

Template Name	Description
Default1	Specialized look for Parallel Tabs

3.4 Questions

We use the “highlight active question” feature while navigating through the questionnaire. In addition, for questions with a radio answer list, we make the answers selectable and change the color when advancing to better identify the answer that has been selected.

3.5 Edits/Errors

For displaying edit/error message information, we did not want the text to expand the screen or to incorporate scrolling to display errors. Based on these requirements, the following were considered: The use of a pop-up box similar to Blaise 4. However, this functionality is not available due to its incompatibility across multiple platforms.

- Creating our own pop-up. However, the software does not support creating an “overlay” screen that can sit on top of the Master Template.
- Using a box, reserving the right side of the screen and have it display the active error. However, this impacts the question/response texts on the screen and pushes them to the side
- Removing excess lines at the top of the screen, and reserving one line in that area for error message.

We decided to remove excess lines at the top of the screen, and reserve one line in that area for error messages. This works okay and allows for expanding as necessary.

3.5.1 Date of Birth (DOB) Question

Multiple approaches were investigated for the date of birth question.

1. An attempt to start the year range at a value within reason.

The current Date dropdown provided in Blaise 5 starts with the current year – 120 years. So for 2016, the list starts at 1896. Feedback provided to the committee requested that this date be “seeded”. In other words, start at a different place (e.g. 1950), based on the population being interviewed to prevent a lot of scrolling. This is currently not supported in Blaise 5, so alternative approaches were considered.

2. Before the DOB question, an age dropdown was provided that allowed for up through 120 years of age.

- Based on the age value provided, an approximate start year was determined and used for picking from a drop-down list.

- Month and day fields were pre-filled with January 1 and a signal applied if the default value was not changed. We ultimately decided not to prefill the date field.

Figure 2. Example of DOB Question, Error Icon, and Error Screen

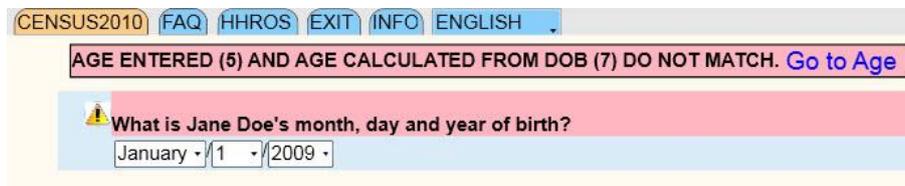


Figure 2 - Example of DOB question, error icon, and error screen. A link is provided in the error area that allows navigation to the Age field for correction.

- An error icon is next to the question to more easily identify what needs attention since more than one error may occur on a page.

3.6 Navigation

We tried several different options for the advancing between pages. As we consider touch screen, real estate, and moving current development to mobile devices in the future, it was decided to use the arrow icons (translatable pictures feature) instead of the previous and next buttons. The up and down arrows on the keyboard did not work initially. We were finally able to implement the up and down arrow navigation by using shortcut events CtrlDown (down arrow) for NextField, and CtrlUp (up arrow) for Previous Field. The shortcut is set at the MasterPage and will work wherever the MasterPage is used.

To implement the shortcut key assignments, you have to:

- In the resource editor, select your master template(s), the very first row (Master Page Template), select the events tab.
- Click on the icon for edit shortcuts.
- Add OnCtrlUp (keystroke) and add OnCtrlDown(keystroke).
- For OnCtrlUp, assign the action Previousfield, for OnCtrlDown, assign the action Nextfield.

3.7 Swiping

We used the Blaise 5 default of SwipeLeft = NextPage and SwipeRight=PreviousPage. During our testing, we found that while using one particular tablet, the longer Swipe Left and Right options would actually move us 2 pages forward or backward. We only discovered this happening on this device, other devices this feature worked fine.

3.8 Parallel Tabs

The parallel tabs were originally developed as just rectangular boxes. We modified these so they displayed as round edged buttons with some spacing between, which allowed for easier selection. The final design kept only the rounded top edges with spaces in between. The language selector was placed in the same area, but with a different look and feel. (See Figure 2 above)

3.9 Help/Flashcards

Initial viewing focused on the help being presented similar to the way it is in Blaise 4, using an external file. The belief is that when help is executed it should be consistent.

The syntax used in Blaise 4 doesn't work in Blaise 5. We were using an internal hyperlink that required a local channel to the local service that is running on the laptop. There is a separation between the Blaise program running and the server service.

For Flashcards, we were able to successfully use an individual .pdf file for each of the screens that use a flashcard. However, this required us to generate a lot of .pdf files and didn't seem very efficient. Our initial thought was to use one .pdf file and pass a parameter to the appropriate destination in the file. We could not successfully implement this as Blaise 5 was not accepting parameters.

- Alternatively, we implemented the external call for Help using .chm. We are exploring the use of the Tooltip, question help, and html files to display for help since .chm will not function properly on mobile devices. Not all help text needs to call an external browser window; as a result, some fields incorporated the use of the ToolTip option to display help text including a large amount of information. (See Figure 1, above for an example of ToolTip help.)

If help is available for a question, the help icon is positioned to the left of the question text. Both Help and Flashcard display content will be accessible by touching a clickable link near the question.

3.10 External Lookup Tables

We added an external State lookup table to our solution. To accomplish this, we did the following:

Using the Blaise4to5 converter, convert the stateslist.bla to stateslist.blax.

Listing 2. External Lookup Table Source Code

```
STATESLIST.BLAX
DATAMODEL
  SECONDARY
    StatesName = STATEText
  FIELDS
    STATEABREV : STRING[2]   {pos 1 - 2 }
    STATEText  : STRING[40]  {pos 3 - 42 }
  ENDMODEL
```

Next, we convert the stateslist database from Blaise 4 to Blaise 5 using the data converter for use in the code. Add the StatesList to USES.

Listing 3. External Lookup Table Source Code

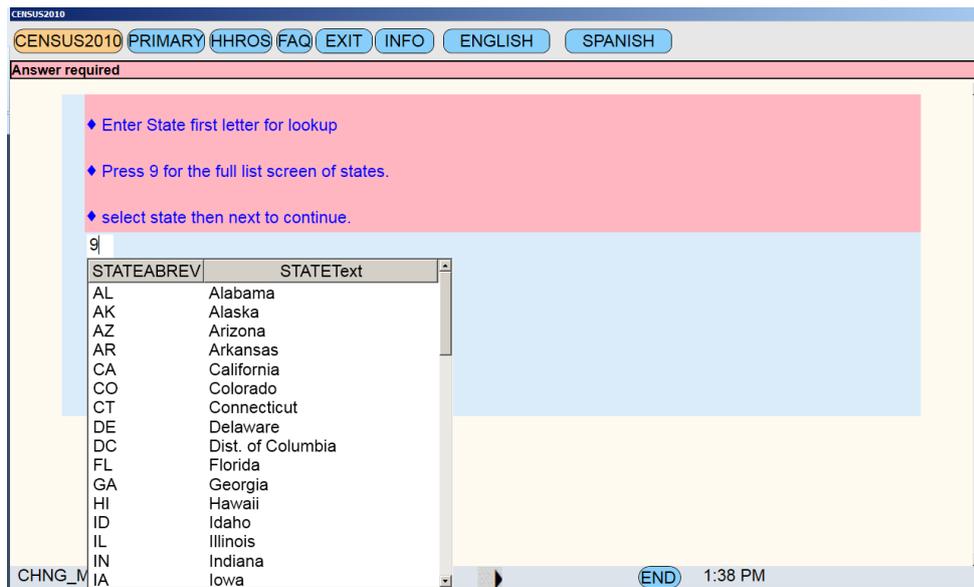
```
USES
  StatesList

{Lookup from statelist database file}

CHNG_MSTATE | STATESLIST.LOOKUP(StatesName).STATEABREV
IF CHNG_MSTATE = RESPONSE THEN
  MST:= CHNG_MSTATE
ENDIF
```

The above code calls the "stateslist" database when the field has an entry. The selection table pops up. The user can click on a selection which then closes the table. This puts the selected state abbreviation in the field.

Figure 3. Example of the State Lookup Table



Initially (5.0.4b875), it did not seem that the look-up was accessing the entries as we had intended based on the value we typed for searching. It was difficult to select with the touch screen due to size. To offer options, we made two fields. One field allowed the user to type the entire state abbreviation in to initiate the search.

The other field allowed for the look up. Typing a 9 on the first field goes to the second field. We also allowed for the entry of a '9' to bring up the entire State list. Alternatively, entering the 2-character state abbreviation displays the list of entries that start with that letter.

We found that using the arrow keys to go to the state and pressing enter caused problems. The selection table would start at the first state with the letters typed in. There was no option to go back in the list. It goes to the next item if there is not a match available for what you typed. For example: If you typed in "MA" for Maine, it would sometimes put in MA for Massachusetts. If you typed in "MD" for Maryland, the list would start at Michigan. The best way to get the item you desired was to type in the first letter and click with the mouse for the correct selection.

Selecting with the mouse click worked correctly each time. Initially, the display window only displayed about 10 rows. We inquired on the possibility of opening up the window so that it displays more of the list that is available from top to bottom. A later version of the software provided this capability.

3.11 Tables

Figure 4. Example of a Table Using Buttons

CENSUS2010 PRIMARY HHROS FAQ EXIT INFO ENGLISH SPANISH

What is firstname4's Last name?

##	Status	First Name	MI	Last Name	Gender	Related
1	Add	firstname1	1	lastnamefamily	M	Reference person
2	Add	firstname2	2	lastnamefamily	F	Husband or wife
3	Add	firstname3	3	lastnamefamily	M	Biological son or daughter
4	Add	firstname4	4	lastnamefamily	F	Biological son or daughter
5	Dele	firstname5	5	lastnamefamily	F	Biological son or daughter

LNAME END 1:02 PM

The table in figure 4 demonstrates the use of a button in column “##”. After entering all of the information for a household member, the corresponding number button becomes active by turning Yellow. Once active, the button allows navigation to the demographics section for that person. The button color change to orange once partial demographics information is entered. A green button is displayed if the demographic information is complete for that household member. Displayed above the table is the current active field question text. The status bar, defined in the Master template, will display the Don’t Know and Refusal buttons when they are valid for the active field.

The use of tables was simple once we learned how to do the grouping. Initially we had trouble attempting to implement the following features:

1. **Changing the background color when a field was active.** This issue was solved in later versions. Our temporary fix was to use a watermark on the active field.
2. **Displaying the question text in a table.** This was temporarily within four master templates. This was also solved in a later version of Blaise. We started with multiple master templates, but with more knowledge, we were able to reduce this down to one template.
3. **Jumping issue when the answer was a different size.** The column would grow and shrink based on the answer in that column. This was solved by using the stretch setting and setting the size to the maximum size needed for the field.
4. **Displaying the entire answer of a drop down.** The display window didn’t seem to be wide enough. Alternatively, we put an abbreviated answer option at the front of the answer list.

Figure 5. Example of Table with Dropdown Content

#	Status	First Name	MI	Last Name	Gender	Related
1	Added	firstname1	1	lastnamefamily	M	Reference person
2	Added				F I	Husband or wife
3	Added				M	Biological son or daughter
4	Deleted				F I	Biological son or daughter
5	Deleted				F I	Biological son or daughter

Figure 5 –The dropdown contents are larger than the answer in the table causing truncation of display. DK/RF will also display in the dropdown if they are valid answer options.

5. **Making the connection of rules and template variables.** We added field references. This linked the fields to the templates and the templates could associate the field value from the rules.
6. **Placing buttons on a table.** We were able to master this with assistance from “Statistics Netherlands”.

We found that setting up the layout setting is still time consuming. Using “promote” for each setting proved to consume allot of time. For example, we had a table with 18 items and used promote 5 times per item. This required 90 promote picks and if the design was changed, we had to do again. Another challenge we encountered is identifying the best way to handle different data types in the table.

Figure 6. Example of Templates

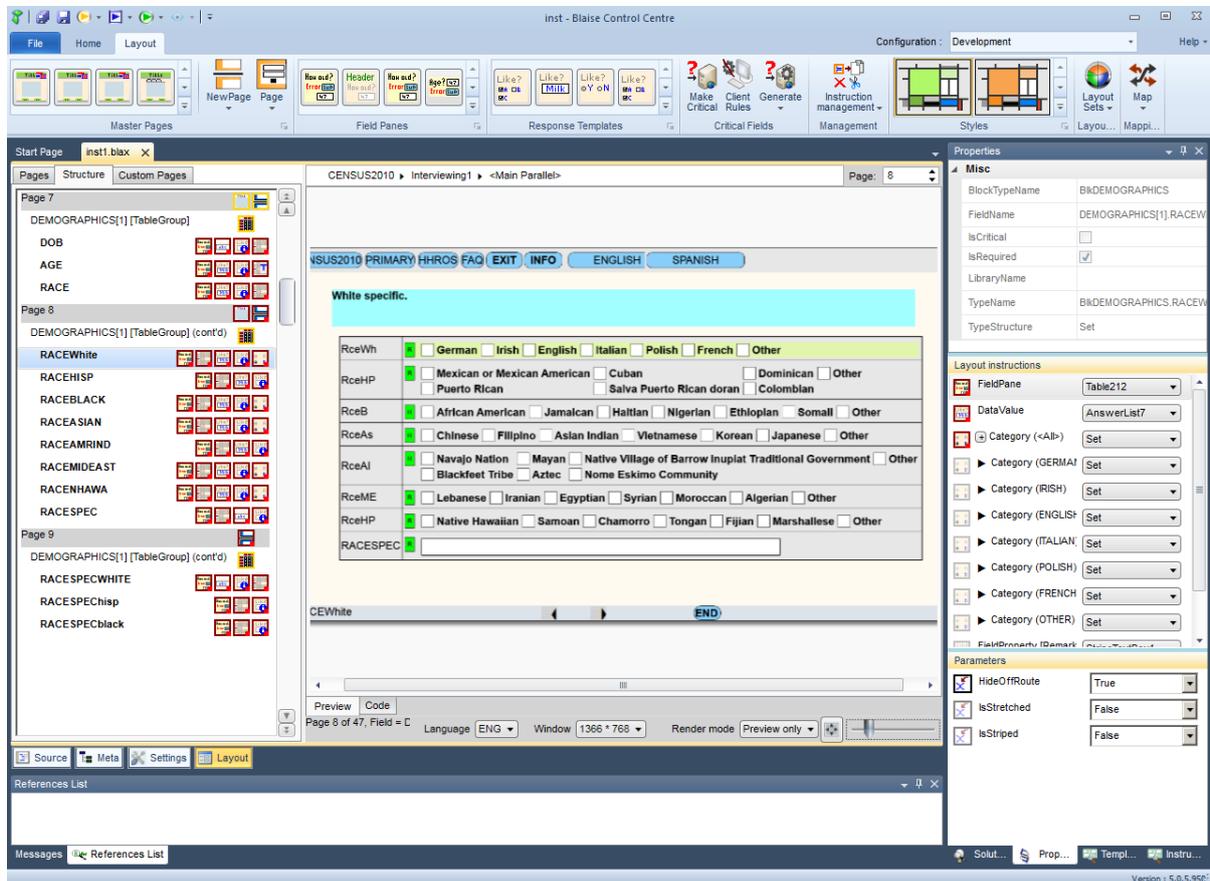


Figure 6 – Demonstrates the use of Templates on top of templates. There is a Master Page template, a table template, and 5 other templates used to display the row for RceWh.

Guidance is needed when it comes to adjusting for drop down list, button answers, select all that apply, enumeration, and other in a defined table. Alternatively, we made data value templates. We set the number of columns for each template. This optimized the display space for different answer size and the number of selections.

3.12 Languages

The languages indicator is displayed in default format near the parallel tabs on one prototype. Once a cell in the template is defined to activate the languages, incorporating this feature is easy. Using the Page Controls under the design tab, clicking on the language selector icon drops the parallel tab to the specified location. The languages are automatically picked as defined in the source code. A drop down list at runtime allows switching between the languages.

We looked at offering separate language buttons on one of the prototypes (see figure 5 above). The separate buttons makes it easier to switch between the two languages. However, that approach utilizes a lot more space in the tab area especially if a lot of languages are defined.

3.13 Status Bar

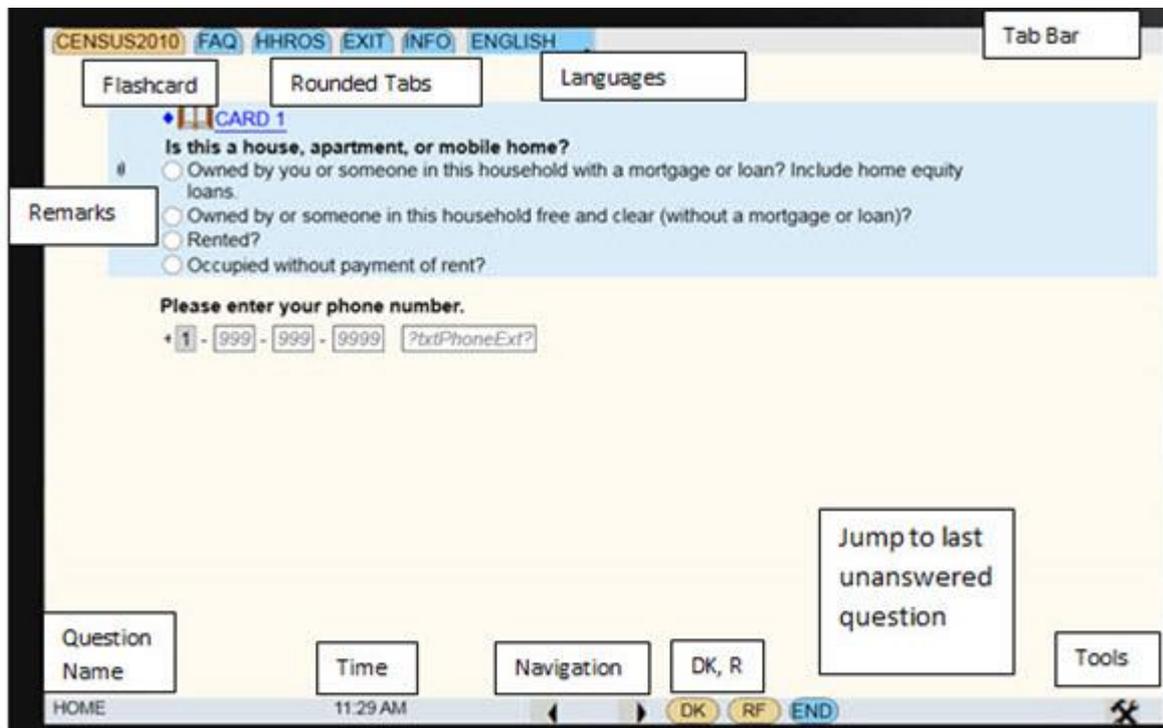
Once the information was defined to be placed on the status bar, it was important that these items were always displayed in the same place, and the displacer used only one line of real estate. Due to information presented in this area and the amount of real estate available, we decided to use front and back arrows instead of the default buttons with text. This helped reduce the “wordiness” of the status bar. Other, minimal data was also placed here (question name, respondent name, time, navigation, DK, R, and tools), with all other relevant data being accessible in either the INFO or HHROS parallel tab. (See figure 7 for an example of information in the status bar)

3.14 Phone Masking

We discovered that masking capabilities typically used for formatting phone numbers was not available yet, but would be implemented in the October 2016 release of Blaise 5. In the meantime, we created a template layout to handle the collection of the telephone number. (see figure 7). This involved the following steps.

