

The use of Blaise in Organisations

The Challenge in Balancing Data Collection Innovations, Remaining Practical, and Being Cost-Effective	3
Use of Blaise in Today’s Survey Research Environment	14
The Centralized Survey Experience at National Agricultural Statistics Service	23

Paradata

Using Computer Audio Recorded Interviewing (CARI)	31
Blaise Audit Trail Data in a Relational Database	38
Paradata at the U.S. Census Bureau (and Where Blaise Fits In)	47
Using Paradata to Investigate Food Reporting Patterns in AMPM.....	61

Using Blaise for specific survey requirements

Cyprus Blaise Integrated Census System (CY-BICS)	69
Implementing an Office & Field Survey Management System for PIAAC using Blaise	85
Integrating Biometric Measurements in a Blaise Instrument.....	98
Multiple Overlapping Waves - Challenges in Supporting Blaise Instruments Simultaneously for Four Waves of Data Collections	103

Blaise in Mixed Modes

The Labour Force Survey, a Mixed Mode Household Survey	113
Developing and Testing A Complex Mixed Mode Questionnaire Instrument.....	131
ONS Web Development Project: Tackling the Difficulties of Social Survey Data Collection	142
Web CATI (Part of NatCen’s Multi-Mode Approach).....	153

Blaise in CATI

Blaise Multiple Contact Interface for Telephone Interviewing on a Complex Household Survey.....	160
A system allowing sequential interviewing of household members, where the household members are individual cases, within the Blaise CATI-framework.....	170

Blaise Tools

Extending Blaise Capabilities in Complex Data Collections	178
Programming and Use of Blaise to SAS Transformation Tool for Streamlining Processing in the Panel Study of Income Dynamics.....	185
New Tricks with Old Tools.....	197
Using metadata in Manipula and Manipulus.....	211

Blaise IS

Optimal Ways of Developing Blaise IS Instruments.....	220
Performance and security enhancements on the Blaise IS standard stylesheet.....	232
Blaise On-the-Go: Using Blaise IS With Mobile Devices	237

Working with Blaise Data & Case Management

Blaise Translation Challenges: Versioning, Multimode and Exporting	249
Leveraging the Capabilities of the Blaise Editor Add-In Tool to Improve the Readability of Datamodel Source Code	256
Data Transfer in Blaise Surveys with CAPI Collection Method	262
CCMS: CAPI Case Management System.....	271

Miscellaneous

Implementing Audio Computer Assisted Self Interviewing in Basil.....	282
Challenges and Lessons Learned Using Blaise IS with SQL Server	300
Servicing Blaise instrument needs in a multi-mode environment: Some technical examples.....	312

Posters

Event History Calendar program development and anticipated effects - Main functions and characteristics of the KLI CAPI-EHC -.....	321
The Application of Blaise CARI in Chinese Family Panel Studies	329
Using Audit Files To Get The Time Of Each Question (TIEQ) In China Family Panel Study(CFPS).....	334
Social Data Collection Transformation Project: Transforming CATI data collection and data processing for the Labour Force Survey.....	338

The Challenge in Balancing Data Collection Innovations, Remaining Practical, and Being Cost-Effective

Leonard Hart, Scott Reid, and Erin Slyne, Mathematica Policy Research

Presented at the 14th International Blaise Users Conference, April 2012—London, United Kingdom

Survey organizations continuously develop innovative and complex data collection designs, which in these increasingly difficult economic times have to be implemented as efficiently and cost-effectively as possible.

This paper describes how we use the built-in capabilities of the Blaise programming suite, both out of the box and integrated with custom-built applications, to meet this challenge. We also explain our plans to expand our use of Blaise in order to keep up with the ever-changing landscape of information technology (IT) and the impact it can have on our future data collection efforts.

1 Background

1.1 Philosophy on Efficiency and Cost-Effectiveness of Data Collection Efforts

Mathematica Policy Research's mission is to "... improve public well-being by bringing the highest standards of quality, objectivity, and excellence to bear on the provision of information collection and analysis to our clients."¹ We are also widely recognized for having experts in the areas of social science research and evaluation on staff; we have received numerous awards and honors for the contributions they have made in providing high quality research using innovative methods.²

However, Mathematica is more than just a mission statement and staff looking to provide high quality analytical results. It is also an employee-owned company attempting to maintain and develop long-term effectiveness and growth opportunities. To accomplish these goals, while remaining self-sufficient financially, we have to be as efficient and cost-effective as possible when it comes to our data collection efforts and using Blaise as one of our primary data collection tools.

Before discussing Blaise topics in detail, we will examine current technological trends, the U.S. political and economic landscape, and the concerns that all organizations face in these difficult economic times. Finally, we will illustrate how we use Blaise to handle these data collection challenges while still meeting our goals.

1.1.1 Remaining Competitive and Capable

As we began our research for this paper, we realized Mathematica is not alone in its concern to remain competitive while meeting the research goals of our external and internal clients, especially during difficult economic times.

A number of conference panels and town hall meetings at the Gartner Symposium/ITxpo 2010 focused on the desire of chief information officers (CIOs) to keep their operations running despite

¹ "Our Mission." Mathematica Policy Research corporate website. Available at http://www.mathematica-mpr.com/About_Us/mission.asp. Accessed February 14, 2012.

² "Awards and Honors." Mathematica Policy Research corporate website. Available at http://www.mathematica-mpr.com/About_Us/awards.asp. Accessed February 14, 2012.

reduced budgets, while also wrestling with rapidly evolving technology.¹ In his article covering this event for *eweek.com*, Nicholas Kolakowski brought attention to research notes released during the conference by Gartner analysts that broke down several primary concerns of chief executive officers (CEOs) that their CIOs should address.² Two concerns caught our attention:

1. **Fading confidence.** The global recession has apparently left many CEOs in a pessimistic frame of mind, causing them to reduce their more optimistic investments in technology and other areas. To address this concern, CIOs should assume their IT resources will either decline or remain static in 2011.
2. **Investing in new cost efficiencies.** CEOs increasingly focus on systemic efficiencies. As a result, CIOs should examine how to contribute to saving costs (that is, they should introduce automation policies).³

1.1.2 “Smarter, Faster, Cheaper”

Pessimism over the global economy and the realization that investments in systemic cost-efficiency methods are needed were two of the prime reasons the mantra of “Smarter, Faster, Cheaper” was often quoted in the past half decade. We hear this refrain recited in all business sectors, not only in areas of public policy research or IT.

IBM stated in a recent white paper, “Smarter, Faster, Cheaper: An Operations Efficiency Benchmarking Study of 100 American Cities,” that “All large organizations harbor inefficiencies. When IBM embarked on its transformation program in the early 1990s, the company eliminated \$6 billion in costs, primarily by simply being smarter about what we did and how we did it.”⁴

By being smarter, faster, and cheaper—as IBM did when facing its own economic troubles in the 1990s—companies functioning in these difficult economic times enable themselves to remain competitive in their bids to win new work while enticing existing clients to come back for more. This is something Mathematica constantly strives to achieve.

1.1.3 Cutting Budgets, Expecting Quality

The social science research community should expect to see its budgets cut, or at the least to remain frozen, for the foreseeable future. Despite the flat or reduced funding, there will be no cuts when it comes to the expectation of receiving top-quality results from our clients.

In a letter from Jeffrey Zients, acting director of the Office of Management and Budget, to Vice President Joseph Biden, in his role as President of the United States Senate, as part of the “Statistical Programs of the United States Government, Fiscal Year 2011” report, Zients stated:

“As we aim to tackle longstanding challenges in an era of scarce resources, it is especially critical that we support our ongoing efforts to provide unbiased, reliable, and timely data. Having access to quality, unbiased data allows us to make reasoned, disciplined decisions about where to target our

¹ Kolakowski, Nicholas. “CIOs Need to Address CEO Concerns in Major Ways: Gartner.” *eweek.com*, October 21, 2010. Available at <http://www.eweek.com/c/a/IT-Management/CIOs-Need-to-Address-CEO-Concerns-in-Major-Ways-Gartner-371815/>. Accessed February 14, 2012.

² Kolakowski 2010.

³ Kolakowski 2010.

⁴ Edwards, David. “Smarter, Faster, Cheaper: An Operations Efficiency Benchmarking Study of 100 American Cities.” IBM Global Business Services White Paper. Somers, NY: IBM Global Services, February 2011. Available at <ftp://public.dhe.ibm.com/common/ssi/ecm/en/gbw03132usen/GBW03132USEN.PDF>. Accessed February 14, 2012.

resources to get the biggest return for our investment, and to identify where we've been spending consistently but yielding underperforming results.”¹

The total spending in non-Census years on supporting federal statistics is just 0.02 percent of the entire United States gross domestic product.² However, the estimated budget requested for fiscal year 2010 to carry out all fiscal year 2011 statistical work involving the 13 agencies whose principal missions revolve around statistical activities, plus the 80 other agencies that carry out statistical activities as part of their program missions, was \$6.8 billion.³

Federally sponsored statistical work might account for just a small part of our overall government expenditures, but government officials are looking to cut budgets of all agencies across the board. “My committee went line-by-line through agency budgets ... to negotiate and craft deep but responsible reductions in virtually all areas of government,” said House Appropriations Committee chair Hal Rogers in a statement quoted by CNN in April 2011.⁴

Zients' statement also recognizes the push by federal agencies to gain the best possible research return on their investments. We see no evidence why privately funded foundations or individual state agencies also conducting similar research would not expect the same results in this era of scarce resources.⁵

1.2 Challenge: Implementing Efficient Data Collection Efforts in a Budget-Conscious Economy

One of the primary challenges faced by the Computer-Assisted Interviewing Support Group (CAISG) at Mathematica is handling multiple types of Blaise data collection efforts and at the same time remaining cost-conscious. We work with a wide variety of computer-assisted telephone interviewing (CATI), computer-assisted personal interviewing (CAPI), and computer-assisted web interviewing (CAWI) projects, some of which are conducted simultaneously across these multiple data collection modes. Several of these projects have also provided us with innovative and forward-thinking data collection designs, which our internal researchers and external clients have brought to our attention.

1.2.1 Meeting Expectations While Remaining Within Budgetary Constraints

Completing a project efficiently is one of the major issues computer-assisted interviewing (CAI) programmers consider when taking on a challenging Blaise-based data collection task. Ideally, we would all have as much time as necessary to troubleshoot and try out new concepts, but in the business world that is not possible and definitely not an efficient use of potentially scarce programming time. Time equals money and time not properly spent is a client's budget wasted.

Our primary goal is to persuade programmers to apply out-of-the-box Blaise solutions as much as possible. However, when we cannot do so we look to develop custom applications with an eye toward designing them with as much reusability potential built in as possible.

¹ Zients, Jeffrey D. “Statistical Programs of the United States Government, Fiscal Year 2011.” Washington, DC: Office of Management and Budget, 2010. Available at http://www.whitehouse.gov/sites/default/files/omb/assets/information_and_regulatory_affairs/11statprog.pdf. Accessed February 14, 2012.

² Zients 2010.

³ Zients 2010.

⁴ Riley, Charles. “2011 Budget Cuts Revealed.” money.cnn.com, April 12, 2011. Available at http://money.cnn.com/2011/04/12/news/economy/2011_budget_cuts/index.htm. Accessed February 14, 2012.

⁵ Zients 2010.

We might have built some of these custom applications thinking they would be feasible for only one project's purpose, but they later developed into valuable tools we have repeatedly applied to other projects, ultimately saving on new development costs. We realized such a benefit because we kept this reusability concept in mind at the start.

1.2.2 Remaining Innovative and Forward-Thinking

If all CAI projects were exactly the same, our work would be boring and devoid of challenges; but, as we all know, very few actual projects are exactly the same, in spite of what the client tells you! There are always opportunities in which we have the chance try out new data collection concepts or improve upon existing ones, as long as we find a budget-friendly way to implement them.

Mathematica's CAISG strives to remain innovative, forward-thinking, and cost-effective. One of the areas in which we have developed expertise involves the use of real-time processes. In these cases, data collected from other systems are accessed and written to by the Blaise instrument immediately, rather than through the use of an external series of off-hours processes. These real-time processes in turn facilitate the simple synchronization of data between disparate systems: for example, a Blaise instrument collecting data linked to a SQL-based sample tracking database.

We have also developed an area of expertise in programming multimode instruments with a unimode design—for example, a CATI/CAWI instrument that shares one .bla and .bdb file.

Whether it be the out-of-the-box, built-in capabilities of Blaise or integrating it with custom applications, you can balance your data collection innovations, apply your allocated and limited programming resources to the data collection tasks at hand practically, and wind up being cost-effective for your clients.

2 Benefits of Using the Built-In Capabilities of Blaise

In our quest to balance cost-efficiency with forward-thinking data collection solutions, Mathematica's CAISG has used the Actions and Events capabilities of Blaise to produce dynamic and reusable applications. Their flexibility facilitates the easy synchronization of external systems with data collected in our Blaise instruments, resulting in reduced development costs. We have also used Actions to implement a system in which one instrument updated data in a second instrument before launching it in the data entry program (DEP). Lastly, we have extensive experience programming multimode surveys using one instrument and one centralized database because of the capabilities available to us right out of the box in the Blaise programming suite.

2.1 Actions and Events

Using Blaise's Actions and Events capabilities adds valuable functionality to our survey instruments. They enable us to seamlessly integrate external systems, such as our survey management systems, with our Blaise instruments. Their capabilities also enable us to incorporate other Blaise functions, such as executing a question-by-question look-up help file into an instrument's DEP session. We have used Actions and Events to invoke reusable component object model (COM) applications and out-of-the-box Actions, including calling Manipula and Manipulus programs from within the DEP. Although Actions and Events function differently they are similarly defined.

An Event can be linked to a data type using an Event handler defined in the data model properties file (.bdp). The COM object is referenced in the Event handler and is executed when the DEP encounters the field with the associated data type. The COM object specifies when to execute the code for the associated Blaise field by using the predefined methods of the application programming interface (API): pre-edit, executed when entering the associated field; edit, executed when switching to the edit mode; and post-edit, executed when leaving the associated field. We have used Events in our

instruments to execute COM objects that invoke external applications. We are able to reuse these custom-built COM objects for multiple projects.

Actions can be linked to a data type, but unlike Events they can be linked to a menu selection in the Blaise Menu File (.bmf) and can define the end-of-parallel behavior. In addition to calling COM objects, Actions can be specified to perform traditional DEP functions, such as executing Manipula programs and closing forms. The Blaise system has 49 predefined Actions originating from the traditional DEP menu.¹ We have Actions declared in our base instrument that we use as a starting point.

Events are used if the COM object should be invoked automatically at a specific time. Conversely, Actions are not automated, but are instead invoked by the user.

2.1.1 Invoking Built-In Actions

Based on the needs of a specific project, we discovered that using Blaise's built-in Actions to link two separate Blaise instruments was the most efficient way to conduct the project's data collection. The interview was split into two sections in which a selected respondent completed one section and another household member completed the second. We have also had similar projects in which one respondent acting as a proxy for another could complete a secondary survey. We decided that because each section had potentially different respondents it would be easier to manage CATI calling if the interview was separated into two instruments. Additionally, we had to link the instruments together because the logic for the second section depended on responses from the first. We used an Action to update data in the secondary instrument and to launch it in the DEP from inside the primary instrument.

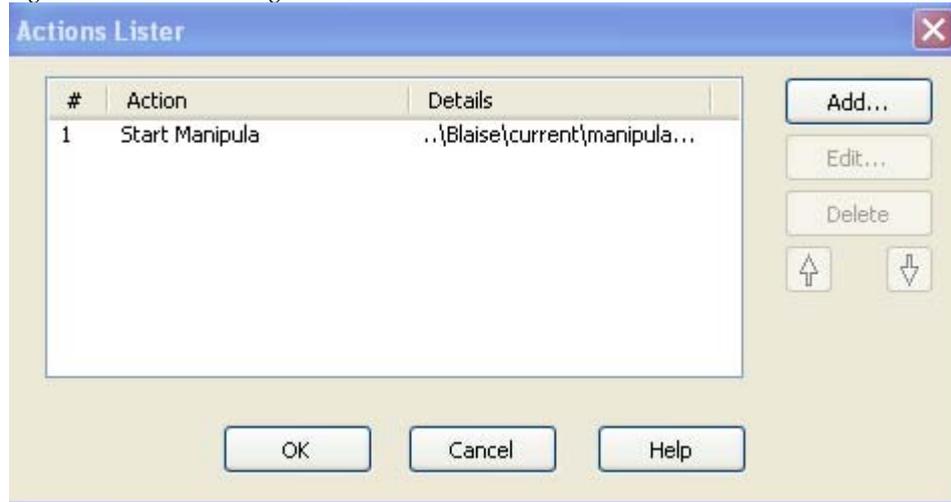
To implement the transition from the primary to the second instrument, we linked an Action to a data type that executed a Manipula program launching the second DEP. To invoke the Action, the interviewer clicks on a button located inside the input box of the field (Figure 2.1). Before launching, we defined an Action calling a Manipula program to import data from the primary instrument to the secondary instrument (Figure 2.2). Because the secondary instrument cannot function without the data from the primary instrument, we added an edit check to ensure the interviewer executed the update Action before the transition Action.

Figure 2.1. Input Box to Invoke Action



¹ "Predefined Actions." Blaise 4.8 Online Assistant. Available at <http://www.blaise.com/doc/doc4.8/index.html>. Accessed February 14, 2012.

Figure 2.2. Action Dialogs – Action Lister



2.1.2 Invoking Actions and Events Through COM Objects

In addition to invoking built-in Blaise Actions, it is possible to invoke COM objects in a Blaise instrument. Mathematica has built several COM object applications activated through Actions and Events, including a stopwatch and an error tracking system.

The stopwatch application is invoked as an Event linked to a specific data type. The application is automatically activated when the user lands on the associated field and deactivated by an Event within the COM object, in this case someone stopping the stopwatch. The timing collected by the stopwatch is captured and stored as a field in the data set.

The error tracking system links an Action with a selection in .bmf and calls an external application allowing the entire Blaise object to be available for the application to read applicable data via the Blaise API. These applications have served as a prototype launching pad for development of our larger systems.

2.2 Multimode Instrument Design

In our previous IBUC conference paper, we discussed our challenges and experiences with conducting multimode CATI and CAWI instruments sharing a centralized Blaise database.¹ Our multimode instruments use most of Blaise IS's out-of-the-box capabilities with minimal customization of Blaise's Internet Workshop settings, style sheets, and .asp pages in order to work with our authentication process.

We also use Blaise's Automatic Language Mapping capability to declare each mode as a separate instrument language. This enables both modes to share the same variables, field declarations, and logic in a single instrument. The capabilities also give the programmer the flexibility to specify unique text and logic for each mode. The end result is the ability to store all instrument data in one centralized Blaise database, thereby eliminating the need to transfer data between separate Blaise instrument databases. This single database storage also activates Blaise's case-locking mechanism so that a case can be accessed by only a single user in a particular mode. For example, a web user is not able to access a case that a CATI interviewer currently has open and vice versa. In this scenario, the locking mechanism and single database eliminates the dilemma of deciding which data should be used

¹ Hart, Leonard, Scott Reid, and Erin Slyne. "Challenges of Developing and Supporting Multimode Survey Instruments." Blaise Users Group website. Available at <http://www.blaiseusers.org/2010/papers/6a.pdf>. Accessed February 14, 2012.

if there is a conflict across modes, thereby maintaining data integrity and saving on additional programming and data reconciliation time.

2.3 Integrating with Custom Applications and SQL

Mathematica's CAISG has implemented two system designs in which external applications are seamlessly integrated with the DEP. One is a COM object used to execute stored procedures in a SQL database through an Action and the other is an on load Event in which the COM object interface loads automatically upon form selection.

2.3.1 SQL and Blaise

One of our continuing goals is to find ways for Blaise to communicate in real time with an external non-Blaise database, such as a SQL server database.

Several years ago, we explored the Blaise Datalink component and, although Datalink has the potential to be a powerful tool, it did not meet our all of our requirements. We needed the capability to store selected instrument data into a native Blaise database while simultaneously storing other selected data into a SQL database. Datalink provided only the capability to save all of the instrument data into a single SQL database. To meet our requirement of writing data once to a centralized sample management database, we needed the capability to share the SQL data tables that Datalink utilizes with non-Datalink SQL-based systems. Although Datalink can be flexible, it is rigid when it comes to sharing tables with other systems.

Our solution was to build triggers that listened to the Blaise SQL tables for updated data. When data were modified in the Blaise SQL database, the triggers would execute updates to the central tables used by our other systems. These triggers were also used to listen for and transfer data from the central tables to the Blaise tables.

Although our solution worked, the difficulty of supporting multiple databases and maintaining individualized triggers was inefficient and costly because of the additional programming time needed to document and implement them. To work around these limitations, we built an application that calls a COM object that executes stored procedures in the SQL server database to move data between the Blaise data set and our centralized sample management system SQL database. This design proved to be the most efficient and cost-effective way to proceed as we can control which specific instrument's data elements are stored in a particular necessary database. Through an easy-to-maintain COM object, we are also able to read and write data between the SQL servers and native Blaise databases. We built two stored procedures for use by our SQL databases to make this tool reusable and dynamic. One procedure accepts a listing of the Blaise fields where information will be exchanged and updated between the Blaise and SQL databases. The other procedure contains the SQL statements that move the data. Because the COM object is driven by the stored procedures, only the stored procedures have to be modified for a specific Blaise instrument.

2.3.2 On Load Event

A challenge encountered during our quest to implement real-time updates between systems was to build an instrument in which the COM object was automatically loaded upon the form's selection. It was suggested to use the predefined Events, in particular the "OnDialBegin" Event in the CATI specification file; however, we realized that this would work only for our CATI projects that used the call scheduling capabilities of Blaise and would not be applicable to our CAWI or CAPI instruments.

We tried to implement an alien router procedure in which individual parameters are passed between the DEP and the COM object, but discovered it would be difficult to maintain a list of parameters because our data models can often change, especially during the instrument development phase.

In order to establish flexibility and reusability, we decided to invoke the COM object using an Event linked to a data type. The external application was granted unlimited access to the entire Blaise survey instrument and data records. As a result, we can perform real-time updates seamlessly through the

Blaise API. No parameter declarations were required and the implementation by a Blaise programmer is straightforward. Using the Event handler, we specified the COM object method to execute at the pre-edit time using the “blrsPreEdit” property found in the API. The COM object is invoked immediately after opening a case and is completely transparent to the user.

Real-time updates, although powerful, carry risks such as increased data vulnerability if they are not implemented with caution. Because the entire Blaise data instrument and data set are available to the outside application, data can easily be modified unintentionally. In order to minimize this risk, we implemented explicit specifications and comprehensive test plans. Any changes to the system are thoroughly evaluated and carefully tested and when these guidelines are followed, the rewards easily outweigh the risks.

3 Areas We Are Exploring Using Blaise to Help Us Remain Cost-Effective and Efficient

The technologies available for use in survey data collection advance with each innovation that enters the marketplace. Recent innovations include the push for a national broadband plan;¹ always expanding and increasingly capable mobile devices (devices based on the iOS used for the iPhone and iPad and Android™ operating system); and improved ways for transferring data across multiple systems allowing for reporting in real time.

Implementing systems that use the latest technologies can be rewarding, but it is also risky and can have significant cost implications. Comprehensive planning, such as performing a detailed cost-benefit analysis, is essential to balancing the risks against the rewards of implementing successful and innovative systems.

In spite of the risks, there are new areas of technology in which we feel Blaise can have a strong impact on data collection process efficiency.

3.1 Real-Time Processing Using the Blaise Datalink Component and SQL Databases

The Blaise Datalink component in the Blaise 4.8 series allows for reading and writing of Blaise records to Object Linking and Embedding Database (OLEDB) data sources. Because this Blaise component is available, we can start planning the implementation of real-time processing between our survey management and data collection systems. As the data shared between these disparate systems become instantaneous and centralized, we can eliminate the scheduled data transfer and synchronization processes between these systems and instead provide up-to-the-second statistics of our data collection efforts to clients.

As previously mentioned, Mathematica tried using the Datalink component to share the SQL tables used for storing the data collected in a Blaise instrument with our survey management systems; although we were successful, it required substantial programming time and effort to implement. The end result produced an inefficient and costly system requiring intricately programmed updates to multiple databases. However, we plan to revisit the Datalink component as part of the plan for the upgrade to Blaise 5.

Because Blaise 5 will not store data in the current Blaise database (.bdb) format, we will have to link Blaise records collected in our instruments to another database structure. The current Datalink component enables us to design a system into which we can easily integrate our eventual Blaise 5 instruments that will be stored in the new database format.

¹ “Recent FCC Broadband Initiatives.” Federal Communications Commission website. Available at <http://www.fcc.gov/guides/recent-fcc-broadband-initiatives>. Accessed February 14, 2012.

We also recently conducted an evaluation of our Blaise case management system to determine how to collect and process survey data as efficiently as possible. Because respondent sample information can be tracked and manipulated across multiple systems, it seems extremely inefficient to maintain data mirrored in distinct systems, which, if not properly handled, can easily get out of sync. After our success with using a SQL server database to communicate with Blaise, we decided to design and eventually implement a process that stores all respondent sample information in a centralized database shared by all the systems that need access to it. This would eliminate processes that update data on a scheduled (usually overnight) basis between systems and make updated information available to the interviewing process almost instantaneously.

By successfully using the Datalink component for our CAWI surveys by linking a .boi file on our web server with a .bdb file on our data server, we are operating with a secure, real-time processing system for our multimode CATI/CAWI data collection instruments. Using our past experience with the success of these .boi files and the expected advances of the Datalink component, we anticipate that our re-exploration will produce desirable, cost-effective, efficient, and innovative results.

3.2 Blaise on Mobile Devices

During the past decade, the wireless internet and mobile telephone markets have exploded in the United States and shows no signs of slowing down any time soon. According to the Pew Research Center,

“About four-in-ten Americans (41%) connect to the internet wirelessly using a laptop or hand-held device when away from home or work. This is up from 36% in April 2009. Far more Millennials than those in older generations use wireless connections to surf the internet. About six-in-ten Millennials (62%) connect to the internet wirelessly when away from home or work, as do 48% of Gen Xers.¹

Because of this growth, especially when contacting younger survey respondents, clients are demanding that we offer new ways of collecting data from respondents who use the latest wireless and mobile devices. This has added complexity to our goal of remaining flexible but cost-effective and efficient.

Developing dynamic data collection systems to adapt to the seemingly endless variety of mobile technologies is an ongoing challenge. From January to December 2011, mobile platforms have gone from 0.7 to 1.2 percent of the browsing marketplace² and we expect this percentage to skyrocket in the years to come. From iPhones, iPads, and Android-enabled devices to tablets and e-book readers, each platform also has its own browser to support. In the near future we will have to determine how to design efficient, real-time systems that can optimize Blaise’s data collection capabilities on mobile devices. It appears the trend in data collection is moving to an entirely internet-based system, in which one browser-based system that can handle many distinct types of browsers can manage CATI and CAWI (and possibly CAPI). Supporting the vast variety of browsers will be a challenge. We hope our experience with multimode data collection, especially in using Blaise IS, can give us a leg up as the world of mobile technology moves to the forefront of our data collection efforts.

¹ Keeter, Scott, and Paul Taylor, editors. “Millennials: A Portrait of Generation Next. Confident. Connected. Open to Change.” Pewsocialtrends.org, February 24, 2010. Available at <http://pewsocialtrends.org/files/2010/10/millennials-confident-connected-open-to-change.pdf>. Accessed February 14, 2012.

² “OS Platform Statistics.” W3 Schools website. Available at http://www.w3schools.com/browsers/browsers_os.asp. Accessed February 14, 2012.

4 Conclusions

Striving to be innovative, practical, and cost-effective should be key parts of any organization's business model if it wishes to remain economically viable in today's competitive marketplace. Through proper planning, applying innovative ideas and implementing complex data collection efforts should not be a difficult task, even when your funding is restricted or you face obstacles presented by a difficult economic climate.

Organizations have to look outside of themselves to companies that excel at change to help in spotting leading technology trends and how to adapt to them. If you look at companies that have been around a long time, such as IBM, you will see they had to change their business models just to remain in business. IBM has migrated from making business tabulating and time recording machines to being on the leading edge of computer technology today. If IBM did not change and evolve over the years it would probably be out of business today. Today, IBM makes none of the products it sold 100 years ago.¹

Eastman Kodak is an example of a company that has not kept up with technology and market changes. Kodak struggled to stay relevant as the world moved from film to digital cameras and it is possible Kodak might go out of business in spite of all the innovative products it developed in the past. Kodak was once admired as a technical and marketing marvel, the gold standard of American business success for decades, yet it filed for Chapter 11 bankruptcy protection in early 2012 and has seen its worldwide workforce reduced from 130,000 in the mid 1980's to just 17,000 today.²

4.1 Remaining Cost-Effective in a Cost-Conscious Economy

The challenge to keep costs low while producing a quality product is very demanding in today's environment. One way of being cost-conscious is to build flexible and dynamic systems that can be reused repeatedly. This helps lower overall costs by spreading the cost out over several projects. Before building new features, we must research the use of out-of-the-box utilities to their maximum capabilities. If these out-of-the-box features do not meet the project's needs, we must figure out how we can add reusable value to them without re-creating the wheel.

Too often organizations are resistant to change and continually implement the same methodologies for all projects, regardless of cost and efficiency. Therefore, it is important for us to periodically evaluate the efficiency of our systems, determine if we are using the most suitable development tool(s) for each task, and decide if an update is necessary. The process of updating systems to keep up with changing technology can be expensive in the short term but cheaper in the long run.

4.2 Future Improvements

Technologies in the data collection field are continuously changing. In order for us to succeed, we have to design innovative systems and meet the needs of our clients while keeping our costs in line with or lower than our competitors. We have high expectations that the next generation of Blaise, version 5, will meet these demands and enable us to continue to add flexibility while writing code once and sharing data for real-time data collection and sample management in an easy-to-use fashion. Along with this, we must continue to develop reusable products that are sufficiently dynamic that they can be used with other systems.

¹ "An Exploration into Making the World Work Better." IBM at 100 website. Available at <http://www.ibm.com/ibm100/us/en/thinkexhibit>. Accessed February 14, 2012.

² Sink, Steve. "Kodak Files for Chapter 11 Bankruptcy." *USA Today.com*, January 19, 2012. Available at <http://www.usatoday.com/money/industries/retail/story/2012-01-19/Kodak-bankruptcy/52660342/1>. Accessed February 14, 2012.

This leads us to future improvements we feel Blaise has to make to remain a leading product in the area of survey data collection. Datalink should have functionality for working with external databases that are defined outside of its default structure settings. The ability to write once and share between systems contributes to major cost savings and reduces data problems that could occur by moving data around. It will also be important for Blaise to run on any number of mobile devices. This market is growing and changing rapidly as people find new ways to use these devices in their everyday lives.