1. Identify the field names (Country, Area, Prefix, Suffix, Ext)
2. Add the Type Reference in the BLRD for each of the fields. (Example: TPhoneArea String)
3. For each of the fields, add User Defined text. (Example: txtPhoneArea text:999)
4. Add a Template for the data fields.
5. Add a data value template in the Input Control Option.
6. Add a response value template. Use WaterMark to display format.
7. Add a table template in the Grouping option.
8. Define a Type for each of the fields.
9. Place all of the fields in a Group FullPhoneMask to collect the phone number value.
10. Define the format in the rules within the Group.
11. In the layout set, apply the PhoneMaskTemplate to the FullPhoneMask[TableGroup].

Figure 7. Example of Telephone Template and Other Features



3.15 Don't Know and Refusals

Age

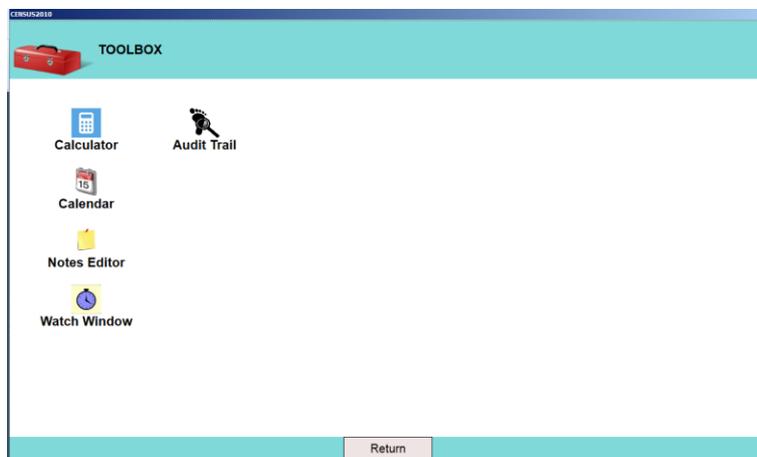
The remarks feature works well.

3.17 Tools

We asked about the possibility of adding a drop down picklist similar to the Language selector. We think this would be beneficial for providing a list for the various tools available as well as handle surveys with multiple parallel tabs incorporated for interviewing sections. The response received was that this kind of functionality, dropdowns, menu-bars in general will be available when Maniplus has been implemented.

Alternatively, a tools icon was placed on the far right edge of the status bar (see Figure 7). This is a link, that when pressed directs focus to a custom page. The custom page displays all of the tools that have been made available and are identified below.

Figure 10. Example of Tools Available via the Tools Icon



3.17.1 Basic Tools

Based on use in the Blaise 4 instruments, some basic tools were incorporated. These tools included the calculator and calendar. In addition, an attempt was made to incorporate the Audit Trail, Watch Window, and a Notes Editor.

3.17.2 Audit Trail

An external program was written to access the audit trail data from the SQLite database "AuditTrailData.db".

3.17.3 Watch Window

Utilizing Software from University of Michigan, a C# project was created to demonstrate how to use the API's to generate a Watch Window Application, resembling the Blaise 4 Watch Window. A Windows Presentation Foundation (WPF) uses the same services to secure the applications. The goal was to run the Blaise 5WatchWindow.exe when the browser of the Wpf DEP is active for the instrument having a primary key.

Example:

```
Blaise5WatchWindow.exe -KeyValue:100 -InstrumentID:5ebfd908-c4c0-4959-bf0b-663829c287e2
```

Then you can choose fields from a field selector. The values of these fields will be displayed in the window; possibly changing whenever the rules of the interview are executed.

4. Features that worked well

While considering the redesign of CAPI instruments and future of touch screen devices, we found a number of features that worked well:

4.1 Buttons

Work well and are very flexible. They can be used for navigation, running routines, and have the ability to assist with implementing the look and feel of the questionnaire.

4.2 Fonts

Appear to be unlimited to how these can be defined. However, a lot of existing Census instruments use Wingding fonts, which are not currently supported. We understand this will be available once Blaise 5.1 is released.

4.3 Layouts

Once set, adjustments are easier.

4.4 Tooltip

Offers flexibility in amount of text allowed as well as positioning on the screen.

4.5 Watermarks

The Watermarks can be easily defined. They can also be used for hints, and prefilling help on how data should be entered. To save real estate, this feature can be used for our interviewer instructions.

4.6 Multimode

We like that this is supported. This feature can be pretty powerful. We are exploring this more based on devices and potential navigation.

5. Challenges Encountered

We, of course, encountered some challenges along the way. The following are some attempts that we made without success:

1. **Sub screens** - If we were allowed the option to create another screen smaller than normal, then we think this will allow us some additional flexibility. This could be used for studies that have a large amount of parallel tables, an “Info Tab Screen”, or even a table with a dropdown list containing a large number of answer options. This is similar to what is displayed when multiple languages are available.
2. **Numeric Dropdown** - We noticed that the dropdown feature is only available for character fields. We think a dropdown for numeric will also be useful. This can be used on touchscreen to select number of people in a household.

These are some features we feel can be useful, but didn't fully work for us.

5.1 Use of .pdf for help

We found that we really needed the ability to pass parameters from the command line for this to work as we would like.

5.2 Print

We thought this was a neat feature, but found that we could not use the feature outside of the Blaise 5 software. For example, we could not use this feature from a scripting language.

5.3 Tables

We would like to know the best way to display a table with multiple data types and the behavior of tables in a group.

5.4 Answer Required

We think it would be helpful if we could personalize the "Answer required" to include field name, so it would say "Answer required for FNAME". The "Answer required" is a system message when a field is defined with the "EMPTY" parameter. We would like to modify so it shows "Answer required for" + Field.Name

5.5 Fills

We could not use field references/fills in system texts.

6. Future Research

We think we could use the ability to add a drop-down box in the template.

We are still looking forward to the masking feature for use with phone and date of birth. We also need the ability to pass parameters into executables.

The ability to use the print option from a script and produce the questionnaire output may be helpful.

New releases often bring software improvements. Updating Help contents could prove most helpful as we explore the new capabilities.

7. Conclusion

The Blaise Authoring Team Layout Group began researching options in June, 2015. We began using Blaise Version 5.0.3 Build 810. In the early stages, we ran into bugs, but being new to the software, often thinking that we were not setting things up correctly, it was time consuming to work our way through the process and conduct the research. During the course of our investigation, a number of enhancements and bug reports were submitted. The support provided was excellent. We received quick turnaround on our feedback, and a number of suggestions and enhancements have been implemented in subsequent versions.

We have made a lot of progress in our understanding of Blaise 5 layouts and templates. We have a draft Master Template that we plan to use and maintain moving forward. This template will be modified as new features are made available with the new releases of Blaise 5 software and as additional decisions are made on implementing a Blaise 5 survey.

8. Acknowledgements

The author would like to acknowledge the work and knowledge of Richard Squires, Michael Johnson, and Daniel Moshinsky as their assistance on the Census Form Prototype instruments was invaluable in researching and testing the Blaise 5 features and identifying the challenges that were presented.

The views expressed in this paper are those of the authors and not necessarily those of the U.S. Census Bureau.

From Blaise 4 to Blaise 5: Systems in transition

Leif Bochis Madsen, Statistics Denmark

1. Abstract

For more than 15 years Statistics Denmark has used Blaise 4 as a tool for data collection for a wide range of surveys. During this period of time a lot of effort has been made to tailor systems and utilities covering Questionnaire Development, Survey Management, Sample Management, Case Management, Post Processing, etc. in order to support fast, efficient and automated processes.

When moving to a new generation of the basic software it is important to assert that the production system supports all the necessary sub systems in order to keep productivity high, as well as considering which parts of the system that should be revised.

This paper aims to describe the general data collection system at Statistics Denmark/Survey division and the work needed identifying, prioritizing and migrating or changing the sub processes of this system.

2. History

The general Data Collection System used by DST SURVEY (the Interview Services Division of Statistics Denmark) consists of a number of loosely coupled subsystems. The system has been developed, maintained and extended since 2000 with the purpose of administering the collection of data via questionnaires written in Blaise 4.

The first part consisted of a menu system designed to support CATI data collection and it's various tasks like setting up new surveys, import of samples, daily CATI administration, export of completed data, etc. It was designed as a role-based system, i.e. the individual user was presented with the options relevant for this particular kind of user, like interviewer, researcher, supervisor, etc.¹

In 2007/2008 CAWI was introduced as a general mode in our data collection and dual mode (CAWI and CATI) soon became the main data collection scheme. This led to several amendments to the data collection system in order to support deployment of web questionnaires on a web server and integration of data collected in the two different modes.

Also, a number of extensions have been made in order to manage automated tasks and to support the questionnaire development process² and testing of data models.

In 2014/2015 the system was extended in order to support CAPI data collection following the fusion of the interview services division of Statistics Denmark with SFI SURVEY – the data collection organization of the National Institute for Social Research.

3. System Overview

All these systems and subsystems are based on a data collection with Blaise 4, so the challenge is to support data collection with Blaise 5.

Fortunately, only parts of the sub systems are closely connected to Blaise 4 which means, we can actually make a plan for a gradual incorporation of Blaise 5 into our data collection system.

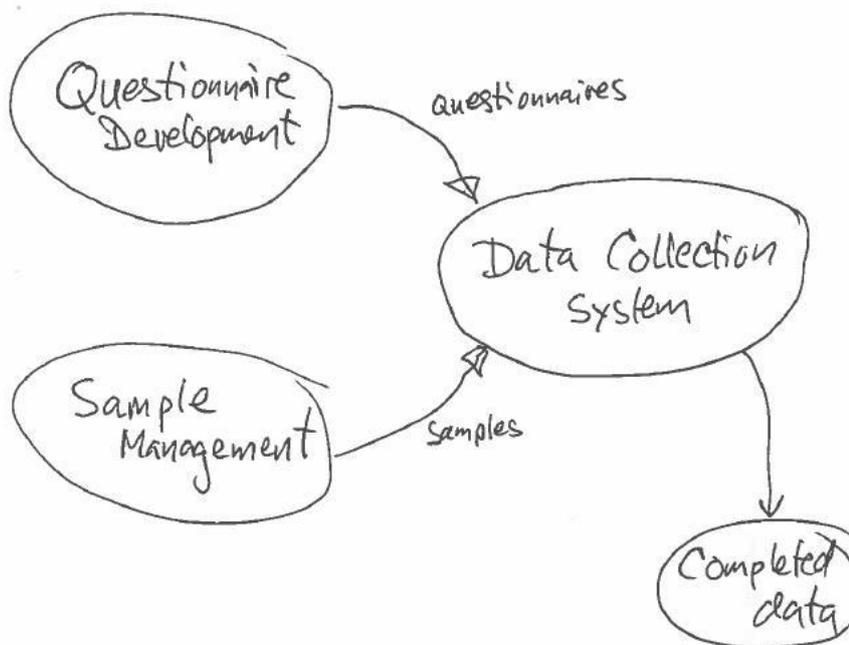
¹ The CATI Survey Management System was presented at IBUC/2003 in Copenhagen.

² The Questionnaire Generator was presented at IBUC/2013 in Washington DC.

The subsystems of the Data Collection System covers:

- Questionnaire Development (Code generation, setup files, source folders, scripts etc.)
- Datamodel Testing (Code inspection, Random Walk)
- Deployment (survey installation, sample import, setup of automated tasks)
- Removal of completed forms from web servers
- Survey Management (CATI Menu System)
- Automation Management (Subsystem for carrying out automated tasks)
- Automatic procedures
 - Day batch creation
 - Transfer of web forms to CATI system
 - Transfer of CAPI forms to CATI system
 - Status information retrieval
- Export of completed forms
- Management of collected CAPI forms (upload, import, reporting, accounting)

Figure 1. Systems Overview



3.1 Questionnaire Development

Development of questionnaires comprises generation of Blaise code from the specification and creation of survey specific folders and setup files for development, test and data collection.

Through a Maniplus program general settings are filled in and folders and setup files and a set of scripts used for e.g. testing are all created automatically.

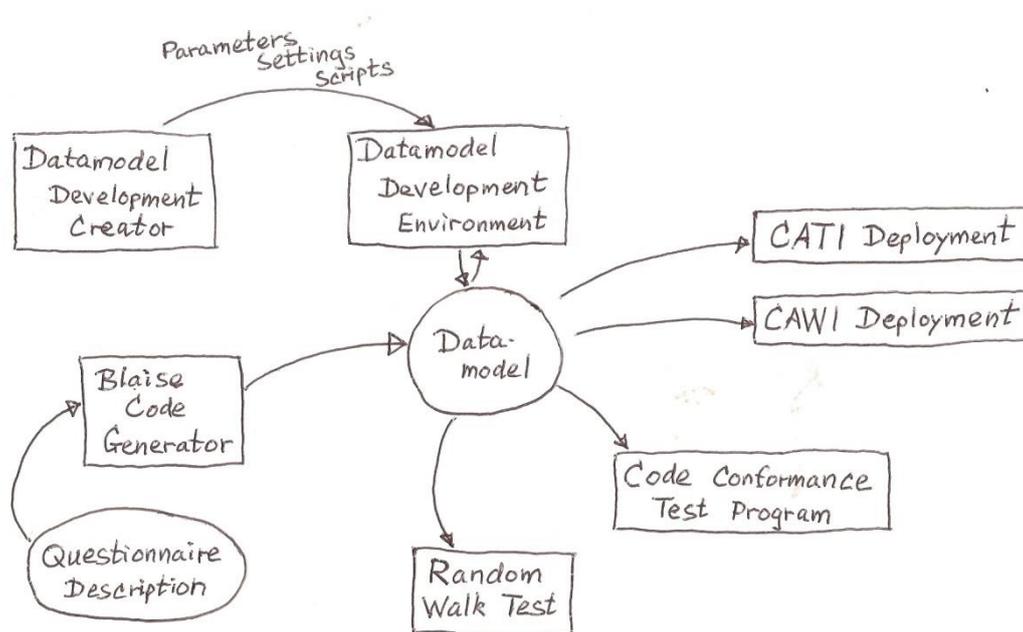
The specifications are written in a MS Word document with a table-based structure and by using an XSL script it is transformed into Blaise code which is afterwards edited by a Manipula program into an (almost) preparable Blaise datamodel. The former also generates a supplementary datamodel without rules sections and uses this datamodel as a repository for checking the references in the rules sections (field names, their corresponding types and proper values of expressions). Thus, it is possible to generate the main version of the same datamodel including rules instructions that are ready for preparation.

In order to manage this automatic creation all datamodels that should be used in our system are created from templates to ensure that properly structured models are created.

The generator creates the basis for adjusting the questionnaire through the Development and Test phase (it is a “90 percent generator”: it automates generation of the 90 percent most trivial parts of a Blaise datamodel).

Blaise 5: For Blaise 5 we have already made a preliminary version of a generator able to handle groups. Also, a C# program managing the editing using the Blaise 5 Meta API has been written in order to replace the Manipula program used for creation of Blaise 4 datamodels. The Manipula program has been extended in order to manage setup of files and folders for development of Blaise 5 datamodels.

Figure 2. Questionnaire Development



3.2 Test of datamodels

Test of datamodels comprises a general tool for checking Code Conformance and a Random Walk program for automatic generation of a larger number of randomly filled forms. Code Conformance checking ensures that the developed datamodel is structured according to our standards and that field and type names comply to our naming conventions,

Both of the tools are written in Manipula 4. General test settings are generated automatically together with the generation of files and folders. Test deployment is carried out by the same program that supports production deployment.

Blaise 5: None of these tools are ready for Blaise 5 at present though it should be possible to convert them into Manipula 5 accessing the metadata via API calls.

3.3 Sample Management

Samples are taken care of by a SAS-based system. Samples are delivered as xml files into the proper survey folders. The xml files are in the format delivered by SAS and a C# program takes care of importing the data into a Blaise database during CATI or CAWI deployment.

Blaise 5: For Blaise 5 we have tentatively made a conversion utility (XSL script) that transforms SAS xml into Blaise xml in order to use Manipula for importing.

3.4 Web deployment

CAWI surveys are deployed through a web application that carries out the following three tasks on the web server:

1. Installation of CAWI survey (upload and install of Blaise Internet Package file)
2. Creation of folders and files for automatic retrieval and deletion
3. Import of samples file(s) using a C# program able to read SAS xml files and create or update Blaise databases (BDB). In most cases the mapping can be determined from a comparison of the xml file and the relevant Blaise datamodel. Optionally, a mapping file can be supplied.

Blaise 5: For Blaise 5 we have written a standard Manipula script that can update the survey database via the BDIX with data in Blaise xml format – as mentioned in the previous paragraph.

3.5 CATI deployment

The oldest and simplest function: First copy the metadata files into the instrument folder then open the SMS Menu System. The rest is point-and-click operations in order to import sample files and set up CATI specifications.

Blaise 5: So far, we have no support for Blaise 5 CATI. Blaise 4 source code may, however, be generated from Blaise 5 Meta. We have carried out a couple of studies where we have used slightly modified versions of these automatically generated Blaise 4 datamodels for post processing.

3.6 Survey Management System (CATI Menu System)

CATI Survey Management is carried out using the menu system, i.e. tasks like import of (supplementary) samples, interviewing, progress supervision etc. Some tasks are carried out by automatic procedures.

All collected data end up in a CATI database which is the basis for the export and the overview of the collection process – whether or not CATI is a mode used for the particular survey. The advantage is that in a large part of the system we do not have to deal with more than one datamodel for each survey – and may use standard blocks and fields for the generation of overview tables etc.

Automatic procedures comprise:

- Day batch creation
- Transfer of web forms to CATI system
- Transfer of CAPI forms to CATI system
- Status information retrieval
- Preparation of updated analysis data sets (for SAS)

3.7 Management of collected CAPI forms

The CAPI data collection is parallel to the CAWI data collection and comprises a similar set of procedures. These will not be described further in this paper.