The bottom line is no organization wants to become the next Kodak.

Use of Blaise in Today's Survey Research Environment

Jane Shepherd and Ray Snowden, Westat

International Blaise Users Conference, April 2012, London, UK

Summary: *Blaise is used in a variety of survey research environments by Westat and many other organizations. Surveys typically involve multi-mode data collections and organizations are required to use commercial off-the-shelf (COTS), integrate multiple platforms and applications to solve complex problems, and assure the security and confidentiality of the respondents and collected data. Software selection is a key issue as organizations have to invest efforts in training staff and supporting and managing software and systems infrastructures over time, while offering cost-effective solutions. These factors are discussed in relationship to the use of Blaise in this evolving environment.*

Dynamic Survey Research Environment

One of the major challenges for software developers today is the evolving and dynamic nature of the survey research environment. In recent years there has been a continuing increase in the number of data collection modes and the types of activities and measures being incorporated into these data collections. In conjunction with these needs survey researchers have seen increased pressures for real time reporting and integration across modes for the exchange of data and performance of quality control tasks.

Some of our most common modes include CAPI, CATI, CAWI, PAPI, CASI/ACASI, and IVR. These modes are often inter-mixed with other presentation media such as brochures, mail outs, show cards, permission forms or other materials, and the data collection software may need to interact across platforms and operating systems. The need to support integration with other types of software, for example GPS or biomedical components, is increasing. The role of the interviewer in administering these various modes of data collection may vary from the traditional model. Interviewers may be required to do more or to change roles during the data collection. The completion of the survey may be interviewer mediated or respondent mediated or a combination of both. For example, the interviewer may arrive at the household and do the initial contact and consent task, and then move the data collection into a CASI, IVR or other mode. In other situations a hard copy questionnaire or IVR may be used to initially solicit the respondent's participation, and then the data collection mode may switch to some type of interviewer mediated mode. Survey designers are taking advantage of the multiple available modes to maximize respondent compliance with the data collection and offer respondents the option to complete the data collection at the time of their choice and in the mode of their choice.

In addition to the complexities described above, the administration of the multiple data collection modes may be sequential, concurrent, or variable (for example, changing from sequential to concurrent at some point in time). Adaptive designs that take advantage of feedback during the data collection can alter the operational plan for the survey. Paradata captured during survey administration and survey results including refusals and item nonresponse rates provide valuable feedback to researchers looking to optimize the use of resources and maximize response rates.

In short, the survey research environment is very complex and changing rapidly by incorporating new technology and techniques to improve the quality of research data, improve response rates, reach more respondents, and lower costs. No single product can meet all of these emerging needs. In addition to a strong set of core survey support features, survey research organizations require software with other key characteristics that allow products and tools to be adapted, extended, and integrated in creative new solutions. We define some of these characteristics in this paper, the manner in which Blaise reflects many of these characteristics as a leading survey research tool, and examples of how Westat has integrated Blaise capabilities in numerous custom applications.

Key Characteristics of Survey Research Software

Data collection software has to meet the needs of the current multimode landscape and be flexible enough to adapt to future challenges. The following characteristics represent both the core capabilities required in survey research software plus those characteristics that permit managers and developers to creatively extend and adapt the software for new applications:

- Fulfills core survey functions - the tool allows the user to author and execute survey instruments that support a wide range of question types, complex data structure such as rosters and loops, and complex routing while maintaining data integrity through the interviewing process.

Blaise is perhaps the leading product of its type with respect to these core features. The Blaise data model records the metadata that defines questions, responses, and flow and is interpreted by the Blaise rules engine. These two components are central to the Blaise solution and ensure a high level of functionality and consistency in the collection and interpretation of data collected using any of the Blaise supported modes, platforms, or tools.

- Provides a rich set of features, tools, and utilities – this includes tools to author and customize instruments, deploy and execute instruments in different operational and technical scenarios, and access and manipulate data.

Blaise provides a wide variety of features, tools, and utilities including developer tools and documentation, multimedia support, multi-language support, customizable layout, ancillary programming languages such as Manipula, Maniplus, and Basil for data manipulation and interface control, a CATI management system, audit trails, support for data entry verification, and integrated user help. These features and tools all build on the core datamodel and rules engine of Blaise and provide a wide variety of options to creatively and efficiently extend the core capabilities of the product and allow Blaise-based functions to be used throughout the survey life cycle.

- Reliable performance and scalability – in order to maximize response rate and interviewer efficiency it is critical that a large complex survey performs well in a single user environment (e.g., CAPI) and that a survey intended for a large audience functions well in a multi-user environment (e.g., CATI or CAWI).

Blaise has a long history of reliable performance in a wide variety of large and complex surveys. The Blaise datamodel and rules engine has a huge capacity for large numbers of questions, edits, and hierarchies and can apply these rules to responses with sub second response time. The Blaise architecture has evolved to a more flexible multi-tier architecture that can support large concurrent population of respondents.

- Integration with other technology, products, and applications – no single product can provide all of the support required for a large scale, end-to-end, survey operation. The ability to interoperate and share data with other applications and platforms, often in real-time, is an essential characteristic to permit survey research organizations to build complex and unique solutions.

Blaise provides numerous features and mechanisms for integrating with other applications including the use of RDBMS for data storage, APIs to invoke Blaise functionality, and the use of Active X components and DLLs to extend Blaise functionality.

- Supports a secure data collection environment – given that confidential data is frequently collected in survey research projects, organizations may fall under numerous legal and regulatory mandates including FISMA, HIPAA, FDCC, and others relating to information security and data confidentiality. This is a complex and growing area of concern and can consume significant time and resources through the survey life cycle.

Blaise provides numerous features and capabilities that directly support or are compatible with many of the controls required to meet these requirements (Rhoads and Snowden, 2010).

- Availability of support – when undertaking large and complex survey that may be active for years, it is important that the technology and tools used to support the operation are supported and will be supported in order to adapt to new technology, build in new features, fix bugs, train staff, answer questions, and solve problems. Support of this type is particularly important for a product as complex and extensive as Blaise.

There are a wide variety of support mechanisms in place for Blaise including technical support for the core product by Statistics Netherlands, the International Blaise Users Group (IBUC), the Blaise Corporate Users Board (BCLUB), and 3rd party support providers such as Westat.

Blaise in the Survey Research Environment

Blaise is a core system used at Westat throughout the survey lifecycle. In addition to the core features and capabilities listed above, Westat has extended Blaise and integrated Blaise with various applications and technologies to provide full-service support for a wide variety of survey modes and scenarios and to support the full life cycle of the survey life cycle. Some of these extensions are shown in the following diagram and described below:

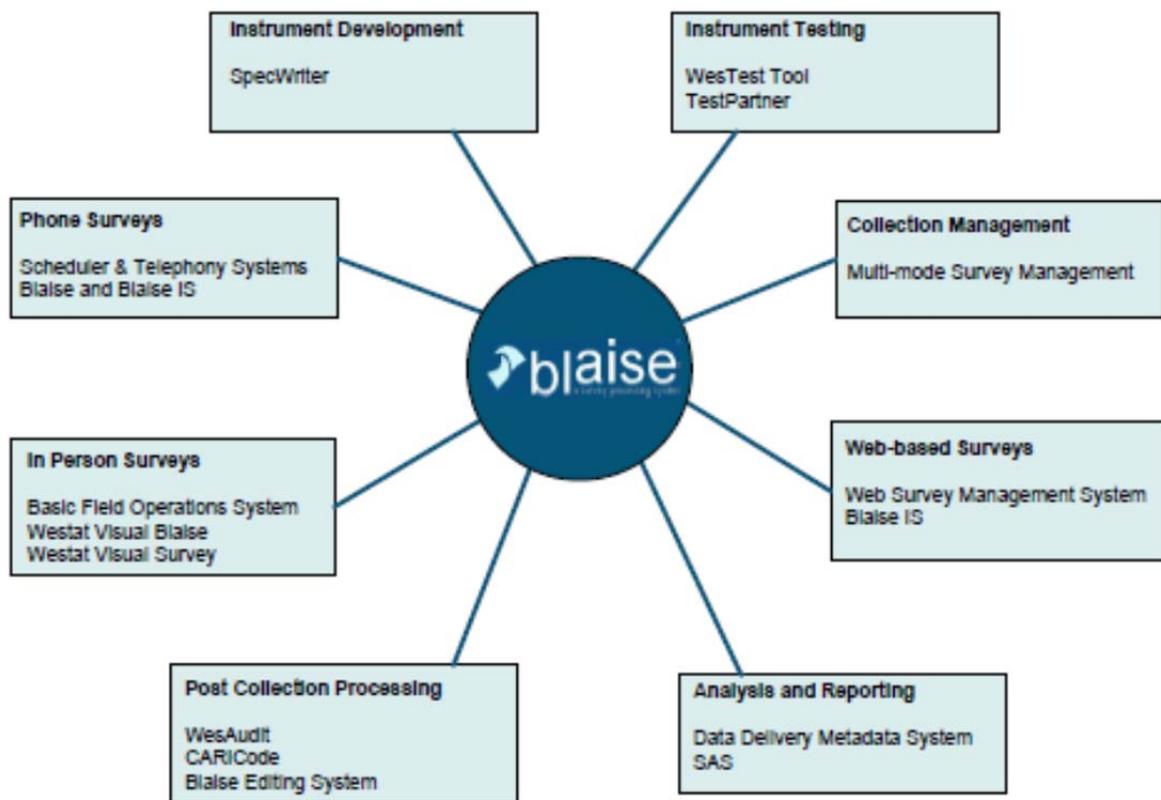


Figure 1. Survey instrument life cycle: Blaise and Westat tools, extensions and applications

Instrument development – the core Blaise product includes a full set instrument development suite that permits developers to specify the data model and customize the look and feel of the Blaise survey. Westat has modified a custom specification authoring tool to generate output that is fully compatible with the Blaise authoring process.

- Specification development and maintenance – A SpecWriter tool is used at the specification stage in the development life cycle. It provides designers with dynamic templates for considering and specifying all required elements for key item types, where items are defined as questions, boxes, or edits. SpecWriter allows hard-copy generation of the specifications and generates Blaise block templates that Blaise programmers use to create a Blaise instrument (Gowen and Clark, 2007). The output generated from this tool is customizable and contains the detailed specifications for the instrument in terms of question text, response text and values, questionnaire flow, and whether interviewer help is available for the question. SpecWriter creates HTML files, Blaise template code, and detailed specifications used by coders. It also streamlines the process of documenting and updating instrument specifications. Testers can use detailed specifications to compare data entry screens and functions against requirements and expected results. It increases efficiency by automating some of the questionnaire programming and helps reduce errors in creating and implementing the instrument.

Instrument testing - the testing of Blaise instruments is a crucial part of the lifecycle to insure that the instrument will collect and store data correctly and that the instrument flow properly reflects the survey design. This is particularly important with a large and complex survey instrument where the number of test scenarios required to reflect the numerous navigational paths through which the instrument may pass can grow quite large. Automated testing support tools are absolutely essential to

permit comprehensive testing on a reasonable schedule and cost. Westat has used the following COTS and custom tools in the testing of Blaise instruments:

- Testpartner is a COTS testing product that can be used to automate testing for Blaise Internet projects including Westat Visual Survey. Automated testing provides a more cost effective way of conducting effective regression testing and eliminates much of the manual testing work (Brenner and Manoj, 2010). Repeatable scripts can be executed against code after updates are made to assure that the data are being captured correctly.
- The WesTest tool has made problem reporting during testing easier by implementing an automated reporting system that is integrated with a COTS change management system. When a problem is discovered during execution of a Blaise instrument, the user can invoke an executable that produces a dialog box that the user completes to report a problem. The user need only enter a summary, description, and “type” of the problem. Items such as date, field tag, primary key, field value, instrument version, etc., are automatically recorded.

Collection Management – systems to manage the overall data collection, particularly in multi-mode studies, are essential. These systems are sometimes provided within products, but more often the user needs to provide a layer of integration across independent, mode-based platforms and applications. Westat has integrated Blaise with a multi-mode management system to support these needs (Frey, et. al. 2012).

Phone Surveys – Blaise is used in a wide variety of CATI surveys at Westat. Westat has integrated Blaise with custom CATI management software which permits Blaise to be integrated with the existing Westat CATI operational model. In order to provide web-based access to Blaise instruments for our remote CATI interviewer workforce, Westat uses two different technical approaches.

- Blaise integration with Westat CATI Scheduler and Telephony Systems -- Westat has implemented Blaise for CATI instrumentation and interviewing under a custom CATI Scheduler through the use of our Blaise Mainline software, which provides the link between Blaise CATI instruments and the CATI Scheduler. Operationally, the interviewer sees only Blaise screens, including screens for receiving or requesting cases, viewing interview management information about a case, setting status codes, etc. The integration of these Blaise screens with the CATI Scheduler is transparent to the interviewer and also to the supervisors, who are provided with Westat-created Blaise utilities for examining cases combined with their respective scheduler information, all within a Blaise user interface. Another critically important system for CATI surveys is the telephony system that supports all telephone operations in coordination with interviewer workstations. Westat has adopted a commercial computer-telephony integration (CTI) system for use in telephone interviewing. Through the Blaise Mainline, Westat has integrated this CTI system with the interviewer workstations.
- Browser-based access to Blaise – Westat provides browser-based access to Blaise instruments for our remote CATI staff through Blaise for Windows launched within a Citrix environment and through the use of Blaise IS. These two options provide flexibility in terms of platforms, performance, integration, and end-user functionality. This flexibility permits us to choose the environment that best meets the needs of a particular project.

In Person Surveys – Blaise is also used in a wide variety of in person surveys including CAPI, CARI, and ACASI. Westat has integrated Blaise with custom CAPI management software which permits Blaise to be integrated with the existing Westat CAPI operational model (Hill, 2004; O’Reagan et.al, 2010). In addition, Westat has developed several significant functional extensions using the Blaise interoperability features to meet complex and unique needs.

- Blaise integration with Westat CAPI management system-- Westat implements CAPI studies with our Basic Field Operations System (BFOS), using a web server and a web browser on field staff computers. Westat's BFOS system is designed for use with Blaise, and all management information coordination between Blaise instruments and the BFOS management system is handled through a software component known as the BFOS Mainline. This Mainline software resides on field laptop computers and provides the interface for field staff to review, select, update, and start Blaise instruments for all cases on the laptop computer. BFOS also provides functions to synchronize data between the laptop and the home office.
- Westat Visual Blaise (WVB) is a Blaise-based CASI software system that includes features for self-interviewing, with audio (ACASI) and video file incorporation. WVB has a user-friendly screen design suitable for touchscreen operation by survey respondents.
- Westat Visual Survey (WVS) takes advantage of the underlying technical architecture that captures the benefits of the Blaise open architecture and uses the Blaise rules engine to incorporate the features of Blaise, while writing the data to a COTS database such as SQL server. The system displays an enhanced user interface via an Internet browser. This architecture results in a multimode product, allowing WVS to run on an individual laptop computer, networked workstation, or via the web. Using this architecture facilitates more complex data collection by allowing a virtually unlimited number of records to be stored interactively at any database level during program execution. WVS provides better programmer control of instrument navigation, enabling the system to fulfill more complex navigation and data flow requirements, by allowing the developer to override the Blaise route and force the application to go to a specific point in the instrument. This also provides for more robust functionality by incorporating summary screens and jumpback navigation screens and provides for better navigation options for interviewers because of enhanced programmer control over what appears on the screen. These features allow developers to program navigation menus in more complex instruments (Segel and O'Reagan, 2012).

Web-based Surveys – CAWI has become an increasingly important mode as a cost effective means of reaching a large and distributed population of respondents. Westat has used Blaise for instrumentation support on web-based surveys particularly in cases where multi-mode support is required and the same instrument can be deployed for multiple uses.

- Blaise integration with Westat custom web survey management systems – Westat implements web surveys with our management system that provides support for respondent authentication, survey task workflow, and instrument integration.
- Use of Blaise IS – with the changes to the architecture of Blaise IS, the product now can provide a level of performance with increased concurrent usage that is required to support complex web-based surveys with a medium to large respondent population.

Post Collection Processing – there are numerous back-end processes through which instrument data may pass before it is ready for final analysis and reporting. In many cases, the ability to utilize the Blaise data model and rules engine to help guide and validate these efforts is crucial for ensuring data consistency and quality throughout the survey lifecycle. Westat has been able to utilize and extend core Blaise capabilities to meet a variety of post collection processing needs.

- Managing audit trail data – The Wesaudit tool is an extension of the Blaise standard audit trail feature that tracks movements in the instrument and records field values. The tool's visual interface provides an organized means for viewing audit information recorded by the basic Blaise audit feature. WesAudit makes it easier for programmers to debug code and follow flow through complex instruments. It also facilitates the process of data recovery if Blaise

data files develop problems that cannot otherwise be fixed. In addition, WesAudit can be used as part of interviewer quality monitoring and to facilitate the exploration of audit trail data in order to set up more extensive methodological analyses of audit trails

- CARI coding – CARICode is a web-based coding tool that works in conjunction with the CARI feature of Blaise. Coders listen to digital recordings of interviews while viewing an image of the Blaise screen and complete the coding tasks. Westat is using this technology to address several different components of survey error thus improving the quality of our data collection efforts. Field data collections use CARI technology as an alternative to supervisor observations to review interviewer performance, identifying any skill concerns in coding of audio recordings, and providing detailed feedback and coaching to individual interviewers or more global re-training on survey specific concepts across the broader field staff. Westat also pairs the audio recordings with other survey paradata (e.g., interview timings, case disposition history) as a monitoring tool for identifying possible falsifiers. The tool integrates applicable paradata and enables coders to quickly and comprehensively perform their work. Other uses of CARI include monitoring data quality, validating interviews, recording open-ended responses, conducting usability reviews, and testing instruments.
- Post-collection editing – the Blaise Editing System (BES) is designed to take advantage of the ‘replay’ features of Blaise COTS to perform editing and coding operations (Laidlaw, et. al. 2010; Allan, et. al., 2001). As data are received from the field, the CAPI data are loaded into Westat’s BES. Comments are auto loaded into a Data Decision Log (DDL) module and can be viewed by the editor. As editors review the comments, they can enter a status for the comment, reason for data update if needed, and launch the Blaise instrument from the DDL module. The Blaise instrument will ‘replay’ in interviewer or editor mode, so that the editor may post a data update. The Blaise replay mode assures that data updates do not alter the integrity of the Blaise datamodel and associated skip, range and formats in the dataset. The metadata for the study are also maintained throughout the editing process and available for review by the BES editors. A Blaise audit trail is automatically updated when data changes are made so that all data changes are tracked and known. The BES system reduces editor and post-processing labor significantly by assuring that the data updates are made in conformity with the skip and flows within the CAPI instrument and CAPI metadata. The reasons for data update are noted in the decision log. Edited and archival versions of the data files are maintained throughout the process. BES generates reports by editor ID, case ID, and variables updated for QC review. A verification module assures that the work of editors is reviewed and QC processes are observed. Coding of other specify responses or other items can also be handled by the system. The BES system also generates frequencies and cross tabulations of data for review.

Data Analysis and Reporting – wherever the information in the Blaise data model can be used to automate ancillary processes or validate data this results in substantive benefits in both productivity and data quality. Westat utilizes the information in the Blaise data model to drive a variety of analysis and reporting functions.

- Metadata and Delivery Data Support – the Data Delivery Metadata System (DDMS) is a web-based tool that supports delivery of metadata associated with datasets including raw, analytic, and public use files. Blaise data files and XML can be used with the tool which provides descriptive information about the context, quality and condition and characteristics of the data. This information helps one to understand how the data were collected and any post-collection handling that might have affected its use and interpretation. The tool also provides documentation of each delivery, version control of data deliveries, and searchable linked metadata.

- SAS software is often used for aspects of post-collection processing. Blaise data are easily transformed into SAS or other database systems (Allan, et. al., 2012).

Conclusion

In this paper we have discussed and illustrated two important dimensions of Blaise that allow it to function as a core technology for survey research organizations:

- Blaise provides a stable and secure platform with extensive and powerful core functionality, a large set of features and tools, and numerous support services. As a result, there are a wide variety of solutions that can be developed using out-of-the-box Blaise capabilities.
- The Blaise open architecture and related features permit Blaise functionality to be extended or integrated with other applications to create unique and innovative solutions that go beyond the core Blaise functionality. As a result, the significant value of the Blaise data model and rules engine can be reused to improve efficiency, reduce cost, and ensure a very high level of data consistency and integrity.

The survey research environment will continue to evolve and change at an increasingly rapid pace. Some of the factors that will drive this change include:

- Multi-mode surveys will become increasingly common in order to reach a greater number of respondents, control costs, and achieve desired response rates.
- Real-time feedback loops between instrumentation and management systems will become increasingly important to implement more effective adaptive design and multi-mode control.
- Different types of data, media, embedded links, etc. will become more commonly integrated into survey instruments.
- Different types of end-user devices, e.g., tablets, smartphones, etc. will be used to access CAI systems.
- Browser-based access to surveys will become a more common access mechanism for surveys of all modes.
- The use of virtualized and cloud-based infrastructure to support survey systems will continue to expand.
- Security will continue to be a central issue, particularly with Federal or other governmental surveys, and will continue to evolve as the underlying technology and platforms change.

Blaise has performed extremely well in the survey environment most organizations face today and is strongly poised to meet the challenges of the future. Blaise will continue to be an excellent platform for survey research organizations in the future due to its strong set of core features, sound and flexible architecture, and the commitment of the Blaise support team to continue to extend and evolve the product.

References

Allan, B., O'Reagan, K., and Lohr, B., Dynamic ACASI in the field: Managing all the pieces, Proceedings of the 7th International Blaise Users Conference, September 2001.

Allan, B., Frey, R., and O'Reilly, J., Using metadata in Manipula and Maniplus, Proceedings of the 14th International Blaise Users Conference, April 2012.

Brenner, K and Manoj, S., Automated regression testing of Blaise Internet: A case study, Proceedings of the 13th International Blaise Users Conference, October 2010.

- Frey, R., O'Reagan, K., and Brown, N., Servicing Blaise instrument needs in a multi-mode environment, Proceedings of the 14th International Blaise Users Conference, April 2012.
- Gowen, L. and Clark, P., Lifecycle processes to insure the quality of Blaise interview data, Proceedings of the 11th International Blaise Users Conference, September 2007.
- Hill, D., Deploying Blaise to tablet PCs for mobile use, Proceedings of the 9th International Blaise Users Conference, September 2004.
- Laidlaw, M., Rhoads, M., and Shepherd, J., Post-collection processing with Blaise in a distributed environment, Proceedings of the 13th International Blaise Users Conference, October 2010.
- O'Reagan, K., Frey, R., and Robbins, K., Impressions of Basil, Proceedings of the 13th International Blaise Users Conference, October 2010.
- Rhoads, M. and Snowden, R., Security considerations in Blaise environments: Options and solutions, Proceedings of the 13th International Blaise Users Conference, October 2010.
- Segel, P. and O'Reagan, K., Extending Blaise capabilities in complex data collections, Proceedings of the 14th International Blaise Users Conference, April 2012.

The Centralized Survey Experience at National Agricultural Statistics Service

Roger Schou and Emily Caron, National Agricultural Statistics Service, USA

1 Introduction

The National Agricultural Statistics Service (NASS) is an agency of the United States Department of Agriculture. NASS is responsible for collecting, editing, and summarizing agriculture data. We are the sole agency for producing Agriculture Statistics for the United States. NASS primarily uses CATI data collection, but also paper questionnaires for mail and field interviews, CAPI field interviews, and web data collection methods.

NASS is made up of forty-six field offices across the United States, a headquarters (HQ) location in Washington, D.C., a research division in Virginia, and as of recently a National Operations Center (NOC) in St. Louis, MO. The NOC is intended to serve as a primary calling center for NASS with a capacity of up to 150 phone interviewers. Six of our forty-four field offices also serve as Data Collection Centers (DCCs), ranging from a 20 to 60 seat phone interviewer capacity.

When we last presented at IBUC XIII, we reported on the successes and lessons learned from converting our first survey to a Centralized environment: the Mink Survey. Since that time we have converted ten disseminated Blaise instruments and added fifteen new surveys into Centralized solutions all while facing numerous challenges such as an agency-wide upgrade to virtual desktops; adapting to and in some cases assisting with the centralization of other systems which interact with Blaise; updating code to incorporate our new NOC; doubling the number of employees in our section with somewhat “green” programmers who are sitting one time zone away at the new call center; and others.

2 Blaise Programming... Times Two

Eighteen months ago, the NASS Blaise development group was called the CASIC Section and it included six developers and one section head all sitting in HQ. Today, the CASIC Section no longer exists, but there is a Blaise Questionnaire Design and Editing Section (BQDES) in HQ with four developers and one section head. There is also an Enumerative Survey Development Group (ESDG) sitting at the NOC, consisting of seven Blaise developers. Three of the seven at the NOC had never touched Blaise coding and had little or no programming experience in any other language before their arrival in ESGD. A 3½ day training session was conducted for new NOC personnel and group meetings are held when needed over Virtual Teleconferencing equipment. Even though the Blaise group has grown, the overall total number of NASS employees is declining intentionally.

Plans are to add at least another 25 surveys into the Centralized solution during 2012. In a number of cases, there is more involved than just converting existing surveys from decentralized to centralized solutions or introducing new surveys. Some of the surveys are weekly surveys, which we have never attempted in Blaise at NASS. These offer new challenges. For example, some of the weekly surveys are expected to offer a place for revisions for the previous weeks’ data within a given week’s data collection.

Another set of five surveys has a need to be coordinated to allow any given respondent to respond to one or more of these five surveys, depending on the surveys for which they were sampled. A Virtual Survey Coordinator was developed using VB.NET so that a group of surveys could be completed by the same respondent in a single phone call. So even though the size of our group developing Blaise instruments has nearly doubled, the workload and new challenges have more than kept the staff busy.

3 Centralized Blaise Infrastructure

In the past with the decentralized environment, there were challenges with the physical Blaise datasets being located in every field office (FO). We had the six DCCs collecting data and sending completed forms home to the client states via Blaise datasets. It was a stable scenario, but it was very cumbersome with a large number of physical file handoffs. The maintenance rested almost entirely on the CASIC Section.

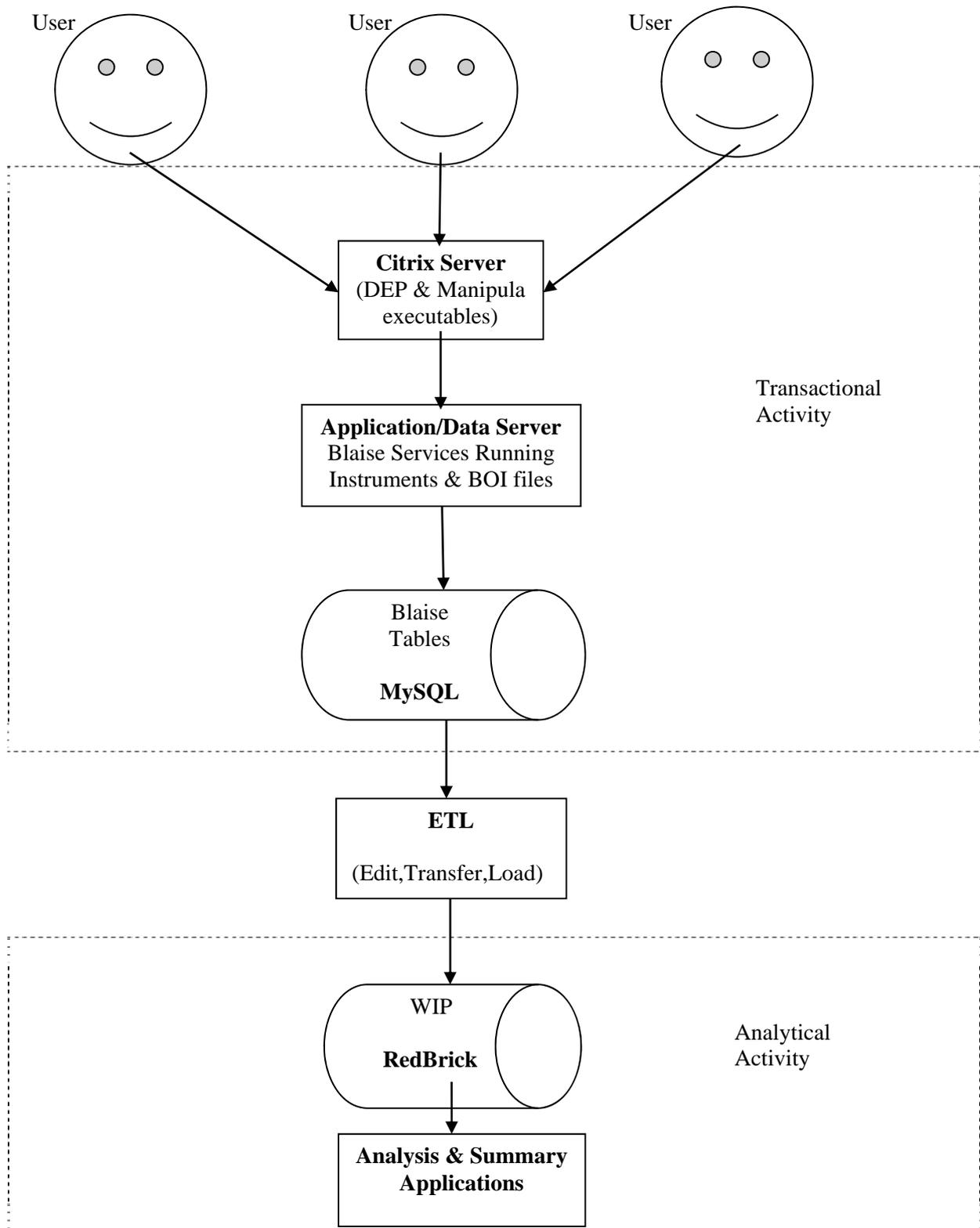
With the centralized environment, there are many more players involved in the maintenance of the systems. Figure 1 shows the many different layers of infrastructure involved in our Centralized Blaise environment.

Figure 1. Infrastructure Table

Component	Description
Citrix server	Gateway to all applications. Conversion of all users to Citrix was completed in 2011.
Application servers	Blaise applications and services reside here. We use three different app servers: Development, Beta and Production.
Database servers	The databases themselves reside here. We access three different database servers: Development, Beta and Production.
Storage	There must be physical storage for the databases
Communications	The Wide Area Network must be available for users across NASS to reach the centralized applications
MySQL database	Transactional activity takes place here
ETL to WIP	The ETL (Extract, Transfer, Load) moves data from MySQL to WIP on a timely basis so analysis and summary reflect the latest data
WIP database	RedBrick analytical database - Analysis and summary programs pull data from here. We actively “write” to two different WIP databases: Beta during testing, and Production during live data collection.
Windows AD Table	Each NASS employee that needs to access Blaise must have current information in Windows AD about their ID, location (FO or HQ Div-Branch-Section), and role in order for their rights to the data and menu options to be set correctly.

Figure 2 gives a picture of the system flow. It also identifies the areas of transactional activity on the MySQL database and analytical activity on the RedBrick database.

Figure 2. System Flow



4 Determining Rights in Centralized Blaise

The Blaise software is accessible to users through a VB.NET menu system, written and maintained by the Blaise programming staff. One of the first responsibilities of the menu is to see which user is attempting to access the system and check that individual's information in the Windows AD table. If any pertinent information about the user is missing, or if the user not found, the menu will not allow the user to go any further. However, if all information is present and accounted for, the user will be granted access and the location and role of the user are stored. Role could be statistician, stat assistant, supervisory interviewer, or interviewer.

In order to maintain a common structure across all centralized Blaise instruments, we use Generic BOI files with the in-depth data partition type. This allows us to use one ETL for all surveys to copy the data from the Blaise MySQL tables to the WIP analytical database. In addition to the eight standard Blaise tables that accompany Generic BOI files, we found the need to create a few other flat tables: CASIC_SurveyInfo, CASIC_Management and CASIC_FAT. As other systems in NASS centralize, this same type of information will be needed, so these tables may end up being shared or moved so that many can glean information from them.

The CASIC_SurveyInfo table holds critical information about each survey, such as survey code; year, month and day values; Blaise instrument name; different survey type indicators; data collection dates and other useful information. The CASIC_Management table contains many key Blaise fields on which we often need to sort or limit the datasets. These fields are defined as indexes and used in Record Filters in Manipula. The fact that these fields are indexed allows the queries to the database to be quick and efficient. The CASIC_FAT table (FAT = FIPS Assignment Table) is necessary to track which NASS locations are collecting and/or editing survey data, or simply identifying a state that is being serviced by other states. The acronyms we use for these three office level roles are as follows: DCC = Data Collection Center, EC = Estimation Center, and CS = Client State. HQ staff can interactively assign CSs to DCCs and ECs for each survey using the CASIC_FAT. Once these assignments are created, the resulting data is populated into the CASIC_FAT table and the menu will know which rights to provide to each office. **Every different survey we conduct at NASS has a different mix of DCCs, ECs, and CSs.**

Every survey instrument in the centralized solution has three fields defined in it to store the DCC, EC, and CS responsible for each form. These fields are populated when we initialize the sample for the survey. Once the menus have identified the user's attributes, Manipula can be used to retrieve the appropriate forms by using Record Filters.

The buttons on the VB.NET menu interface will appear and disappear dynamically based on a combination of the roles and location of the person attempting to access them. The access to the centralized database is controlled entirely by the menu. The design of the menu, as seen in Figures 3 and 4, allows whole tabs to be invisible based on the needs of the survey and the roles of the user accessing them. The design also allows group boxes to be made invisible within a certain tab. The goal is to only supply a given user the necessary buttons/functions for a given survey.

Figure 3. Data Collection Menu Tab

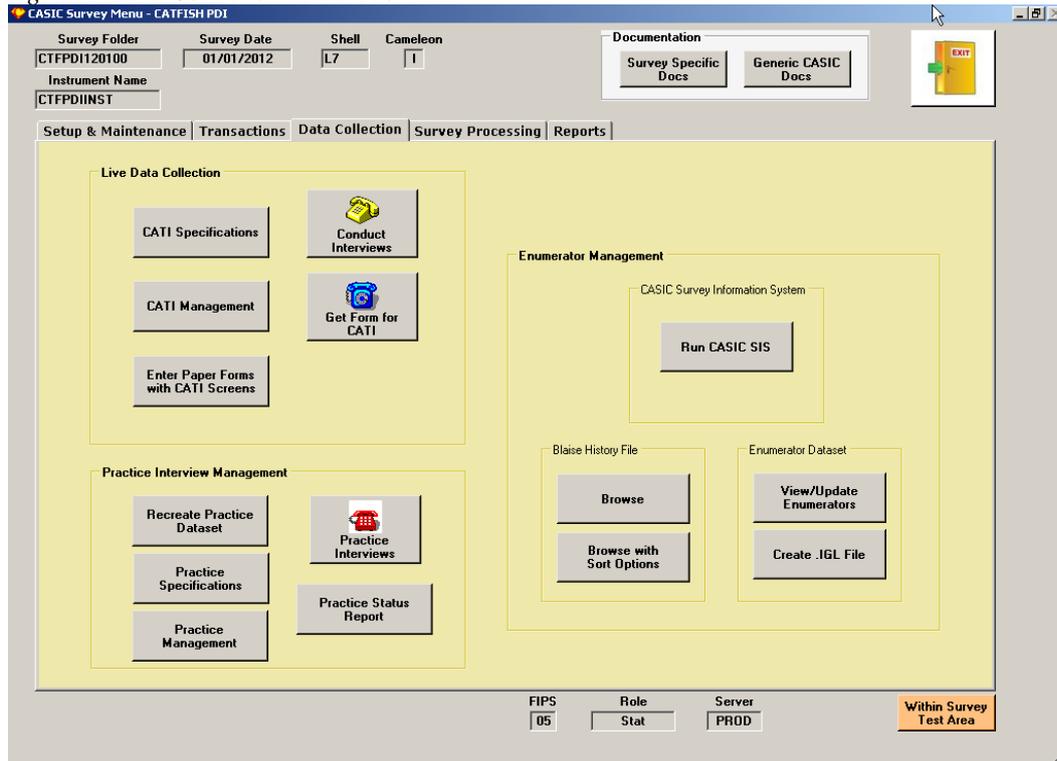
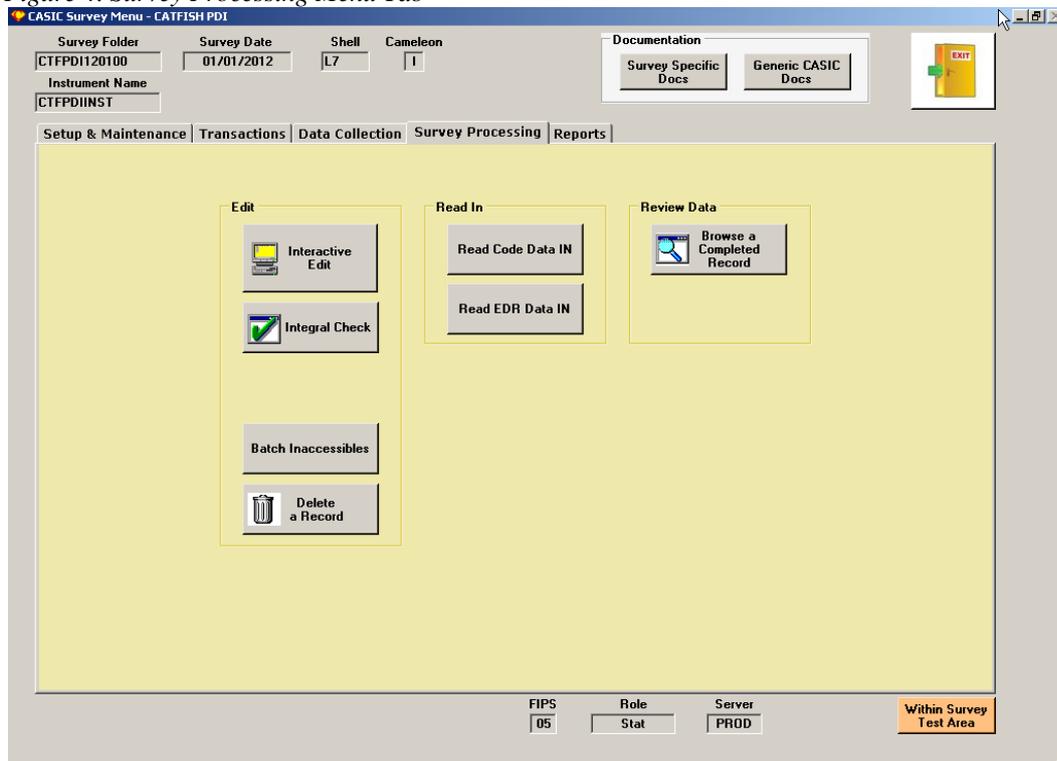


Figure 4. Survey Processing Menu Tab



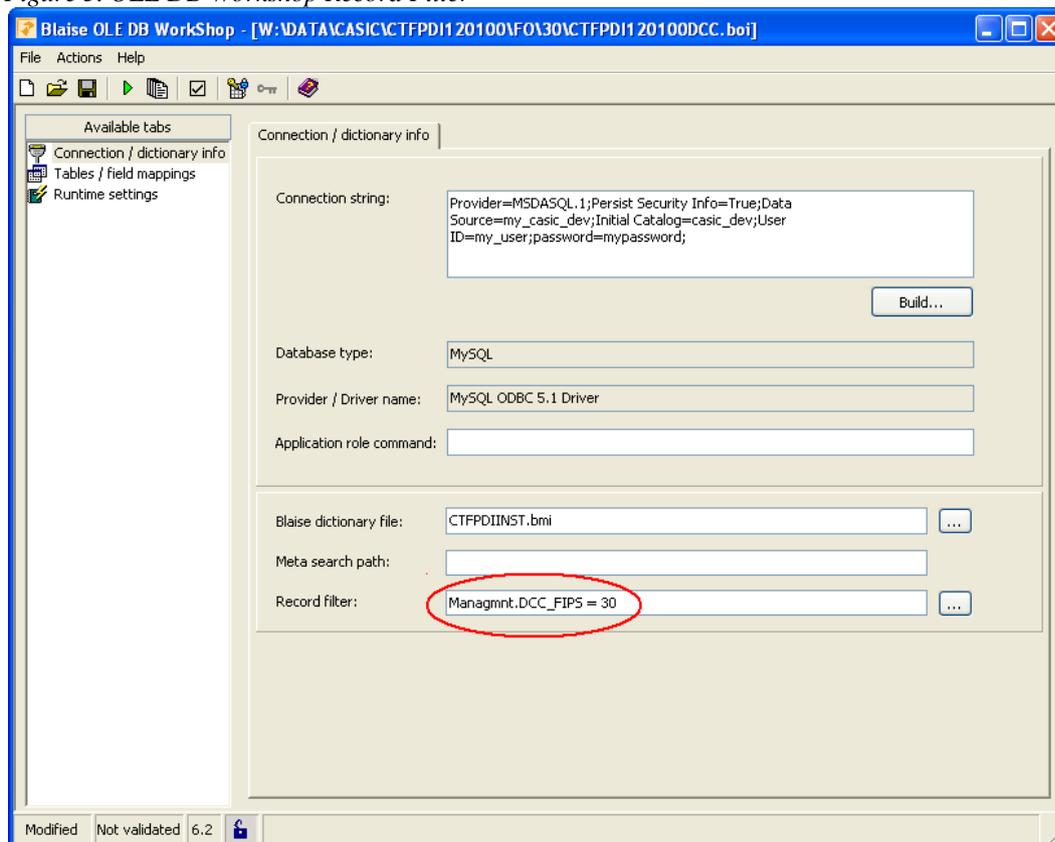
So to summarize, our VB.NET menu needs to react based on where a user is sitting (AD table), the user's role in the office (AD group), the office's role in the survey that is selected (CASIC_FAT table), and the specifics of the survey itself (CASIC_SurveyInfo table). All of this information together serves as a map of the different rights and functionalities we are providing to our users at the

survey level. So each piece of information is critical to ensure security and functionality of our Blaise instruments.

5 Managing CATI

For each survey, we have a main .BOI file defined for the entire sample. However, we need to virtually separate the data for collecting as well as editing the data. With all of the data centrally located in the same MySQL database, there is a need to manage the daybatches for each FO that collects CATI data. We also have to take into account six time zones. We create a folder for each FO involved in the survey for either data collection or editing. In this folder exists a filtered .BOI file that is used for creating daybatches as shown in Figure 5. This limits the daybatches as well as the forms that are accessible within CATI Management to the forms that are assigned to that DCC.

Figure 5. OLE DB Workshop Record Filter



For all other processing like interactive editing, reports, posting transactions, etc. we use the main .BOI file. The reason being, Blaise .BOI files are unaware of locked forms in other .BOI files. So in order to limit any possible collisions between users, we use the main .BOI file (many times with Record Filters) so that Blaise is aware of the potential locked records.

In the decentralized scenario, there were only two time zones of which Blaise needed to be aware: the time zone of the respondent and the time zone of the interviewer (and incidentally the Blaise dataset which was always the same as the interviewer). With the centralized scenario, Blaise now needs to keep track of three time zones: the time zones of the respondent, the interviewer, and the database. We learned that there are two places in the CATI Specifications where the time zone of the database plays a role. The first is the Crew Times. These must be defined in the time zone of the database, which is not necessarily the same as where the actual crew is sitting. The second is the Time Zone differentials. These must be defined with the time zone of the database as the “home” time zone – the one with a differential of zero.

6 Split-State Scenarios

Roughly a year ago we came across a situation where one of our FOs wanted to send part of its sample to another FO to be collected. We had tossed around the idea of split-state samples before but had never had to deal with them. To complicate matters, the FO needing to take part of the sample wasn't defined in our CASIC_FAT table as having DCC capabilities for that particular survey – yet another FO was defined as collecting data for the receiving FO's own sample!

We found the best solution was to add another column to the CASIC_SurveyInfo table called Add_Func (short for “add functionality”). The FO needing to take part of the other FOs sample for phoning received a value of 1 in this column, to indicate to our VB.NET menu that it needed to have the data collection related buttons activated, in spite of the fact that they were not designated as a DCC. Then all we needed to do was to run a Manipula program to update the field in the dataset to assign that FO as the DCC_Fips for the subsample of records needing to be transitioned. Our Manipula Record Filters took care of the rest.

7 Growing Pains

We would be remiss if we did not mention some of the growing pains that we suffered through to get where we are. The Blaise programming group at NASS was one of the first groups to move toward a centralized environment. So as we moved forward, we had to continually build bridges to the legacy systems so that the overall survey process would still function normally. As other pieces of the process began to centralize, new links to these pieces needed to be developed while maintaining the bridges so that parallel testing was possible. It is our goal that we will eventually no longer be passing data to the analysis and summary systems. The ETL will copy what is needed from the Blaise MySQL tables to WIP, and all post editing systems will retrieve what is needed directly from WIP.

The pre-survey processes are in the beginning stages of being centralized. The NASS Survey Management System (SMS) where the name and address files are loaded to create labels as well as the input files to our Blaise initialize process is being redesigned into a centralized environment. Once this is achieved, our front-end process in Blaise will need to be altered to accept a nationally generated file instead of multiple state-level generated files. Also, it is not impossible to imagine the Blaise database being populated for each survey by directly reading the centralized SMS databases.

We found a major bug in the services part of the Blaise software, which halted most of our forward momentum. The services would lock up and freeze all of the users until the services were restarted. It took us awhile to even diagnose the problem. Once we could reproduce the problem (although sporadically), we had to send Statistics Netherlands a set of files so they could hopefully reproduce the problem. They were able to set up a survey in the closest environment to NASS that has ever been implemented outside of NASS. They were successful in reproducing the freeze, and from that point they were in high gear to solve the bug. Our faith in the Blaise Team at Statistic Netherlands never faltered, but the task was not a trivial one. Several months later, a new beta version was sent in which they could no longer reproduce the error. The testing with this version will begin on the day this paper is due, February 29, 2012. We anticipate that the tests will prove that the system is sound and a production version can be released soon. This will allow us to move our more time critical, high-profile surveys into the centralized solution. Up to this point we have not abandoned our forward progress, but limited it to smaller surveys and surveys with a large data collection window.

As systems become more and more centralized, concepts of databases working together arise. “Why can't these ASCII files be eliminated?” “Why can't this database be aware of this other database?” “This information should be stored in a common table shared by all.” All of these exciting questions and observations are valid and usually good ideas. As the overall staffing numbers continue to decline at NASS, and the need to continue the integrity of the on-going survey programs remains, the challenge becomes reigning in those who want to implement these ideas immediately. Our goal is to create a sound survey process utilizing all of the efficiencies possible with enough forethought that the

systems communicate with each other effectively and minimal rewrites are needed. We want to avoid the feeling that we have jumped into a rowboat and started floating only to realize that the oars are still on the shore.

Using Computer Audio Recorded Interviewing (CARI)

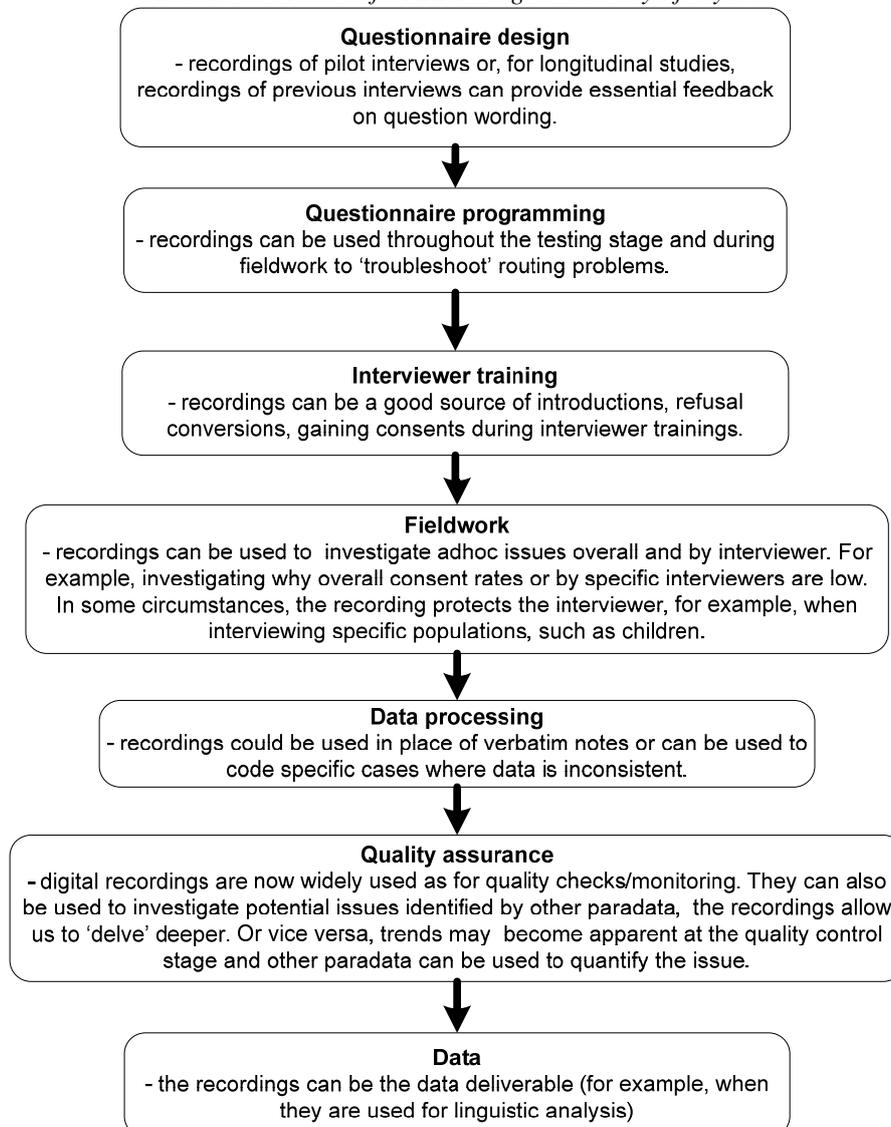
Rebecca Gatward, Gina-Qian Cheung and Patty Maher,
Survey Research Center, University of Michigan

1 Introduction

Computer Audio Recorded Interviewing (CARI) is now among the standard set of tools used during survey data collection. Prior to digital recording monitoring systems in centralized telephone interviewing facilities provided valuable and instant feedback on interviewer performance. This tool also provided researchers with the ability to collect timely feedback on the design of questions. However, until CARI, there was no equivalent systematic system for face to face or decentralized telephone interviewing – other than using portable recording devices.

CARI is now used to provide a valuable insight into factors that affect data quality throughout the survey lifecycle, from question design to responding to queries following data delivery. The following illustration describes some of the ways CARI is used during the survey process.

Illustration 1. Use of CARI through the survey life cycle



Although CARI has clearly developed into an important (and almost standard) tool in the survey data collection process, little has been shared across the industry about best practices or challenges implementing this technology and utilizing the resulting data. University of Michigan, Survey Research Center, Survey Research Operations organized a two day meeting to bring together five survey research organizations. The key objective of the meeting was to share best practices and challenges encountered whilst implementing Computer Audio-Recorded Interviews (CARI) and utilizing the data. The event was held on 14-15 November 2011. The following organizations participated:

- Research Triangle Institute (RTI),
- Statistics Canada,
- Statistics Netherlands,
- University of Michigan-Survey Research Center
- US Census Bureau and
- Westat.

The purpose of this paper is to share details of the information and discussion that was exchanged during the two day meeting. All omissions or interpretations are the responsibility of the authors. The paper should provide a useful status report of how CARI is currently being used within the participating organizations and an insight into some planned developments and new ways they would like to use CARI in the future.

2.1 Methods: Operations and Sampling

The focus of the first part of the two day meeting was on sharing experiences of administering and implementing CARI within each organization. To facilitate this, participating organizations were asked to prepare presentations around two sets of questions. The first set is based on administration and are listed below;

Sharing survey administration experiences with this technology –

- Recording across samples. What percentage of sample is recorded? Does this percentage change during the production period? What are the reasons for the change?
- Recording across data collectors. How is this controlled? Are there attributes of the interviewers that are taken into account when recording? Does this change during the production period? What are some of the reasons for the change?
- Selection. What are the recording variables? Full interview, selected questions, or random questions?
- Reviewing the recording. How are the recorded interviews reviewed? Who does this and do the reviewers go through the full recorded interviews or segments?
- Evaluations. What are the main criteria for reviewing the recording?
- Feedback. How are the results of the review communicated to interviewers and researchers?
- Implementation. How does this technology affect your organization's quality of data collection? Has there been feedback from users about the technology implementation? (i.e., does recording slow down the survey instrument during the interview?)

Common themes emerged as each organization shared details around the administration of CARI and specifically, CARI as a tool to facilitate quality control. Some of the key themes were discussed in more detail during the meeting and will be covered later in this paper, the following is a brief summary of all the themes.

- **Rate of recording vs. rate of evaluation**

A key decision is setting the rate of recording i.e. how many interviews are recorded (n=completes) and then of these, what proportion are evaluated (% of completes). Factors that influence these decisions included practical concerns, such as, the size of the audio files and possible effect on transmission or interviewing speed. The rate of recording varied across

organizations; ranging from, recording all interviews and evaluating a proportion of these recordings, record at a predetermined rate and then evaluating a percentage of these recordings to only recording what will be evaluated.

Frequency of interviewer monitoring/evaluation also varied by organization – this variation seemed to be based on how field interviewers were organized – if they work across different projects, then the cycle of evaluations is based on time – for example, they are evaluated so many times a month. If they typically work on one project then evaluations are based around the cycle of a project – for example, the first two interviews at the start of fieldwork and then a set number or interval (unless there are specific quality concerns).

- **Selection of cases to be evaluated**

Following on from determining the rate of recording, the selection of cases to be evaluated is also a key decision. The typical rate across organizations represented at the meeting was ten per cent, however, criteria used to select the cases and how/whether this rate varies by interviewer was not consistent. Issues around recording and evaluation across samples and data collectors were discussed in some detail during the meeting. Key points from the discussion are summarized below:

- Evaluation rates can vary by type of project or by type of interview. Some clients may also request a higher level of monitoring on some projects. Generally, 10 per cent was the norm.
- Selection of cases for evaluation can also be based on answers to survey questions (rural cases, high/low income, and other demographic variables). Although there was strong support for selection of cases based on questionnaire content for evaluating the questionnaire, whereas selection of cases for evaluation should be focus on interviewer attributes or sample distribution.
- Is it sensible to preselect cases to be evaluated at the interviewer level based on workload? For example if the rate of evaluation is set at 15 per cent of assigned cases – which means you actually get less than 10 per cent because they do not result in completed interviews. Should the sample be selected by recording 15 per cent and then evaluate them all?
- An alternative is to record a set number of an interviewer’s initial interviews completed and then evaluate a proportion of subsequent cases completed. Using this method results in all interviewers having the same number of cases evaluated early on in the field period.
- Some organizations also set a target of a certain number of interviews per interviewer to be evaluated per week.
- There was some discussion about adjusting evaluation rates by experience of interviewer (for example, new to survey organization, experienced interviewer or new to a project). It was felt that this is useful to an extent, but some methods literature shows experience is not always correlated with better quality and fewer errors.
- Selecting a small percentage of non-interviews who were then followed up by telephone was a procedure followed by one organization – others thought this was good practice.
- It is possible to identify those interviewers who are not following procedures to save time by looking at data. For example, higher levels of missing respondent telephone numbers, missing data, and higher refusal rates for CARI, item level timings – faster timings could indicate falsification or skipping more complicated or lengthy sections of the interview.

- **Selection of content to be recorded**

The selection of variables to be recorded also varies by organization – the various methods are listed below;

- Questions or groups of up to three questions are selected from across the entire interview – totaling between around 8 to 12 questions. If more than 12 are recorded then probabilities are set so only 8-10 of the items are scored.
- Pre-defined set of questions are recorded, the questions are selected based on key analytic variables, such as, new items, challenging questions or sets of questions, key routing variables, consent questions and incentive payment scripts.
- Variables are chosen to be recorded but specific variables are used for each component of an evaluation, for example, different variables are used for quality assurance than for behavior coding.
- One organization keeps track of which variables are being evaluated, from this it is possible to identify those variables which are being missed.

Again, the discussion around how we select what is recorded generated some topics for further discussion by the group after the meeting. These include;

- Stratifying for selection of segments – how can paradata inform the selection of segments to be recorded?
- Best practices for selection of items (including, electing across the entire interview, optimum length of recording).
- Difference in CATI vs. CAPI.

- **Evaluation Criteria**

As might be expected, each organization uses a standard set of evaluation criteria, which are, broadly, consistent. Interviewers are evaluated according to adherence to standard interviewing techniques (skills and behavior) and knowledge and implementation of survey specific procedures. Generally, interviewer errors are rated according to the type and severity of an error – rating errors as critical, major or minor is common. An overall rating is then based on the number and severity of errors. Feedback is provided to interviewers via a standard report – and shared via an interviewer supervisor. It was noted that, in some organizations, aggregate performance data are produced by project, supervisor and across projects.

The following two ideas emerged during the discussion on evaluation criteria which were noted for future discussion (post-meeting) by the group.

- Could a common framework for evaluation be conceived?
- How to best use paradata attached to CARI files?

- **Training – monitors and interviewers to collect and evaluate recordings**

Training provided to interviewers in the quality control process ranged in detail. The lowest level of training was simply informing interviewers that they are required to ask the respondents the consent to record question at the beginning of an interview. More detailed training included interviewers being trained on the standard QC approach – i.e. outlining the evaluation criteria, how interviews will be evaluated, what they should expect from the feedback process and their role in the process. Interviewers are also trained, as necessary in the practical aspects of CARI for example, setting up an external microphone (if used) or setting up equipment for recording when interviewers are carrying out decentralized telephone interviewing.

The training provided to those completing the evaluations was generally, more extensive. These staff is generally very experienced interviewers so they have knowledge of interviewing skills but need to be trained in the evaluation criteria and the evaluation tool or

system. Inter-rater reliability is used by some organizations to monitor and achieve consistency of scoring, although this is not the norm.

- **Legal Issues**

Legal issues is a topic that was not discussed in detail but was identified as one that should be included in further post-meeting discussions. Items for future discussion include:

- best practices to share;
- file destruction practices;
- are there appropriate ways to train interviewers to respond to R questions regarding consent and legal issues regarding recordings; and
- Long-term storage of recordings.

- **Affects of CARI**

The overall consensus amongst the group was that CARI has facilitated an improvement in data quality – primarily through enhanced feedback on interviewer performance but it also encouraged ease and frequency of monitoring by clients and project teams which assisted the questionnaire development process which contributes to better survey measures and data quality.

2.2 Methods: Operations and implementation of CARI

Organizations were also asked to share their experience around implementing CARI - the following questions were designed to facilitate this.

Sharing the implementation experiences with CARI

- Tool. What tool did you use? Why did you select it?
- Technology. What is the recording technology used? Audio only, audio and video?
- Storage. What is the recording storage medium? Are the recordings sortable by interviewer ID or questionnaire number?
- Quality of Sound. What problems or challenges have you faced on the quality of the sound file?

A summary of the discussion of the key issues around implementation of CARI is provided below.

- **Audio or Audio and Screen (or video)**

Blaise CARI is used by all organizations who participated in the meeting – the output from CARI is an audio file and, if requested, a screen shot of the corresponding screen. University of Michigan also extensively uses a system that is controlled from outside the Blaise questionnaire via a capture list. The system records a video of the sections of an interview as determined by the capture list (using, Camtasia®, the video capturing software).

There was some discussion during the meeting about the added value of having a video over a screenshot – and varying views on this. It was generally agreed that the major advantage of video above audio and a screen shot is that video captures interviewers' actions and interaction between questions – for example; 'off script' discussions which might highlight poor interviewer behavior, how soft checks are handled, what feedback is provided and frequency of feedback by the Interviewer. It also captures interviewer: respondent interaction when an external (i.e. non-Blaise) program is launched. For example, the completion of an event history calendar or cognitive test which is launched via a DLL from within the Blaise questionnaire can still be captured and displayed for evaluation.

Having a video image of the laptop screen allows staff to see if an interviewer has any other programs open on their laptop at the time of the interview that they should not be using. The video can also provide a useful insight – especially at the piloting stage – of any usability issues of new question design or procedures.

- **Storage/file size/performance/sound quality**

One of the main practical concerns organizations faced during the implementation of CARI was file size and the possible impact on other systems – such as interview speed and transmission of files. However, these issues and risks were controlled by saving files in formats that produced files of a smaller size, setting a minimum file size which removed empty recordings and compressing files to a lower but still usable quality.

Sound quality was not an issue, the only problems are a result of unavoidable circumstances during face to face interviews.

The discussion turned to ways organizations would like to use CARI that they are not able to currently and planned developments of current systems. The following were identified;

- **Record all interactions between a face to face interviewer and the respondent/informant**

A complete record of interactions with the respondent is available for centralized telephone interviewing, however, for face to face or decentralized telephone interviewing this stage of the interview is not yet recorded. Organizations agreed that recording the full interaction between respondent and interviewer from doorstep to the start of the Blaise questionnaire would be incredibly useful. In addition, to complete the entire process, the recordings would need to include any telephone contact they make with the respondent prior to arriving on their doorstep. Ideally, it could be presented like an audit trail of all contact attempts during the entire process to gain cooperation. There was some discussion about the type of technology that would be used for CARI outside the interview and legal and ethical issues were also noted.