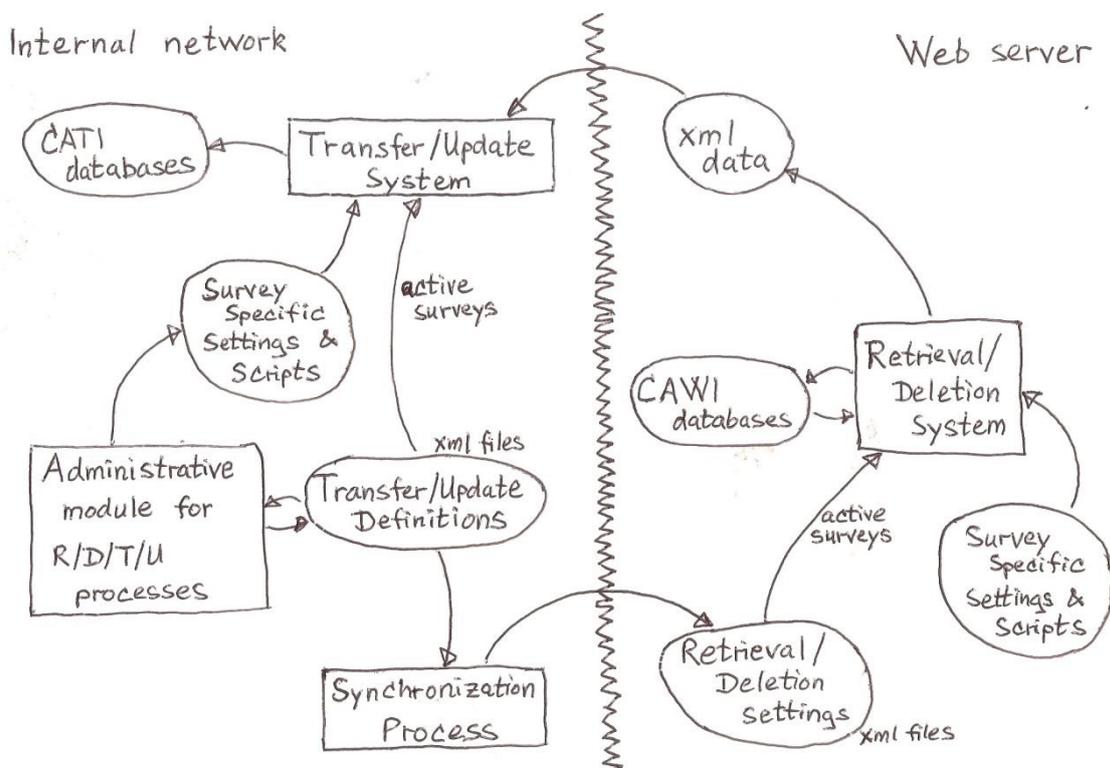
3.8 Automatic day batch creation

Every morning after Transfer/Update processes are completed, day batches are automatically generated for every active survey. Also Status tables and SAS data sets for analysis are made ready during this process.

3.9 Automation management - System for maintaining Retrieval/Deletion/Transfer/Update (RDTU)

At specified times of the day web surveys are checked for completed interviews. They are extracted into xml files and the entries in the web databases are deleted (Retrieval and Deletion process). A few minutes later another process is carrying out the transfer of the xml files to the internal network and update of CATI databases – including setting the proper CATI result codes.

Figure 3. Retrieval/Deletion/Transfer/Update System



The survey definitions are maintained via the administrative module, where surveys are defined with respect to data collection period and other settings. Through creation of an R/D/T/U-definition it is also possible to generate survey specific programs used by the Transfer/Update system.

Standard Retrieval/Deletion is a general program for ‘emptying’ Blaise web databases for completed forms and converting the data into xml. It is built into this program that it can detect different kinds of datamodels, e.g. with or without a primary key (e.g. anonymous respondents), and select the suitable retrieval/deletion procedure. It should be possible to build in automatic detection of Blaise 4/5 data source.

Standard Transfer/Update is also a general program (written in Manipula 4) and handles the output from the Retrieval/Deletion process in order to update CATI databases. This part of the RDTU-system currently only handles Blaise 4 data.

The survey handling definitions are described in xml files and are handled by VB scripts. The automatic execution is fully independent of Blaise.

Blaise 5: Though the management program is written in Manipula 4 this module is fully independent of Blaise and may in fact be used to control any kind of automatic process. Little effort should be necessary to make it support Blaise 5 surveys. For example, by extending this program so it is able to generate Blaise 5 settings and programs for the Transfer/Update system.

For a specific study a retrieval/deletion program was written in Manipula 5. This can be either generalized or automatically created so we can also automate this first part of the RDTU process. Because the xml formats used by Blaise 4 and 5 are the same we are able to handle the retrieval/deletion process with Blaise 5 and the transfer/update process with Blaise4.

3.10 Export and Dissemination

Scripts for exporting data and metadata to SAS are automatically generated by a Cameleon script when needed (via menu or via automated procedures). It comprises a Manipula program and a SAS program controlled by a VB script. All dissemination is carried out by SAS systems.

All data are exported from the CATI data format.

4. Conclusions

The current organization of our Survey Management System comprises:

- standard templates for generation of almost anything
- standard format of completed survey data (i.e. CATI data model)
- automatic procedures based on standard modules
- loosely coupled subsystems

This organization allows us to gradually implement support for the next generation of Blaise in our production environment and thus we can, for example, introduce Blaise 5 for CAWI and keep CATI management in Blaise 4.8 while we are making plans for the introduction of Blaise 5 CATI.

We have conducted our first Blaise 5 study – a web questionnaire for an external customer with special layout requirements – using this scheme. Lots of lessons were learned with this study, but the division between Blaise 5 on the web server and internal processing in Blaise 4 actually turned out well.

There will, however, be some extra work developing and maintaining datamodels in Blaise 4 as well as in Blaise 5 and it is probably more complicated with multi-alphabet surveys. Also, this scheme requires maintained compatibility of Blaise 4- and Blaise 5-xml files which might not be the case for ever.

A setup like this, while useful during a transition period, will only be an option in the short run. We shall be looking forward to conducting our dual mode surveys also in Blaise 5 CATI.

5. Future studies

The basic architecture of Blaise 5 promotes storage of data in external databases. It should therefore be considered which tools we shall use to exchange data with Blaise in the future. Parts of the system might integrate more efficiently through another approach than the current one. We will have to study this thoroughly.

Also, a number of tasks still need to be done before we have a production line ready for Blaise 5 data collection which includes:

- Redesign of templates needed for setup of Blaise 5 datamodels

- Revision of the questionnaire description
- Implementation of a resource database that reflects our requirements for the layout of questionnaires – including layout sets for different kind of devices
- Implementation of the automated procedures still lacking support for Blaise 5

However, at present we are at a state where we can conduct studies in Blaise 5 using the core of our existing data collection environment.

6. References

Leif Bochis Madsen, CATI Survey Management System, in: Papers from the 8th International Blaise Users' Conference, Copenhagen 2003, [link](#)

Leif Bochis Madsen, Saliha Zayoum and Lars Peter Jørgensen, Automatic Creation of Blaise Data Models, in: Papers from the 15th International Blaise Users' Conference, Washington DC 2013, [link](#)

Blaise 5 Data In/Data Out

Andrew D. Piskorowski, University of Michigan Survey Research Center, United States

1. Abstract

The focus of this presentation is to demonstrate various methods used to move data in and out of the Blaise 5 databases. We will show you how the team at the University of Michigan is combining custom tools with Manipula to improve the process of accessing the Blaise 5 data.

1.1 Survey Preload

We have a windowed application that will preload a Blaise 5 data model. In addition to simplifying the preload process, this application allows for incremental updating and adding of sample rows to the project database.

1.2 Survey Migration

Sometimes during data collection, a new version of the survey questionnaire may need to be released. This may happen from an error in original programming, an update in question text, or a change to response options.

This data migration (from the old version to the new version) is needed can be challenging if respondents have already started and suspended on the old version. We want to ensure that we do not lose their data from the old version, and we want to make sure that they can resume the survey where they broke off. We will demonstrate our process for this complex data migration.

1.3 Survey Download

We have created various processes for getting data out of Blaise 5.

- **Manipula and SAS/Metadata** - We have developed automated processes for delivery of raw data from Blaise5. This includes export to a SAS database which integrates code frames and other metadata.
- **MQDS** - Our latest version of MQDS also allows for the downloading of the survey question text and other metadata. This is useful in creating a survey crosswalk - a spreadsheet that maps the survey data responses to the survey questionnaire. MQDS also allows for exporting selected fields into a text file.

2. Introduction

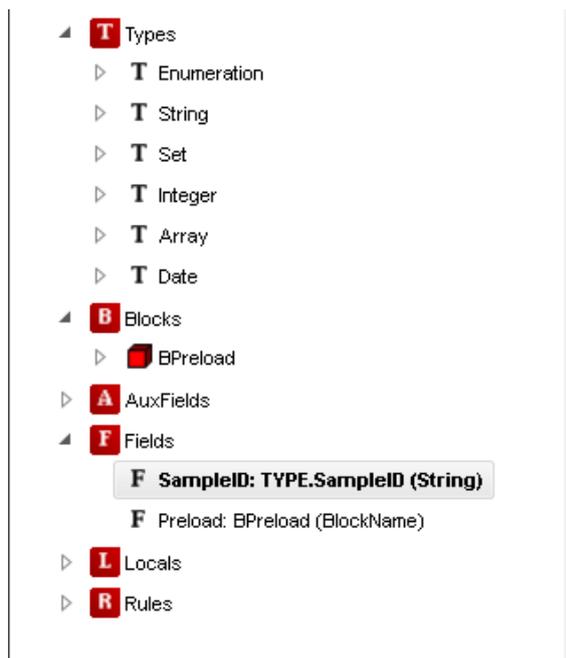
Over the evolution of Blaise 5 there have been many avenues to access the data. In this paper we will review various ways we populate the Blaise databases prior to data collection and how we export data during and after data collection – i.e., data in, data out.

3. Survey Preload

With known survey populations we know some information about the people we are gathering information from and we often look to use this during data collection. This includes respondent attributes like name, gender, number of children, etc. With Blaise 4.8, Manipula is used to preload the Blaise databases to be used by survey logic to control different actions and display functions within a survey. In the early rounds of Blaise 5 there was a learning curve to both programming surveys but also to converting many of the traditional functions and routines previously completed with Manipula.

With the new structure of Blaise 5, a unique compiled data model [.bmix] containing a subset of the main instrument blocks is required to map preload information back to the main instrument database.

Figure 1. Survey Preload



At SRO, creation of this separate preload data model structure is created by our Blaise programmers. Our data manager group has created a simple Manipula script that imports the preload information including the SampleId which is used to access the instrument. Below is an example script. This script uses the preload bmix file to load the values appropriately into the main data model instrument.

Listing 1. Manipula Script Source Code

```
SETUP MySetup

settings
  DATEFORMAT = 'MMDDYY'
  DATESEPARATOR = '/'

USES
ImportMeta 'MyProjectPreload'

MyMeta 'MYPROJECTProd'

INPUTFILE
MyInputFile: ImportMeta ('MyProjectpreloadtestwithouthheaders.txt',
ASCII)
SETTINGS
  SEPARATOR = ','

UPDATEFILE
MyOutputFile: MyMeta ('MYPROJECTProd', BLAISE)

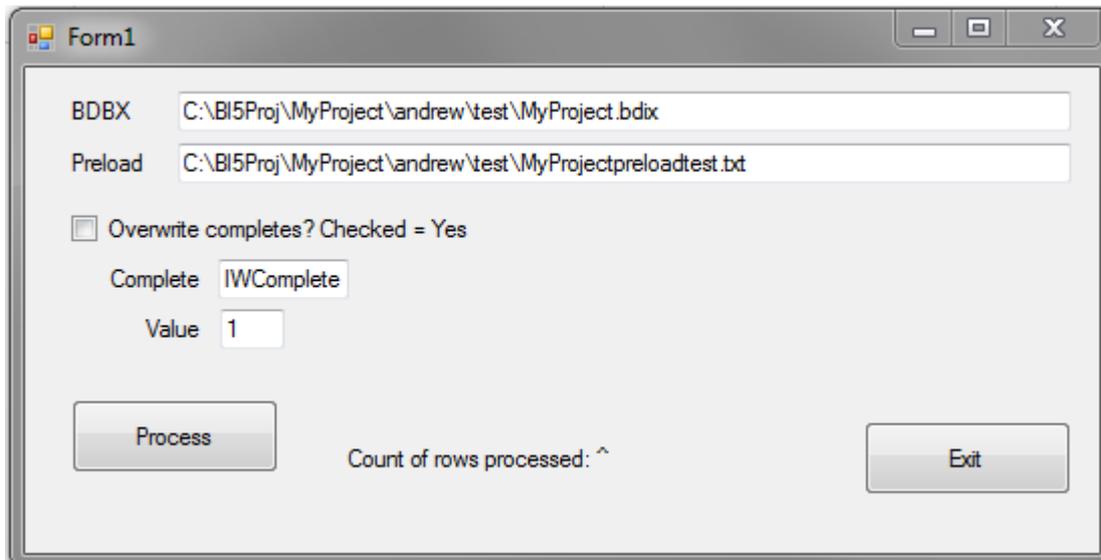
MANIPULATE
MyOutputFile.WRITE
END
```

This method allows for customization of the preload import for different data formats and a more hands-on approach to working with the data.

There are a few downsides of the Manipula approach. The script needs to be recompiled each time the data model is updated. This is required even when the information you are preloading has not changed. The file name containing the preload information is also hard-coded into the script.

As an alternative to Manipula, we created a windowed application, using the Blaise DataLink and DataRecord API , that will preload a Blaise 5 data model for use of the data in a survey administration. In addition to simplifying the preload process, the application allows for incremental updating and adding of sample rows to the project database.

Figure 2. Manipula Alternative Preload



The program takes two arguments: a Blaise Data Interface file [.bdix], and a caret delimited file to be loaded. The program overwrites entries with the same primary key but only if not complete, unless otherwise specified. A verification of the number of records (rows) updated is also provided. The program allows for one not as familiar with Manipula to load an instrument with preload.

4. Survey Migration

At times during data collection, a new version of the survey questionnaire may need to be released. This may happen because of an error in the original instrument programming or because of a needed update to question text or response options.

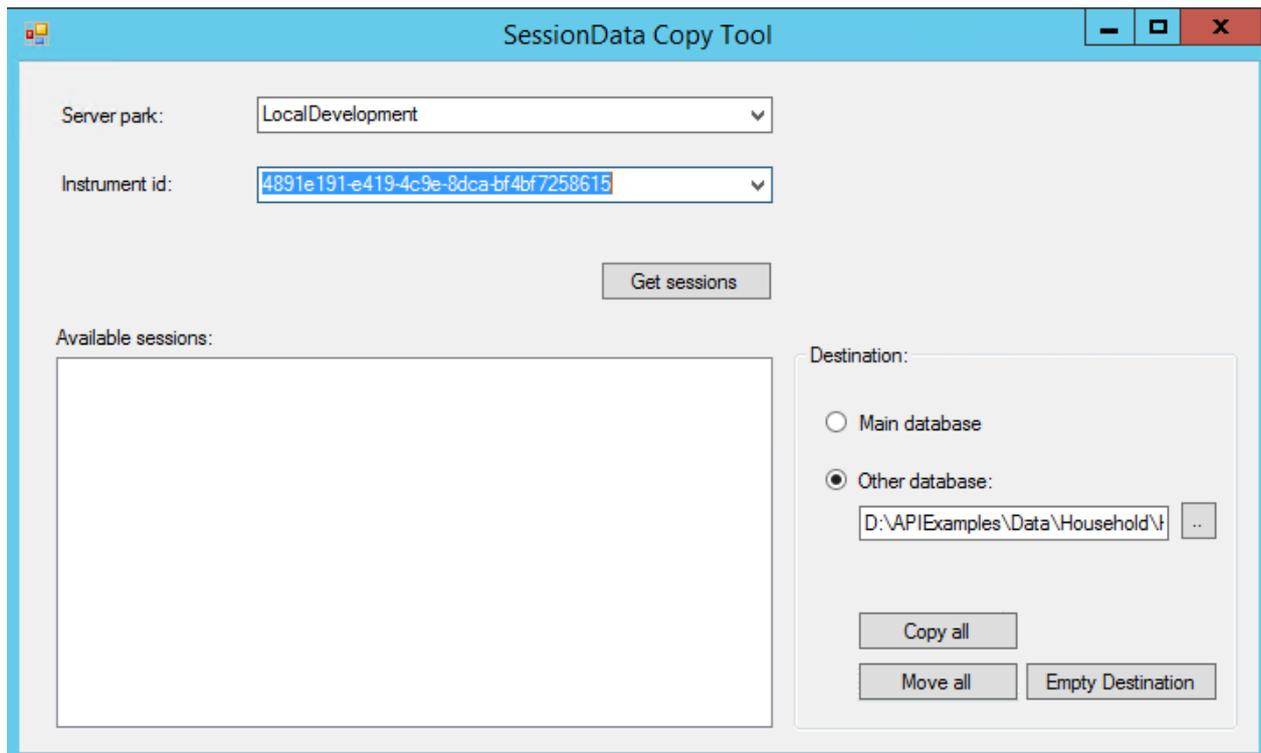
To implement this change, a data migration from the old version to the new version is needed. If a respondent has already started and suspended on the old version, we need to ensure that we do not lose their data from the old version, and we want to make sure that they can resume the survey where they broke off. This complex data migration process involves:

1. Accessing the partial, not yet completed, data stored in the Session database
2. Merging them with the completed data
3. Finally, migrating both the old version completed and partial cases to the new version of the survey questionnaire.

Programming logic was added to ensure partial respondents did not have to resume from the beginning.

To access the partial data stored as blob files in the RuntimeSessionDatabase we developed a custom application to populate an empty database of old instrument.

Figure 3. Populate Application



To use this application we created a separate server environment for data managers to utilize as this requires the installation of a copy of the survey to be migrated.

Migration Process

1. We install the current version of the survey.
2. Overwrite the empty session database with the current production session database.
3. Run the SessionData Copy Tool
 - a. Enter the Instrument Id of the sessions to be downloaded.
 - b. Clicking 'Get sessions' verifies there are sessions available.
 - c. Clicking 'Main database' then 'Copy all' copies the available sessions to the empty database.
4. Run a Manipula [bdbx] to [bdbx] script that copies the partial cases to the new instrument database.

This allows us to preserve respondents data who have already started the instrument and eliminate their burden of having to start the survey over.

5. Survey Data Download

We have created various processes for getting data out of Blaise 5. Many of our projects need a daily delivery of raw data from Blaise 5 to use for analysis and data QC. We developed an automated process that uses a standard Manipula script for data export, batch file processes, and SAS.