- **New technologies and capturing CARI using them?**

The following new technologies were mentioned as possible ways to capture CARI, all organizations expressed an interest in investigating these new technologies for CARI.

- Capturing CARI on smart phones
- Capturing CARI on other hand-held or non-traditional devices
- Using Skype or other systems to capture video files.

- **Flexibility**

The ability to make changes ‘on the fly’ varied across the organizations and was dependent on the systems they have developed or if they are using CARI or another recording system. Organizations commented they would like to have more flexibility around making changes to what is being captured – such as changing the questions or sections that are being recorded during data collection, change the rate at which interviews are being recorded overall or vary the rate by interviewer. In Blaise CARI it is possible to change what is being recorded – but it is not possible to vary what is being recorded for different cases or by interviewer. It was agreed that the goal should be to capture recordings of all interviewers across the full questionnaire. The ability to randomize the capture list would facilitate this.

- **Other uses of CARI**

- Can or should CARI be used in lieu of verbatim recording on (some) open-ended items?
- Key word searches from database of recordings to identify cases to evaluate

3 Conclusion

Recent versions of Blaise have supported CARI and Blaise CARI is now well developed and supports the current use of CARI as outlined above. Blaise CARI has been developed with the user in mind i.e. providing a tool that has many options (in the CARI settings) that address the practical issues that need to be considered when implementing CARI – such as, the ability to control the maximum and minimum file size, recording time and file format. It is also possible to operationalize many of the requirements essential to making CARI a useful tool within the CARI settings in Blaise. For example, the response to an assigned consent question will control if recording is activated and how much of an interview is recorded. Blaise CARI cannot be controlled in anyway by the interviewer and as is preferable with any CARI system, interviewers are not aware when the system is recording. Blaise CARI also produces a log file, providing a trail of when the recording starts and stops during the interview and the name of the corresponding sound and image file (a screen shot).

A limitation of the Blaise CARI system is that it only records the interaction which occurs between the interviewer and respondent when they are within the Blaise questionnaire. Blaise does not capture any interaction which occurs when external programs are used for data collection, from within Blaise.

During the meeting at the University of Michigan an idea was formulated to open up CARI in such a way that the user can specify to use another recorder than the built-in audio recorder. This would make it possible for the user to rely on their own recorder component while having the full benefits of Blaise CARI. This idea, also known as ‘the CARI hook’, has by now been implemented in a beta version of Blaise 4.8. The CARI hook can for instance be used to record screen video (using for instance Camtasia®) but it can also be used to record VOIP. Note that the user will need to create an ActiveX® component to interface between Blaise CARI and their recorder component.

Participating organizations agreed to continue to meet and share updates periodically.

Blaise Audit Trail Data in a Relational Database

Joel Devonshire, Youhong Liu, and Gina-Qian Cheung
Survey Research Center, University of Michigan

1 Introduction

One principle that has become increasingly clear to many technical team members at the University of Michigan's Survey Research Center, Survey Research Operations (SRO) is that paradata are only as useful as the ease with which they can be accessed, manipulated, and analyzed by end users. Even for a relatively small survey research study, the sheer volume of paradata that can potentially be collected can be overwhelming. A conundrum often exists between needing to have high-level questions guide decisions about what data to collect versus needing to see all of the data first in order to know what questions are possible to answer. The end result is often the collection of large quantities of data that are never examined in any meaningful way. To the extent that this happens, it may represent an inefficient use of technical resources and project funds, as well as potentially less robust survey methods. In other words, such an approach may not be optimally reliant on principles of responsive survey design. The concept of responsive design implies that all available paradata inputs be evaluated during the survey design phase as potential indicators of cost and error, and that decisions be made during data collection according to the insights gleaned from such inputs (Groves & Heeringa, 2006). This, in turn, implies that the paradata inputs are readily available for real-time analysis during data collection.

This paper focuses on one specific and very important element of paradata: the Blaise Audit Trail, or ADT. In particular, we describe the development of a new way to store ADT data that we hope will enable end users to more easily access the raw data as it is collected and to write their own custom queries using tools of their choice. By storing most of the raw elements of the ADT file in a relational database format, and doing so on a nightly basis as data collection is happening, a wide range of options become available for data analysis that are more flexible and efficient.

1.1 Background

In 2004, SRO created an application called "ATReport." The main purpose of the ATReport application was to consolidate and process individual ADT files for analysis. It functioned by creating a local Microsoft Access database on the user's desktop, and storing aggregate data in a handful of predefined tables. The application was also able to generate standard summary reports (e.g., timings by item, section, or interviewer), and could identify specific keystrokes during the interview (e.g., backing up and changing answers, invoking help, switching languages, etc.). While this application significantly advanced the ability to use ADT data, it had some distinct disadvantages, ones we surmise are related to why the application is used only rarely at SRO.

For example, when a user wishes to analyze ADT data with the existing system, she must first gain access to the individual ADT files, which she may or may not already have access to. She then needs to install the ATReport application on her local machine, and process the files of interest. The processing itself can take many hours, especially for a large number of lengthy interviews. When the processing is complete, the user is left with a local database that only contains predefined aggregated data. While the user can then go on and write her/his own queries, the results are limited to what can be gleaned from the ATReport tables. If a user wants to isolate one specific case and examine it in more detail, then she needs to use a different SRO application called Playback, which uses the ADT and BDB files to recreate the individual interview session.

To illustrate why these aspects of usability are important, consider a hypothetical scenario in which a survey director is concerned about reports in the field that interviewers are getting stuck on one particular field in the data model. She finds out about this problem over the weekend and she would

ideally like to have a follow-up plan to recommend at a team meeting on Monday. Specifically, she'd like to know the overall amount of time all interviewers spend in this field, and obtain a report by interviewer so she can identify specific staff members of concern. She also wants to assess what behavior interviewers engage in immediately before, during, and after entering this field. Are they backing up, for example, or entering an F2 comment? Finally, she wants to know how interviewers resolve the issue; do they tend to suspend out of the instrument at that point, or fill in random data, or something else?

While it is true that we currently have the tools to answer all of the questions raised in this scenario, it would be a somewhat complicated process. For example, it is currently the role of one of the technical team members at SRO, the Data Manager, to have access to the data and to create data sets that can be analyzed by others. In this scenario, the Data Manager would need to be contacted, and arrangements would have needed to be made to allow access to a range of ADT files. The processing and analyzing of them would have needed to wait until Monday morning and would have been time-consuming and inefficient.

Compare this with a vision in which the survey director is able to open up a web browser at home, log in to our project management site, WebTrak, and run a query on the ADT data by selecting a few menu options. The results are automatically up-to-date, and the query can easily be saved to run again during the next week. After running the query, she sees that the "problem" is really only happening with 5 out of 60 interviewers, and they are all new staff. It seems that they are repeatedly encountering a hard check in Blaise due to incorrect data entry a few questions earlier. They try repeatedly to enter the same data, and then give up and suspend out of the instrument. Going into Monday's meeting, she therefore knows that this is a staff training issue rather than a programming issue.

1.2 A New Approach: The Centralized Relational Database

To better utilize the audit trail data and to provide more security and efficiency, users have requested that we consolidate all the individual audit trail files into a centralized relational database. The advantages of this approach include:

- Users will not need direct access to the ADT files themselves, which are often located in protected server locations. This helps to ensure that ADT files, and the potentially identifying respondent information included in them, are not unnecessarily copied to multiple – and undocumented – locations.
- Instead of requiring users to create their own ADT database in MS Access, having a central relational database located on a secure SRO server is more efficient, and specific user permissions could be granted as needed. Users link to the tables instead of creating new ones. Unlike point #1, this speaks to the efficient use of the end-data, not the ADT files themselves.
- By processing the ADT files through our automated nightly interview data merge process, we greatly reduce the time it takes to analyze data, since the ADT file consolidation would already be complete. In addition, the nightly processing time itself would be reduced since only a small number of ADTs would be processed each night.
- Nightly processing the ADT data into other nightly batch reporting systems, such as WebTrak, to provide automatic detail on problem cases or interviewer behavior (e.g., aggregate timings by interviewer).
- Users could connect to the ADT database using familiar tools such as SAS or MS Access. They can create their own queries and more easily merge to other databases (e.g., SurveyTrak paradata). This is possible with the current implementation, but would be made easier with a central database.

2 Database Structure

In its current state of development, the database is minimal, comprising only five operational tables. For the purposes of this paper, only two will be described, tADTField, which contains the raw data from the ADT files, and tSuspendVariables, which captures the specific Blaise fields in which instrument suspensions occurred.

The table tADTField contains columns for most of the individual pieces of data for the ADT, and these are listed in Table 1. Notice that the table is able to capture information about the interviewer, the data model, the number of visits to the data model and each field, the time spent within each field, and a number of other Boolean values related to hot keys and events within each Blaise field.

Table 1. Structure of table tADTField

Field Name	Data Type	Size	Description
ProjectID	Text	30	Unique project identifier
CaselD	Text	30	Sample ID within a project
FldSeq	Long Integer	4	Sequential counter; increments by one for each ADTrow
MetaName	Memo	-	Name of Blaise data model
MetaTime	Text	200	Date/Time Blaise data model was created
UserID	Text	30	Interviewer ID for this particular instrument entry
FldName	Memo	-	Blaise long field name
BlockName	Text	100	Blaise block name
FormVstNum	Long Integer	4	Instrument visit number; increments by one for each entry
FieldVstNum	Long Integer	4	Field visit number; increments by one for each entry
PrevLeaveLineNo	Long Integer	4	Previous field Blaise line number
EnterLineNo	Long Integer	4	Current field entered Blaise line number
LeaveLineNo	Long Integer	4	Current field exited Blaise line number
ISLEAVEFORM	Long Integer	4	Interviewer exited Blaise at this field (Yes/No)
EnterDate	Text	30	Date interviewer entered the Blaise field
EnterTime	Text	30	Time interviewer entered the Blaise field
LeaveDate	Text	30	Date interviewer left the Blaise field
LeaveTime	Text	30	Time interviewer left the Blaise field
Fld_Time	Text	30	Time (sec) spent within the Blaise field
Fld_SS	Text	30	Time (sec) spent within the Blaise field (different format)
Fld_Time_Mss	Long Integer	4	Time (milisec) spent within the Blaise field
Btw_Time	Text	30	Time (sec) spent between last Blaise field and this one
Btw_SS	Text	30	Time (sec) spent between last Blaise field and this one
Btw_Time_Mss	Long Integer	4	Time (milisec) spent between last Blaise field and this one
Adj_Time	Text	30	Fld_Time + Btw_Time
Adj_SS	Text	30	Fld_Time + Btw_Time
Adj_Time_Mss	Long Integer	4	Fld_Time + Btw_Time
RspLat_Time	Text	30	Time (sec) between entering field and first keystroke
RspLat_SS	Text	30	Time (sec) between entering field and first keystroke
RspLat_Time_Mss	Long Integer	4	Time (milisec) between entering field and first keystroke
Key_Count	Long Integer	4	Number of keystrokes while in Blaise field
Enter_Value	Memo	-	The value of the Blaise field upon entry
Leave_Value	Memo	-	The value of the Blaise field upon exit
Leave_Cause	Text	50	Action that initiated interviewer to leave Blaise field
Leave_Status	Text	50	Field leave value is normal or DK/RF
Prev_Lang	Long Integer	4	Language was switched to previous while in field (Yes/No)
Next_Lang	Long Integer	4	Language was switched to next while in field (Yes/No)

Set_Lang	Long Integer	4	Language was set while in field (Yes/No)
CtrlL_SetLang	Long Integer	4	Language was changed with hot key while in field (Yes/No)
ALTXExit	Long Integer	4	Alt-X interview suspension was initiated at this field (Yes/No)
RemClk	Long Integer	4	Interviewer remark was initiated at this field (Yes/No)
RemChng	Long Integer	4	Interviewer remark was changed at this field (Yes/No)
QHelp	Long Integer	4	OXQ Help was initiated at this field (Yes/No)
BlaiseHelp	Long Integer	4	Blaise Help was initiated at this field (Yes/No)
Error_Esc	Long Integer	4	Blaise check was closed at this field (Yes/No)
Error_Esc_Text	Memo	-	Text of the Blaise check encountered before closing
Error_Supp	Long Integer	4	Blaise check was suppressed (Escape) at this field (Yes/No)
Error_Supp_Text	Memo	-	Text of the Blaise check encountered before suppressing
Error_Jmp	Long Integer	4	Blaise field that interviewer jumps to after Blaise check
Error_Jmp_Text	Memo	-	Text of the Blaise check encountered before jumping
Media_Start	Long Integer	4	Blaise launched media file while in this field (Yes/No)
Mouse_Click	Long Integer	4	Any mouse click detected while in this field (Yes/No)
F1	Long Integer	4	F1 hot key was pressed while in this field (Yes/No)
F2	Long Integer	4	F2 hot key was pressed while in this field (Yes/No)
F3	Long Integer	4	F3 hot key was pressed while in this field (Yes/No)
F4	Long Integer	4	F4 hot key was pressed while in this field (Yes/No)
F5	Long Integer	4	F5 hot key was pressed while in this field (Yes/No)
F6	Long Integer	4	F6 hot key was pressed while in this field (Yes/No)
F7	Long Integer	4	F7 hot key was pressed while in this field (Yes/No)
F8	Long Integer	4	F8 hot key was pressed while in this field (Yes/No)
F9	Long Integer	4	F9 hot key was pressed while in this field (Yes/No)
F10	Long Integer	4	F10 hot key was pressed while in this field (Yes/No)
F11	Long Integer	4	F11 hot key was pressed while in this field (Yes/No)
F12	Long Integer	4	F12 hot key was pressed while in this field (Yes/No)
CtrlD	Long Integer	4	Ctrl-D hot key was pressed while in this field (Yes/No)
CtrlR	Long Integer	4	Ctrl-R hot key was pressed while in this field (Yes/No)

The table tSuspendVariables contains columns to track up to 10 instrument suspensions, and also easily identifies where the last suspension occurred. Table 2 lists these columns.

Table 2. Structure of table tSuspendVariables

Name	Type	Size	Description
ProjectId	Text	30	Unique project identifier
CaseID	Text	30	Sample ID within a project
vTotalSuspend	Long Integer	4	Total number of suspends for this sample line
vCaseComplete	Yes/No	1	Is the case completed?
vSuspendedVariable1	Memo	-	Blaise field at which first suspension occurred
vSuspendedVariable2	Memo	-	Blaise field at which second suspension occurred
vSuspendedVariable3	Memo	-	Blaise field at which third suspension occurred
vSuspendedVariable4	Memo	-	Blaise field at which fourth suspension occurred
vSuspendedVariable5	Memo	-	Blaise field at which fifth suspension occurred
vSuspendedVariable6	Memo	-	Blaise field at which sixth suspension occurred
vSuspendedVariable7	Memo	-	Blaise field at which seventh suspension occurred
vSuspendedVariable8	Memo	-	Blaise field at which eighth suspension occurred
vSuspendedVariable9	Memo	-	Blaise field at which ninth suspension occurred
vSuspendedVariable10	Memo	-	Blaise field at which tenth suspension occurred
vDate1	Text	30	Date on which first suspension occurred
vDate2	Text	30	Date on which second suspension occurred
vDate3	Text	30	Date on which third suspension occurred
vDate4	Text	30	Date on which fourth suspension occurred
vDate5	Text	30	Date on which fifth suspension occurred
vDate6	Text	30	Date on which sixth suspension occurred
vDate7	Text	30	Date on which seventh suspension occurred
vDate8	Text	30	Date on which eighth suspension occurred
vDate9	Text	30	Date on which ninth suspension occurred
vDate10	Text	30	Date on which tenth suspension occurred
vTime1	Text	30	Time at which first suspension occurred
vTime2	Text	30	Time at which second suspension occurred
vTime3	Text	30	Time at which third suspension occurred
vTime4	Text	30	Time at which fourth suspension occurred
vTime5	Text	30	Time at which fifth suspension occurred
vTime6	Text	30	Time at which sixth suspension occurred
vTime7	Text	30	Time at which seventh suspension occurred
vTime8	Text	30	Time at which eighth suspension occurred
vTime9	Text	30	Time at which ninth suspension occurred
vTime10	Text	30	Time at which tenth suspension occurred
vLastSuspendedVariable	Memo	-	Last Blaise variable at which a suspension occurred
vLastSuspendedDate	Text	30	Date on which the last suspension occurred
vLastSuspendedTime	Text	30	Time at which the last suspension occurred

3 Integration with Other SRO Systems

As mentioned above, ADT files will be processed on a nightly basis when all active production projects at SRO undergo the Blaise data merge. Prior to this implementation, nightly ADT processing at SRO was limited to a calculation of overall interview length, which was then written to a column in our production sample management system, SurveyTrak. This new process will allow a tight integration of detailed ADT information into all other aspects of daily project management.

For example, SurveyTrak currently has the capability to capture a wide array of paradata, including things such as contact attempts, appointment times, incentives and letters sent to respondents, characteristics of the sample (e.g., mode and language of collection and interviewer assignment), and so forth. One direct advantage of processing ADT files nightly is that keystroke data can then be joined with any of the other “real-time” SurveyTrak paradata that we collect. One example of how this might be useful is in the early stages of data collection. If reports were coming in from the field that interviews were taking much longer than usual, one could easily query interview lengths by mode, for example, to see if face-to-face interviews were taking longer than telephone, or whether the increase in length was due to some other factor.

This integration also allows these kinds of questions to be included in the reporting specifications during the initial project design. Our daily programmed Field Progress Reports (FPR) could include information on keystroke data, and key queries could be built into WebTrak, our web-based project management application. Such integration encourages project staff to consider questions related to keystroke data, and could potentially become a key component in the work we do at SRO.

4 Practical Applications

As is often the case with paradata, the practical applications of ADT data in a relational database depend significantly on what questions are of interest to survey directors and project managers. We have already mentioned above a few relevant questions or scenarios in which this data could be applied. In this section, however, we present a more detailed workflow diagram of how a few key questions might potentially be approached with the new system.

4.1 Estimating Length of Recorded Interviews

A real-life scenario for one of the authors of this paper involved the following scenario. We were asked by project management to take a proposed digital recorded interview (DRI) capture list specification¹ and estimate how long the actual DRI recordings might be if the capture list were used (and before the recordings were viewed by our quality control team). The project managers wanted capture list timings summaries by sample ID and instrument section, using existing data that had recently been collected.

Because the capture list is a document that can be easily imported into a database system such as MS Access, the Blaise fields specified in the capture list can be joined with the ADT database data to pull out only the fields included in the capture list. In SAS, this was handled by a simple PROC SQL statement:

```
PROC SQL;
  SELECT DISTINCT a.*
  FROM tADTField a JOIN CaptureList b
  ON a.fldname = b.BlaiseName_Full
  ORDER BY a.caseid, a.fld_name;
QUIT;
```

From there, instrument sections can be defined via queries that examine the Blaise field name prefixes, and section and interview subtotals can be calculated with simple SQL queries:

```
PROC SQL;
```

¹ A DRI capture list is a specification document (text) that defines which Blaise fields should be digitally recorded during the interview. When a sample line is flagged to record, Blaise uses the capture list to control when the recording software pauses and resumes the recording of sound and screen activity.

```

SELECT caseid,
       sum(Adj_Time) as CaptureList_Length
FROM capture_list
GROUP BY caseid;
QUIT;

```

The resulting data might look something like this:

Table 3. Interview Length Calculated Using Capture List Specification

CaseID	Length of Interview
6000040020	0:28:26.274
6000260010	0:37:58.675
6000470010	1:11:30.885
6000510010	0:47:37.999
6000540020	0:53:29.781
6000540021	0:56:10.560
6000600010	0:34:11.024

4.2 Examine Specific Keystroke Behavior

In a relational database format, certain kinds of questions become incredibly easy to answer. For example, finding out what fields were associated with F2 comments by the interviewer is as simple as writing the following query:

```

PROC SQL;
  SELECT DISTINCT fldname
  FROM tADTField
  WHERE F2 = 1
  ORDER BY fldname;
QUIT;

```

To determine what specific fields had values changed by the interviewer (as opposed to retaining their preloaded values), one could use the following query:

```

PROC SQL;
  SELECT DISTINCT fldname
  FROM tADTField
  WHERE Leave_Value <> Enter_Value
  ORDER BY fldname;
QUIT;

```

To determine what fields were associated with backing up to the previous field, and to count how many times this occurred for each field, one could use this query:

```

PROC SQL;
  SELECT DISTINCT fldname,
  Count(fldname) as Count

```

```

FROM tADTField
WHERE Leave_Cause IN ("Move Left","Move Up")
GROUP BY fldname
ORDER by fldname;
QUIT;

```

If one wished to focus in on a particular field and develop a summary keystroke report, examining how these changed between versions of the data model, one could use the following query:

```

PROC SQL;
SELECT ProjectID,
       MetaTime AS DataModel_Version,
       Count(MetaTime) AS N,
       Avg(Fld_Time_Mss) AS Avg_FldTime,
       Avg(Btw_Time_Mss) AS Avg_BtwTime,
       Max(FieldVstNum) AS Num_Visits,
       Avg(Key_Count) AS AvgKey_Count,
       Max(Key_Count) AS MaxKey_Count
FROM tAdtField
WHERE fldName = "SecB.Marriage2.B055_"
GROUP BY ProjectID, MetaTime;
QUIT;

```

The resulting output might look something like this:

Table 4. Keystroke Summary for One Blaise Field By Data Model Version

DataModel_Version	N	Avg_FldTime	Avg_BtwTime	Num_Visits	AvgKey_Count	MaxKey_Count
Friday, January 06, 2012 1:08:04 PM	5	36781	48	3	1	3
Tuesday, January 24, 2012 1:48:20 PM	2	8944	48	2	1	2
Wednesday, February 01, 2012 9:50:26 AM	1	592	43	1	2	--

4.3 Examining Interview Suspensions and Checks

A final example that will be considered in this paper relates to the analysis of particular events during the interview session, namely suspensions and hard or soft checks. While we currently have tools at SRO to examine individual cases to see where or how a check or interview suspension occurred, we have had no easy way to analyze the entire data set. By having the ADT data stored in a relational database, we can now easily find patterns of interviewer behavior, such as common suspension points, where errors or checks tend to occur, and how interviewers tend to respond to these checks.

Using the table tSuspendVariables, for example, one could determine how often the interview is suspended before a critical junction, or within a complex set of questions. One could see whether interview suspensions are, on average, associated with longer total interview times, and one could determine how long, on average, the duration is between entries into the instrument.

For soft checks, one could see how many times interviewers suppress the check, and whether this changes as a function of interviewer experience, data model versions, or other factors. Since the

database stores the text of the check, one could also calculate summaries of the checks themselves, seeing if certain ones are triggered more than others. One could see whether checks and suspensions as a whole were more common toward the beginning of data collection, or occurred more frequently among certain interviewers.

One interesting example involves the detection of possible interviewer falsification, in which the interviewer attempts to enter erroneous data. For example, if a query were written to isolate all of the hard checks encountered during the interview, it might look something like this:

```
PROC SQL;
  SELECT ProjectID,
         CaseID,
         UserID,
         FldName,
         Enter_Value,
         Leave_Value,
         Error_Jmp,
         Error_Jmp_Text
  FROM tAdtField
  WHERE tAdtField.Error_Jmp =1
  ORDER BY tAdtField.CaseID;
QUIT;
```

Let's say that this query finds that the following check occurred several times: "This is not a valid zip code. Please check." The analyst then looks at the value stored in the field Enter_Value and discovers many instances of the value "99999." In this case, the interviewer is attempting to bypass the zip code question by entering in fake data.

5 Further Development and Conclusion

The above examples are just a few of the kinds of things that potentially could be examined using the ADT data in this format. However, this initiative is still in its early stages, and one could imagine the development of additional tables to hold aggregate data based on the raw data in the ADT. For example, if project managers knew in advance that they wanted to focus on a particular group of interviewers or sample from a particular geographic area, summary tables could be automatically updated daily with this aggregate data. In another example, if regular timings reports for each section of the interview were required, a crosswalk table could be created to link each field name within the data model to a section name (if the Blaise instrument was not already programmed in straightforward section blocks). Stored procedures could then be run daily to update a table of aggregate section timings. These types of enhancements would make the raw ADT data easier and faster to analyze.

Ultimately, the development of such a tool will be driven by the questions that need to be answered to assist in collecting high quality survey data. However, paradata often involves a chicken-and-egg scenario in which some questions are not asked because people do not know the data exist or do not readily have access to them. We hope that by making audit trail data more easily accessible, the data will be used more often and become a standard part of responsive survey design decisions.

6 References

Groves, R. M. and Heeringa, S. G. (2006). Responsive design for household surveys: tools for actively controlling survey errors and costs. *Journal of the Royal Statistical Society, Series A, Statistics in Society*. Volume: 169, Issue: 3, Pages: 439-457.

Paradata at the U.S. Census Bureau (and Where Blaise Fits In)

Michael K. Mangiapane, U.S. Census Bureau

1 Introduction

Paradata, or data about the survey process itself, has been collected in various forms for decades. In recent years it has become a topic of interest among statistical agencies as they look for ways to perform their survey operations more efficiently and at a lower cost. The U.S. Census Bureau is pursuing a number of different efforts to collect and analyze current and future paradata to improve the quality of the data they collect, reduce burden on their Field Representatives (FR's), and bring the costs of a survey down in a time of leaner budgets.

2 Current and Future Paradata Efforts

There have been a number of paradata-related efforts at the Census Bureau in use for several years.

- **CHI** – We have been collecting and analyzing information about interview contact attempts for some surveys using our Contact History Instrument (CHI) since 2004. In 2011 we implemented CHI for all of our demographic surveys and developed a facility-based CHI for our Ambulatory Medical Care surveys. The CHI will be discussed in more detail in this paper.
- **PANDA/Giant PANDA** – The Demographic Surveys Division (DSD) developed the Performance and Data Analysis (PANDA) tool to summarize key metrics of the quality of data collection, such as item missing rate and length of interview (from audit trail fields). This system is being expanded to handle more of our demographic surveys.
- **CARMN** – The Technologies Management Office (TMO) developed the Cost and Response Management Network (CARMN) system to report on field representatives' cost and performance, using data from the CAPI case management system and the FR's payroll system.
- **CARI** – The Census Bureau has been testing Computer-assisted Audio Recorded Interviewing (CARI) for several years now and last year conducted a test with a new CARI monitoring system with the American Community Survey (ACS). We have yet to put any CARI projects into full-time production though.

However, there are some new, more comprehensive efforts taking place now.

- **Efforts Related to Field Restructuring** – In 2011, the Census Bureau embarked on a major restructuring of our field offices and management structure. Our twelve (12) regional offices (RO's) will be reduced to six (6) by the end of 2012, and many of the survey supervisors who previously worked out of an RO will now be working out of their homes. The supervisory chain for the FR's has also changed to be a Field Supervisor (FS) instead of a survey supervisor in the RO. Due to this significant change, specific efforts have been undertaken to monitor the data quality and response data during this period of change. Some of the work done with the Data Combing Process (DCP), described in more detail in this paper, was related to providing data for these efforts.
- **UTS** – The ultimate goal for the Census Bureau's paradata efforts are the development of a Unified Tracking System (UTS) that will provide integrated survey progress, cost, and data quality metrics in a centralized repository for use in survey management and analysis. This system is discussed in more detail in this paper.

3 Objectives of This Paper

This paper will go into more detail about the various ways that the Census Bureau is using Blaise to collect paradata and how using this data will continue the Census Bureau's mission to serve as the leading source of quality data about the United States' people and economy. The topics to be covered include:

- Using CHI for gathering household, facility, or person-based contact attempt information for a case.
- Creation of a Data Combing Process (DCP) that runs in Manipula and uses a list of variables provided by a stakeholder to capture and output paradata to a Blaise Audit Trail or a separate ASCII file.
- Parsing and analysis of Blaise Audit Trails and CARI files and logs to provide paradata.
- Using the above-named tools to provide input to the new Unified Tracking System (UTS) under development at the Census Bureau.

4 Contact History Instrument (CHI)

In 2002, the Census Bureau and a group known as the Interagency Household Survey Nonresponse Group (IHSNG) held a conference to discuss a concern about the decline of response rates and a rise in refusals for household surveys. The summit primarily focused on the National Health Interview Survey (NHIS) and Consumer Expenditure Quarterly (CEQ) survey. Among the topics discussed was the idea to collect "call records" that would contain data about contact attempts for a case. They wanted to see the history of a case with information such as how many times and when a household was contacted, reasons that the respondent was reluctant to complete the survey, and the contact strategies the FR implemented in attempted to complete the interview.

They hoped to be able to use this data to better discern why respondents tended to refuse surveys, when was the best time to attempt an interview and what strategies should be implemented to get responses. The group decided that it would be best to automate this process despite the initial costs. They also decided that the process should contain standard questions and answers across surveys so that data could be compared and evaluated the same way. The questions and answer choices could be modified if all areas agreed the modification was needed. From this effort, the Contact History Instrument (CHI) was created.

4.1 CHI Process

The CHI automatically runs after the FR exits the survey instrument so that the interviewer keys in the contact attempt while it is still fresh. It can also be called manually from Case Management for situations where the FR does not open the survey instrument during a contact attempt. After the FR exits the Blaise CHI instrument, an ASCII output file is generated and processed by Case Management so the data can be transmitted back to the Regional Office Survey Control (ROSCO) system.

There are three main sections of CHI used to collect information about a contact attempt – 1. Noncontact information, 2. Concern/Behavior/Reluctance information, and 3. Contact Strategies Attempted.

4.1.1 Noncontact Information

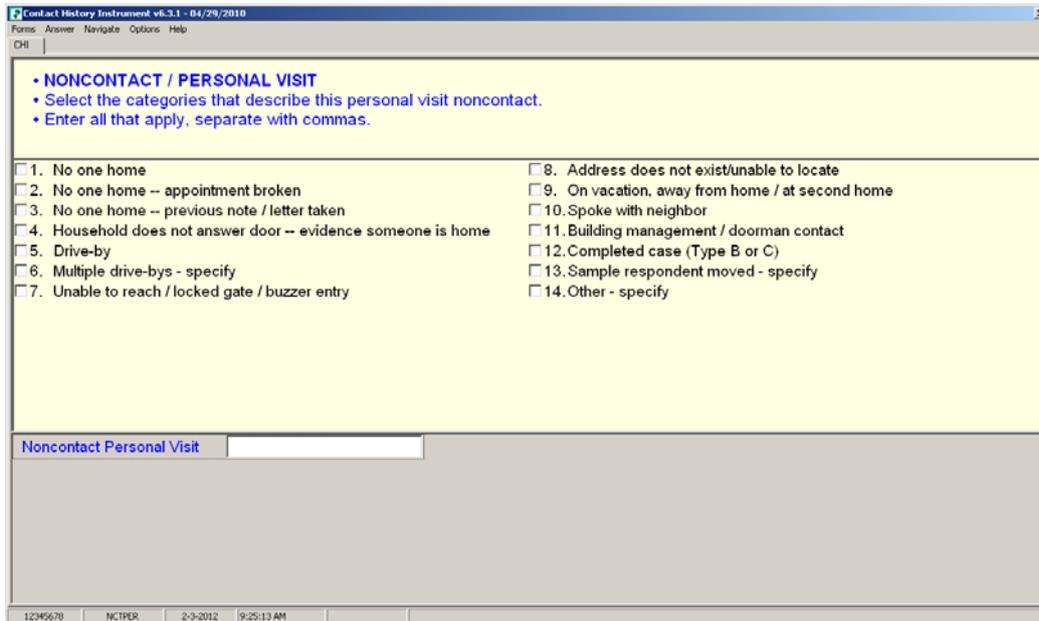


Fig. 1 - CHI Non-Contact screen

Inside the CHI, the FR is asked how they attempted to make contact with the respondent, by personal visit or by phone. If they did not make contact with the respondent, they are taken to this section to mark the outcome of their non-contact (e.g. No one home; locked gate; spoke with neighbor). They then enter any strategies they attempted to make contact in the “Contact Strategies Attempted” section. The CHI then exits and creates output.

4.1.2 Concern/Behavior/Reluctance Information

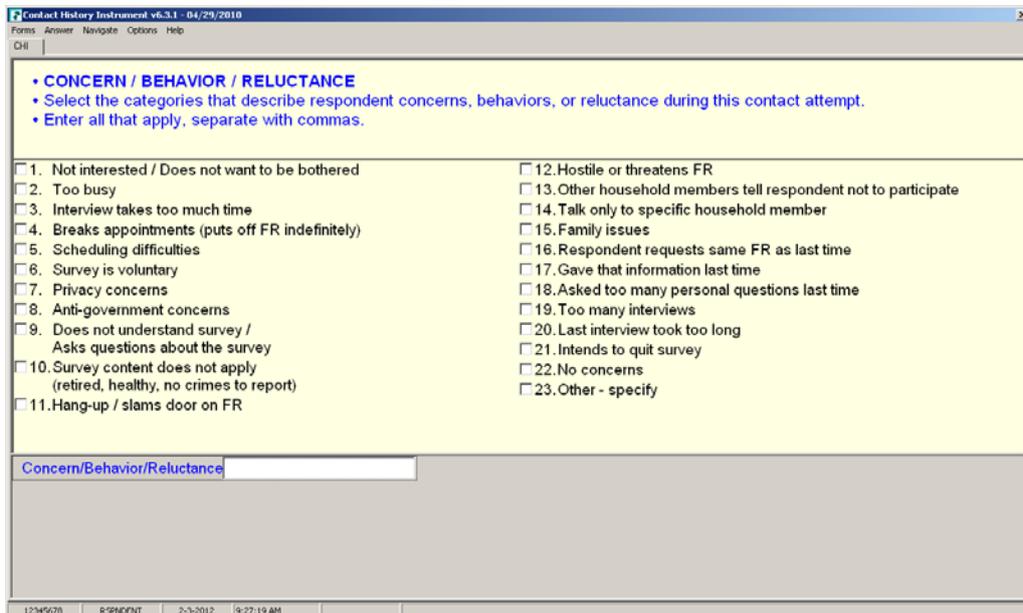


Fig. 2 – CHI Concern/Behavior/Reluctance Screen

If the FR does make contact with a respondent, they are taken through this section to answer a few questions about the nature of the contact. They are asked to enter in all Concerns/Behaviors/Reluctances that apply to this contact attempt.

4.1.3 Contact Strategies Attempted

CONTACT STRATEGIES ATTEMPTED

- Select the categories that describe the strategies used on this contact attempt.
- Enter all that that apply, separate with commas.

<input type="checkbox"/> 1. Advance letter given	<input type="checkbox"/> 13. Contacted other family members
<input type="checkbox"/> 2. Scheduled appointment	<input type="checkbox"/> 14. Contacted property manager
<input type="checkbox"/> 3. Left note / appointment card	<input type="checkbox"/> 15. Visited county assessor / post office / permit office
<input type="checkbox"/> 4. Left promotional packet / informational brochure	<input type="checkbox"/> 16. On-line tracking database
<input type="checkbox"/> 5. Called household	<input type="checkbox"/> 17. Sought help from SFR / RO
<input type="checkbox"/> 6. Left message on answering machine	<input type="checkbox"/> 18. Reassignment
<input type="checkbox"/> 7. FR will request No One Home Letter	<input type="checkbox"/> 19. Offered incentive
<input type="checkbox"/> 8. FR will request Refusal Letter	<input type="checkbox"/> 20. CED double placement
<input type="checkbox"/> 9. FR will request Better Understanding Letter	<input type="checkbox"/> 21. Used MAF or ALMI
<input type="checkbox"/> 10. Called contact persons	<input type="checkbox"/> 22. None
<input type="checkbox"/> 11. Stake-out	<input type="checkbox"/> 23. Other - specify
<input type="checkbox"/> 12. Checked with neighbors	

Strategies attempted:

12345678 STRATEGIES 2/3/2012 9:32:28 AM

Fig. 3 – CHI Contact Strategies Attempted Screen

Regardless of the outcome of a contact attempt, the FR will need to complete the “Contact Strategies Attempted section. Here they will mark all strategies that apply to this contact attempt.

Once the FR completes and exits the CHI, output is created.

4.2 CHI Output

Upon exit, the CHI transaction process creates an ASCII file containing information from the CHI. The output is coded for processing efficiency and so that future enhancements can be easily implemented.

```
#caseid
12345678
#frdate
02/02/2012
#howcontacted
T
#contactstatus
C
#contactperson
1
#casestatus
1
#types
C01
#reluctant
R22$A200$A#71
#strat
S22
```

Fig. 4 – Example CHI Output

For this particular case, the output shows that for this contact attempt, the respondent was contacted by telephone and the attempt resulted in a completed case (How contacted, Contact Status, Types). The FR reported no concerns with respondent behavior (R22 in #reluctant), and did not employ any strategies (S22 in #strat). It is also reported that the CHI case was open for 71 seconds. This CHI data and thousands of others like it are collected and analyzed to learn about the ever-changing nature

of our survey processes and respondents. The CHI data also stays with the case so when the case comes back to an FR in the following month, quarter, or year, the FR can review it before they attempt to contact the respondent.

Below is an example of returning CHI data in Case Management. The CHI data was recorded when a contact attempt was made for this case when it was previously interviewed. (November 2008).

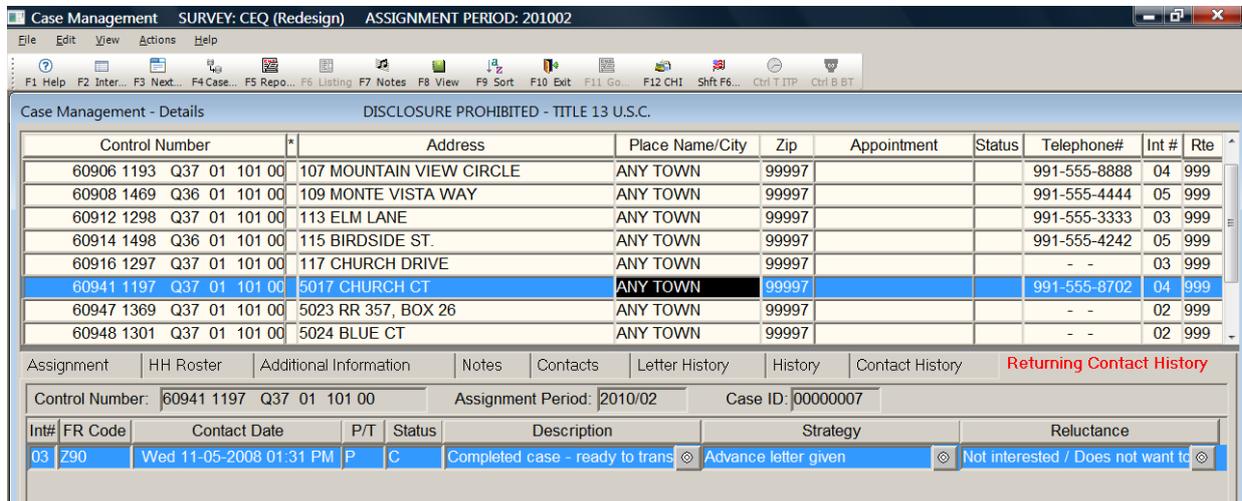


Fig. 5 – Returning CHI Data in Case Management

The FR is able to see which FR last made a contact attempt, when the contact attempt was made, the type of contact and the status of the contact. They can also review any attempted strategies and behavior or reluctance concerns. The FR may click icons next to the Description, Strategy, or Reluctance fields for more information.

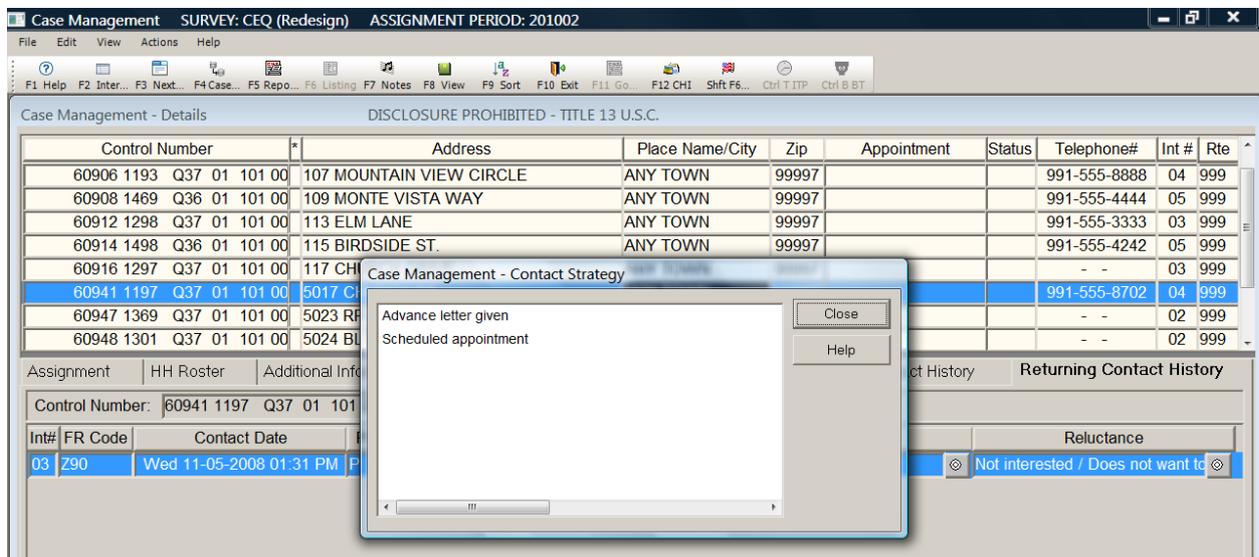


Fig. 6 – Detailed look at previous contact strategy in returning CHI data in Case Management

4.3 CHI in the Field

CHI was first put into production for NHIS in 2004. It is now in production for every one of the Census Bureau’s household surveys, and we have a facility-based CHI in production with our Ambulatory Medical Care surveys. There is an undertaking to create a person-based CHI, which will be explained later in this paper. During the testing process prior to production, there were some concerns from the FR’s that CHI data would be used against them (i.e. “Big Brother”) but this was not what the sponsors were interested in. The FR’s found that the CHI was useful as a quick way to

review their contact attempts with the household, or to review contact attempts that other FR's had made with the household. Prior to CHI, they were entering notes about their contact attempts in their case-level notes for each case. CHI standardized the process of recording their contact attempts, and it meant they spent less time typing notes about their contact. On average, a new FR spends 90 seconds in CHI, and an experienced FR spends 60 seconds.

4.4 CHI Data Usage

The CHI data collected has become key paradata for the Census Bureau in how they make contact with households and the responses they get. There have been a number of papers written about usage of the CHI data and what they have found from it, including which behaviors are most likely to get an interim refusal, meaning the FR will be refused before they get a survey response, and which ones will be a final refusal, meaning the FR will not get a survey response. The data also indicated that some regions of the United States are more likely to have final refusals than interim refusals, and that personal visits had a better success rate of getting survey responses than telephone contacts (Bates, Dalhamer, and Singer 2008:591). Other research has shown which times may be the best times for making the first contact attempt and the average number of contact attempts made that result in an interview or non-interview (Tan 2011:9).

CHI made it easier for the survey sponsors to review respondent behaviors and effectiveness of contact strategies because they were not reviewing case notes for thousands of cases to record responses vs. non-responses, the CHI data can be aggregated and analyzed instead.

4.5 Person-based CHI (pCHI)

When the National Crime Victimization Survey (NCVS) implemented CHI, the sponsor was concerned about collecting contact attempt data for each member of the household. NCVS interviews all persons 12 and older in the household; that is, every eligible household member has to provide their own answers to the survey questions. Since the standard household CHI assumes that you are only attempting to contact one person at the household, enhancements were requested so that contact attempt information could be collected for each respondent in the household. A new person-based CHI instrument is being developed at the Census Bureau.

The person-based CHI will read in the household roster when the instrument is started. When the FR reaches the question to ask about contact or non-contact, they will indicate if they made contact with one or more eligible persons, no eligible persons, or if it was a non-contact.

Person Based Contact History Instrument v8.00 - 02/06/2012

Forms | Navigate | Options | Help

pCHI | Roster Information

pCHI - CONTACT

Select outcome that best describes this contact attempt.

LNO	NAME	AGE	BIRTHDATE	SEX
1	Dejuan Loe	46	03/22/1965	M
2	(I) Derrek Loe	31	02/20/1980	M
3	(I) Larry Boe	32	05/30/1978	M
4	Thomas Loe	54	01/22/1957	M
5	(I) Sixfeet Under	100	12/21/1911	M

End of Roster

1. Made contact with **one** eligible person
 2. Made contact with **more** than one eligible person
 3. Made contact with only non-eligible persons
 4. Noncontact

Continue CHI / Exit: **Attempt**
 Immediate Attempt: **Yes**
 Date:
 Time:
 Personal/Telephone: **Personal**
 pContact:

00000003 | pCTSTATUS | 2-16-2012 | 3:58:07 PM

Fig. 7 – Person-Based CHI Contact Screen

If contact was made, the FR is taken to a table that displays the full roster and puts on-path the person(s) who were eligible for the survey and require CHI data to be collected. The CHI data is then collected for each person.

Person Based Contact History Instrument v8.00 - 02/06/2012

Forms | Navigate | Options | Help

pCHI | Roster Information

PERSON CONTACT

Did you make contact with Dejuan Loe?
 (C)omplete (I)nactive HH Member

1. Yes
 2. No

	Contact	Non-Contact	Contact Attempt	Partial /Unable to Conduct	Language Issue	Language List	Specify Language /Dialect	Other Contact Category	Concern /Behavior /Reluctance	Other Reluctant Respondent	Strategies Attempted	Other Strat. Attempted
Dejuan Loe	[1] <input type="checkbox"/>		<input type="checkbox"/>								<input type="text"/>	
(I) Derrek Loe	[2] <input type="checkbox"/>											
(I) Larry Boe	[3] <input type="checkbox"/>											
Thomas Loe	[4] <input type="checkbox"/>		<input type="checkbox"/>								<input type="text"/>	
(I) Sixfeet Under	[5] <input type="checkbox"/>											

00000003 | PCONTACTPER | 2-16-2012 | 3:59:30 PM

Fig. 8 – CHI data collection table for eligible household members.

In the above example, only two members of the roster are eligible household members in NCVS so they are the only ones that CHI will collect data for. Just like CHI, Person-based CHI will collect the same information asked at the household-level, but for each eligible person in the roster. After the CHI is finished, an output file is produced for each line of the roster that had CHI information collected. There are a few lines that are different from CHI, but the codes used for “reluctance” and “strategies” are the same.

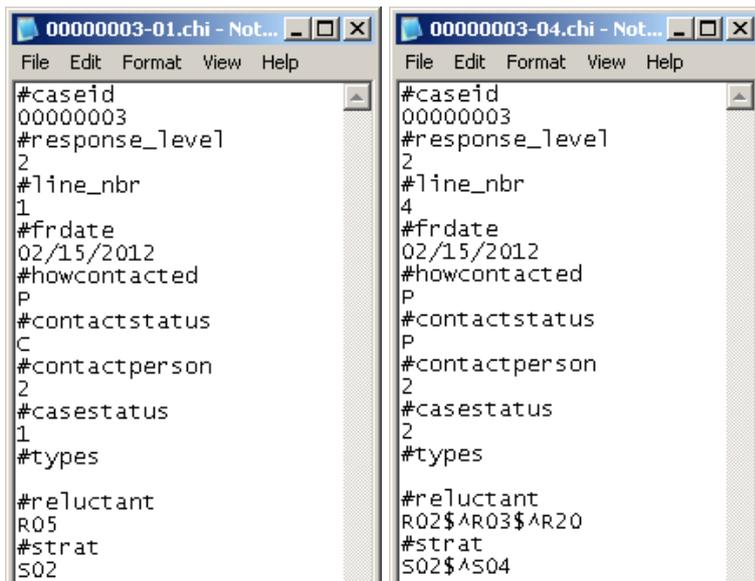


Fig. 9 – Example CHI output for person-based CHI for two eligible members

5 Data Combing Process (DCP)

Prior to the creation of the Unified Tracking System (UTS), the sponsors of the NCVS were interested in capturing their own paradata from Blaise. The original request was to capture the values of certain variables and include them in the audit trail file. They also wanted to know which FR completed each question if the case had been reassigned. Originally we pursued writing our own data to the audit trail file as the interview progressed using Manipula¹. For example, upon starting the interview, we would write a line with our own defined tag that included the FR’s ID code so we could know that the questions from that point forward were completed by that FR. Our thinking then evolved to capturing the data we needed each time the FR exited the instrument and append it to the end of the audit trail file

5.1 Initial Research

In trying to implement the second approach, our first thought was to start with an existing Manipula script that our Master Control System (MCS) uses in their data processing to output data for a defined list of variables. This script pulls requested data from the Blaise database and outputs it as a text file for our Demographic Survey Methods Division (DSMD) to pick up key variables they need for the sample control system. With a few modifications to the code, this script would be placed on an FR’s laptop where it would run and output the variables to the audit trail. Unfortunately, that script only read variables that were at the datamodel level and it required using Chameleon to create a separate datamodel and variable list before the script would run. Since this meant extra time and overhead for processing and changing our instruments (to pass variables to the instrument datamodel level) just to be able to produce output, the output script from MCS would not be ideal for running on the laptops. It was clear that creating a new script would have a better chance of fulfilling these requirements.

Since Blaise 4.8 had introduced many new features within Manipula, we decided to research its methods of obtaining values from a Blaise database to see if there was one that would fulfill our needs. The discovery of the GETVALUE method intrigued us since as a function it “Retrieves the value of a field of which the name is not known at prepare time” (Blaise Help File). Based on that

¹ To allow an external program to write to the audit trail, we set the “CloseFile” toggle in the .aif file to “1.” This will open and close the audit trail every time a line is written to it, thus removing the Blaise generated lock file associated with that case and allowing Manipula to write directly to the audit trail.

statement and the examples given, it meant that yes, we would be able to use any variable that existed in a Blaise database, so long as we used the fully-qualified field name.

5.2 Proof of Concept

The first DCP script used some hard-coded variable names as a proof of concept, and was designed to run on the FR laptop. It would open the audit trail and append the values at the end of the audit trail, in a line that looked something like this:

```
#PARADATA;value1;value2;value3;value4;...;#ENDPARADATA
```

By including the #PARADATA and #ENDPARADATA, it would make it easier to parse the entire audit trail for a case and capture the included paradata.

5.3 Redesign

As this proof-of-concept was created and tested, the UTS was entering its design phase and the UTS designers wanted to capture similar data from all surveys rather than just for NCVS. At about the same time, DSMD needed to get data for numerous variables from several instruments, in order to monitor the effect of field restructuring on interview data quality. We decided to combine efforts and “comb” Blaise data for either UTS, DSMD, or a survey sponsor. With this combined effort came a redesign of the output of the script as both parties wanted to collect many variables. To reduce case processing and data transmission times on the laptops, we decided to comb the data after a case was checked back into headquarters. This also changed when the combing process would run since MCS would now handle the process rather than the FR laptops.

5.4 DCP Script Process

The DCP was built in Manipula using Blaise 4.8.2 Build 1606. It must be compiled with the datamodel of the instrument it will query, and it is set up to query a consolidated Blaise database that contains all of the case records. For our purposes we pass in the name of a Blaise database that contains all of the cases that have been checked in from the field, a list of ID’s of cases that were checked in that day, and a “qualified variable list” (QVL) of fields to be collected (provided by the sponsors in a text file). After creating or opening an ASCII file for output, the DCP steps through the list of case ID’s and begins to comb through the data to collect for output.

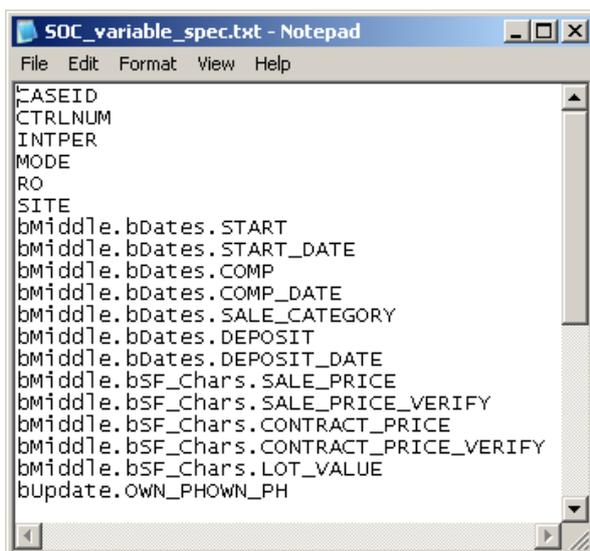


Fig. 10 - Example QVL

To collect the data, the script first checks on the status of the field. If there is a response, the script uses GETVALUE to add the value from the field to the output.

ConsolidatedDB.GETVALUE(InputFields.VarName, UF)

We also use the UF option in GETVALUE so that enumerated fields are output as a number rather than the mnemonic of the number. For non-responses, we print DK, RF, or nothing but the delimiter if the field is Empty.

After the script finishes, the output generated is one file with all the data combed out of the cases, or an individual file for each case. A typical output file will look like this (individual files will only have one line)

```
3003901N|222222222N|201010SI|0|30|DK|DK|203|20111208|20111208|121924|121945|3.28|  
3003902N|333333333N|201010SI|0|30|1|1|DK|203|20111208|20111208|102202|102321|1.19|
```

We decided to use the pipe character as a delimiter because typical delimiters such as a comma or semicolon may be a part of a string field, and that would throw off the processing of the output since the values are in the order of the variable list.

A test of the script showed that it runs very fast to produce output. In a scenario of combing for 306 data values per case, the script processed 270 cases in only 32 seconds. This was much faster than expected and meant it would not have a great impact on the overhead of our processing systems.

Recently a request to enhance the script has been requested by the survey sponsors, asking if it could provide a count of the number of don't knows, refusals, and fields that were asked but left empty in an interview. While Manipula does have built-in functionality to count don't knows, refusals and responses, it does not have the functionality to count fields that were asked but left empty. We are currently exploring using ROUTESTATUS to find the empty fields, even if there is a keep on that field.

6 The Unified Tracking System (UTS)

The Census Bureau is attempting to drastically change the way it manages its survey data collection processes, including using responsive design to improve quality and/or reduce costs. There was a need for a system which would gather all of the paradata that is currently being collected by various processes such as CHI data, data extracted using the DCP, and the expansion into collecting more paradata for a survey, such as CARI data. Not only should this system be able to store data from all of these different processes, it needs to be able to present it to the users in a user-friendly format for analysis. An initiative is underway at the Census Bureau to create such a system, which we call the Unified Tracking System.

6.1 UTS Design

The UTS collects data from many different sources at the Census Bureau, including CHI data, DCP data, Regional Office Survey Control System (ROSCO), Cost And Response Management Network (CARMN), the Census Bureau's financial system, the Census Bureau's payroll system, Blaise audit trail files, and TMO's CATI case management system, WebCATI. Other sources may be added in the future as systems develop and data requirements change, including CARI.

The users of the UTS will be Field survey managers, internal survey sponsors, and even some of our external survey sponsors (with limited views).

All of the data presented by the UTS is inside a SAS Enterprise Business Intelligence (SAS-EBI) system in the form of "data cubes." An Oracle Data Warehouse serves as the data repository for UTS cost, progress, and quality paradata, and SAS-EBI uses this data warehouse to form the data cubes. Within a data cube, a user can create and customize views and of the data they wish to see and analyze; they have a lot of control over what they see while they are in the UTS. With a few clicks,

they can go from a table of how many cases a regional office has for a survey to how many cases each FR in the region has assigned. This cuts down on waiting time to create and generate a separate report for each query the user has since SAS is doing the aggregation for them in the background. As data is added to the system, users will be able to look across all surveys rather than one at a time, and analyze data on the system as far back as five years or more, which was not possible before. The UTS allows for easy export of their data to other formats such as Microsoft Word or Excel.

6.2 Using the UTS

Upon logging in, the user is taken to their own personal portal page in the UTS, which contains links to any previously saved reports, and further access to elements of the data cube.

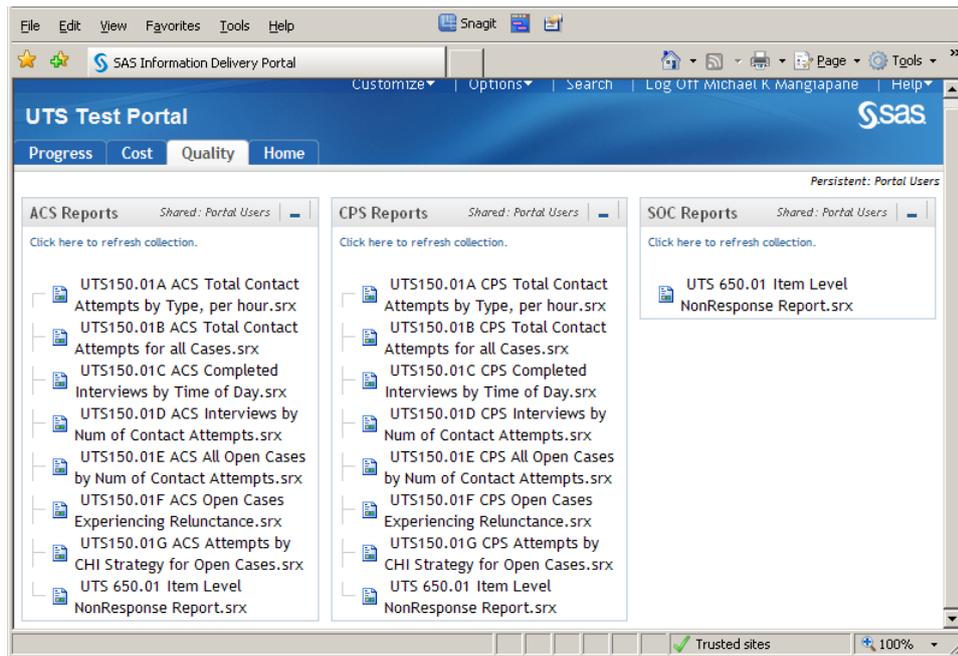


Fig. 11 - UTS Portal Page

Not all users will have access to all of the data saved inside the cube. Field Supervisors may only see cost, progress, and quality reports related to their assigned geographic areas, while a survey sponsor may see the same reports for all geographic areas.

Selecting a report brings up a screen to filter which data they want to see in the report.

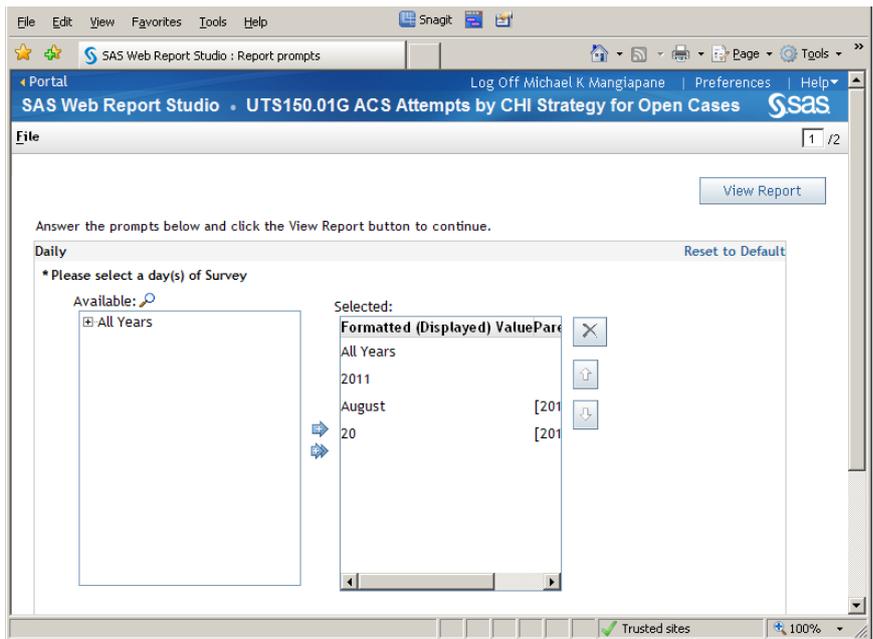


Fig. 12 - Report Filter Screen for the CHI Strategy Report

The view report button will then display the report with the selected options.