Below is a standard script to export completes:

Listing 2. Export Source Code

```
SETUP MYPROJECT
SETTINGS
DESCRIPTION = "My Manipula Setup"
```

```

{ CHECKRULES = YES}
USES
  MetaID 'MYPROJECT.bmix'

INPUTFILE
  InputID: MetaID ('MYPROJECT.bdbx', BLAISE)

OUTPUTFILE
  OutputID: MetaID ('MYPROJECT_CompletesNASK.txt', ASCII)
  SETTINGS          HEADERLINE = YES
                   SEPARATOR = '^'

MANIPULATE
IF InputID.IWComplete = 1 THEN
  OutputID.WRITE
ENDIF
END

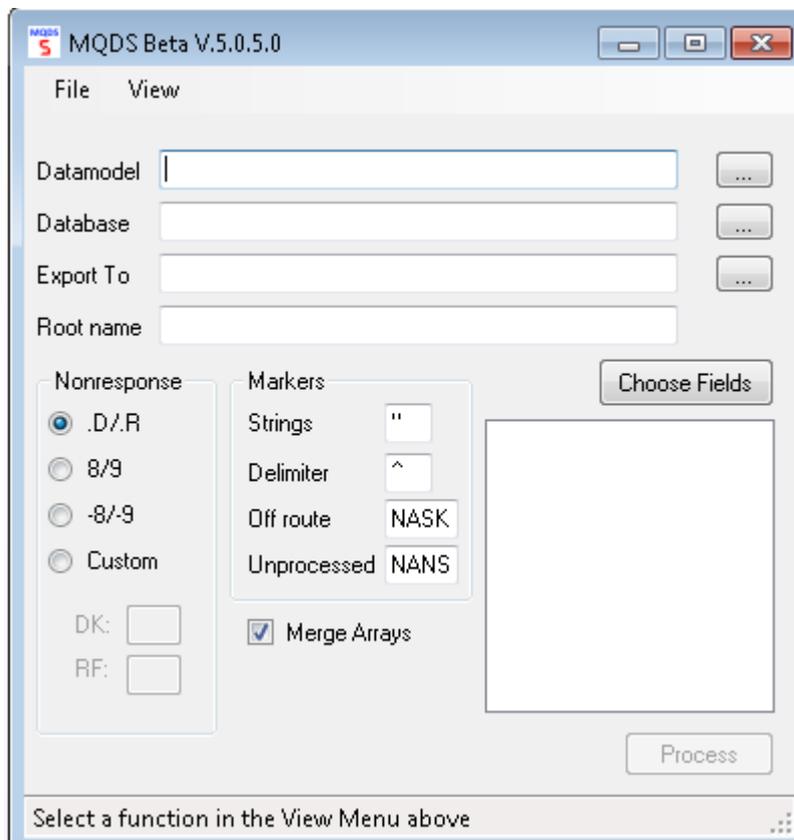
```

A batch file runs the manipula and SAS import to create a new file each day. This file is a raw dataset that can be used by our analysts.

6. MQDS

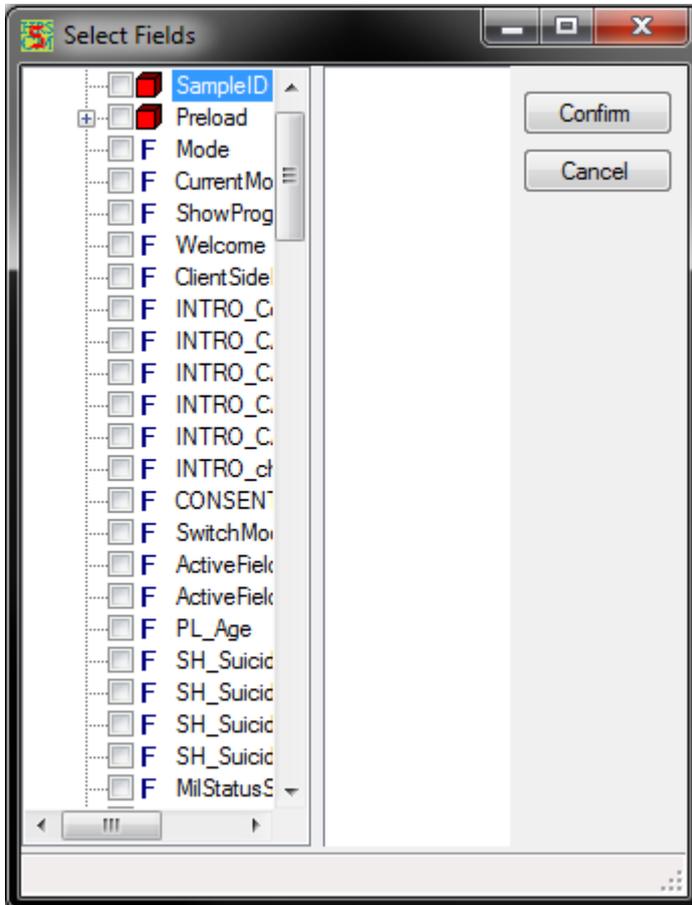
Our latest version of MQDS allows for the downloading of the survey question text and metadata for use in creating a survey crosswalk - a spreadsheet that maps the survey data responses in the dataset created above to the survey questionnaire. The program currently exports the data into separate files for each block within the instrument.

Figure 4. MQDS



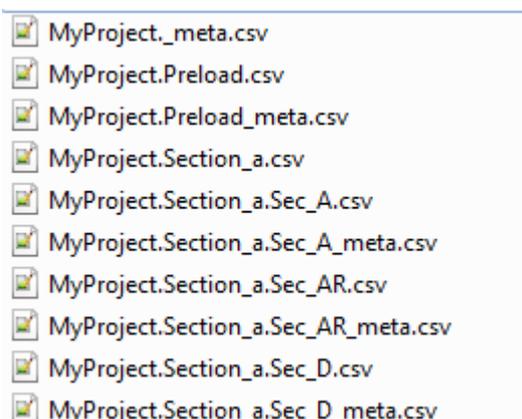
Once the Datamodel is selected, Database, Export To, and Root name are prefilled using the same directory. All can be edited, however. The utility allows for exporting non-response from a Don't Know and Refusal to a few common values or custom options. Under Markers, we are able to select a string identifier and a delimiter. The utility also allows for selecting values for off-route or unprocessed. The utility offers the ability to select only certain fields as well.

Figure 5. Select Fields



As stated earlier, the utility provides separate files for each block but also separate files for the meta data and data values.

Figure 6. Files



Here is an example of the questionnaire that is exported.

Figure 10. Exported Questionnaire

```

***** Section_a.AA_MilStatus_b *****
Current Military status - b
*****
Are you currently on active duty in the Regular Army, another branch of t

1. Regular Army
2. Other branch of the military
3. Army Reserve
4. Army National Guard
5. Other reserve component

Answer not required/Don't Know allowed/Refusal allowed

***** Section_a.AA_MilStatus_c *****
Current Military status - c
*****
Are you currently in the reserves, separated from the Army, retired from

1. In reserves (not active duty)
2. Separated from the Army
3. Retired from the Army

```

Using all the files created above we create an Instrument crosswalk that is used by analysts and clients as documentation of the data. Below are the fields we include in our crosswalk.

Table 1. Crosswalk Fields

COLUMN	MEANING
#	Row number indicating a unique data point
Instrument	Instrument Name
Phase	Wave of data collection
CATI	Indicates whether this data point appears in CATI mode
WEB	Indicates whether this data point appears in WEB mode
Section Label	Section label given to each block or group of questions
Section Designation	Abbreviation for section or block
In DataModel	Indicates whether the field is included in data delivery (contains a data point)
Variable Name	Variable name from the instrument or renamed if changed during processing
Variable Label	Label created from variable name and description of question
Blaise Tag	Unique Blaise tag from Blaise
Section Letter	Larger grouping of similar blocks/sections
Question Number	Question ID
Code Frame	Code frame as programmed in Blaise instrument
Data Type	Data type
Data Range	Data range of values

Table 2. Crosswalk Fields

Instrument	Phase	CATI	WEB	Section Label	Section Designation	In DataModel	Variable Name	Variable Label	Blaise Tag	Section Letter	Question Number	Code Frame	Data Type	Data Range
MyProject	Pilot	0	1	Your Army Car	AA	1	AA_MilStatus	Current Military status	A01_Q	A	A1	MilStatus	Integer	1-14
MyProject	Pilot	0	1	Your Army Car	AA	1	AA_Checkpoint_01	Your Army career and beyond	A02_X	A	CKPT.A1.1	CKPTA11	Integer	1-2
MyProject	Pilot	0	1	Your Army Car	AA	1	AA_MilStatus2	Current Military status 2	A03_Q	A	A1.1	MilStatus	Integer	1-14
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_a	Current Military status – a	A06_Q	A	A1	Yes_No	Integer	1-5
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_b	Current Military status – b	A07_Q	A	A2	MilStatus2	Integer	1-5
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_c	Current Military status – c	A08_Q	A	A3	MilStatus3	Integer	1-3
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_d	Current Military status – d	A09_Q	A	A3a	MilStatus4	Integer	1-3
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_e	Current Military status – e	A10_Q	A	A3b	MilStatus5	Integer	1-3
MyProject	Pilot	1	0	Your Army Car	AA	1	AA_MilStatus_f	Current Military status – f	A11_Q	A	A3c	MilStatus5	Integer	1-3
MyProject	Pilot	1	1	Your Army Car	AA	1	AA_OverallFeel	Overall feeling about militar	A04_Q	A	A1.2	PosNeg	Integer	1-5
MyProject	Pilot	1	1	Your Army Car	AA	1	AA_Checkpoint_02	Your Army career and beyond checkpoint	A	A	CKPT.A1.2	CKPTA12	Integer	1-5
MyProject	Pilot	1	1	Your Army Car	A_AA	1	A_AA_ETS	Army - Do you have an ETS da	A_A01_Q	A	A.AA1	Yes_No	Integer	1-5
MyProject	Pilot	1	1	Your Army Car	A_AA	1	A_AA_Checkpoint_01	Active duty regular army chec	A_A02_X	A	CKPT.A.AA2	CKPTAAA2	Integer	1-2
MyProject	Pilot	1	1	Your Army Car	A_AA	1	A_AA_ETSMonth	Army ETS - month	A_A03_Q	A	A.AA2	Month	Integer	1-12
MyProject	Pilot	1	1	Your Army Car	A_AA	1	A_AA_ETSYear	Army ETS – year	A_A04_Q	A	A.AA2		Integer	2015-2021
MyProject	Pilot	0	1	Your Army Car	A_AA	1	A_AA_ThinkReEnlist	Army - Will be given the optic	A_A05_Q	A	A.AA3	Option	Integer	1-5

7. Audit Data Processing

The AuditTrailDatabase is a useful source of data providing valuable paradata that, once extracted and processed, can be used in many ways. At the University of Michigan we extract the data using SAS, perform data manipulation, and then export the data into a SQL Server to reside alongside the rest of our study data, including data from our survey management system. This allows us to use the data collected in the audit trail in conjunction with other data sources during production.

The AuditTrail is originally stored in a sql lite database. We used the following connection string to connect the database in SAS and parse the 'Content' field into the table that follows.

Listing 3. Connection String Source Code

```
libname Audit odbc complete="dsn=SQLite3 Datasource;
Driver=SQLITE3 ODBC Driver;
Database=Z:\DataManagers\LS\PilotEndGame\Audit\AuditTrailData.db";
```

Table 3. Audit Trail

VARIABLE	DESCRIPTION	HOW ITS CALCULATED
Content	Raw Content from Blaise	
LoginId	Unique primary key from Blaise	
SessionNumber	Computed (LoginId_SessionNum)	Each start session indicates a new session
InstrumentId	From Blaise	
SessionId	From Blaise	
TimeStamp	From Blaise	
Survey	Login or Survey Data Model	If multiple data models are used, we identify from instrument
Seq	Sequence of rows	Row count in order recorded
SessionStart	Computed using Start Session Events	Flag for StartSession events
SessionEnd	Computed using Start Session Events	Flag for last row within session
IWComplete	Computed from EndQ	Flag for EndQuestionnaire event
Width	Width of screen	Parsed from Content
Height	Height of screen	Parsed from Content
Browser	Browser used for session	Parsed from Content
AgentString	Parsed from Content	Parsed from Content
EnterField	Computed from Events (Enter Field)	Flag for EnterField event
PageUpdate	Computed from Events (Page Update)	Flag for PageUpdate event
LeaveField	Computed from Events (Leave Field)	Flag for LeaveField event
FieldName	Computed from Events (Enter Field)	Computed from Events (Enter Field)
Block	Computed from Events (Enter Field)	Computed from Events (Enter Field)
AnswerStatus	Computed from Events (Leave Field)	Computed from Events (Leave Field)
Answer	Computed from Events (Leave Field)	Computed from Events (Leave Field)
EventSeq	Sequence of Events by SampleLinId	
SessionNum	Computed Session Number	Number of current session by LoginId
SessionSeq	Sequence within Session	
SuspendField	Last field seen by respondent	Last Field in Audit Trail for that Session
FieldSeq	Sequence of Enter Fields entered	Count of all enter fields within a session
FieldTimeSec	Time spent on field	Time between EnterField to EnterField

EventTimeSec	Time spent on particular field	Time between current event to next event
PageLoadSec	Time for page to load (server)	Time between navigation click and PageUpdate event
SessionMode	From Start Session	Using parameter passed in through Language to indicate
AuditLayout	From Layout, to pick up/filter out Login	Parsed from PageUpdate indicating layout used for

Further, we aggregate some of this data at the respondent level along with data from our survey management system into the following table:

Table 4. Aggregated Data

COLUMN	MEANING
LoginId	Unique primary key from Blaise
SampleLineID	Unique key within survey management system
FirstString	First agent string provided per respondent
FirstPlatform	First platform used respondent
FirstBrowser	First browser used respondent
FirstWidth	First width of browser
FirstHeight	First height of browser
LastString	Last agent string provided per respondent
LastPlatform	Last platform used respondent
LastBrowser	Last browser used respondent
LastWidth	Last width of browser
LastHeight	Last height of browser
CATIDone	Indicates respondent completed instrument on CATI mode
WebDone	Indicates respondent completed instrument on WEB mode
SurveyComplete	Overall complete indicator
AuditCompleteDT_EST	Complete time of instrument in EST from audit trail
AuditCompleteDT_UTC	Complete time of instrument in UTC from audit trail
LastField	Last field respondent scene from audit trail
LastQAnswered	Last field respondent answered from audit trail
CompletionMode	Overall variable to indicate which mode completed on
AuditvMSMS_Check	Compares survey status of audit trail (complete or not) with status of MSMS (Michigan Survey Management System)
AuditEnd_Check	Flag to indicate if EndQuestionnaire is last event for a complete case
TotalTimeSurvey	Total time spent in survey
LoginAttempts	Number of times they attempted to login
LoginSuccess	Number of times they successfully logged in
LoginFails	Number of times they failed to login
AuditStatus	Overall indicator of survey status (Not started, started, complete)

This data allows us to follow closely, in almost real time, how a respondent is progressing through the survey. It also allows us to QC the data against our survey management system in some key areas including completion status, time to complete, and in which mode.

8. In The End

Data out of an instrument and its connected systems is only as good as the data in. It's very similar to the art of painting the inside of your house - the end results are directly correlated to the prep work

one puts in. As both Blaise and client's needs evolve we have been able to use custom applications, native tools and creative thinking to ensure we're creating a product that provides the good end data all of our users desire.

Data-out Experience and Challenges in Blaise 5

Mohammad Mushtaq and April Beaulé, University of Michigan Survey Research Center, United States

1. Abstract

Blaise 5 web interface was used to collect data from Opt-In Pilot 2015. At the time of data collection, a limited number of data-out utilities were available. The data extraction from Blaise 5 became a challenging task. The available data-out options were flat file and xml data format and previously developed tool for Blaise 4 were not working successfully with these options. Using available data-out options in Blaise 5, the data were extracted into Blaise 4 compatible flat file. Then it was imported into SAS, the wide tables had 11664 and 11970 variables from two survey instruments V1 and V2 respectively.

Using knowledge of core interview instruments from prior years, Blaise 4 extraction and reverse engineering method of data transformation, the wide data files were transformed in relational data structure. First, identify interview sections (Housing, Employment, etc.) and create data files by sections – this is household level file with sample id (SID) as primary key. Second, identify arrays and loops from each section and stack data by loop instance into level-one tables from each section (parent table), remove loop-one variables from parent table. Third, some sections had nested loops (loop inside of loop), therefore, data from these sections were further stacked by loop-two instance and loop-two variables were removed from loop-one table. As a result of this relational transformation, the table structure is matching with PSID 2013. Using same method, the data from All Stars 2016 can be compared with data from PSID (CATI) 2015.

The data-out task would have been much simpler if ASCII-Relational output option in Manipula can be made available sooner so that projects like the PSID can benefit from existing data processing tools.

A second major hurdle that we have to find a solution for with Blaise 5 is identifying questions that are ‘on the route’ but not answered versus questions ‘not on the route’. For self-administered web questionnaires where respondents may just skip a question, it is imperative for data processing that we can easily identify which questions were presented and not answered so that we can properly document the universe of each question in the codebook and generate the appropriate frequencies for valid responses.

2. Introduction

The Panel Study of Income Dynamics (PSID) is a nationally representative longitudinal study of approximately 9,000 U.S. families. Since 2001, the PSID has used Blaise as its main software for its data collection. Due to the size and complexity of the instrument, PSID staff work with tables extracted in looped blocks rather than a single flat file. As the PSID moves to more web-based or mixed mode data collections we have begun to run pilot projects using Blaise 5.

This paper will discuss the challenges we encountered in extracting data from Blaise 5. We also discuss potential future challenges in data handling and documentation for web-based interviews, for which respondents’ ability to skip questions creates challenges for variable documentation.

We begin by giving you an overview of specific challenges of Blaise 5 extraction where are goal was to extract in the same format as Blaise 4.8 CATI. Secondly, we will review the issues surrounding determining an acceptable partial interview in the web self interviewing environment. Finally we will

discuss the data processing challenges that Blaise 5 self-administered web projects present for final variable handling and documentation.

3. Background

In the early years of the study, the data collection instrument was a paper and pencil questionnaire. In 1992, the PSID automated the survey instrument using SurveyCraft software. By 2001, the Survey Research Center (SRC) at the University of Michigan began using Blaise and the PSID was re-programmed at that time based on the new software. Just as the sample continues to grow, so has the instrument. In 1968, the first year of the PSID, the interview took approximately 20 minutes to administer and the staff released 447 variables. By 2015, the average interview length was 78 minutes and we plan to release 5,493 variables on the Family File.

The PSID is a family level instrument meaning that one respondent reports details about all the individuals in the family. In order to capture individual-level information, we have to set the threshold for array sizes higher than the average family size to allow data collection for larger families. These large array sizes as well as the need for many embedded arrays makes flat extraction unwieldy and cumbersome given the large number of blank positions in most interviews.

In all previous versions of Blaise, we extracted ‘relational blocks’ using an extraction tool based on Manipula that was designed ‘in-house’ at the University of Michigan. This has allowed us to handle the data efficiently and keep only rows which contain data while dropping blank rows. However, with our Blaise 5 pilot projects that have launched over the last year we have faced several challenges extracting data in the same format given the limited number of extraction tools available in Blaise 5 at the time of data collection.

4. Blaise 5 Data Extraction and Relational Transformation

At the time of Opt-In Pilot 2015, Blaise 5 Manipula extraction with ASCII-relational output was not available. As previously developed tools did not work with Blaise 5 data out options, therefore, data extraction became a challenging task.