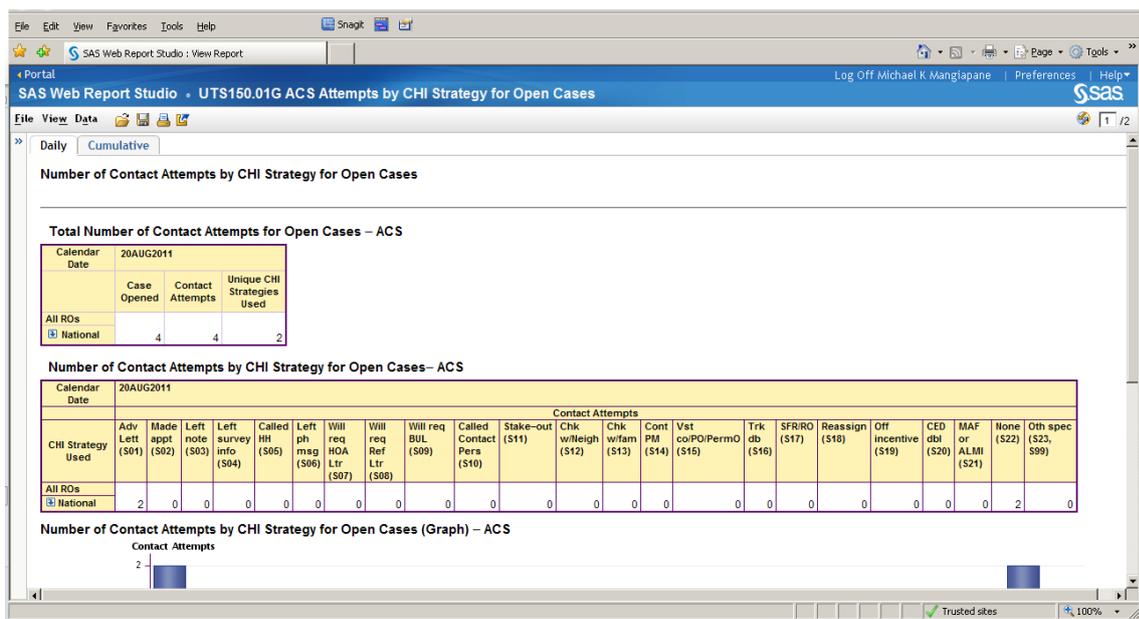


Fig. 13 - Example report generated after selecting all of the options in the filter screen:

With the default options selected, the UTS displays all of the retrieved data collectively as a national sample. However, the user can use the “drill down” icon to the left of National and separate the data by RO. They can also right-click in the chart and set up a filter or sorting to organize the data.

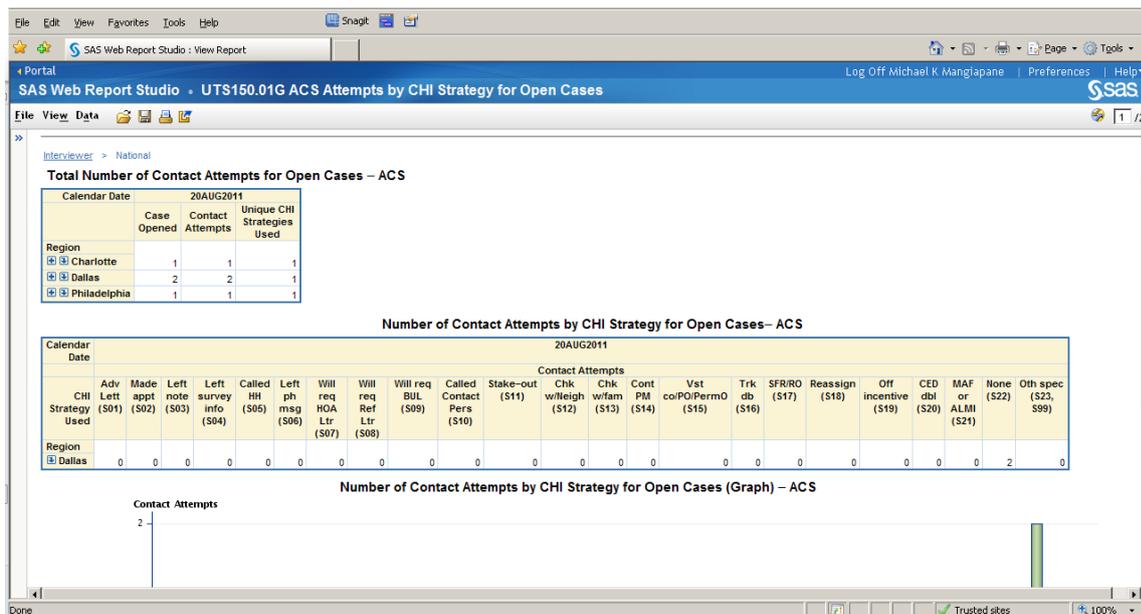


Fig. 14 - Report after user clicks the Drill Down under “Total Number of Contact Attempts for Open Cases - ACS” and applies a filter to “Number of Contact Attempts by CHI Strategy for Open Cases – ACS”

In the above example, the “Number of Contact Attempts by CHI strategy for Open Cases – ACS” was filtered to only show regions where the CHI strategy of “None” was greater than zero. If a user wanted to take it further, they could view and filter strategies by Field Supervisor (FS) and by the FRs that report to the FS. This is just one of many ways that the UTS can be customized to display data for analysis by the user.

6.3 Plans for UTS

The ultimate goal of the UTS is to formulate dynamic response design for our surveys and help bring down the costs of conducting surveys at the Census Bureau. Since the user will be able to see how much each case is costing them, they can remove any outliers that are well above the average cost per case with little response or useful data. It will also allow a survey sponsor to identify when a survey has reached a statistically significant sample and no more interviews are needed for that interview period. In terms of responsive design, the sponsors will be able to immediately measure how a change to a survey is having an impact in the field rather than waiting until all of their data comes back and is processed.

Besides storing the CHI data and DCP data, the UTS is also developing a process for parsing Blaise audit trails to satisfy a request for calculating the amount of time that an FR spends in each section of the instrument. We have tried in the past to perform this calculation within Blaise, but have found that turning the appropriate section timers on and off presents a difficult challenge if a user backs up from one section into another.

7 Conclusion

For a number of years the Census Bureau has collected and analyzed paradata from their surveys through CHI, case level notes, Blaise audit trails, and critical items in the instrument data. Now our systems are evolving to effectively collect, track, and use more paradata for responsive survey design. Even with all of the data that we will collect from a number of different systems to put together in the UTS, we anticipate that Blaise will have a significant role in collecting paradata for our surveys.

8 References

Bates, N., Dahlhamer, J., and Singer, E. (2008). Privacy Concerns, Too Busy, or Just Not Interested: Using Doorstep Concerns to Predict Survey Nonresponse. *Journal of Official Statistics*, 24, 591-612.

Maitland, A., Casas-Cordero, C., Kreuter, F. (2009). An Evaluation of Nonresponse Bias using Paradata from a Health Survey. *Section on Government Statistics – JSM 2009*.

Tan, L. (2011). An Introduction to the Contact History Instrument (CHI) for the Consumer Expenditure Survey. *Consumer Expenditure Survey Anthology*, 2011, 8-16.

9 Acknowledgements

The author would like to acknowledge the following people for input into this paper and input on some of the features and usability of the UTS and CHI.

Nancy Bates, U.S. Census Bureau

John Wilen, U.S. Census Bureau

William E. Dyer Jr., U.S. Census Bureau

Malcolm Robert Wallace, U.S. Census Bureau

Tom Spaulding, U.S. Census Bureau

Karen Bagwell, U.S. Census Bureau

The views expressed in this paper are those of the authors and not necessarily those of the U.S. Census Bureau.

Using Paradata to Investigate Food Reporting Patterns in AMPM

Lois C. Steinfeldt, Jaswinder Anand, and Theophile Murayi, United States Department of Agriculture, Agricultural Research Service, Beltsville Human Nutrition Research Center, Food Surveys Research Group

1. Introduction

USDA’s Automated Multiple-Pass Method (AMPM), written in Blaise version 4.8, collects 24-hour dietary recalls for the What We Eat in America (WWEIA), National Health and Nutrition Examination Survey (NHANES). Each year it is used in approximately 10,000 interviews which ask individuals to recall the foods and beverages that were consumed the day before the interview. WWEIA collects two days of dietary intakes 3 to 10 days apart. The first interview is an in-person interview which takes place in the NHANES Mobile Exam Center. The second interview is by telephone. AMPM uses a research-based, multiple-pass approach designed to encourage complete and accurate food recall and to reduce respondent burden. The 5-step multiple-pass method used in AMPM is the result of more than 10 years of research by USDA to improve the 24-hour dietary recall methodology. The basis of the multiple-pass is to lead the respondent through the 24-hour period of the recall more than once using different approaches to assist the respondent in recalling foods and beverages without being repetitive (Raper et. al., 2004). AMPM provides a structured interview with standardized questions combined with unstructured opportunities for respondents to use their own individual strategies to remember and report foods. The success of this approach was demonstrated by the AMPM validation study conducted by the Food Surveys Research Group in collaboration with the Food Components and Health Laboratory, in which the energy intake collected by the AMPM was underreported by < 3% of the energy expenditure as measured by doubly-labeled water for normal weight individuals (Moshfegh et. al., 2008). Individuals report foods, at each of the 5 steps in AMPM on both days of the interview. “Food” is used throughout this paper to represent all foods and beverages, including water. The percent of foods and daily energy intake contributed by each step is shown in Table 1.

Table 1. Comparison of Contribution of AMPM Steps

AMPM Step	Day 1 Interview		Day 2 Interview	
	Percent of Foods*	Percent of Energy	Percent of Foods*	Percent of Energy
Quick list	65	77	73	84
Forgotten foods	9	6	5	5
Time and occasion	3	2	2	1
Detail cycle	22	12	19	10
Final probe	1	< 0.5	1	< 0.5

WWEIA, NHANES 2007-2008, Ages 12 and over, MEC weights

*“Food” comprises all foods and beverages, including water

The AMPM 24-hour dietary interview with the multiple-pass approach and flexibility to add foods at any point in the interview is a large and complex instrument. In order to maintain the coherence of the interview, it must keep track of a large amount of interview process information. This information about the process of the interview or paradata can be used to explore the dietary interview process and how individuals report foods. AMPM keeps track of the step in which the food was reported and also the order in which foods are reported in the interview. It also designates foods as primary foods such as coffee or as additions to foods such as milk added to the coffee. This data can be used to describe how individuals report foods during the AMPM interview and how the reporting changes between the first and second interviews.

2. Collecting Foods

2.1 Quick List

The AMPM interview begins with the Quick List where respondents are asked to report all the foods and beverages including water consumed from midnight to midnight the day before the interview. The Quick List is an unstructured, uninterrupted listing of foods which the respondent can report in any order. This allows respondents to use their own strategies to recall and report the foods consumed. A number of memory cues are included within the question. The day before the interview is referred to both as yesterday and by the day of the week. Memory cues in the question suggest that the respondent think about whom they were with and what they were doing such as working, eating out, or watching television. The question also includes references to foods eaten at home and away, and foods such as snacks, coffee, soft drinks, water, and alcoholic beverages. Approximately 69% of foods are collected during the Quick List making up 80% of the energy intake for the day.

2.2 Forgotten Foods

The next step is Forgotten Foods in which the respondent is asked 6 questions about specific types of foods and beverages that respondents frequently forget to include (beverages; alcoholic beverages; sweets; savory snacks; fruits, vegetables, and cheese; breads and rolls). The foods listed in each question provide memory cues to the respondent to think about these specific types of food. The respondent can report any type of food recalled at any of these questions. The seventh and last question is open asking the respondent if they can remember any other foods not yet reported. This step collects about 7% of foods accounting for 7% of daily energy intake.

2.3 Time and Occasion

The time the food was eaten and the name of the eating occasion are collected in this step. Examples of eating occasions include breakfast, lunch, dinner, snack, and beverage. This information is collected for each food reported in the first two steps. While reporting this information, the respondent can also report additional foods. Foods reported during this step account for about 2% of foods and 2% of daily energy intake.

The foods are stored in the Respondent Food List which is a block containing an array of foods. Each food has 54 fields associated with it. As each food is reported it is stored in the next position of the array. At this point in the interview, AMPM sorts all of the foods reported by the time of day they were consumed. It does this by assigning a sort order value to each food and does not change the position of the foods in the array. Since the position of the foods in the array is not changed, the order the foods were reported is maintained. AMPM also groups foods by time and eating occasion and establishes intervals between midnight and the first eating occasion, between each eating occasion, and between the last eating occasion and midnight. It should be noted here that foods consumed at the same time are not allowed to be assigned more than one eating occasion by the respondent.

2.4 Detail Cycle

AMPM then begins the next step which is the Detail Cycle. This step has two different types of questions. The first type is standardized questions which collect descriptive details about the food, additions to the food (such as milk to coffee), the amount consumed, the source of the food and whether it was eaten at home. The second type is review questions for each meal and each interval between eating occasions. For each interval, the respondent is asked if they had anything between the start of the interval and the end of the interval. The first interval is between midnight at the beginning of the 24-hour period and the first eating occasion. Then the question is asked for the interval between each eating occasion and finally between the last eating occasion and midnight at the end of the 24-hour period. When the standardized detail questions have been completed for all the foods in an eating occasion, the respondent is read a list of the foods reported for that eating occasion and asked if they ate anything else. Here the foods already reported serve as memory cues. At any

point during this step, if a respondent reports a new food, that food can be added to the Respondent Food List. All foods added during this step, including additions, are stored as separate foods in the next position in the food array. If the food was an addition it is assigned several values which mark it as an addition and link it to the food in the food array to which it was added. Additions are assigned the same eating occasion and time as the food to which they were added. Foods added as a result of the review questions may be part of an eating occasion for which foods were already reported or they may be a new occasion. Although most of the foods reported at an interval question will create new eating occasions, the respondent may also have remembered a food to add to an existing occasion. This is also true of the questions which review the eating occasion. Most of the foods will be added to that eating occasion, although respondents will also report new eating occasions.

Foods reported for an occasion or time of day which has already been asked, are held until the end of the detail cycle. At this point all remaining foods are asked. When the food added has created a new occasion, there are no new intervals created and no interval questions are asked but the review question for a new occasion is asked. Approximately 21% of foods and 11% of daily energy intake is collected during the detail cycle.

2.5 Final Probe

The last step is the Final Probe question which asks the respondent if they can remember any other foods including water and emphasizing even small amounts of foods which the respondent may not have felt were worth reporting. This question includes memory cues about foods consumed in the car, at meetings, or while shopping, cooking, and cleaning up and accounts for about 1% of foods and less than 1% of daily energy intake.

3. Using the Paradata

Information on where in AMPM foods are reported and the order in which they are reported does not impact the results of the survey which monitors food and nutrient intake in the U.S. population. However, this data can be used to investigate how the methodology of the AMPM succeeds in collecting complete and accurate 24-hour dietary recalls.

For this analysis, variables stored in the Respondent Food List food array were selected. These variables include the position of the food in the array or the array food number, the AMPM step in which the food was reported, and the type of the food (primary or addition). A food in AMPM is any food or beverage, including water. The foods were sorted by the eating time and occasion which produced a listing of foods in the order they were consumed within the 24-hour recall period. Each eating occasion was numbered sequentially beginning with 1. An eating occasion is defined as a unique combination of the time the food was consumed and what the respondent called it (breakfast, lunch, dinner, snack, beverage, etc.). Each food within an eating occasion was assigned the same number.

The foods were then sorted by their position in the food array which represents the order in which they were reported. Foods were grouped by eating occasion using the same definition, and sequential numbers were assigned to each eating occasion as it was reported beginning with 1. Foods from the same eating occasion that were reported in different steps on the AMPM have different reported order values. The exception is foods that are additions which were assigned the same reported order of the food to which they were added. For example, for a breakfast eaten at 7 am consisting of coffee, cereal, and milk, if the cereal was the first food reported on the Quick List, it was assigned a reported order of 1. The milk added during the Detail Cycle, because it was an addition, was also assigned a reported order of 1. If the coffee was reported during the Forgotten Foods step, it received a different reported order which was the next sequential number. This allows the analysis to look at how both foods and eating occasions are reported.

Figure 1 is an example of order assignment and the calculation of order difference. The respondent had 4 eating occasions: a 7 am breakfast with coffee and cereal with milk, a 9 am beverage which was a soft drink, a 1 pm lunch of pizza, and a 3 pm snack which was a cookie. The consumed and reported orders for the cereal and the milk are the same. The order values for the other foods are different. Order difference was calculated by subtracting the consumed order from the reported order.

Figure 1. Example of Order Assignment and Calculation of Order Difference

<u>Reported Order Number</u>	<u>Reported on AMPM Quick List</u>	<u>Consumed Order Number</u>	<u>Order Difference</u>
1	7 am Breakfast: cereal	1	0
2	1 pm Lunch: pizza	3	-1
3	3 pm Snack: cookie	4	-1
<u>Forgotten Foods</u>			
4	7 am Breakfast: coffee	1	3
<u>Detail Cycle</u>			
1	7 am Breakfast: milk added to cereal	1	0
5	9 am Beverage: soft drink	2	3

4. Statistical methods

Statistical analyses were performed with PC-SAS software (version 9.2; SAS Institute Inc, Cary, NC). Patterns in which individuals report foods were tested between the order that foods were reported and the order in which they were consumed using the nonparametric sign test provided by the Univariate SAS procedure for two related samples. Mean order differences were classified by selected demographic and food reporting variables. A two by two mean order difference comparison was performed using Tukey's multiple comparison method within the General Linear Model SAS procedure.

5. Results

This analysis was done using males and females age 12 and over from the 2007-2008 WWEIA, NHANES. Intakes for respondents younger than age 12 were not included because they are collected using assisted and proxy interviews. All the analysis was done separately by the day of interview. There were a total of 6,575 respondents with day 1 interviews and 86% (5,663) of these respondents also had a day 2 interview for a total of 12,238 dietary intake interviews. There were a total of 160,609 foods reported in the AMPM during these interviews. Overall, respondents reported an average of 13.2 foods in the day 1 interviews and 13.1 foods on day 2 showing no difference in the number of foods reported between the two interviews. As Table 2 shows, females report slightly more foods than males and older individuals report more foods than do younger.

Table 2. Average Number of Foods Reported by Day of Interview

Age (years)	Day 1		Day 2	
	Males	Females	Males	Females
12-29	11.0	11.6	10.3	10.8
30-59	13.3	13.5	13.0	13.6
60 and over	14.5	14.9	15.0	15.4
12 and over	12.9	13.4	12.8	13.4

Table 3 shows the percentage of interviews by day and gender in which respondents reported new foods (i.e., foods not reported in the Quick List) for each subsequent AMPM step. More respondents reported new foods on these AMPM steps on day 1 than they did on day 2. For both days, a higher percentage of male respondents reported new foods on the Forgotten Foods step than did females. On the other 3 steps, there is little difference between the percentages for males and females on both days. The report of new foods in the Detail Cycle includes additions to foods and, as expected, there were more interviews with new foods added in this step. During the last step in AMPM, which is a single final probe with memory cues, 13% of males and 12% of females reported new foods in the day 2 interviews. Therefore, these data demonstrate that the final step of the AMPM is important in helping to collect complete dietary recalls. Furthermore, even though these foods counted for only 1% of total foods reported and <0.5% of daily energy intake, they contributed nutrients to at least 12% of the dietary intakes.

Table 3. Percent of Interviews with New Foods Reported during each AMPM Step

Interview	Forgotten Foods		Time and Occasion		Detail Cycle		Final Probe	
	Males	Females	Males	Females	Males	Females	Males	Females
Day 1	70%	65%	27%	26%	90%	91%	17%	15%
Day 2	53%	45%	17%	16%	87%	90%	13%	12%

In the day 1 interview, about 66% of individuals began reporting foods on the Quick List with their first eating occasion for that day. However, only about 17% reported all their eating occasions on the Quick List in the order they were consumed. In the day 2 interview, with respondents now familiar with the AMPM, these percentages increased to 83% and 35%, respectively. The reference to the time frame from midnight to midnight in the Quick List question leads respondents to begin reporting with their first eating occasion of the day. However, these numbers show that few respondents continued reporting foods in the order they were consumed, even on the Quick List.

Over the entire interview, only 8% of respondents on day 1 and 20% of respondents on day 2 reported all their foods in the order they were consumed. These correct order reporters were evenly distributed between males and females.

Statistical comparisons of the reported order and the consumed order are shown in Table 4. The table shows the mean differences between the reported and consumed order of foods, the standard errors and the significance levels for day 1 and day 2, males and females, and 5 age groups. Significance levels are indicated by asterisks as described in the table footnotes. Smaller mean differences indicate that respondents reported foods closer to the order in which the foods were consumed. In all of the age and gender groups for both days, the mean difference is statistically significant at either $p < .001$ or $p < .01$, as shown in Table 4, except for females age 60 and over on day 2. This does not appear to be due to reporting fewer numbers of foods, because they are also the group who reported the most foods on the AMPM (15.4) as shown in Table 2. In an analysis comparing mean daily energy intake between day 1 and day 2, females age 60 and over are also the only age and gender group in which the within-person variation is lower than the between-person variation as shown in Table 5 (Anand, et. al., 2011). Although these factors may not be related, they do indicate that this age gender group

is different. These women may pay more attention to foods since they are likely to have been responsible for years of shopping and meal preparation. The effect may be less in younger women because time spent on meal preparation and related activities has decreased by 7 hours per week for full time homemakers and 3.5 h per week for employed women between 1965 and 1999 (Rose, 2007).

Table 4. Mean Differences in Food Ordering Between Reported and Consumed

Interview	Gender and Age (years)	N	Order Difference Mean (SE)
Day 1		6,575	.78 (.03)**A
	Males	3,268	.84 (.05)**C
	12-29	1,044	.94 (.08)**
	30-39	472	.88 (.12)**
	40-49	417	.87 (.13)**
	50-59	439	.75 (.12)**
	60 and over	896	.76 (.08)**
	Females	3,307	.73 (.04)**D
	12-29	1,000	.81 (.08)**
	30-39	485	.63 (.11)**
	40-49	465	.72 (.12)**
	50-59	432	.58 (.12)*
	60 and over	925	.77 (.08)**
Day 2		5,663	.5 (.03)**B
	Males	2,750	.51 (.04)**E
	12-29	847	.52 (.06)**
	30-39	379	.58 (.11)**
	40-49	346	.57 (.11)**
	50-59	385	.51 (.10)*
	60 and over	793	.46 (.07)**
	Females	2,913	.49 (.04)**E
	12-29	869	.5 (.06)**
	30-39	404	.48 (.09)**
	40-49	415	.5 (.09)**
	50-59	388	.48 (.10)**
	60 and over	837	.48 (.07)

** Significantly different from 0 at $p < .001$

* Significantly different from 0 at $p < .01$

A,B,C,D,E Means with different letter superscripts are significantly different at $p < 0.01$

Table 5. Within- and Between-Person Variance in Energy

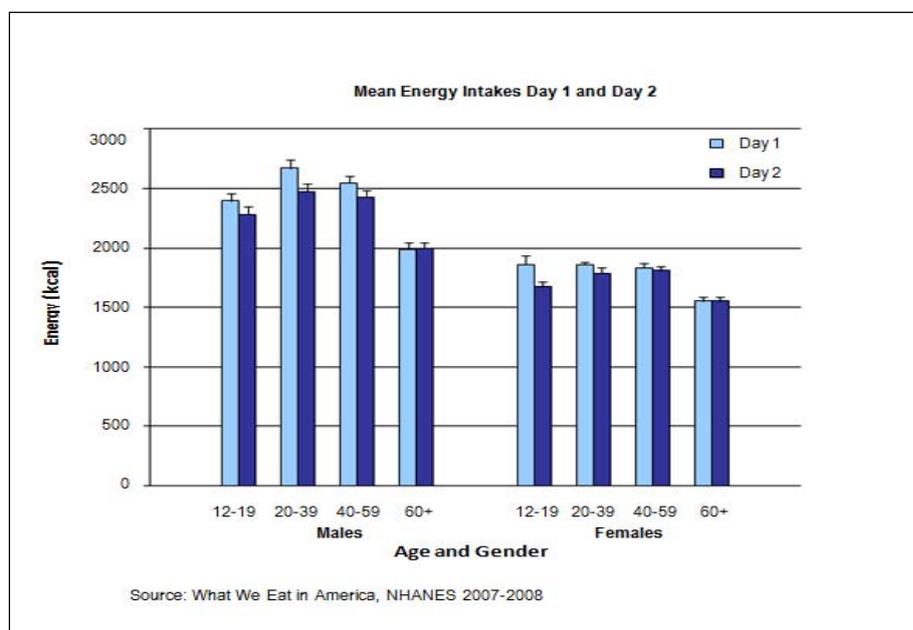
Age and Gender	Two day mean energy (kcal)	Within-person variance %	Between-person variance %
Males 12-19	2341	56.5	43.5
20-39	2575	62.6	37.4
40-59	2487	50.6	49.4
60 +	1994	57.2	42.8
Females 12-19	1768	73.6	26.4
20-39	1825	56.1	43.9
40-59	1821	63.5	36.5
60 +	1553	40.5	59.5

Source: What We Eat in America, NHANES 2007-2008, all individuals with 2 days of dietary data, 2 day weights

Comparing the order difference means between day 1 (.78) and day 2 (.5) shows a significant difference ($p < .01$). This means that respondents reported foods closer to the order they were consumed on day 2. The order difference means are also significant between males (.84) and females (.73) on day 1 ($p < .01$). But the order difference means are not significant between males (.51) and females (.49) on day 2. These significance levels are indicated by the letters A-E in Table 4. Although both males and females reported foods closer to the order they were consumed on day 2, there was a larger change in the reporting behavior of males between day 1 and day 2.

Although there are differences in how respondents report foods in terms of order on the AMPM between day 1 and day 2, there are no significant differences in the mean energy intakes as shown in Figure 2 (Anand, et. al., 2011). This demonstrates that different food reporting patterns in AMPM between day 1 and day 2 did not impact mean energy intakes within age and gender groups.

Figure 2. Comparison of Mean Energy Intake Day 1 and Day 2



6. Conclusion

Analysis of food reporting patterns from AMPM interviews shows the importance of the multiple-pass methodology to produce complete and accurate dietary intakes. Regardless of day, gender, and

age, new foods are reported on every AMPM step and few respondents report foods in the order they were consumed.

This investigation, using paradata from AMPM to look at food reporting patterns, suggests that this analysis could be extended in a number of directions. These include types of foods and beverages reported at each step, the number of new eating occasions added at each step, and the effect of the time of day the interview is conducted. Since most respondents do not report foods in order, describing patterns in food reporting could glean information that would lead to improvements in the AMPM.

References

Raper N, Perloff B, Ingwersen L, Steinfeldt L, Anand J. An Overview of USDA's Dietary Intake Data System, *J Food Compos Anal.* 2004; 17(3-4):545-555.

Moshfegh AJ, Rhodes DG, Baer DJ, Murayi T, Clemens JC, Rumpler WV, Paul DR, Sebastian RS, Kuczynski KJ, Ingwersen LA, Staples RC, Cleveland LE. The US Department of Agriculture Automated Multiple-Pass Method Reduces Bias in the Collection of Energy Intakes, *Am J Clin Nutr.* 2008; 88:324-332.

Anand J, Montville JB, Ahuja JKC, Goldman JD, Heendeniya KY, Moshfegh AJ. What We Eat in America, NHANES 2007-2008: comparing day 1 and day 2 dietary data [abstract]. *The FASEB Journal* 2011; 25:348.5.

Rose D. Food stamps, the thrifty food plan, and meal preparation; the importance of the time dimension for US nutrition policy. *J of Nutr Educ Behav* 2007; 39, 226-232.

Cyprus Blaise Integrated Census System (CY-BICS)

Costas K. Diamantides, Statistical Service of Cyprus

Lon Hofman, Statistics Netherlands

1 Introduction

Aim of the census of population is to enumerate the population, the households and the dwellings and to collect information on the demographic and socio-economic characteristics of the population and households, on the size and amenities of dwellings and the geographic distribution of the population, households and dwellings. This information is essential for governmental policy-making, planning and administration. The census is in fact one of the main sources that provide statistical data which is used in policy development in such fields such as education, employment and manpower, housing, rural development etc.

For the 2011 census the Statistical Service of Cyprus (CYSTAT) had set the following main objectives:

- i. Collect data of high quality
- ii. Timely publication of results
- iii. Minimize the cost
- iv. Full and accurate coverage

Secondary objectives derived from the need to monitor the performance of all the personnel involved as well as to obtain a flexible and scalable system, or parts of it, so that can be configured easily for other censuses and surveys.

To achieve the census objectives, CYSTAT in cooperation with Statistics Netherlands (CBS), developed the Cyprus Blaise Integrated Census System (CY-BICS). The aim of CY-BICS was twofold. First, to capture the census data by applying the Computer Assisted Personal Interviewing (CAPI) method and second, to incorporate innovative and highly automated technologies for the production of accurate data quickly and efficiently at a low cost.

2 Background

The first census of population in Cyprus was conducted in 1881 and recorded a total population of 186.173 inhabitants. Until 1931, censuses were carried out every 10 years. Due to the World War II, the next census was conducted in 1946 and then in 1960, the year in which the Republic of Cyprus was established. In the 70s there were two censuses, one in 1973 and another in 1976. This exception was due to the 1974 Turkish invasion. The following censuses were carried out in 1982, 1992 and 2001. In 2001 a population of 703.529 inhabitants was recorded in the government controlled area (Statistical Service of Cyprus, 2011).

The 2011 census was conducted for the first time in the framework of a European regulation¹. The Regulation stipulates that National Statistical Offices should provide reliable, detailed and comparable data on population and housing, following basic principles, definitions and specific deadlines for the submission of data and metadata.

¹ Regulation 763/2008 of the European Parliament and of the Council of 9 July 2008 on population and housing censuses

In order to safeguard the data quality, traditionally the censuses of population in Cyprus are carried out with personal interviews. Although the data capture is a small part of a national census project it is one of the most critical, costly and time consuming activity (United Nations, 2009). In order to fulfil the objectives of the census, CYSTAT considered three options for carrying out the data capture in 2011.

The first option examined was the “traditional” method i.e. the manual entry from paper. In this approach the operators type in the responses they see on the physical census form into the computer system. Before the data entry all questionnaires are manually edited and coded. This method has significant disadvantages such as: large number of staff needed for editing, coding and data entry, high cost of manpower, time consuming process, potential for errors during data entry affect data quality and finally, large volumes of paper.

The second option considered was the optical data entry method which was applied in the 2001 census of population. Based on the 2001 experience this approach has several disadvantages such as: it requires specially printed and cut forms, sophisticated hardware/ software, errors in recognition affect data quality and finally, no significant difference in the total cost compared to the traditional method.

The third alternative was the use of Computer Assisted Personal Interviewing (CAPI). CYSTAT has been using Blaise for more than 10 years in household surveys. Although there is some experience in Blaise the knowledge is very basic and definitely not sufficient for the needs of a census. For this reason, CYSTAT sought the cooperation of CBS. A feasibility study prepared by CYSTAT and CBS showed that such a system would fulfil all the census objectives. It would be cost efficient as all the costs for the manual data entry, editing and coding are eliminated. The data collected is of high quality as the manual input errors are reduced, there is data validation during the interview and consequently the logical error are reduced. Finally, the census information is processed faster leading to timely publication of results.

Alea jacta est! CYSTAT in cooperation with CBS would develop the Cyprus Blaise Integrated Census System (CY-BICS) with the aim to capture the census data by applying CAPI and to incorporate innovative and highly automated technologies for the production of accurate data quickly and efficiently at a low cost.

3 Census Organisation

3.1 Organisational Structure

The census’ geographical breakdown parallels the organisational division of the fieldwork. The geographical division into enumeration blocks facilitates the better organisation of the census as well as the collection of data at a very detailed level. In this way it is possible to make a precise indication in any geographical subdivision and in addition, double entries or omissions are avoided.

The government controlled part of Cyprus was divided into clear geographical limits according to the official administrative boundaries as defined by the Department of Lands and Surveys. In particular, within each of the five districts of Cyprus (Lefkosia, Lemesos, Larnaca, Ammochostos (rural only), Pafos) the following geographical levels were specified: Municipality/ Community, Quarter, Enumeration Block (Figure 1. Geographic Hierarchy). Each enumeration block had a maximum area of 1km² and around 300 inhabitants based on the data of the previous census. A total of 3.145 enumeration blocks were formed.

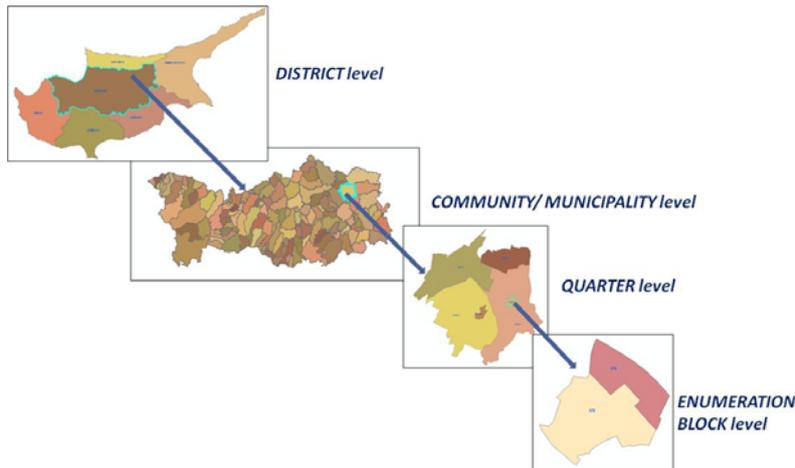


Figure 1. Geographic Hierarchy

To facilitate efficient fieldwork, a hierarchical organisational structure was designed (Figure 2. Hierarchical Organisational Structure). The field setting comprised the enumerators peaking at 800, 80 supervisors, 13 Assistant District Officers (ADO), 4 District Officers (DO) (Larnaka District Officer was also responsible for Ammochostos) and the managerial team comprised of the Director of CYSTAT, the Chief Statistics Officer who had the overall responsibility of the census and the IT Director. Technical support was provided by the support team at each level of the hierarchical system.

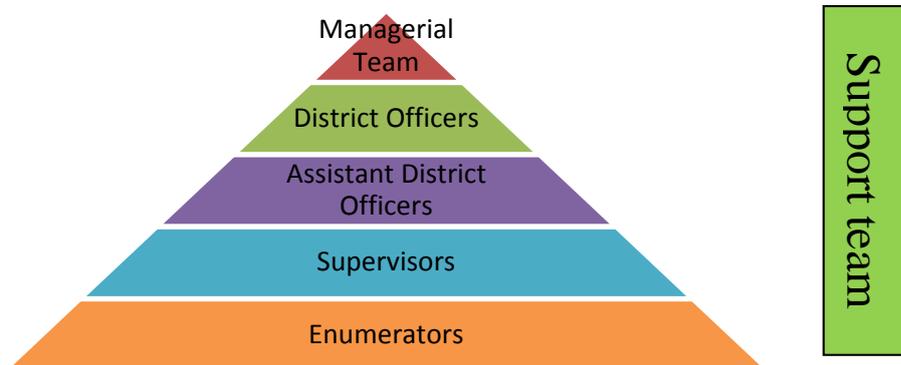


Figure 2. Hierarchical Organisational Structure

DO/ ADO were permanent staff of CYSTAT, while supervisors and enumerators were temporary employees recruited for the census. The duties of DO/ ADO included the organisation and supervision of the fieldwork in the district. The task of the supervisors was to co-ordinate, supervise and check the daily work of the enumerators. On average each supervisor was responsible for 10 enumerators. Each enumerator was responsible to enumerate on average four enumeration blocks.

Recruitment of temporary staff was carried out in each district by DO/ ADO. Training was carried out by the 'ladder method', each rung of the hierarchy training the one below it. In particular, DO/ ADO were trained by the development team and users team (see section 4.1), the supervisors by the DO/ ADO and the enumerators by the supervisors.

3.2 Geographical Information System (GIS)

Throughout the history of population censuses in Cyprus maps were generated and provided to each district office. The maps covered the whole urban area, which was divided into the administrative levels (municipality-quarter) and enumeration blocks. These base maps constituted the reference of

the supervisors and interviewers in each district. At the same time a map of the relevant enumeration block with well defined boundaries was given to each interviewer.

For the first time in the history of census-taking in Cyprus, the geographic part was based on computerized mapping through the use of the GIS. The digitization and encoding of the entire road network in the government controlled area of Cyprus was assigned to a private firm by CYSTAT and the Department of Lands and Surveys.

The GIS has significantly assisted in the organisation and implementation of the census, by improving the coverage as well as in supporting and facilitating the process of data collection. Furthermore, the GIS facilitate the presentation of the results in the form of thematic maps.

Each enumerator was given two maps for each enumeration block. One map was a satellite image of the enumeration block (Figure 3. Satellite image) and the other displayed the streets of the enumeration block (Figure 4. Map with streets). Each map displayed the boundaries of the enumeration block in red colour and each street type (motorway, asphalted, loose surface, under construction, pedestrian) by a different colour.



Figure 3. Satellite Image

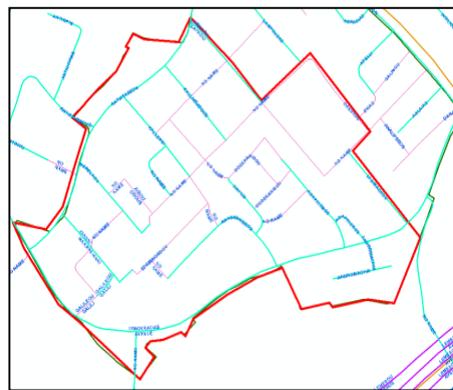


Figure 4. Map with streets

4 The Cyprus Blaise Integrated Census System (CY-BICS)

CY-BICS aimed to offer users a friendly environment to work with and to carry out automated procedures the maximum possible extent. The inclusion of all validation and consistency checks in the electronic questionnaire facilitated the capture of high quality data. The use of automation within and between applications enabled the near real time quantitative and qualitative monitoring of the data collection and the identification of coverage problems.

CY-BICS consists of four applications as follows:

- i. Electronic Questionnaire (EQ)
- ii. Supervisors' Application (SA)
- iii. District Officers' Application (DOA)
- iv. Central Application (CA)

This section provides information on the development of CY-BICS, the dataflow and the four applications.

4.1 Development

The development of CY-BICS started two years before the census. CY-BICS was designed and built in a stepwise approach, i.e. first the electronic questionnaire followed by the supervisors application, the district officers application and finally the central application.

During the development period two teams were setup, the development team and the users team. The development team had the responsibility for constructing the system and the users team for specifying the requirements, evaluating and testing the system during its build up. The development team was composed of three persons, two from CYSTAT and one from CBS who was the expert in programming with Blaise. One person from CYSTAT had the responsibility for the design and analysis of the system and acted as the mediator between the development team and the users team. The other person from CYSTAT contributed to the programming in cases where no expert knowledge was required and was also involved in the testing of the system. The users team comprised of four persons all CYSTAT staff.

During the two year period, three missions to Cyprus of the Blaise expert were carried out. During these missions it was possible to discuss in person the development process as well as to resolve several issues on the spot. The development mainly was carried out by the Blaise expert in the Netherlands and files were exchanged through the internet.

The first version of the EQ was tested in a pilot survey that took place in October 2010. In this survey four enumerators were trained in the handling of electronic questionnaires and personal interviews were carried out in households which were randomly selected. The survey proved to be very useful as it gave the opportunity to identify and rectify errors of the application. Moreover it enabled in the determination of the required settings of the netbooks and in the adjustment of the users manual. One month later a second pilot was carried out. This pilot took place within CYSTAT, i.e. the respondents were CYSTAT employees, and its aim was to test whether the errors identified in the first pilot were corrected. The role of the enumerators was taken by the members of the users team. The outcome was very positive and the electronic questionnaire was considered as completed.

The SA and DOA were mainly tested by the members of the development and users teams. A final test of all applications, except the central, was carried out in June 2011 at the presence of the Blaise expert (third mission to Cyprus). This test was carried out by all district and assistant district officers (all CYSTAT permanent staff). The aim of the test was twofold. First, to give the opportunity to all permanent staff to get acquainted with the different applications and second, to make a final test of the electronic questionnaire, the supervisors and district officers application. As a result all participants got familiar with the census applications, all problems were rectified and suggestions for improvement were implemented.

For each application the development team prepared a users manual which was used both as a guide during the training course and as reference during the census.

4.2 Dataflow

The setup in each district was similar the only difference being the size (Figure 5. Dataflow in each district). Dataflow within each district office was based on the available LAN infrastructure. In each district there was a central computer which acted as a server. The technical specifications of the 'server' were exactly the same as those of the supervisors PCs and are described in Appendix A'. The DO/ ADO used their own PCs and consequently there were different kinds of specifications. However, this heterogeneity in the PCs imposed no restrictions to the use of the different CY-BICS applications. In general, the technological infrastructure applied was very 'basic' and there was neither need for expensive hardware nor for other 'means' for data transfer rather than cables.

Although all computers were connected through the LAN the data transfer between the computers was determined by the different CY-BICS applications. Each netbook could connect at any point of the LAN but SA could only receive data from specific netbooks and send data only to the district

‘server’. There was no data transfer from/ to the district officers’ application but there was only access to the Blaise database installed on the district ‘server’.

A number of supervisors were responsible for the supervision of data collection in rural areas only. In those cases the supervisors were provided with a laptop on which SA was installed. As the meetings of the supervisors with their enumerators were taking place in different places each time, away from the district offices, the data transfer between the netbooks and laptops was carried out with the use of cross cables. The data were transferred to the district ‘server’ from the supervisors laptops each time the supervisor was visiting the office, about twice a week.

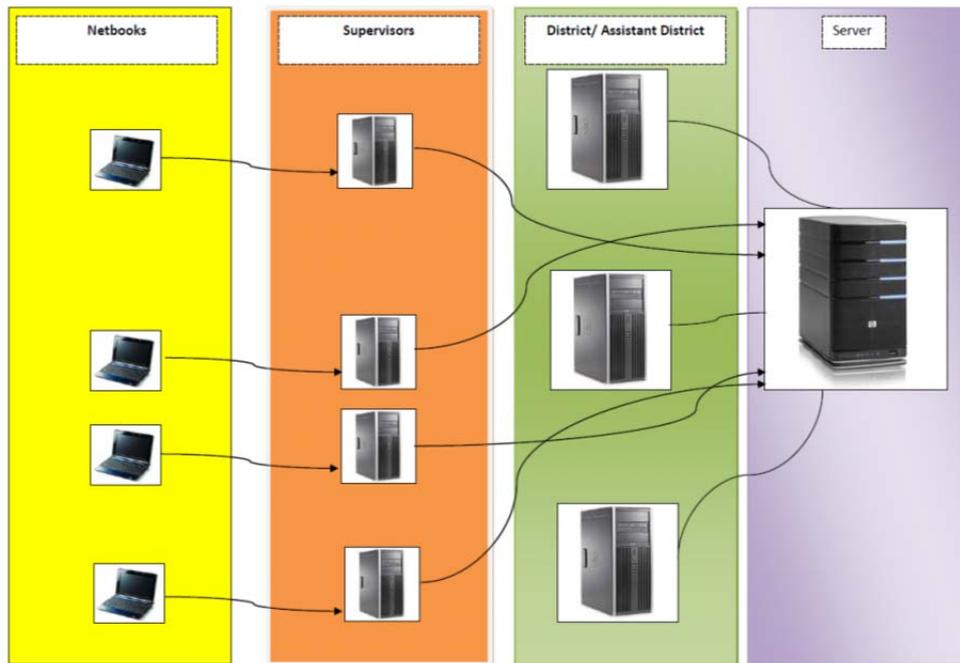


Figure 5. Dataflow in each district

Data from districts were transmitted to a central ‘server’ located in the main offices of CYPSTAT in Lefkosia through the Government Data Network (GDN) which provides a fast and secure way for data exchange (Figure 6. Dataflow within Cyprus). Data were transmitted to the central ‘server’ automatically every evening according to a predefined transmission schedule for each district. The technical specifications of the ‘server’ were the same as those of the district ‘server’.



Figure 6. Dataflow within Cyprus

4.3 Electronic Questionnaire (EQ)

The working unit of the census of population was the enumeration block. Each enumerator was responsible for the completion on average four enumeration blocks. EQ was designed in such a way as to map the logical sequence of the actions of the enumerator. The enumerator should first handle the street information and then the dwelling questionnaire which consists of five parts as follows:

- PART A: Housing Unit questionnaire
- PART B: Household questionnaire
- PART C: Household Roster
- PART D: Personal questionnaire
- PART E: Completion Status

In Blaise there were two main datamodels, one for the housing unit (Part A) and another for the household (Parts B-E). The last question of Part A concerned the number of households residing in the housing unit. Depending on the answer given to that question the corresponding number of household questionnaires were created.

The adoption of the GIS for the census enabled the unique identification of all street segments within each enumeration block. Consequently for each enumeration block there was available a list of streets with unique codes and names. Before the beginning of the census each netbook was uploaded with all the streets information for all enumeration blocks of the district. The enumerator after selecting the enumeration block was then given the option to select a street from the streets list (Figure 7. Streets within an enumeration block). Each enumeration block was password protected and the enumerators were provided only the passwords of the blocks they had the responsibility of.

Street code	Street name	Dwellings?	#Dwellings	#Completed	#Started	Status	Original name	Remark
1012030006	Αθαλάσσης Λεωφόρος		22	22				
1012030089	Γρηγορίου Λουβιά		22	22				
1012030090	Γυθείου	No						
1012030093	Δασομπέλειας		33	33				
1012030141	Καλλιόπης		2	2				
1012030235	Πεντέλης		31	31				
1012030239	Πέτρου Ηλιόδη		19	19				
1012030315	Φώτη Πίπτα		12	12				
1012030318	Χαράλαμπος Καλαϊτή		5	5				
1012030320	Χριστοδούλου Εγγλέζου		16	16				
1012030321	Χριστοδούλου Καννάουρου		10	10				
1012030495	Χ-0495	No						
1012030678	Χ-0678	No						

Figure 7. Streets within an enumeration block

Due to the time lag between the last updating of the GIS with the streets information and the census it was expected to find differences such as addition of new streets and change of street names. For this reason EQ facilitated the addition of new streets and the correction of the street names. Both procedures could be carried out very easily at the ‘Streets in enumeration block’ table (Figure 7. Streets within an enumeration block). This table contains several information at the street level such as the existence or not of housing units in each street, the number of housing units and the number of ‘complete’ and ‘incomplete’ cases.

A case was considered as ‘complete’ if one of three conditions were satisfied. First, the housing unit was occupied by one or more persons and all five parts of the dwelling questionnaire were completed in full. Second, the housing unit was vacant and third, the housing unit was used as a second home (holiday). A case was considered as ‘incomplete’ if any of the five parts of the dwelling questionnaire was not completed in full.

After selecting the street the application displays the table ‘Housing Units’ which includes all the currently listed housing units of this street (Figure 8. Housing units listed within a street). For each housing unit the table displayed several information such as the status (‘complete’/ ‘incomplete’), if the housing unit is occupied or not, the household/ flat number, the number of households, there presence of any remarks/ DK and whether the questionnaire had been checked by the supervisor.

HousingID	Status	Occ status	Housenumber	Flatnumber	Household count	Remark
0001	COMPLETED	Inhabited	133		2	ΠΑΝΩ ΔΕΞΙΑ
0002	COMPLETED	Inhabited	135		1	
0003	COMPLETED	Inhabited	133		1	ΑΡΙΣΤΕΡΑ ΤΟ ΔΕΥΤΕΡΟ
0004	COMPLETED	Inhabited	133		1	ΑΡΙΣΤΕΡΑ ΤΟ ΙΣΟΓΕΙΟ
0005	COMPLETED	Inhabited	129	103	1	
0006	COMPLETED	Reserved	129	101		ΚΕΝΗ
0007	COMPLETED	Inhabited	123		1	
0008	COMPLETED	Vacant	133			ΚΕΝΗ ΔΕΞΙΑ ΙΣΟΓΕΙΟ
0009	COMPLETED	Inhabited	119	1	1	
0010	COMPLETED	Inhabited	119	3	1	
0011	COMPLETED	Inhabited	129	102	1	
0012	COMPLETED	Inhabited	117	11	1	ΠΡΩΤΟΣ ΟΡΟΦΟΣ
0013	COMPLETED	Inhabited	117	41	1	ΤΕΤΑΡΤΟΣ ΟΡΟΦΟΣ
0014	COMPLETED	Vacant	111	111		ΚΕΝΗ
0015	COMPLETED	Inhabited	109	1	1	ΚΑΤΩ ΑΠΟ ΤΟ ΣΠΙΤΙ ΕΧΕΙ ΚΑΤΑΣΤΗΜΑΤΑ
0016	COMPLETED	Inhabited	111	301	1	

Figure 8. Housing units listed within a street

After adding/ selecting a housing unit the dwelling questionnaire must be completed. Part A includes questions for both resident and empty housing units. Specifically the questions refer to the building type, the number of rooms and the various available facilities such as kitchen, bathroom, heating etc. Part B includes questions about the ownership, rent etc. Part C is the list of names of household members and the relationship among members. Part D includes questions for all members (one questionnaire for each individual) usually resident in the household such as marital status, religion, employment/ unemployment etc. Finally, Part E includes questions on the completion of the various parts of the questionnaire and finally, the details of the contact person.

In the cases where the questions referred to country, municipality, citizenship and spoken language the data entry was carried out through trigrams. As a result the selection of the correct description and coding were carried out very efficiently.

In general, the comments of all users of EQ were very positive. Nevertheless, there is always room for improvement. Although the enumerators made a few suggestions for improvement during the census those were not possible to be implemented as the process of data collection was running. Those suggestions concerned the ability to delete streets that were created accidentally and the availability of the option to enter the code directly in the cases where the code was known instead of following the procedure of trigrams. As regards the general design of CY-BICS and in particular the integration of EQ and SA the restriction of one enumerator for each enumeration block and one supervisor for each enumerator caused some additional burden in the cases where the enumerators resigned and their job had to be assigned to another enumerator.

4.4 Supervisors' Application (SA)

The supervisors were responsible for checking and correcting the questionnaires completed by the enumerators. They were also responsible for the transfer of 'completed' cases from the netbooks to their PCs and for the production of summary reports at the enumeration block level. The tool for carrying out all the tasks was SA which can be considered as the 'heart' of CY-BICS.

Although the interface of SA looks very simple there is a lot of automation behind it. For instance, the completed cases are transferred by the press of a button but behind this function there are several processes carried out like the regeneration of a new unique id for the dwelling questionnaire on the supervisors' database, the change of mode of the dwelling questionnaire that stays on the netbook into read only and the creation and storage on the supervisors pc of a zip file with all the data of the specific enumeration block stored on netbook.

The supervisor have the option to work through SA either on the netbook of the enumerator or locally. In the former case (Figure 9. Supervisor application in 'netbook' mode) the supervisor is able to perform various tasks related to the handling of streets and the editing of the questionnaires (those not transferred from the netbook), produce summary reports for all 'complete' and 'incomplete' cases of the enumeration block and finally transfer the 'completed' cases from the netbook to the supervisor's pc. As soon as a dwelling questionnaire is transferred to the pc then it is not possible to make any changes on the netbook which it is now only available for viewing. The functions that are available in each application are displayed in the table of Appendix B'.

A total number of six reports can be produced by a click of the button as follows (sample of reports is available in Appendix C):

- a. Report 1: Number of 'completed' questionnaires within an enumeration block, by date
- b. Report 2.1a: Summary statement of completed housing unit/ household questionnaires within an enumeration block, by street name
- c. Report 2.1b: Summary statement of completed housing unit/ household questionnaires within an enumeration block, by transferred date
- d. Report 2.2: Summary statement of 'incomplete' housing unit/ household questionnaires within an enumeration block, by street name
- e. Report 3: Summary statement of housing units by street name
- f. Report 4: Summary statement on the status of each street, by enumeration block



Figure 9. Supervisor application in 'netbook' mode

When the supervisor chooses to work locally (Figure 10. Supervisor application in 'local' mode) the application allows to either work on the database with all the transferred 'complete' cases or to view the last copy of the netbook's database. In the former case the supervisor can perform the same tasks as an enumerator and some additional such as deletion of housing units and household members. In the latter case, i.e. when the supervisor wants to consult the last copy of the netbook's database, no changes are possible.

In both cases it is possible to produce the same reports as those described above with the exception of report 2.2 in the case when the supervisor accesses the database with the 'complete' cases. In all reports the source of data is clearly indicated i.e. netbook or 'complete' database.



Figure 10. Supervisor application in 'local' mode

The decision on whether an enumeration block was declared as completed or not left on DO/ ADO. SA facilitated the transfer of 'completed enumeration blocks' from the supervisor's pc to the district 'server'. However, this procedure could only be carried out by DO/ ADO. Depending on the login details, i.e. DO/ ADO/ supervisor, the application enabled or disabled the 'Complete enumeration block' button accordingly. As soon as the enumeration block was transferred to the district 'server' no changes could be made on the supervisor's database.

SA worked smoothly during the census and no problems or suggestions for improvement were reported.

4.5 District Officers' Application (DOA)

DO/ ADO through DOA were able to edit/ view data, produce summary reports and create backups of the census data (Figure 11. District officers' application). The application had direct access to the data that was stored on the district 'server'. On the district 'server' there were two sets of data. The first set was a copy of each enumeration block database with 'complete' questionnaires stored on the pcs of the supervisors. The copy on the server was created automatically each time the supervisor was exiting the database. The second set included all the completed enumeration blocks transferred from the pcs of the supervisors by DO/ ADO. DO/ ADO could only edit the latter data set but were able to produce the same type of reports for both sets.

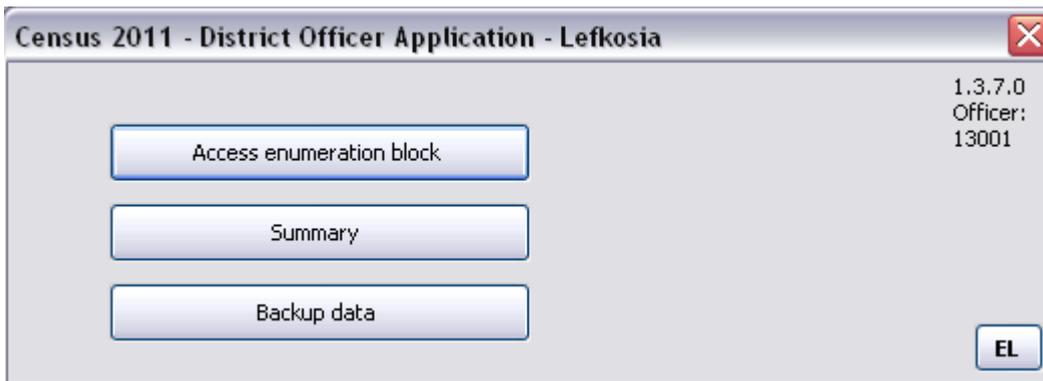


Figure 11. District officers' application

The access to the first data set was only for consultation purposes and in particular, to assess the work progress. Although DO/ ADO were responsible for declaring an enumeration block as complete the application was designed in such a way as to facilitate the capability of making changes on the transferred completed enumeration block. In this way, DO/ ADO were given full authorization to edit the data before transferring the enumeration block to the main census 'server' in Lefkosia. In practise, it proved to be a wise decision. The selection of the data set (Figure 12. Selection of data set) is made as soon as DO/ ADO choose to work either on the data or to produce reports.

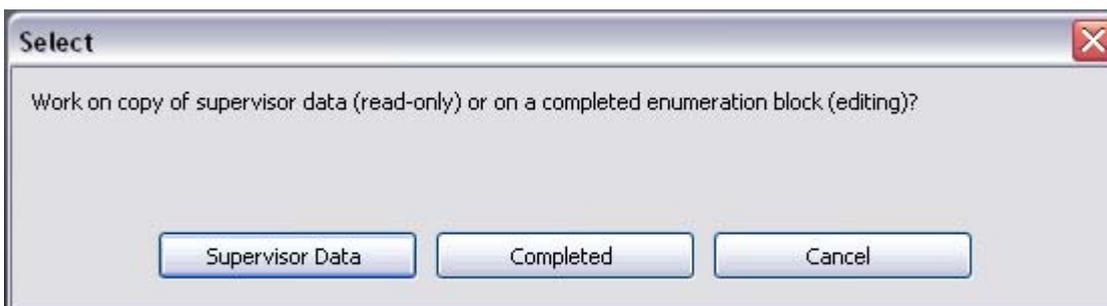


Figure 12. Selection of data set

The reports produced for the two datasets are similar. The first report produced by DOA contains information for each enumeration block such as the total number of households, housing units and number of persons. The information is sorted in three ways by enumeration block, enumerator or supervisor. There is also available the total figures for the whole district. The second report contains the same information but only for a selected supervisor. A sample from the reports produced by DOA is available in Appendix D'.

Similarly to the rest of applications DOA worked smoothly without any problems. The only suggestion received during the census which was not possible to be implemented at that time concerned the capability to produce the reports of SA. As explained by DO/ ADO that option would allow them to check the supervisors and enumerators work more efficiently.

4.6 Central Application

The Central Application was implemented for the exclusive use by the census managerial team (Figure 13. Central Application). The members of the team were located at the main building of CYSTAT in Lefkosia. The census central 'server' was also located at the same building.

The server contained the data from all districts that were replicated automatically from district 'servers' every evening. The data included both the completed enumeration blocks and the copies of the supervisors databases with the complete questionnaires.



Figure 13. Central Application

The members of the team were able to produce reports for each district separately as well as for the whole Cyprus. In this way, it was possible to monitor the progress of the data collection in each district and consequently in Cyprus (Figure 14. Data collection progress in Cyprus). Similarly to DOA it was possible to select one of the two data sets, i.e. database with ‘complete’ cases or database with completed enumeration blocks. The reports for each district were similar to those produced by DOA whereas in the case of whole Cyprus the total figures were just displayed on screen (Appendix E’).

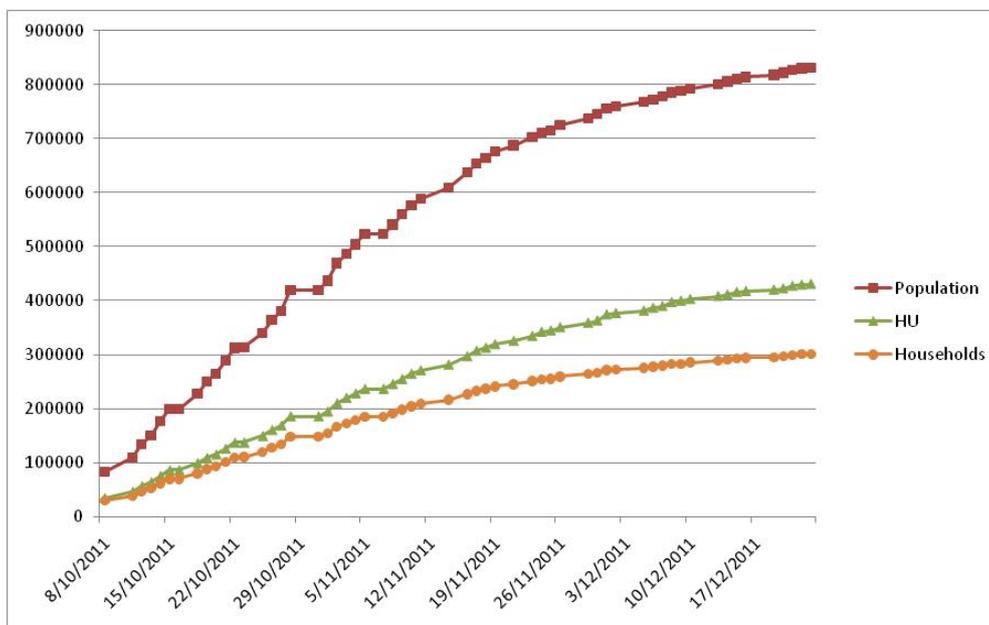


Figure 14. Data collection progress in Cyprus

5 Conclusions

The increasing demand for statistical data from the different categories of users such as policy makers, academics, journalists and others, result in application of such methods and procedures by

CYSTAT with which to achieve timely production and dissemination of reliable official statistics. Furthermore, the significant reduction in available funds requires the adoption of innovative methods through the use of modern technology.

A census of population is considered as the biggest challenge for a Statistical Office due to the large volume of data collected and the organization needed to coordinate the work of a large number of people employed. Traditionally, a census is considered as a good opportunity to implement new methods and techniques that will form the basis for the carrying out of subsequent surveys.

CYSTAT's response to the above challenges was the development and implementation of the innovative system CY-BICS. CY-BICS has fulfilled all the census objectives and in particular it is considered to be a big success for the following reasons:

Significant cost savings

By applying the CAPI method significant savings in cost have been achieved estimated at 25% of the total budget. The direct data entry, editing and coding in netbooks prevented the manual work after the data collection as well as the printing of questionnaires (only a small amount was printed).

Data quality and improved coverage

The data editing during the interviews but also the editing in near real time by the supervisors and the assistant district officers contributed to the collection of high quality data and improved coverage.

Timely dissemination

The public was informed about the progress in the data collection during the census. The first preliminary results were published immediately after the end of the census and the first analytical data were released within three months.

Development and implementation of similar systems in CYSTAT

All knowledge and experiences gained in the development and implementation of CY-BICS will be applied to future surveys and this can bring additional benefits for CYSTAT.

Environmental protection

By using electronic instead of printed questionnaires significant savings in paper have been achieved contributing to the protection of the environment. It is estimated that a volume of 28m³ of paper has been saved.

Immediate availability of sampling frame for household surveys

For the first time a sampling frame is available immediately after the completion of the census. Sample was drawn for three household surveys a few days after the end of the data collection.

6 References

Statistical Service of Cyprus. (2011). *Demographic Report 2009*. Nicosia: Statistical Service of Cyprus.

United Nations. (2009). *Census Data Capture Methodology, Technical report*. New York: United Nations.