The goal of this task was to create SAS data files which are relational and have same structure as PSID 2015 so that data can easily be compared between two waves of data collection. In order to achieve this goal, wide format extraction and reverse engineering approach were used to transform data into desired relational structure.

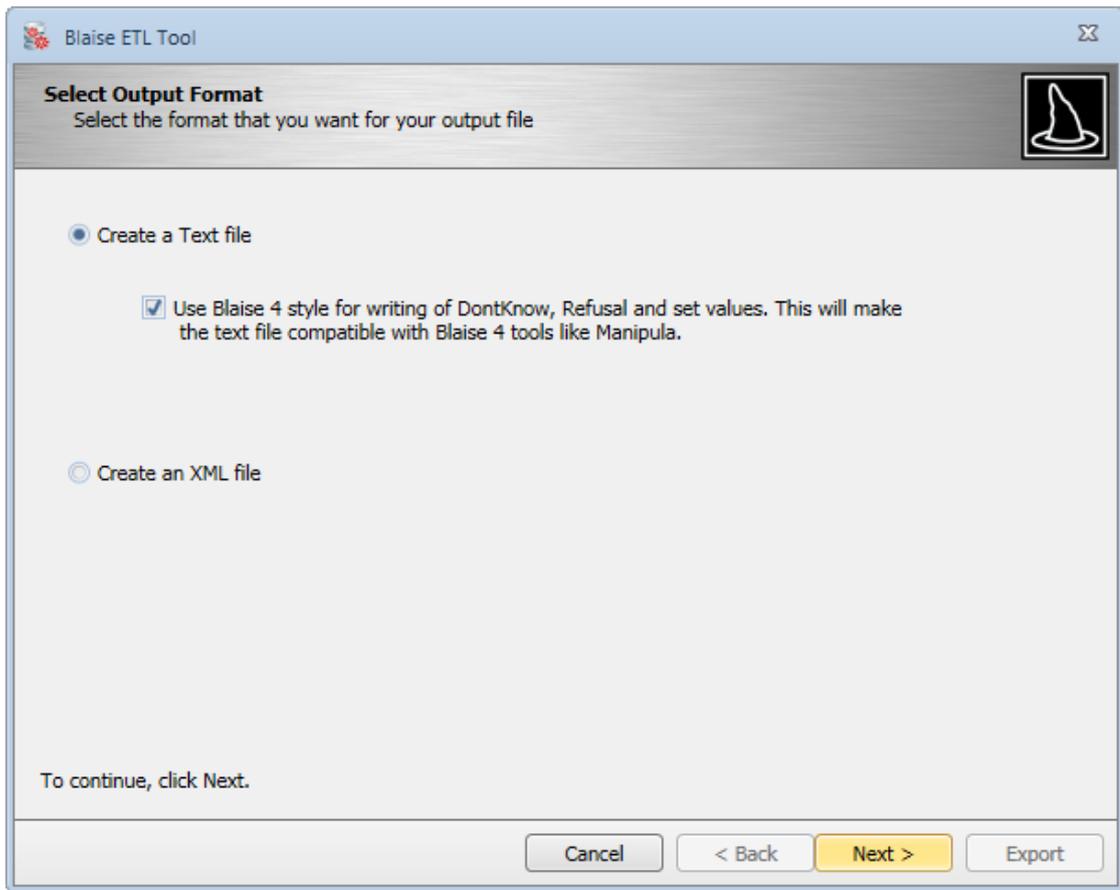
Blaise ETL tools (export functions) are available from “Home” and “Data Viewer” tabs. ETL Export function from “Home” tab does not export all cases. It only keeps 400 cases from the database – the first 400, the last 400 or a random set of 400 cases, we don’t know.

4.1 Data Extraction

After careful examination of all tabs and available options, below are steps that we have used for successful extraction.

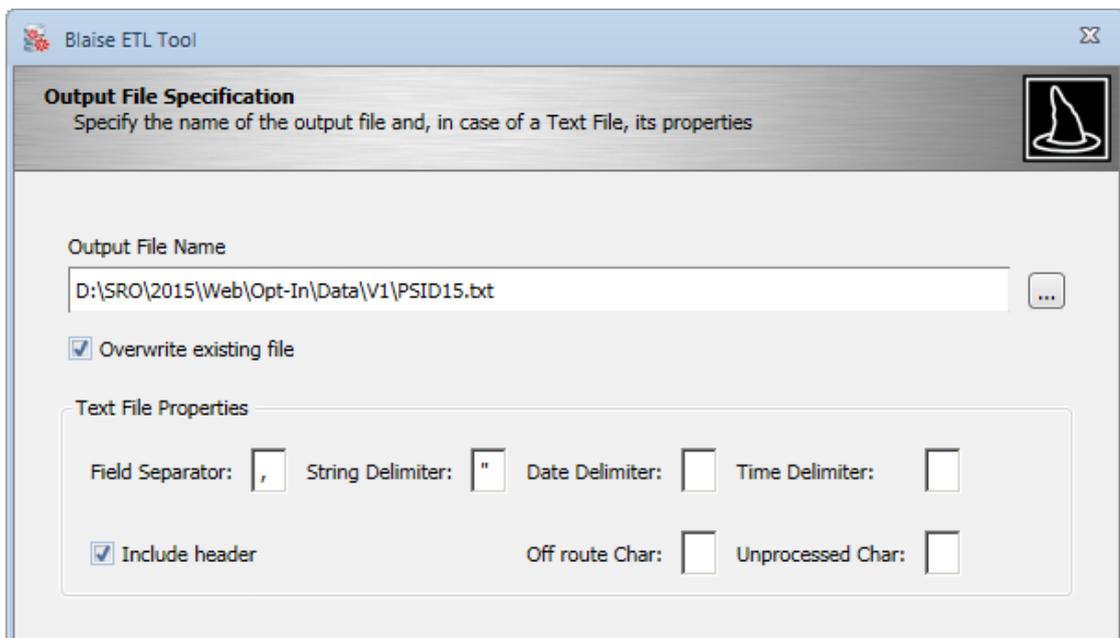
1. Open Blaise 5 database by “Browse Database” option from “Home” tab. With database opened, it also opens “Data Viewer” tab.
2. From “Data Viewer” tab, click “Export Data” and Select Output Format as below.

Figure 1. Output Format



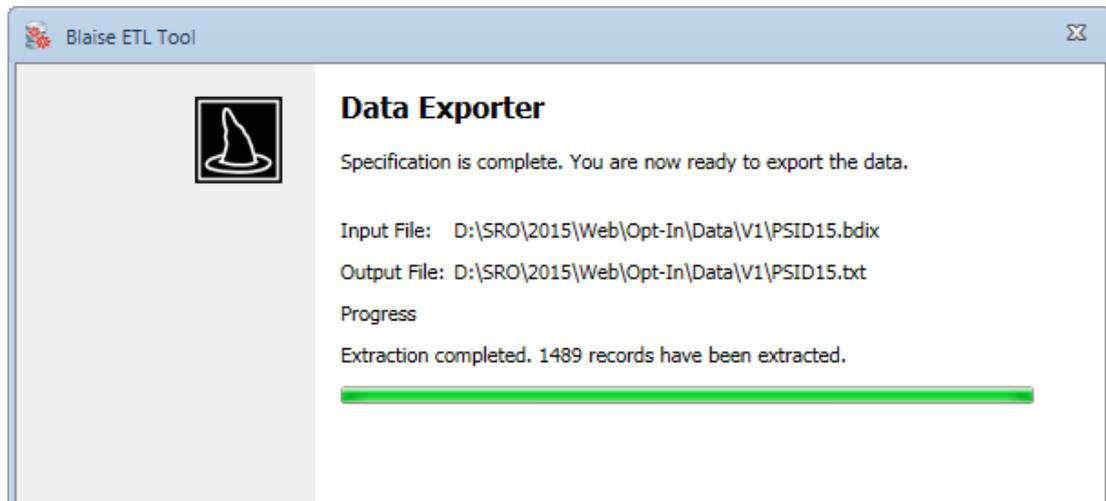
3. Select Output File Specification as below and click Next

Figure 2. Output File Specification



4. Skip Blaise Data Interface File (click Next), select Export from “Data Exporter” tab

Figure 3. Data Exporter



With these steps, data and open text have been extracted in two files. The wide data file PSID15.txt has 1489 data rows (observations) and 11664 columns (variables). The delimiter is as specified in step #3 above. The first row is column header, which will be processed further to create sections and sub-section from the wide file.

4.2 SAS Transformation

Using knowledge of previous year core interview instruments and reverse engineering method, wide data file was transformed in relational data structure. First, identify interview sections (A, BC, F, G, R, W, P, H, M, KL and J1/J2) and create data files by sections – this is household level file with sample id (SID) as primary key. Second, identify arrays and loops from each section and stack data by loop-one instance into child level tables from parent table, remove loop-one variables from parent table. Third, sections P, H and J has nested loops (loop inside of loop), therefore, data from these sections were further stacked by loop-two instance and loop-two variables were removed from loop-one table. As a result of this transformation, V1 database has 52 tables and 2554 variables. The resulting table structure is matching with PSID 2015 core, therefore, the relational structure can be used to compare data between CATI 2015. Below are details of transformation process:

1. Create SAS table from wide file, the table has 1489 rows and 11664 columns
2. Identify sections and sub-section from column headers extracted by the extraction process. In wide file, section A has 149 variables, it has three sub-sections – mortgage, foreclosure and computer usage –, each sub-section has two loops.

Figure 4. SAS Table

30	Section_A.A3	
31	Section_A.A5CK	section A
.....		
43	Section_A.MGTE[1].A23a	
44	Section_A.MGTE[1].A23aSpec	mortgage instance #1
.....		
62	Section_A.MGTE[2].A23a	
63	Section_A.MGTE[2].A23aSpec	mortgage instance #2
.....		
81	Section_A.A28	
82	Section_A.a31a	
.....		
91	Section_A.A37FOR[1].A37B	
92	Section_A.A37FOR[1].A37C	foreclosure instance #1
.....		
98	Section_A.A37FOR[2].A37B	
99	Section_A.A37FOR[2].A37C	foreclosure instance #2
.....		
107	Section_A.A45	
108	Section_A.A46	
.....		
109	Section_A.A57[1].A57a	
110	Section_A.A57[1].A57b	computer usage instance #1
.....		
122	Section_A.A57[2].A57a	
123	Section_A.A57[2].A57b	computer usage instance #2
.....		
164	Section_A.A40_[1]	
165	Section_A.A40_[2]	
166	Section_A.A40_[3]	
167	Section_A.A40_[4]	
168	Section_A.A40_[5]	
169	Section_A.A40_[6]	
170	Section_A.A40_[7]	
171	Section_A.A40_[8]	
172	Section_A.A40_[9]	how heated
.....		
177	Section_A.A46aPer	
178	Section_A.A46aPerSpec	end of section A

Section BC has 1040 variables from wide file and several sub-sections, and each sub-section has 10 loops. Sections P, H and J have second level nesting which makes relational conversion more complicated.

- From our experience with PSID data, Section A (parent table) has one row per sampleid (primary key), and mortgage, foreclosure and computer usage (child tables) are sub-sections where each row is unique by sampleid and instance number. We used SAS data step programming to write SAS code. It created tables for all loops within each sub-section and then all loops were stacked.

a.

Figure 5. SAS code for foreclosure loop #1

```
data A37FOR_01 ;
  length SampleId $24 A37FORInstance 3 ;
  set V1.psid15 (keep=
    Sampleid
    Section_A_A37FOR_1__A37B
    Section_A_A37FOR_1__A37C
    rename=(
      Section_A_A37FOR_1__A37B      = A37B
      Section_A_A37FOR_1__A37C      = A37C )
  ) ;
  A37FORInstance = 01 ;
run ;
```

Figure 6. SAS code for foreclosure loop #2

```
data A37FOR_02 ;
  length SampleId $24 A37FORInstance 3 ;
  set V1.psid15 (keep=
    Sampleid
    Section_A_A37FOR_2__A37B
    Section_A_A37FOR_2__A37C
    rename=(
      Section_A_A37FOR_2__A37B      = A37B
      Section_A_A37FOR_2__A37C      = A37C )
  ) ;
  A37FORInstance = 02 ;
run ;
```

b.

Figure 7. SAS code for stacking data from foreclosure loops

```
data A37FOR (drop=rowsum) ;
  length SampleId $24 A37FORInstance 3
    A37B 8
    A37C 8
  ;
  set |
    A37FOR_01
    A37FOR_02
  ;
  rowsum = sum(A37B,A37C) ;
  if rowsum > . ;
  attrib _all_ label=" " ;
run ;
```

4. The above example is from two loops and two variables and output table is one level below the parent table. The process becomes more complicated when output tables are two steps deeper and/or loops are significantly large number.

4.3 Missing data

1. Numeric variables from loops as specified in Blaise data-model, can assume numeric or characters data type in SAS, when delimited file is read into SAS using data step and “dsd” option. In wide file, each instance is separate set of variables. If a numeric variable has missing data on the first row then its data type is set as character. Therefore, same variable from different instances have mixed data types and the stacking process can’t stack instance tables successfully.
2. Characters variables from different instances can be of different length. If instance-1 variable has smaller length from other variables, then stacking process is going to need additional adjustment, else data will be truncated in the stacking process.

In the absence of ASCII Relational output options from Blaise5 and Manipula, the data extraction task from Blaise 5 database needed lots of resources, good understanding of PSID questionnaire from previous waves and excellent data management skills in SAS.

5. Determining a completed interview

The PSID has started completing supplemental interviews to the main study by offering respondents mixed mode data collection using primarily a Web/PAPI protocol. The web collected records pose several challenges for the data processing team. The first main challenge is the handling of partially completed interviews. In CATI interviews the interviewer codes an interim or final disposition based on whether an interview has gone far enough to be considered a partial. In contrast, with self-administered web interviews we do know when a form has been submitted by the respondent but we do not have a sense of its ‘completeness’. For those cases where an interview has been started but not yet submitted we are also unsure about its status and about whether the respondent will go back and make further attempts to complete it. These ambiguities make it more difficult to monitor data collection while it is in the field and add extra burden for data processing staff to adjudicate which cases we will accept as completed interviews for the final dataset.

In CATI mode, the interviewer develops a sense of item-level refusals and may in fact suspend the interview or code the case out a refusal if the respondent refuses to answer a large proportion of questions. Alternatively, interviewers alert the processing team of problematic cases by triggering a flag in the observation section of the interview. These interviewer-initiated triggers for further scrutiny are not possible in self-administered web surveys and alternative approaches are required to deem an interview an acceptable completed case: For web self-administered surveys, the Data Processing (DP) team works closely with the lead researcher to determine an acceptable level of item nonresponse based on the forms submitted by the respondent that are automatically coded as complete by the sample management system. Generally we may look at the entire sample and determine the average number of variables with non-missing values to determine the acceptable threshold. This approach presents challenges as instruments increase in complexity and the number of variables ‘on-route’ or ‘off-route’ (see below) varies significantly between interviews. In many situations, cases with questionable ‘completeness’ have to be reviewed on a case by case basis, which is often time-consuming.

6. Problem of determining questions ‘on-route’ vs ‘off-route’

One of the most basic fundamentals in data documentation is identifying the universe of each variable. In the PSID, we have long documented the inverse of the universe in our codebooks (*Figure 8*). The INAP (Inappropriate code) lists the criteria the respondent met in order to skip this particular question.

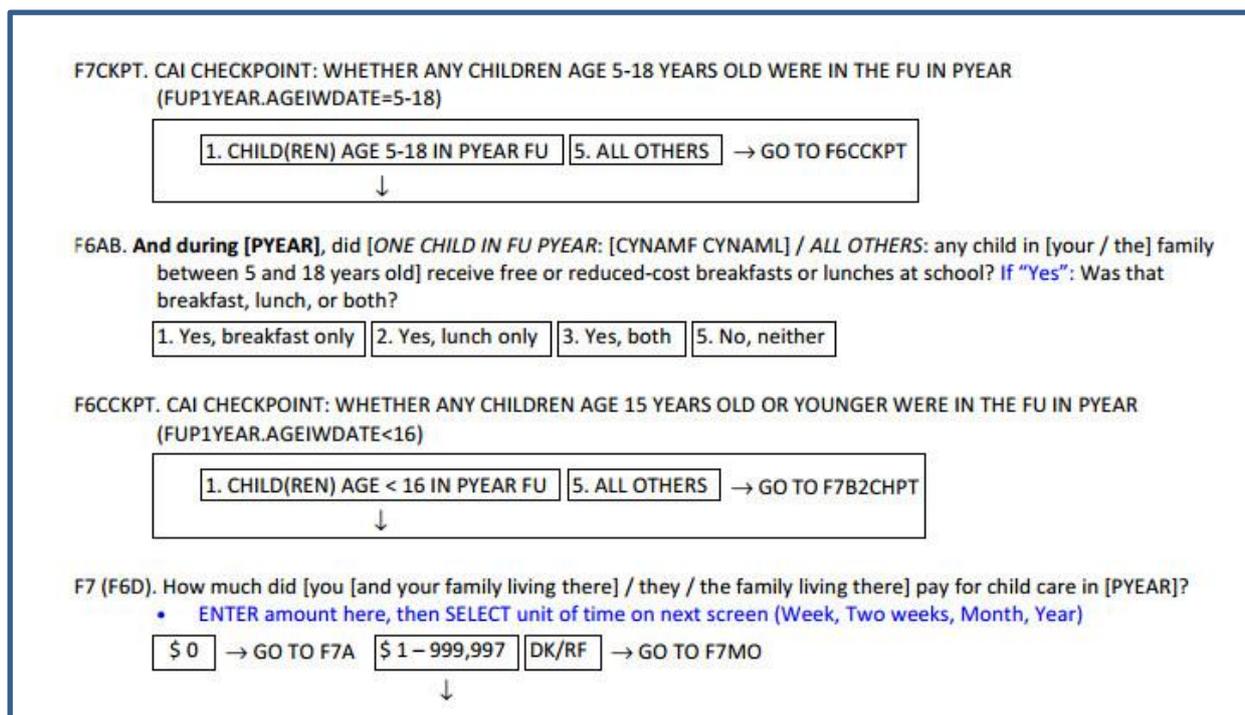
Figure 8. Example Codebook for PSID Family Level Variable in CATI

ER53680			F6AB WTR REC FREE BRKFT/LUNCH LAST YR		
F6AB. And during 2012, did (CHILD NAME/any child in your family between 5 and 18 years old) receive free or reduced-cost breakfasts or lunches at school? If "Yes": Was that breakfast, lunch, or both?					
Count	%	Value/Range Text			
49	.54	1 Yes, breakfast only			
217	2.39	2 Yes, lunch only			
1,193	13.16	3 Yes, both			
1,618	17.85	5 No, neither			
17	.19	8 DK			
26	.29	9 NA; refused			
5,943	65.57	0 Inap.: all FU members younger than 5 or older than 18; all children with current age 5-18 not in the FU in 2012 (ER53679=5)			
Years Available: [13]ER53680					
Index: Family Public Data Index					
Summary: 01>NON-CASH ASSISTANCE					
02>Public					
03>subset selectors					
04>school meals programs, received last year:					
05>type, whether:					

In a CATI interview, we can determine which questions are skipped easily because all questions that are ‘on-route’ (i.e. those that are presented based on pathing) must be answered by the interviewer even if the response is ‘Don’t Know’ or ‘Refused’. In contrast, in self-administered web interviewers the general norm is to simply let the respondent pass a question without an answer. Similarly, ‘Don’t Know’ or ‘Refused’ are not explicit options available to the respondent at all as the aim is to encourage the respondent to provide a valid (non-missing response).

For web data-out in Blaise 5 there is no distinction between variables with a system missing code because that variable was ‘off-route’ (i.e. those variables that are never presented based on pathing) versus a system missing code because the variable was ‘on-route’ but the respondent did not provide a response but instead just skipped the question. Determining the difference between these two scenarios is imperative once we move to the documentation stage that describes the universe of those who did not receive this question because of routing. Because this critical distinction is not currently a feature that we have, the Data Processing (DP) team must essentially re-write all the skip patterns in the Blaise instrument using cleaning/analysis software like SAS. The DP team members use the ‘Box and Arrow’ (*Figure 9*) or visual documentation of the questionnaire often written in a Word document to determine all the possible skip patterns and assign a unique code to questions that are empty because they were ‘off-route’.

Figure 9. Example PSID Box and Arrow Documentation of CATI Questionnaire



For basic questionnaires with few skip patterns, this is not unduly burdensome. However, if we plan to attempt data collection on the web with more complex instruments containing many loops, arrays, and embedded arrays this becomes ever more challenging for the SAS programmer and error prone. Unlike the Blaise programming phase, which includes many rounds of testing by several testers, the SAS programmers do not have the same amount of time or testing resources to check their programs and output for integrity.

Writing the logic of the entire questionnaire by both the Blaise programmer and then again by the SAS programmer is also extremely inefficient and may affect data integrity if the SAS programmer makes an error in determining the correct skip flow (*Figure 10*). It also may offset some of the presumed cost savings of having the respondent rather than the interviewer complete the questionnaire. Since the Blaise system knows which questions were 'on-route' vs 'off-route' at the time of the interview based on the data model version, it appears there should be a way to communicate those differences in the storage and extraction of data from the bdbx automatically.

Figure 10. Example SAS code for Recoding Skipped Variables to Not Ascertained (NA) = 9

```
/*section J*/
/*A1a rule--mother in house*/

  /*to fix 2 zeros to NA*/
    if J1A='0' and A1A ne '9' and A1A ne '7' then J1A='9';
  /*to fix 3 zeros to NA*/
    if J2A='0' and A1A ne '9' and A1A ne '7' then J2A='9';
  /*to fix 4 zeros to NA*/
    if J3A='0' and A1A ne '9' and A1A ne '7' then J3A='9';
  /*to fix 3 zeros to NA*/
    if J4A='0' and A1A ne '9' and A1A ne '7' then J4A='9';
  /*to fix 3 zeros to NA*/
    if J5A='0' and A1A ne '9' and A1A ne '7' then J5A='9';
  /*to fix 3 zeros to NA*/
    if J6A='0' and A1A ne '9' and A1A ne '7' then J6A='9';
  /*to fix 5 zeros to NA*/
    if J7A='0' and A1A ne '9' and A1A ne '7' then J7A='9';
  /*to fix 3 zeros to NA*/
    if J8A='0' and A1A ne '9' and A1A ne '7' then J8A='9';
  /*to fix 4 zeros to NA*/
    if J9A='0' and A1A ne '9' and A1A ne '7' then J9A='9';
  /*to fix 4 zeros to NA*/
    if J10A='0' and A1A ne '9' and A1A ne '7' then J10A='9';
  /*to fix 5 zeros to NA*/
    if J11A='0' and A1A ne '9' and A1A ne '7' then J11A='9';
  /*to fix 4 zeros to NA*/
    if J12A='0' and A1A ne '9' and A1A ne '7' then J12A='9';
  /*to fix 5 zeros to NA*/
    if J13A='0' and A1A ne '9' and A1A ne '7' then J13A='9';

/*A1b rule--father in house*/

  /*to fix 8 zeros to NA*/
    if J1B='0' and A1B ne '9' and A1B ne '7' then J1B='9';
  /*to fix 8 zeros to NA*/
    if J2B='0' and A1B ne '9' and A1B ne '7' then J2B='9';
  /*to fix 8 zeros to NA*/
    if J3B='0' and A1B ne '9' and A1B ne '7' then J3B='9';
  /*to fix 8 zeros to NA*/
    if J4B='0' and A1B ne '9' and A1B ne '7' then J4B='9';
  /*to fix 9 zeros to NA*/
    if J5B='0' and A1B ne '9' and A1B ne '7' then J5B='9';
  /*to fix 9 zeros to NA*/
```

7. Conclusion and Summary

Using knowledge of core interview instruments from prior years, Blaise 4 extraction and reverse engineering method of data transformation, the wide data files from Blaise 5 were transformed into relational data structure. First, identify interview sections (Housing, Employment, etc.) and create data files by sections – this is household level file with sample id (SID) as primary key. Second, identify arrays and loops from each section and stack data by loop instance into level-one tables from each section (parent table), remove loop-one variables from parent table. Third, some sections had nested loops (loop inside of loop), therefore, data from these sections were further stacked by loop-two instance and loop-two variables were removed from loop-one table. As a result of this relational transformation, the table structure is matching with PSID 2015.

It is likely that many large surveys like the PSID will continue to explore data collection via the web given cost-cutting pressures from funders as well as increasing respondent coverage as more and more families gain access to computers and the internet. The expectation from both funders and the user community is that data collected via the web will have the same standards of cleaning and documentation as data collected via CATI.

The current tools offered in Blaise 5 do not allow us to easily check and assign appropriate codes for ‘on-route’ but not answered versus ‘off-route’. This is a significant drawback for data processing and documentation. If tools could be provided to automatically assign fields with this distinction then we could also build on that information to help us come up with algorithms for determining ‘completeness’ of a web interview. Instead of case-by-case checking of completeness we could flag outliers that have more than the average share of missing responses by dividing the number of presented but skipped questions by the number of overall questions that were ‘on-route’.

Census Setup with Blaise 5 Manipula

Michael K. Mangiapane and Roberto Picha, U.S. Census Bureau

1. Abstract

In the process of CAPI and CATI data collection at the U.S. Census Bureau, Blaise is one of a number of parts that work together to ensure we successfully collect data. Before a survey is fully deployed for data collection using Blaise, it needs the appropriate data from our control and case management systems. One of the first steps in this process is to run a Manipula “setup” script that reads a data file and uses the file’s contents to create individual Blaise databases. These Blaise databases are later loaded on an interviewer’s computer to conduct a survey.

When Manipula became available in Blaise 5, we researched how we might use it to implement scripts that have the same functionality as our Blaise 4 scripts. We modified our setup script to run in Blaise 5 and tested its functionality versus the original Blaise 4 script. This presented some challenges since Blaise 5 creates and references databases differently than it did in Blaise 4. This paper will discuss how we overcame this challenge and others so that individual Blaise databases were produced by the setup script. We will also discuss other challenges in using Blaise 5 with the setup program and our progress in converting other Manipula scripts from Blaise 4 to Blaise 5.

2. Introduction

Blaise is one of a number of software programs used by the U.S. Census Bureau in the process of conducting surveys and collecting data. The software applications may be in different languages and exist on different platforms, but they must be able to talk to one another to pass the correct data needed to accomplish their functions. Blaise needs to be able to talk with our Case Management, WebCATI, and our Master Control systems, each one containing their own unique requirements. The way we accomplish this is by using Manipula to translate between Blaise and our other systems.

One of the early steps in our data collection lifecycle is to take a sample input file provided by our sponsors, known as a Sample Control Input File (SCIF), load the data and create individual Blaise databases. These databases are assigned and transmitted to laptops in the field or pushed down to desktop computers in our telephone centers. To accomplish this we create a “Setup” Manipula script which is a standard file created and packaged with all of our Blaise 4 surveys.

Now that Blaise 5 has added Manipula to their suite, moving a survey into Blaise 5 becomes easier for the Census Bureau. We have begun researching how our current Setup script and other Manipula scripts fit in with Blaise 5 and what changes are needed to conduct our surveys. This paper will talk about our current efforts to update the Setup script and other survey related scripts to make them Blaise 5 compatible and the updates required to the code. We will also look at a few challenges that were presented during this process, what we did to work around them, and future plans for our usage of Manipula. As of this paper, we are using Blaise 4.8.4.1861 for most of our current production surveys and Blaise 5.0.5.950 for our current research.

3. Current Blaise 4 Setup

In our current surveys environment, the survey sponsors provide a SCIF that is processed and used by the survey control systems and instrument. A typical SCIF is an ASCII text file that contains “record typed” data that is read in by the appropriate system. Each system (including the instrument) processes only the data that it requires. The control systems use record types that correspond to survey level settings, address information, and contact information. The instrument uses many of these record types along with survey (interview) specific information. Each line of the SCIF corresponds to a “record type” which is like a map that tells our Setup program where to read the data. Most of these record types are standard across our surveys and are identified by a four-digit number that indicates the record type and subtype that is being used for that survey. One exception to the

standardization is the 8500 series of record types. They are known as “dependent data” as they contain large amounts of survey-dependent data so these records are customized for each survey.

In our Blaise 4 Setup, we define each record type as a datamodel, entering the provided variable names and length of each variable so that they correspond to the position of the data for the record. Each datamodel is available as a part of the input file definition, with the output being a Blaise database:

Listing 1. Datamodel Source Code

```
INPUTFILE
  In1060 : InitRT1060 ('lqiinput.dat', ASCII)

ALTERNATIVE
  In2060 : InitRT2060
  In2561 : InitRT2561
  ...
```

Another output file produced by the Setup program is a list of Case IDs, which are unique identifiers that correspond to each case. These come from the SCIF and it serves as a record of which cases were created. We use this for verification that all of the expected cases were created for the interviewing period.

As the Setup script runs, it reads in each line of the SCIF. The first two characters correspond to the record type series (10, 20, 25, etc.) so the script matches which record has been read in and reads the appropriate datamodel to load the script. Each new case starts out with Record Type 10 so that is used as a signal to the script to create a new database to load data at the top of the script. Once that is done the script follows its standard procedure to load data into the case. It will perform a block to block copy of the data into a block at the datamodel level of the instrument (for backup purposes) and it will also load data into individual variables defined in the instrument.

Listing 2. Sample of Blaise 4 Manipula Script.

```
IF In1060.RecType = '10' THEN
  {If not the first record in the file }
  IF ID <> In1060.Rt1060in.CASEID THEN
    Outfile.OPEN(In1060.Rt1060in.CASEID)
    Outfile.INITRECORD
    {Copy Record to Block}
    Outfile.Rt1060 := In1060.Rt1060in
    {Copy to datamodel level variables}
    Outfile.CASEID := In1060.Rt1060in.CASEID
    Outfile.CTRLNUM := In1060.Rt1060in.CTRLNUM
  ...
  {Determine record type and load appropriate data}
ELSEIF In1060.RecType = '25' THEN
  {Copy Record to Block}
  Outfile.Rt2561 := In2561.Rt2561in
  {Copy to datamodel level variables}
  Outfile.PRICEEDIT := In2561.Rt2561in.PRICEEDIT
  Outfile.SALEPRICELO := In2561.Rt2561in.SALEPRICELO
ENDIF
In1060.READNEXT
...
```

The script continues in this manner until it reaches another Record Type 10 line in the SCIF. At that point the script writes to the Blaise database, closes it, and starts the process again until it reaches the end of the SCIF. The result is a set of Blaise databases that are ready to be packaged up and assigned to the appropriate computers to conduct the survey.

Figure 1. Blaise 4 databases created by current Setup script, Cases 22TB001N and 22TB002N.

Name	Date modified	Type	Size
 22TB001N.bdb	8/12/2016 12:12 PM	Blaise Database	12 KB
 22TB001N.bfi	8/12/2016 12:12 PM	BFI File	1 KB
 22TB001N.bjk	8/12/2016 12:12 PM	BJK File	1 KB
 22TB001N.bpk	8/12/2016 12:12 PM	BPK File	1 KB
 22TB002N.bdb	8/12/2016 12:12 PM	Blaise Database	11 KB
 22TB002N.bfi	8/12/2016 12:12 PM	BFI File	1 KB
 22TB002N.bjk	8/12/2016 12:12 PM	BJK File	1 KB
 22TB002N.bpk	8/12/2016 12:12 PM	BPK File	1 KB

4. Creating a Blaise 5 Setup

Since Blaise 5 Manipula contains significant changes in its language, our Blaise 4 scripts must be updated to make them compatible with Blaise 5. In the current Blaise 5 version of Manipula we do not have the ability to do a block to block copy, which was functionality we wanted in our Setup script along with copying data to our datamodel level variables. That was an immediate challenge for running our Blaise 4 scripts, but we were able to come up with an alternate way to load our data and keep that functionality.

One method that is available for both Blaise 4 and Blaise 5 Manipula is the PUTVALUE method. It allows us to feed in a fully qualified field name in the instrument and the value that should be placed in this field, including a Don't Know or Refusal. We decided as a part of the design process to use PUTVALUE to place our data because not only could we feed in the full path to the block where we keep the backup copy of the data, we could call it again and use a substring to copy the data to the datamodel-level variable. Since there is only one block name before the variable name, we can use the POSITION function to find the position of the period that separates the two, resulting in the variable name only. For example:

Listing 3. Position Function Source Code

```
iPos := position('.',iPath)
iLen := LEN(iPath)
//load variable at the datamodel level
Outfile.PUTVALUE(SUBSTRING(iPath,(iPos+1),(iLen-iPos)+1),Mydata)
//load variable at the RT block level
Outfile.PUTVALUE(iPath,Mydata)
```

In a similar fashion to our Blaise 4 Setup, we coded the script to read in each line of the SCIF until it reaches a Record Type 10 and then it writes out the data to the database. This became an opportunity to simplify coding the script and it makes it easier to do later maintenance if any record layouts are ever changed. Each input variable is written into the backup block and into the datamodel-level variable in the instrument so PUTVALUE must be called twice. We could just simply put a call to PUTVALUE for every variable in the instrument but that leaves us with a very long script. Many of our instruments are reading in over one-hundred twenty (120) variables from the standard record types and dependent data could more than double that number.

A best practice in coding is if there is something that is done repeatedly but not in a loop, then place it into a procedure. Rather than call PUTVALUE twice, we could just call the procedure once for each variable. If any records on the SCIF change, adding or removing a single procedure call rather than removing both instances of PUTVALUE is easier for maintenance, especially with larger surveys. It also means changing a single procedure if extra functionality needs to be added or removed instead of changing nearly every line of the script.

Listing 4. Procedure Source Code

```
prc_FormatData('r','rt1060.CASEID', Substring(RTLine[1],5,8))  
prc_FormatData('r','rt1060.CTRLNUM', Substring(RTLine[1],13,24))
```

Another design consideration we added in to our script was to make an array of strings, with each element corresponding to a single record type on the SCIF. This allows all of the data for the SCIF to be loaded and we could examine individual lines in the script if we need to do any debugging. It also makes our script more readable since a programmer can easily spot the variable(s) they are looking for when reviewing and updating the code in the future.

Listing 5. Array of Strings Source Code

```
IF Substring(In1002.OneLineRT,1,4) = '1060' THEN  
RTLine[1] := In1002.OneLineRT  
ELSEIF Substring(In1002.OneLineRT,1,4) = '2060' THEN  
RTLine[2] := In1002.OneLineRT
```

The resulting Blaise 5 Setup performs like the Blaise 4 Setup so it will be familiar to our programmers. In the MANIPULATE section the script reads in the individual lines of the SCIF and places each one in an array until it reaches the next record type 10. At that point it verifies that the Case ID on that record type 10 is a different case ID than what is already held. If it is different then the script calls a procedure to load the variables into the database. Inside the load procedure, we create an empty Blaise database and then call a procedure that takes each variable from the SCIF and uses PUTVALUE to place them in the appropriate instrument variables. Once the load procedure has gone through all of the variables, it empties out the array and starts reading the next set of records and loading them into the next database. This process is repeated until it reaches the end of the file. This results in a set of individual Blaise databases and data interface files

Figure 2. Blaise 5 Databases created by the new Setup script. Case ID's 22TB001N, 22TB002N, and 22TD001N.

Name	Date modified	Type	Size
22TB001N.bdbx	8/9/2016 4:10 PM	BDBX File	7 KB
22TB001N.bdix	8/9/2016 4:10 PM	BDIX File	5 KB
22TB002N.bdbx	8/9/2016 4:10 PM	BDBX File	8 KB
22TB002N.bdix	8/9/2016 4:10 PM	BDIX File	5 KB
22TD001N.bdbx	8/9/2016 4:10 PM	BDBX File	8 KB
22TD001N.bdix	8/9/2016 4:10 PM	BDIX File	5 KB

You can view a snippet of the Setup script code in Appendix A.

One major advantage to the Blaise 5 Setup is how fast it runs when compared to the Blaise 4 script. Using a test input file that creates 74 cases, the Blaise 4 Setup takes approximately 1 minute and 10 seconds before it finishes executing. The Blaise 5 Setup takes approximately 12 seconds. If we extrapolate this speed difference to a number of survey cases for a national survey, like 10,000, it will take approximately 2 hours and 15 minutes for the Blaise 4 Setup to process those cases. The Blaise 5 Setup should take approximately 27 minutes to process the same number of cases. This very large difference will have an impact on the speed of our processing systems. As we move further with testing Blaise 5 instruments we will know more about how much of an impact it will have.

4.1 The Blaise 4 to Blaise 5 Setup Conversion Script

We have the tools available in Blaise 5 to convert our Blaise 4 datamodels, data files, and even Manipula scripts, which saves a lot of time in converting instruments to Blaise 5. However, since we were changing a large portion of the code used in the Setup script, we built our own Blaise 4 to Blaise 5 Manipula conversion script written in Blaise 4 Manipula. This script reads in a Blaise 4 Setup script

and turns it into a Blaise 5 compatible script. With each record type defined as their own block, we used the GETFIELDINFO to find the name of each block, the name of each variable in the block, and the size of each variable as they are needed for the Blaise 5 Setup.

The conversion script starts by creating a setup.manx file for output. It prompts the user for the number of dependent data records that are needed for their script since these vary by survey.

Figure 3. User prompt for the number of 8500 records in the current SCIF.



Next, a “header” that contains the first lines of the Setup script is added, including defining an INPUTFILE and OUTPUTFILE. This continues until it reaches what would be the loading procedure for the Setup script. After that our conversion script searches the Blaise 4 database for any blocks that start with “BRT” or “BRT85” as those are standard names for our record type blocks in our instruments. When it finds the block, it stores the block name into an array of names. After that search is done the script calls a procedure to pull the name of the variables and characteristics of each variable for each block that was stored in the array. As it pulls each name, the procedure writes out the characteristics with other string constants to produce the lines that call the procedure to copy data into the Blaise 5 database. Once all of the fields have been written to the new script, the “footer” is written. The “footer” contains the rest of the Setup script, including the MANIPULATE section.

Blaise 4:

Listing 4. Blaise 4 Source Code

```
Outfile.COLLECT_START      := In2561.Rt2561in.COLLECT_START
Outfile.COLLECT_SALES      := In2561.Rt2561in.COLLECT_SALES
Outfile.COLLECT_COMP       := In2561.Rt2561in.COLLECT_COMP
Outfile.COLLECT_MCDNOTES   := In2561.Rt2561in.COLLECT_MCDNOTES
Outfile.COLLECT_FRREQUEST  := In2561.Rt2561in.COLLECT_FRREQUEST
```

Blaise 5:

Listing 5. Blaise 5 Source Code

```
prc_FormatData('r','rt2561.COLLECT_START',
Substring(RTLine[3],1361,1))
prc_FormatData('r','rt2561.COLLECT_SALES',
Substring(RTLine[3],1362,1))
prc_FormatData('r','rt2561.COLLECT_COMP',
Substring(RTLine[3],1363,1))
prc_FormatData('r','rt2561.COLLECT_MCDNOTES',
Substring(RTLine[3],1364,1))
prc_FormatData('r','rt2561.COLLECT_FRREQUEST',
Substring(RTLine[3],1365,1))
```

The conversion script removes a lot of the time required to convert our Setup script from Blaise 4 to Blaise 5, though the user does still have to enter some variables manually. We left off pulling the dependent data variables and placing them into the Setup script at the time of the conversion because

not all of our surveys copy the dependent data into their own blocks at the datamodel level of the instrument. Most of our surveys directly copy dependent data into their matching variable in the instrument so it makes more sense to have the user supply the fully qualified variable names as they can pull those from the Blaise 4 Setup script.

4.2 Setup Script Challenges

One of the immediate challenges that was noted earlier in this section was being able to replicate some of our Blaise 4 script functionality in Blaise 5. Manipula is a fairly recent addition to the Blaise 5 and much like Blaise 5, it is being completely overhauled to take advantage of new technologies and additional capabilities that are now available in Blaise 5. Since it is a work in progress, not everything that we had in Blaise 4 Manipula is available.

4.2.1 Maniplus Not Available

One significant feature not available yet in Manipula is Maniplus. This means that commands such as RUN and FILEEXISTS are not available for us to call out to processes that rename files. Besides these commands, many of the Manipula scripts that we use in our survey instruments to perform functions like calling the instrument, sampling, unduplication of data, and some navigation within the instruments are not available for use in Blaise 5 yet. This affects a number of our surveys that use these Maniplus scripts.

4.2.2 Block to Block Copy

As mentioned earlier in “Creating a Blaise 5 Setup,” the current version of Manipula does not allow for a block to block copy. We were able to find a workaround by using procedures to tell the Setup script to directly place each variable value into the record type blocks in our instrument. We could simplify our Setup script if we had the ability to use a block to block copy since only one line of code would be needed to copy all of the values into the block versus a line of code for each variable. It would shorten our script as well, making it easier to maintain.

4.2.3 Calling Cases Individually

A challenge that is currently under research is the ability to call a specific case (individual BDB) to run and store data. Blaise 5 surveys expect to connect to a central database to retrieve and store their data, with each case being a record in this central database. We have noticed that when we call the DEP, it expects to find one database to connect and retrieve its record, and we cannot specify which database file that Blaise 5 should connect to in order to retrieve the form and run the case. There are a few potential options that we could consider to work around this, including creating a single Blaise database on each laptop as a placeholder where each case could be placed. That option may or may not be available since it creates the potential requirement that all of our control systems and our case management software would need to be updated to handle setting up one Blaise database. Another option would be to have a central database on a server where laptops could connect and retrieve individual cases. However, that is not entirely feasible as we do need our surveys to run in offline (disconnected) mode due to the potential for interviews occurring in areas that have no steady connection to the Internet. We will continue to investigate options for integrating Blaise 5 into our existing control systems.

4.2.4 Other Commands Not Available

At the time of originally coding the new Setup script (5.0.4.875) items like the ALTERNATIVE section and AUTOREAD were not available. Some of these missing features have appeared in Blaise 5.0.5.950 and more will be added. However, our finding of alternatives such as reading each line of the SCIF into an array and parsing the array to load data has resulted in a leaner script that is more efficient. As Manipula is enhanced for Blaise 5, we may find even better ways to run our Setup than what we have now.

5. Converting Other Manipula Scripts

There are three other standard Manipula scripts used in our CAPI interviewing process. These are known as our Case Management In, Case Management Out, and CAPI_trans scripts. The Case Management In script takes information passed from our Case Management software such as the respondent's contact information and updates the appropriate variables in the instrument. The Case Management Out script does the opposite in that it passes information such as updated respondent contact information and outcome codes from the instrument back to our Case Management software. The CAPI_trans script is what calls the Case Management In and Out scripts along with calling the instrument or updating instrument data based on the transaction code that is passed by our Case Management software.

With the success in converting the Setup script, we decided to convert the Case Management In and Case Management Out scripts to Blaise 5 so that we could test their functionality. The Case Management In script is like a simpler version of the Setup script. It receives an ASCII file that is much like a SCIF except all data is on a single line and there is no record type needed. It also opens an existing Blaise database so a new database does not need to be created on the fly; it just has to update the appropriate variables in the instrument. We were able to take the same approach used in our Setup script in that we read in our input file, parse out the data in the appropriate position, and update the instrument variables. In the future, we plan to use an XML file instead of an ASCII file for this process.

The Case Management Out script is still a work in progress. We have been able to create a script that compiles and runs with our Blaise 5 instruments, but it only outputs the XML tags that Case Management uses; it does not output any of the instrument data. Only blank spaces are output where the data would go. We are researching to determine if this is due to the script not actually pulling the data from the case or if it is not writing it to the output file. We are confident we will figure out how to retrieve the data from a Blaise 5 database and put it into a custom output file.

The CAPI_trans script currently needs some of the functionality that is only available in the Blaise 4 Maniplus, such as the CALL command to call other Manipula scripts like the Case Management In and Out scripts and the EDIT command to run the DEP. We are researching how to work around those limitations.

6. Future Plans

We have had some great success so far with our research into Blaise 5 Manipula and converting our scripts, but we are just getting started. We have proven so far that we can make our scripts work and potentially work better, but we always like to push it further to get the most out of Blaise and Manipula.

In regards to the Setup script, there are a few surveys that do things a little differently that will need to be updated if we are going to be able to move them into Blaise 5. For example, the SCIF for the Consumer Expenditures Quarterly (CEQ) survey has cases that have multiples of the same dependent data records, such as for families with multiple vehicles or who pay multiple utilities. Per the current design of the Blaise 5 Setup script, if we had a household with two vehicles it would write the first vehicle into the array and then we would overwrite that array element with the second vehicle. So for CEQ's Setup script, we may modify it to capture the full record type for each used element in the dependent data array and use that to call a procedure to pick the set of instrument variables to load the data. Our Blaise 4 to Blaise 5 Setup conversion script will also be modified to collect some additional information since it may be possible to automate some of the dependent data in the Setup.

A number of our surveys have Manipula scripts that are called during a DEP session. These include scripts that perform sampling functions or that do data quality checking such as finding duplicate data. Those scripts will need to be converted and tested to make sure they retain the same functionality that

they had in Blaise 4. It is expected that we will be able to add new functionality and our scripts will continue to evolve as Manipula evolves.

7. Conclusion

Manipula is a valuable tool in Blaise as it allows us to extend our capabilities with our survey instruments. The Census Bureau has been able to take advantage of its features for years and we continue to discover new ways to improve our instruments using Manipula. As new features are added to the Blaise 5 Manipula there will be new possibilities to try something different and learn more about what our survey instruments can do. The future of our instruments and using Manipula reflect the hard work that has already gone in to creating and researching what Manipula can do now. We have had success so far in our Blaise 5 Manipula research and there is still work to be done. It makes for interesting times as survey instruments and Blaise evolve.

8. Acknowledgements

The author would like to acknowledge the work and knowledge of Mecene Desormice as his assistance in converting Setup scripts from Blaise 4 to Blaise 5 along with converting other Manipula scripts was invaluable in our research into using Blaise 5 Manipula.

The views expressed in this paper are those of the authors and not necessarily those of the U.S. Census Bureau.

9. Appendix A - Blaise 5 Setup Script Code Snippet

```
//Format and load the data into the instrument
PROCEDURE prc_FormatData
  PARAMETERS IMPORT  iFlg,iPath,iVal: STRING
  AUXFIELDS iPos, iLen : INTEGER
  Mydata : STRING

  INSTRUCTIONS
    Mydata:= Empty
    Mydata := Trim(iVal) //Trim all trailing spaces
    IF iFlg = 'r' THEN
      iPos := position('.',iPath) //Find the position of the
variable name
      iLen := LEN(iPath)
      //load variable at the datamodel level
      Outfile.PUTVALUE(SUBSTRING(iPath,(iPos+1), (iLen-iPos)+1),
Mydata)
      //load variable at the RT block level
      Outfile.PUTVALUE(iPath, Mydata)
    ELSE
      Outfile.PUTVALUE(iPath, Mydata)
    ENDIF
  ENDPROCEDURE

PROCEDURE prc_Load
  INSTRUCTIONS
  //Open a bdbx or create it if it doesn't already exist
  Outfile.OPEN(Substring(RTLine[1],5,8) + '.bdbx')
  Outfile.INITRECORD
  //Call the procedure to load all of the data
  prc_FormatData('r','rt1060.CASEID', Substring(RTLine[1],5,8))
  prc_FormatData('r','rt1060.CTRLNUM', Substring(RTLine[1],13,24)
  ...
  prc_FormatData('r','rt2060.ADDR1', Substring(RTLine[2],5,54))
  prc_FormatData('r','rt2060.ADDR2', Substring(RTLine[2],59,54))
```

```

    ...
    //Write the data to the instrument and close the database
    Outfile.WRITE
    Outfile.CLOSE
ENDPROCEDURE

MANIPULATE
    //Read the next line of the input file
    In1002.READNEXT
    Verify_ID := Substring(In1002.OneLineRT,5,8)
    //Read each line of the file until you hit the end of the file
    WHILE NOT (In1002.EOF) DO
        //Match to the record type and fill the appropriate array
        element
            IF Substring(In1002.OneLineRT,1,4) = '1060' THEN
                RTLine[1] := In1002.OneLineRT
            ELSEIF Substring(In1002.OneLineRT,1,4) = '2060' THEN
                RTLine[2] := In1002.OneLineRT
            ...
            ENDIF
        //After reading into array, read the next line
        In1002.READNEXT
        //If the next line starts with 10, it's a new case
        IF Substring(In1002.OneLineRT,1,2) = '10' THEN
            IF Substring(In1002.OneLineRT,5,8) <> Verify_ID THEN
                //Load the data stored in the array into the instrument
                prc_Load
            //Update the verify ID so it only reads in the current case
                Verify_ID := Substring(In1002.OneLineRT,5,8)
            ENDIF
        ENDIF
    ENDWHILE // NOT (In1002.EOF)

```

Adventures of a Blaise 5 API jockey

Rod Furey, Statistics Netherlands

1. Abstract

The Blaise 5 Application Programming Interface (API) can be used to access the various underlying objects of the Blaise 5 system in various programming languages. This allows us to manipulate and present these objects to our own advantage. This paper discusses various cases where Blaise 5 API calls have been used in production systems and testing tools.

2. Manga

Long, long ago, back in the mists of time, Manipula for Blaise 5 didn't exist. Alas there was a requirement from a client who wanted to be able to read incoming data and write it to a Blaise 5 file (.bdbx).

The obvious solution for this was to write a simple program which would let someone give the names of the input file and output file as parameters and then do the work for them, but there's no fun in that. Given that Manipula was at that time a long way off, a more generic solution was needed and so Manga was born.

2.1 Manga v1

Manga has had various incarnations. The original variant for the client handled basic export functions and some displays using a syntax that is familiar to anyone who has used Manipula:

Listing 1. Manga Source Code

```
Manga BlaiseToAscii 'Blaise to Ascii'
USES
    datamodel flight 'Flight'
ENDUSES
INPUTFILES
    iBlaiseFlightIn : flight : BLAISE : "C:\CBSTEMP\USER\Flight.bdbx"
ENDINPUTFILES
OUTPUTFILES
    iAsciiFlightOut : flight : ASCII
ENDOUTPUTFILES
UPDATEFILES
ENDUPDATEFILES
AUXFIELDS
    result : resultok
ENDAUXFIELDS
ELEMENTS
ENDELEMENTS
BODY
// Output an indication that we've started...
conswriteline ("Copying existing Blaise file contents to new Ascii
file...")

// We don't have the appropriate files for the Ascii file at the moment so
// we need to generate them (this sets up the fn.txt and fn.bdix files).
result := iAsciiflightOut.Create("C:\CBSTEMP\USER\FlightOut.txt")
result := BlaiseToAscii(iBlaiseFlightIn,iAsciiflightOut)

consolewriteline ("Finished!")
ENDBODY
```

There were (and still are) a number of restrictions on this, mostly due to the parser. However, it did the job. In this case an API call is used to create the Blaise Interface (.bdix) file. This is followed by the appropriate API calls to the DataLink interface to copy the data from one file to another.

Obviously if Blaise to ASCII is achievable, so is ASCII to Blaise. Equally, given a .bdix that points to an XML file, that is also available for copy actions. Even using SQL Server as a backing store for the Blaise 5 data works.

2.2 Manga v2

Armed with some basic copy actions, a decision was taken to add some extra operations. Having written the first version of Manga as an interpreter, the decision was made to write a compiler for version 2. This version contained a first pass at an expression parser, an arithmetic expression evaluator and a string concatenator.

Implemented as a two-pass compiler, the first pass decomposed the Manga source and generated a list of macro calls and parameters to handle such things as string concatenation, branching, displaying items, arithmetic operations etc. This list of macros was then read and the appropriate Common Intermediate Language (CIL) statements written away to a file which was later fed (by hand) into the Intermediate Language assembler (ilasm.exe). The resulting executable handled these extra functions with aplomb.

2.3 Manga v3

The next step was to add some basic manipulation of the Blaise fields. Given that debugging this was going to be easier if an interpreter was used, the decision was made to adapt the work from Manga v2 (parser, evaluator etc.) so that it could be used in the interpreter from Manga v1.

To facilitate the reading of Blaise fields, a series of API calls was used to open the compiled datamodel file (.bmix) and retrieve the names and types of the various fields. This resulted in some interesting discussions with the Blaise group about the correct call to use and why some of them didn't work as advertised. Coding problems on both sides were analyzed and bugs fixed with the result that read, write/update and delete functions could be carried out via the API calls which retrieved the field, read or assigned its value and read, wrote/updated or deleted a record from the file.

2.4 Manga v4

The final step in the evolution of Manga has been to add boolean algebra and loops to the interpreter.

2.5 Results

Manga has helped a couple of projects a number of times with regards to extracting data from Blaise 5 data sources and writing that data to disk for analysis or debugging purposes. It has also caught a number of bugs in the Blaise 5 codebase which have subsequently been corrected. Manga is still used today as an extended test case for new releases of Blaise 5 in order to trap any regressions that may have occurred in the various API calls that it uses.

3. Login

The Production Statistics questionnaire was chosen for Statistics Netherlands' first production Blaise 5 questionnaire. This questionnaire is protected by a login gateway which uses a userid and password pairing for identification. This gateway basically reflects the login sample which is given in the Blaise 5 help but has to implement some extra requirements from the business. Two of these requirements are:

- do not allow more than one person to be logged in at once

- allow someone else from the respondent's company to "steal" the login after an hour of inactivity

The login questionnaire utilizes an ALIEN call to a C# module in order to satisfy these conditions. When someone logs in, a flag is set in an SQL Server database. This flag is checked on every login attempt to see whether or not login is allowed. A call to the Blaise 5 Session API is made which retrieves any session information for the appropriate primary key of the questionnaire. If any session data exists then a check is made on the *<session>.Creation* property and the *<session>.LastModification* property to see if they were more than one hour ago. If this is the case the new login attempt is flagged as being allowable (it must pass other requirements as well before being carried out).

Stripping out all the exception and error handling, the code for this check effectively simplifies to:

Listing 2. Login Source Code

```
using SDataAPI = StatNeth.Blaise.API.SessionData;
...
SDataAPI.IInstrumentSessionInfo _isi =
    SDataAPI.SessionDataManager.GetInstrumentSessionInfo(
        Guid.Parse(<instrument-id-of-associated-instrument>),
        <server-park-name>
    );

SDataAPI.ISessionInfo _session = null;

// Get the datamodel (is needed to build the primary key):
MetaAPI.IDatamodel dm =
    MetaAPI.MetaManager.GetDatamodel(<path-to-the-appropriate-datamodel>);

// Get an IKey interface for the primary key:
DRecAPI.IKey _primaryKey =
    DRecAPI.DataRecordManager.GetKey(dm,
        MetaAPI.Constants.KeyNames.Primary);

// The key consists of just 1 field, userid:
_primaryKey.Fields[0].DataValue.Assign(<userid>);

_session = _isi.Read(_primaryKey);

if (_session == null)
{
    _rc = true;
}
else
{
    if ((_session.Creation.AddHours(1d) <= DateTime.Now)
        && (_session.LastModification.AddHours(1d) <= DateTime.Now))
    {
        _rc = true;
    }
    else
    {
        _rc = false;
    }
}
...
return _rc;
```

4. BISDar

One of the items that has come up on the "Wouldn't it be nice if...?" list is a route checker. There are times when fields disappear off the display or are set to unexpected values when running a questionnaire. Despite extensive pre-release testing, a respondent may fill in a combination of answers that manages to produce an edge case. A structured test method would be nice to have.

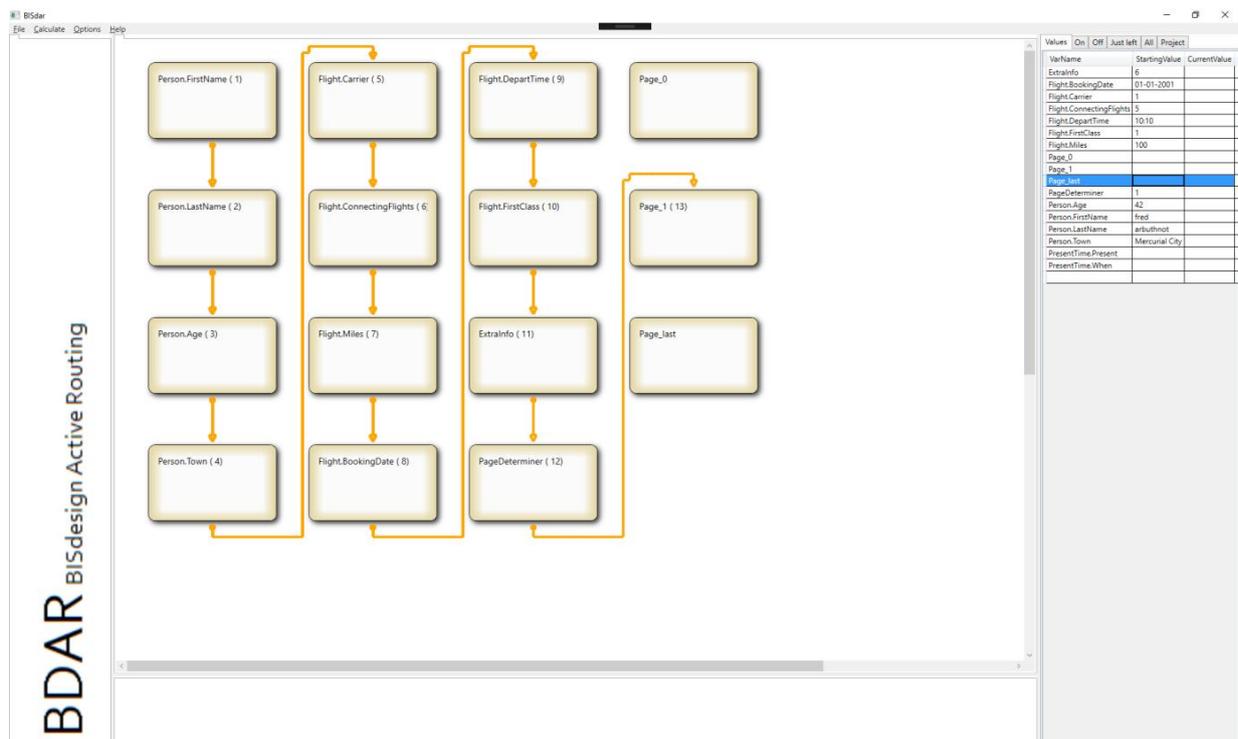
To add an extra layer of testing, BISdar was written. BISdar uses the Blaise 5 API to not only extract the fields and their definitions but also which page the field would be expected to appear on (work in progress; an API call to ensure that this is the same order as the preview has been requested but in the meantime, a small program has had to be written). This is then presented to the questionnaire designer who then has the ability to set various values for the fields and then execute the rules (again, via calls to the Blaise 5 API).

Recently a new display option has been added. Instead of displaying the fields on a page by page basis, the fields are displayed individually and the current route through the questionnaire is shown by drawing arrows between the display boxes of the fields that are on the route. This is far more useful and the original drawing function has now been deprecated.

After the rules have been executed, the status of each field is read and an indication shown in various list views as to whether it is on the route, has just gone off the route in the last execute, or has been taken off the route before that. This latter information is kept locally in the program.

Sets of values can be saved and loaded back and the project saved to disk.

Figure 1. BDAR

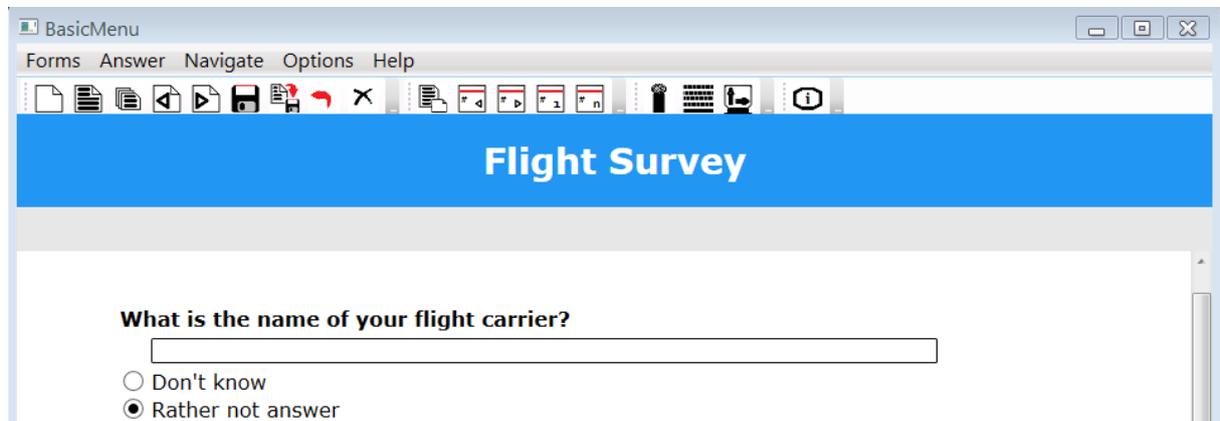


5. BasicMenu

Back in the Blaise 4.8 Data Entry Program (DEP) there is a menu that lets people execute various actions such as displaying all remarks or entering a value of Refusal, amongst others. This menu bar isn't in Blaise 5. The solution? Create a custom WPF DEP and a custom control and use the DataEntry API to recreate the menu bar.

The Data Entry framework utilises the DataEntry, DataEntryWpf, DataEntry.Controls and DataEntry.DataObjects DLLs. The processing code uses the DataEntry, DataLink, DataRecord, Meta, SessionData and ServerManager DLLs to, amongst other things, interrogate the compiled datamodel and session record, execute actions, read records, retrieve the fields on the route, assign values to fields and talk to the server park.

Figure 2. BasicMenu Custom Dep



6. Conclusion

The Blaise 5 Application Programming interface allows access to items from the Blaise 5 milieu from various languages. As discussed, these facilities allow the programmer to create various tools or even other languages to help solve problems in their domain.

It should be noted however, that Blaise 5 is in some respects vastly different in its implementation details from Blaise 4. This affects, for example, the menu bar in subtle and not so subtle ways. One example of this is that the instrument to be queried is expected to be installed in the server park. A bigger difference is that the system-wide implementation of a session means that when navigating between forms, the session information comes back as well which can mean that, for example, the language that is in use in the session can differ from the language that was last set by the user.

However, once these variations in behaviour are taken into account, designing tools such as those discussed becomes a simple matter of programming.

7. And finally...

My serious introduction to Blaise occurred a while ago when I was hired in to work closely with Gerrit de Bolster whose enthusiasm for and knowledge of Blaise is contagious. It has been a pleasure and a privilege to work with him these last 4 years and without him I wouldn't be presenting at the IBUC.

The Blaise 5 Champion Instrument Suite

Mark M Pierzchala, MMP Survey Services, LLC, United States

1. Abstract

A suite of ten demonstration and survey instruments will be part of the system distribution of Blaise 5. The suite is called the Champion series because it features polished instruments that show off Blaise 5 capabilities. Six of these are called B5 instruments. They have the names *B5Basic*, *B5Modest*, *B5Complex*, *B5Scales*, *B5PanEuropean*, and *B5CodeFrame*. They each show a specific range of features. For example, the *B5Scales* instrument shows how various kinds of scale questions can be handled.

There are four realistic survey instruments including *NCSPerson*, a commuter survey in ten languages. The *Trade* survey is based on a Dutch financial survey. It shows non-linear section navigation. There is the *Census* instrument for data collection on mobile devices that is in English and Spanish. Finally, there is the *Annual Survey of Industry*, a North American survey for use in Canada, the United States, and Mexico.

All of the Champion instruments are multimode. They use a common Resource Database. The Layout Designer of each has eight Layout Sets that work for different modes on different devices. New source code keywords are used such as `MODES`, `ROLES`, `SPECIALANSWERS`, and `SPECIALANSWERSETS`.

The range of devices that are targeted includes smart phones, tablets, and computers; the devices use native apps or Dep, or their browsers. The Champion.blrd Resource Database can be used out-of-the-box for your institute. You can also use it as a good start to make your own Resource Database. The Champion Instruments form a test suite of instruments and questions that institutes can use in their tests.

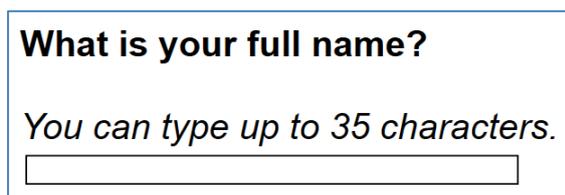
2. Description of Instruments

This section briefly describes selected instrument with a few images from each. All images are from the full-size self-administered browser layout. Pierzchala (2016) describes the uses of the Champion instruments as a test bed for an institute's own standards setting exercise. All Blaise 5 source code, the Resource Database, and other related files will be part of the release.

2.1 B5Basic

The B5Basic instrument demonstrates all the Blaise elementary question types. These include integer, real, string, open, enumerated, set, date, and time question. There are also a few quantity/unit groups. By design, this instrument should easily display correctly on all devices and screens sizes.

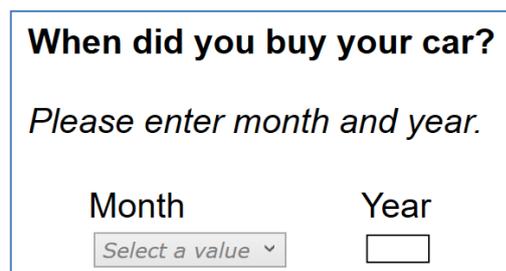
Figure 1. A Basic String Field



What is your full name?

You can type up to 35 characters.

Figure 2. A Quantity/Unit Pair of Fields



When did you buy your car?

Please enter month and year.

Month Year

Figure 3. A Basic Enumerated Field

What is the highest level of education you have attained?

Choose one answer.

- Elementary school
- Some high school
- Graduated high school
- Some college
- Graduated college
- Some graduate school
- Masters or higher degree

2.2 B5Modest

The B5Modest instrument increases the complexity of the questions. For example, Figure 4 shows a lengthier enumerated field that challenges display on smart phones. The question is long, the instruction is long, there are nineteen categories, and some of the choice texts are long. It is also a real survey question. Such a question is called a *stressor* because it stresses the Blaise 5 system in order to improve it.

Figure 4. A More Complex Enumerated Field

We need to assign a standardized educational code to your field of study.

There are 19 categories. Select one that best describes your field of study.

- Agricultural Sciences
- Biological or Life Sciences
- Business Management and Administrative Sciences
- Computer and Information Sciences
- Conservation and Natural Resources
- Education
- Engineering and Engineering-Related Technologies
- Health and Related Sciences
- Languages, Linguistics, Literature or Letters
- Liberal Arts or General Studies
- Library Science
- Mathematics and Statistics
- Physical Sciences
- Philosophy, Religion or Theology
- Psychology
- Social Sciences or History
- Social Work
- Visual or Performing Arts, or
- OTHER Field Not Listed

Additionally, the *B5Modest* instrument shows more complex question structures such as name and address collections (Figure 5) and other specify groups (Figure 6).

Figure 5. A Job Title and Name Collection

Please enter your job title and name.

Job title

First name *

Middle name

Last name *

Suffix

Figure 6. An Other-Specify Group

? Which kinds of employment have you had?
Choose all that apply.

Self employed

Government (exclude military service)

Military service (exclude civilian military)

Private profit-making company

Non-profit organization

Other (please specify)

2.3 B5Complex

The *B5Complex* instrument features complex groups of questions, usually tables (Figures 7 and 8).

Figure 7. A Job Title and Name Collection

	Name	Gender	Age	Relationship to you
Person 1	<input type="text"/>	<input type="radio"/> Male <input type="radio"/> Female	<input type="text" value="Select a value"/>	
Person 2	<input type="text"/>	<input type="radio"/> Male <input type="radio"/> Female	<input type="text" value="Select a value"/>	<input type="text" value="Select a value"/>
Person 3	<input type="text"/>	<input type="radio"/> Male <input type="radio"/> Female	<input type="text" value="Select a value"/>	<input type="text" value="Select a value"/>
Person 4	<input type="text"/>	<input type="radio"/> Male <input type="radio"/> Female	<input type="text" value="Select a value"/>	<input type="text" value="Select a value"/>

Figure 8. A Battery of Yes/No Questions

Thinking back to the time you took community college courses, for which of the following reasons did you take those courses from a community college? Was it . . .

	Earn credits	
	Yes	No
To earn college credits while still attending high school?	<input type="radio"/>	<input type="radio"/>
To complete an associates degree?	<input type="radio"/>	<input type="radio"/>
To prepare for college / increase chance of acceptance to a 4-year college or university?	<input type="radio"/>	<input type="radio"/>
To earn credits for a bachelor's degree?	<input type="radio"/>	<input type="radio"/>
For financial reasons, for example because of the cost of a 4-year school?	<input type="radio"/>	<input type="radio"/>
To gain further skills or knowledge in your academic or occupational field?	<input type="radio"/>	<input type="radio"/>
To facilitate a change in your academic or occupational field?	<input type="radio"/>	<input type="radio"/>
To increase opportunities for promotion, advancement or higher salary?	<input type="radio"/>	<input type="radio"/>
For leisure or personal interest?	<input type="radio"/>	<input type="radio"/>
For some other reason?	<input type="radio"/>	<input type="radio"/>

2.4 B5Scales

The B5Scales illustrates how different scales are handled (Figures 9 and 10).

Figure 9. Phrase Completion Scales

I have faith that the local sports club will win the championship this year.

No faith at all 0 1 2 3 4 5 6 7 8 9 Complete faith 10

I have faith that the local sports club will win the championship in the next five years.

Complete faith 10 9 8 7 6 5 4 3 2 1 No faith at all 0

Figure 10. Some Likelihood Questions

With respect to your research career, how likely is it that you will . . .

	Likelihood			
work at this institution most of your career?	<input type="radio"/> Not likely	<input type="radio"/> Somewhat likely	<input type="radio"/> Very likely	<input type="radio"/> Not sure
conduct research as your main job responsibility?	<input type="radio"/> Not likely	<input type="radio"/> Somewhat likely	<input type="radio"/> Very likely	<input type="radio"/> Not sure
remain in your field of research specialty?	<input type="radio"/> Not likely	<input type="radio"/> Somewhat likely	<input type="radio"/> Very likely	<input type="radio"/> Not sure
work in private industry?	<input type="radio"/> Not likely	<input type="radio"/> Somewhat likely	<input type="radio"/> Very likely	<input type="radio"/> Not sure
work in an academic institution?	<input type="radio"/> Not likely	<input type="radio"/> Somewhat likely	<input type="radio"/> Very likely	<input type="radio"/> Not sure

2.5 NCSPerson

The NCSPerson instrument is a commuter survey. It is programmed in English, Dutch, French, Spanish, Greek, Hebrew, Arabic, Hindi, Chinese, and Japanese. It is meant to be used in fifteen different countries. Figures 11 through 14 show the same set of questions in four languages.

Figure 11. Questions in English

Give a short description of your job with MMP Survey Services, LLC at this location.

What is the distance from home to work? **Unit of measurement?**
You can enter a number with up to 1 decimal place.

Miles
 Kilometers

How do you travel to this work place?
Give up to 4 answers.

Bus Car or van pool
 Tram or trolley Motorcycle
 Subway/metro, elevated light rail Bicycle
 Train, not light rail Walk
 Car by yourself

Figure 12. Questions in Chinese

请简单描述一下您在 MMP Survey Services, LLC 的这个工作地点所从事的工作。

从您家到上班地点有多远？
您可以输入带有一位小数的数字。

计量单位
 英里
 公里

您是如何到上班地点的？
可以提供最多4个答复。

公交车 拼车
 电车，有轨车 摩托车
 地下铁，高架轻轨铁路 自行车，脚踏车
 火车 步行
 自家车

Figure 13. Questions in Hindi

आपके काम के बारे में एक संक्षिप्त विवरण MMP Survey Services, LLC के साथ इस स्थान के लिए दें।

आपके घर से आपके कार्यस्थल का दूरी क्या है?
आप संख्या के साथ 1 दशमलव स्थान तक दर्ज कर सकते हैं।

माप की इकाई
 मील
 किलोमीटर

आप इस काम के स्थान के लिए यात्रा कैसे करते हैं?
चार जवाब तक लिखें।

बस कार या वैन पूल
 ट्राम या ट्राली मोटर साइकिल
 सबवे / मेट्रो या ऊंचा हल्की रेल साइकिल
 रेलगाड़ी पैदल
 अपने आप से कार

Figure 14. Questions in Hebrew – a Right-to-Left Language

לתת תיאור קצר של העבודה שלך עם מעביד במיקום זה

יחידה של מדידות
 מייל
 קילומטרים

מה המרחק שלך מהבית לעבודה?
אתה יכול להזין מספר עם מקום עד אחד עשרוני

איך אתה נוסע למקום העבודה הזה

נסיעה משותפת אוטובוס
 אופנוע חשמלית או עגלה
 אופניים רכבת תחתית / מטרו או רכבת קלה עילית
 ללכת רכבת
 מכונית בעצמך

2.6 Census

This instrument is modelled on a census-type questionnaire. Some question structures are very complex. Figure 15 shows a set (code-all-that-apply) question structure where every choice has a specify field attached.

Figure 15. An Elaborated Set-Specify Question Structure

What is Mark Pier's race or origin? Select one or more boxes, AND enter the specific races or origins.	
<input type="checkbox"/>	White - Enter origin(s), for example, German, Irish, English, Italian, Lebanese, Egyptian, and so on.
	<input type="text"/>
<input type="checkbox"/>	Hispanic, Latino or Spanish origin - Enter origin(s), for example, Mexican or Mexican-American, Puerto Rican, Cuban, Dominican, Salvadoran, Columbian, and so on.
	<input type="text"/>
<input type="checkbox"/>	Black or African Am. - Enter origin(s), for example, African American, Jamaican, Haitian, Nigerian, Ethiopian, Somalian, and so on.
	<input type="text"/>
<input type="checkbox"/>	Asian - Enter origin(s), for example, Chinese, Filipino, Asian Indian, Vietnamese, Korean, Japanese, and so on.
	<input type="text"/>
<input type="checkbox"/>	American Indian or Alaska Native - Enter name of enrolled or principal tribe, for example, Navaho Nation, Blackfeet Tribe, Muscogee (Creek) Nation, Mayan, Doyon, Native Village of Barrow Inuplat Traditional Government, and so on.
	<input type="text"/>
<input type="checkbox"/>	Native Hawaiian or Other Pacific Islander - Enter origin(s), for example, Native Hawaiian, Somonoam, Guanamian or Chomorro, Tongan, Fijan, Marshallese, and so on.
	<input type="text"/>
<input type="checkbox"/>	Some other race or origin- Enter race(s) or origin(s).
	<input type="text"/>

3. References

Pierzchala, M. (2016). Blaise 5 – Is Worth the Wait. Paper presented at the 2016 International Blaise Users Conference, The Hague, The Netherlands, 2016.