Appendices

A. Hardware specification

Netbooks

Acer aspire 10.1" SD 1024X600 (WSVGA)

Intel Atom N455 (512K Cache, 1.66GHz)
 1024MB DDR3 1066 MHz
 250GB Storage Capacity
 Microsoft Windows 7 Starter Multilanguage OEM

PCs (District ‘servers’, main server, supervisors pcs)

Arrow-C2D desktop computers
 Intel Core 2 Duo E8400
 3.0GHz processor
 4GB (DDR2-800MHz) RAM
 Intel GMA 4500 (512MB) Graphics
 Philips 22” TFT multimedia monitor
 Windows 7 Pro

B. Functionalities available to enumerators, supervisors and district officers

	Enumerator - netbook	Supervisor - local (copy of netbook database)	Supervisor - netbook connection	Supervisor - local Complete database (transferred questionnaires)	District Officer - Supervisors copied database	District Officer - Complete Enumeration Blocks
Add Street	√	X	√	√	X	√
Delete Street	X	X	X	X	X	X
Change street name	√	X	√	√	X	√
Add Housing Unit	√	X	X	√	X	√
Delete Housing Unit	X	X	For questionnaires not transferred	√	X	√
Transfer Housing Unit from one street to another	X	X	For questionnaires not transferred	√	X	√
Edit questionnaire	For questionnaires not transferred	X	For questionnaires not transferred	√	X	√
Delete Household Member	X	X	For questionnaires not transferred	√	X	√
Add REMARK/ DK	For questionnaires not transferred	X	For questionnaires not transferred	√	X	√

√: Applicable
 X: Not applicable

C. Reports produced by SA

Report 1: Number of completed questionnaires within an enumeration block, by date

Source: Netbook

Date of the Report: 16-02-2012

Enumeration block: 102400104

Enumerator's code: 11082

Supervisor's code: 12001

DISTRICT: Lefkosia - (1)

MUNICIPALITY/COMMUNITY: Γέφυ - (1024)

QUARTER: ΔΕΝ ΥΠΑΡΧΕΙ ΕΝΟΠΙΑ - (10)

Date	No of Occupied HU	No of Not Occupied HU
02-12-2011	1	0
03-12-2011	2	0
05-12-2011	3	0
07-12-2011	1	0
10-12-2011	11	0
13-12-2011	7	0
Total	25	0

Report 2.1a: Summary statement of completed housing unit/household questionnaires within an enumeration block, by street name

Source: Netbook

Date of the Report: 16-02-2012

Enumeration block: 102400104

Enumerator's code: 11082

Supervisor's code: 12001

DISTRICT: Lefkosia - (1)

No. of households: 25

MUNICIPALITY/COMMUNITY: Γέφυ - (1024)

Occupied housing units: 25

QUARTER: ΔΕΝ ΥΠΑΡΧΕΙ ΕΝΟΠΙΑ - (10)

Not occupied housing units: 0

No. of household members: 93

% Edit: 0,0%

HOUSING UNIT				HOUSEHOLD										
ADDRESS				Housing Unit ID	Occ/Vac	HH serial nr	First name HH	Surname HH	Telephone Nr	Nr of HH members	Start date	Completed	Transfer	Edit
Street name	House nr	Flat nr	Remark											
Άρη Βελουχιώτη	■			0001	1	01	■	■	■	3	05-12-2011	05-12-2011	06-12-2011	
Άρη Βελουχιώτη	■			0002	1	01	■	■	■	3	10-12-2011	10-12-2011	14-12-2011	
Αγίας Λαύρας	■			0001	1	01	■	■	■	4	07-12-2011	07-12-2011	14-12-2011	
Αγίας Σοφίας	■			0001	1	01	■	■	■	5	13-12-2011	13-12-2011	14-12-2011	

D. Report produced by DOA

Summary data for all enumeration blocks - Sorted on Enumerator

Source: Supervisors data

DISTRICT: Lefkosia

Report date: 16-02-2012

No. of households: 117202
 No of Occupied HU: 115294
 No of Not Occupied HU: 26115
 Nr of HH members: 319699
 %Edit: 48,1%
 Total number of completed enumeration blocks: 0

Enumerator	Enum block	Supervisor	No. of households	No of Occupied HU	No of Not Occupied HU	Nr of HH members	%Edit	Completed EB
11001	102103403	12014	141	141	24	506	96,5%	
11002	102103600	12014	57	57	2	209	89,5%	
11002	102103701	12014	43	42	15	136	90,7%	
11002	102103702	12014	92	92	10	292	81,5%	
11002	102103901	12014	96	96	8	334	88,5%	
11003	102103503	12014	205	205	35	714	98,5%	
11003	102200201	12014	91	91	12	228	94,5%	
11003	102200202	12014	83	83	18	172	100,0%	
11004	102200101	12014	102	102	29	234	98,0%	
11004	102200102	12014	63	63	19	169	84,1%	

E. Report for the whole Cyprus

Summary Cyprus: Completed Enum Block ✖

Cyprus totals

No. of households	<input type="text" value="303258"/>	Total number of enumeration blocks	<input type="text" value="3156"/>
No of Occupied HU	<input type="text" value="299294"/>	Total number of completed enumeration blocks	<input type="text" value="3156"/>
No of Not Occupied HU	<input type="text" value="134269"/>		
Nr of HH members	<input type="text" value="836671"/>		
%Edit	<input type="text" value="48,6%"/>		

Implementing an Office & Field Survey Management System for PIAAC using Blaise

Jacqueline Hunt, Central Statistics Office, Ireland

Introduction

This paper describes the Blaise developments undertaken to facilitate the Central Statistics Office (CSO) participation in the Programme for the International Assessment of Adult Competencies (PIAAC). PIAAC is a collaborative project between governments and an international consortium to assess cognitive skills, IT literacy and formal educational attainment. The study is taking place across twenty-five OECD and partner countries in 2011-2012 with results due to be available in 2013. The PIAAC consortium's assessment instrument was designed to maximise cross-cultural, cross-national and cross-language validity. The application was developed as a "black-box" solution designed to run in its own virtual machine. All participating countries had to adhere to common technical standards when implementing the survey but each country was responsible for its own survey and case management functionality. The final CSO solution was developed almost exclusively in Blaise using a combination of Maniplus and Manipula scripts. This paper describes the design approach, the functionality provided in the various GUIs and the data management features.

1 PIAAC Applications

The PIAAC applications supplied by the consortium consist of a PIAAC Virtual Machine (VM) and the TAO platform that runs inside the virtual machine.

The TAO platform contains all the necessary software to run the following three stages of the PIAAC assessment.

1. The global PIAAC interview workflow (WF) describing the case initialisation, the disposition codes, the ICT-Screener, ICT-CORE and ICT-Tutorial; the navigation among instruments; the booklet selection controls, and the interviewer instructions for Paper-and-Pencil booklets;
2. The background Questionnaire (BQ); and
3. The Cognitive Instruments (CI) including general and domain-specific orientations.

The TAO platform collects and manages all the PIAAC data. Statistical offices taking part in the study were instructed that no additional software should be installed inside the PIAAC official VM. External software could be installed on the host machine, outside the virtual machine, to manage the survey processes.

The PIAAC data is stored as case objects; the case objects can be imported to or exported from the TAO platform in a zipped XML file format. If an invalid XML file is provided to the system, it will not be imported and the case will not be available for interview in the TAO. On completion of a PIAAC interview an object based [PERSID].zip archive is produced and exported to an output folder on the host machine.

A series of scripts were provided by the consortium to access the data from the TAO platform. The scripts can be triggered from external applications using the Case IDs as parameters to select specific cases. Table 1 is a list of some of the scripts provided.

Script	Function
<i>Basic scripts</i>	
StartCAPI(optional:'new' or PERSID)	Starts the VM and the TAO application
ResumeCAPI(PERSID)	Starts the VM and the TAO application, resumes the interview specified by the PERSID.
ExportResult(optional:PERSID)	Copies all available data from the PIAAC VM to the Windows environment. If the result data of a certain interview is required the case can be specified by adding the PERSID.
StopVM	Terminates the currently running VM.
HandleCAPI	Starts or resumes the current virtual machine with the given PERSID, dependent on the state of the interview inside the VM.
<i>Advanced scripts</i>	
DropCase(PERSID)	Deletes the interview specified by the PERSID from the TAO database.
DumpCase(PERSID)	Activates to dump an interview specified by the PERSID within the VM and copies the resulting SQL file to the Windows environment. The interview is exported in its current state. The output can be used to Restore a case.
ImportCase(PERSID)	Copies a dump file generated from the DumpCase script from the Windows environment to the VM and imports it to the TAO.
DumpAllCases	Activates to dump the entire TAO database and copies the resulting SQL file to the Windows environment.
RestoreAllCases	Copies a dump file of the entire TAO database to the VM.
GetCaseState (PERSID)	Gets the states of a certain case or of all cases from the VM to Windows. If getting the state of a certain PERSID the return value is set to the cases running state.

Table 1: VM Scripts

2 Lessons from the Pilot

The office had a very tight timetable to prepare for the pilot study scheduled for April 2010 due to late official confirmation to participate in the survey. This resulted in a very short development time of five months, starting at the end of October 2009. The limited time available meant we had to prioritise what could be delivered. A lot of time was required in the beginning to become familiar with the installation and administration of the consortium software. Due to the limited time available for any new development it was decided that the pilot would be conducted with the existing survey management applications that were used for the Quarterly National Household Survey (QNHS) and the Survey of Income and Living Conditions (SILC) see Fig. 1, for the list of software tools used in these systems. The intention here was to reduce the development work required by reusing existing software and prioritise any specific requirements necessary to conduct the survey.

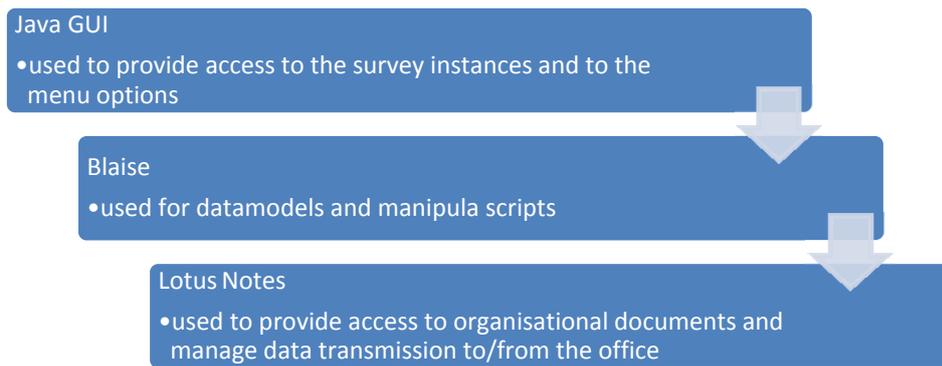


Figure 1: Existing HSCU Survey Management Application Components

Signification changes to the survey management applications were required to adapt them to work with the PIAAC VM and its object based file structure. The PIAAC case files are stored as objects where as the QNHS and SILC survey data are stored as Blaise databases.

The other essential requirements identified for the pilot included the following;

- a preliminary questionnaire to identify the Household composition,
- a selection function to identify the appropriate survey respondent from the Household composition,
- data from this questionnaire to be accessible to the PIAAC assessment,
- an administration block to record progress, and
- report functionality based on the PIAAC disposition codes.

The pilot study was considered a success by the survey area based on the number of completed assessments successfully collected. However, there were a number of problems reported with the field applications that were attributable to the lack of integration between the PIAAC system and the survey management applications. Some of the issues were known in advance of the pilot but there was no time available to correct them. One of the difficulties was the poor transition between completing the Screener and starting the PIAAC assessment. The Screener is the Blaise datamodel that was used to collect the household details and select a random person from the eligible residents to complete the PIAAC assessment. The interviewers had to remember to prepare an input XML file before starting the assessment to ensure individual details would be imported into the PIAAC TAO. The main problems reported by the field interviewers after the pilot were usability and functional issues. There were some issues with performance and screens freezing during the PIAAC assessments, these were resolved by the consortium in advance of the live survey. There was also some confusion and difficulties accessing and closing the VM, which sometimes resulted in multiple VMs being open at the same time. Another issue was that the Screener household data had not been tied down sufficiently and could be changed after the random selection had taken place. This resulted in a number of problems with mismatched data. The link between the PIAAC assessment and the survey management applications was limited which meant that the interviewers could not view the status of their PIAAC cases from the survey management application. Addressing these issues for the live survey required a complete re-design of the survey management applications. The key lesson learned from the pilot was that the field applications required much tighter coupling with the PIAAC “black-box”.

3 Live Applications

3.1 Requirements

The survey area IT requirements for the live survey were similar to the pilot survey, they included

- A facility to capture the household constituents and randomly select one eligible person to complete the PIAAC assessment
- Para data capture to include the number of contacts with the household and the outcome of each contact
- Single point of access for all data entry components
- Laptop GUI for interviewers to manage their workload, display case details, and transfer updated cases to the office
- Data management and administration including management of data transfer from the field to the office
- Field supervisor and office GUI and reports.

Based on the pilot feedback and the difficulties reported, the key features of the new design had to encompass

- Better integration between the CSO data collection, the survey management applications and the PIAAC applications
- Automation of as many user tasks as possible including starting and stopping the PIAAC VM
- Enhanced access and views of the data including an up to date case status for the entire case.

3.2 Sample

How to distribute the sample among the field staff was one of the first issues to address. As this was a fixed survey the sample households were known in advance. A sample file was produced with the Interviewer IDs assigned to each case based on their geographic location. The same sample lookup file was installed on all the interviewer laptops. The interviewer menu was used to filter the sample file to display only relevant cases for an individual interviewer. The interviewers could also filter their own work allocations to create different views of the cases, for example, views based on the case sequence number or the case status. Functionality to amend the sample lookup file and prepare it for redistribution to the field if the work allocation needed to be altered was added to the office menu.

3.3 Data Collection

The data collection had a three phase approach. During the initial contact with the household the interviewer attempted to interview at least one of the residents to identify all the individuals living in that household. When the personal details were collected the PIAAC respondent was randomly picked from the number of eligible people in the household using the Blaise random function. Eligibility was based on the criteria that the individual was between the age of 16 and 65. A Blaise datamodel called the Screener was used to collect the household details, confirm the age criteria and select the person to complete the assessment based on the Blaise random function.

The next stage in the data collection process was to complete the PIAAC assessment. For the live survey we needed to restrict the interviewer's access to the PIAAC VM. During the pilot the interviewers had to select between conducting a *New Assessment* and *Resuming* an existing one. Resuming an interview brought the interviewers to a "Welcome Screen", see Fig. 2, where they could select the case to resume. This screen also contained a button to start a *New Interview* that could not be disabled.

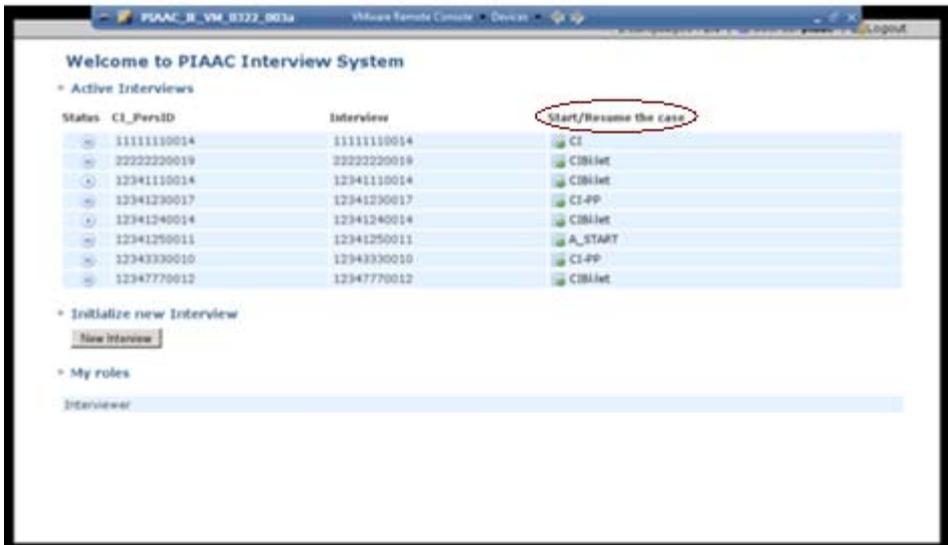


Figure 2: PIAAC Welcome Screen

Allowing the interviewers direct access to the VM resulted in a number of incidents of mis-matched Case IDs as the interviewers could key new Case IDs directly into the TAO via the *New Interview* button. We needed to find a better solution to access the PIAAC assessment regardless of it being a *New* or *Resuming* case. As we did not know in advance who the household constituents were and who would be selected to complete the assessment it was not possible to pre-fill the PIAAC VM with any individuals' details. This meant that the input XML file for all new cases had to be prepared after the Screener details were complete. A Blaise Alien Procedure was used in the Screener datamodel to create the XML file that could be used to pass data directly into the PIAAC VM without any interviewer intervention. The Screener writes the information relating to the selected individual such as age, gender, highest educational qualification to a new input XML file, using the PERSID field as the file name. The file name is then used as a parameter to launch a case in the PIAAC VM with the imported case details.

```

PIAAC2011Scr.bla | Review.inc | Procedures.inc | CreatePIAACInputZipFile.man
Description      : CreatePiaacZip - Used to create a zip file which will be used by the Piaac VM Questionnaire
*****
PROCEDURE CreatePiaacZip
PARAMETERS
  TRANSIT BlockNo, LDU, PiaacID, Name, Age, Phone, Addr , BookID_PPC, BookID_PP1, BookID_PP2, BookID_PRC : STRING
  TRANSIT Sex, SexCode : T$Sex
ALLEN ('CreatePiaacInputZipFile.Msu', 'MakeZip'){ Manipula call }
ENDPROCEDURE

```

Figure 3: Procedure Call Code

A new AutoIt (Automation and Scripting Language) script, called LaunchPIAAC was developed using the consortium scripts to handle the process of opening the VM, starting a new or resuming an existing case, writing the status result to a file on the host machine, exporting the case to the host machine and finally, closing the VM. This restricted and controlled the interviewer's access to the VM and prevented any case being created without a corresponding input XML file.

The consortium scripts used in LaunchPIAAC are:

- **HandleCAPI**
Starts the VM and relevant case at the last point of the assessment. It calls the StartCAPI/ResumeCAPI/GetCaseState/ImportCase scripts with the PersID passed as a parameter from the Blaise screener.
- **GetCaseState**
This script continuously polls the state of the case in the VM while it is open, the last state is exported to the file c:\piaac\piaacstatus.txt file.

- **DumpCase**
If the GetCaseState returns a status of *'paused'* or *'finished'* this script dumps the case and copies the resulting SQL file to c:\piaac\administration.
- **ExportResult**
If GetCaseState returns a status of *'paused'* or *'finished'* this script exports all case data and copies it to c:\piaac\output\PERSID.zip as a completed case.
- **StopVM**
Shuts down the VM.

The PIAAC assessment started with some initial CAPI questions asked by the interviewer followed by the main assessment collected via CASI (Computer Assisted Self-Interviewing). Fig. 4 is a screen shot from the early part of the PIAAC interview.

The screenshot shows a web-based interview form with the following sections:

- I need to verify a few pieces of information:**
 - [CI_Name] Your First Name**: Respondent first name:
 - [CI_Gender] Your gender**:
 - < 01 > Male
 - < 02 > Female
 - [CI_Month] The month of your birth**:
 - < 01 > January
 - < 02 > February
 - < 03 > March
 - < 04 > April
 - < 05 > May
 - < 06 > June
 - < 07 > July
 - < 08 > August
 - < 09 > September
 - < 10 > October
 - < 11 > November
 - < 12 > December
 - [CI_Year] The year of your birth**: Year:
- [CI_Age] Your age**: Respondent age:
- [CI_Telephone] Your telephone number**:
 - Interviewer Instruction**: Do not enter the country code, enter the area code and the full number in a continuous sequence
 - Respondent telephone number:
- [CI_Address] Your address**:
 - Interviewer Instruction**: Enter the respondent address in the following way: Number, Street, Zip code, Locality
 - Address:

Figure 4: PIAAC Screenshot

The third and final part of the data collection was the para data section. Another Blaise datamodel, called the Admin was used to collect the survey case's call histories. Fig. 5 is a screenshot of the Call History section of the datamodel. During the pilot only one datamodel had been used to collect the household components and para data information. For the live survey two datamodels were used to provide more flexible access to the datamodels particularly the Admin as this can be accessed multiple times by the interviewers to record contact and case status information.

Enter a valid date

Visit	Dat	Day	Tim	Typ	Intrn	Scr	Init	BQ	Core	Excr	ExType	Duration	NxtVis
Call[1]	1st Visit	16/11/2011	Wednesday	21:00	2	6	4						1
Call[2]	2nd Visit												
Call[3]	3rd Visit												
Call[4]	4th Visit												
Call[5]	5th Visit												
Call[6]	6th Visit												
Call[7]	7th Visit												
Call[8]	8th Visit												

Figure 5: Visit Data Entry

The interviewer data collection workflow is described in Fig. 6.

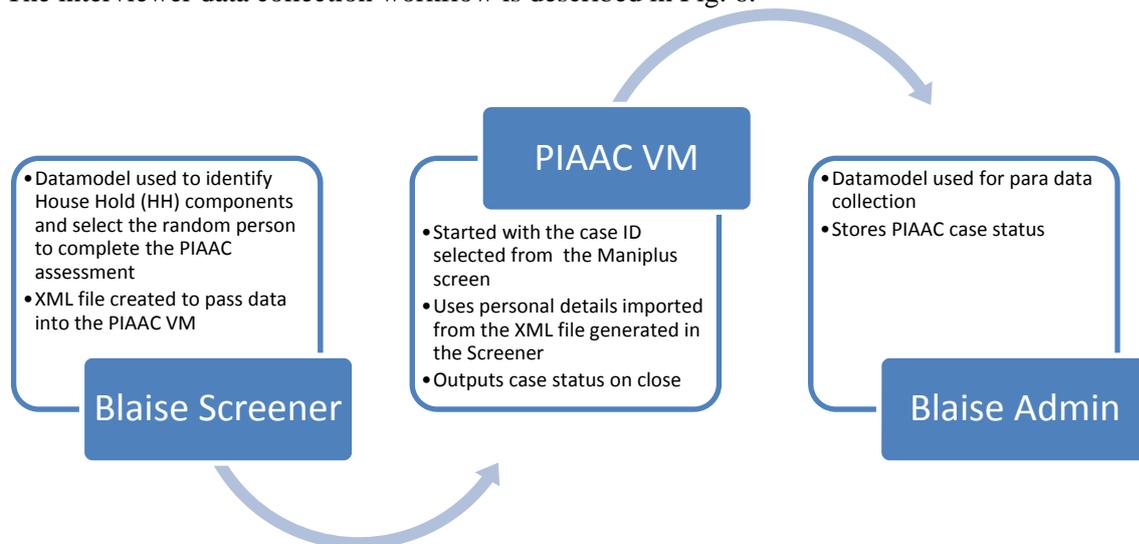


Figure 6: Data Collection Workflow

The interviewer's data collection workflow is described below:

- (i) Complete Screener.
- (ii) Screener application selects eligible respondent for interview.
- (iii) If selected person is available, proceed with PIAAC Background Questionnaire, Exercise and Observation Module (ZZs).
- (iv) Shut down laptop.
- (v) Complete Gratuity Card detail and leave household.
- (vi) Update Admin as soon as possible.
- (vii) Transmit Data to CSO
- (viii) If Selected respondent is unavailable, ensure contact details are correct.
- (ix) Contact person and arrange an appointment as soon as possible.
- (x) Record contact information in the Admin.

The interviewer could access the Screener, Admin and PIAAC components from a TempTable in the Maniplus application, see Fig. 7. Selecting a row in the table followed by the component will identify the Case ID and open the relevant case in either of the Blaise datamodels or the PIAAC VM. The information displayed in Fig. 7 is taken from three sources; the Blaise Screener datamodel, the Admin datamodel and the sample lookup file.

Home Page for Interviewer: 006											
Seq No.	Block No.	Ldu No.	PersId	Contact Name	Screener Status	Piaac Status	Finalised	Visit No.	Last Day	Last Visit Date	Last Tim
▶	1 5074	152	50741520015	JACQUELINE HUNT	Refusal - household member		Yes	1st Visit	Wednesday	16/11/2011	21:00
	1 5074	155									
	1 5074	157									
	1 5074	158									
	1 5074	164									
	1 5074	182									

Search: Key type: Seconds

1:210

Sort Start Screener Start PIAAC Start Admin Cancel

Figure 7: Survey Home Page displaying the list of cases

3.4 Survey Management GUIs

Blaise Maniplus applications were used for all the interviewers and management menus.

3.4.1 Interviewer Application

Interviewers could view their case load, access the survey instruments, view appointments, manage their cases to transfer to the office and view transmission logs and progress reports. Starting any of the data entry components from the Home Page was a straight forward procedure for the interviewer. The interviewer selected the case from the TempTable, see Fig. 7 and the case ID was passed into the datamodel or the PIAAC VM as a parameter.

Traditionally, when transmitting data to the office we have used the Blaise form status of *New* and *Changed* to select the cases for transfer. This worked for the Blaise Screener and Admin data as we retained the Blaise database structure. However, a way of replicating finding *New* and *Changed* PIAAC cases was required. This was resolved by initiating an action in the Maniplus script to copy the PIAAC output zip file to a staging folder on the local machine as the PIAAC VM was being shut down. When an interviewer initiated a data transfer to the office the application would pick up any *New* or *Changed* Blaise records and any PIAAC case files that were held in the staging folder. The staging folder is cleared after every transmission so only cases that had been accessed in the PIAAC VM since the last transmission were transferred to the office. A Transaction Log was created after every upload of data for the interviewers to view the Case IDs and component i.e. Screener, Admin, PIAAC that were successfully transmitted to the office. In addition to the *New/Changed* data transfer the interviewers could also select specific cases to transfer, see Fig. 8. This proved to be a useful option for the field and office staff.

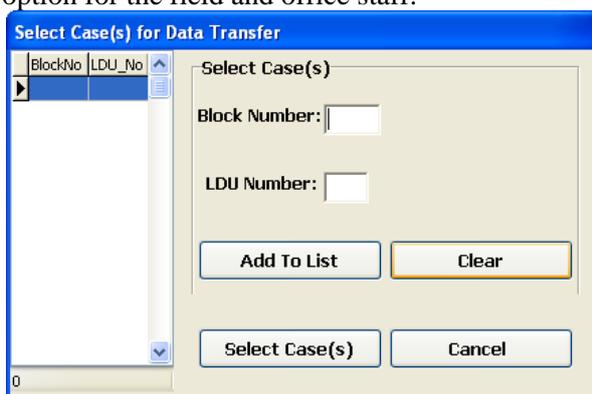


Figure 8: Select Cases to Transfer to the Office

To provide a case status field for the interviewers and management reports we needed to have access to the PIAAC case status. A PIAACStatus.txt file was created when a case was exited that contained the latest status for the last open case. This file had a generic file name and contained only the last active case status. We wanted to find a way to save this information with the rest of the case details. We decided to save this information to the Admin database. If the Case ID already existed in the datamodel the status field was updated. In some circumstances the Admin database would not have a corresponding case, this occurred when the PIAAC selected person was available for interview during the initial contact with the household. In this instance the interviewer would not have had an opportunity to add any details of the case to the Admin database. In this scenario the Case ID i.e. the Block and LDU along with the PIAAC status were written to the Admin database.

3.4.2 Management Applications

The supervisor functionality included the facilities to view their teams' progress and interrogate specific cases. The latest information that had been transmitted to the office was available to the supervisors each morning. Fig. 9 displays the summary information available to the supervisors which is similar to the status information displayed to the interviewers.

Seq No.	Block No.	Ldu No.	Int No.	PersonId	Contact Name	Screener Status	Piac Status	Finalised
1	5047	122	005	50471220087	PERSON X	Complete - 1 sample person selected		No
1	5047	154	005	50471540033	JACK DANIELS	Complete - 1 sample person selected	finished init-q4	Yes
1	5047	180	005	50471800014	ARTHUR GUINNESS	Complete - 1 sample person selected		No
1	5047	183	005	50471830024	JOHN JAMESON	Not at home		No
1	5207	061	013	52070610034	JACK DANIELS	Complete - 1 sample person selected	finished init-q4	Yes

Figure 9: Co-ordinator View of Cases

The supervisors could view a more detailed summary by selecting the case. Different views of the cases were also available and they could be filtered by variables such as the completion status, *Finalised* and *Incomplete*, and Interviewer ID.

Case details for Interview

Reference Data

Coordinator No: 01 Interviewer No: 005
 Block Number: 5047 LDU Number: 099

Respondent details

Name: JACK DANIELS PersID: 50470990028
 Address: Main Street, Cork
 Person: 2 of 5

Case Status

Finalised Screener : Yes
 Piaac Int Status : Finished End
 Disposition Code : Screener Completed

Date of Visit	Time of Visit	Visit Details	Duration
01/08/2011	15:20	1st Visit	
02/08/2011	13:00	2nd Visit	
03/08/2011	12:30	3rd Visit	90

1:10

OK

Figure 10: Case Details

The supervisors could also view and print various reports on the data that were accessed via MS Excel. Outputting the reports in MS Excel allowed the users to filter and sort the data as required. The application used by office based staff had similar views of the data and reports. In addition, they could update the sample file and prepare it for re-distribution to the field staff. Any functionality not available directly in Blaise was accessed from a utility created by our Java team. This utility covers functionality such as encrypting, decrypting, zipping, unzipping and general file management such as copying, moving and deleting files.

3.5 Append Job

A suite of Maniplus and Manipula scripts were used to manage the data when it was received in the office. Each interviewer's lodgement was stored as a folder on the network. A control file with a list of the folder names was used to guide the Maniplus application. The Maniplus script worked through each interviewer's lodgement, first unzipping and decrypting the files. Any Blaise data was appended to the office Blaise Screener and/or Admin databases and the PIAAC zip files were moved to a network folder ready to be imported into the PIAAC Data Management Expert (DME), the data processing application provided by the consortium. Transactions logs were created showing all the records added for each interviewer. The job also prepared four update files for the supervisors based on their own teams' lodgements that were available for them to download each morning.

4 Conclusion

The PIAAC survey has been running since August 2011 and is due to complete in March 2012. Part of the fieldwork finished in February 2012 and some review sessions have already taken place. The initial feedback on the applications is a lot different to the pilot experience. Most of the feedback was very positive with the interviewers reporting high satisfaction levels with the applications. The facility to select cases for transfer was highlighted as a useful feature. The only suggested improvements were to add a Comment field to the Admin datamodel and to revise the wording of the outcome codes, both very minor suggestions. Reviewing the issues logged to the Helpdesk over the data collection period shows most of the incidents reported were due to poor connectivity which is outside the scope of these applications.

Before these developments we had limited experience using Maniplus. Our experience with the software has been very positive and we were very pleased with the functionality we were able to deliver. We are hoping to reuse the applications with minimum amendments for an upcoming new household survey.

5. References

Kuusela, V. & CMS BCLUB Working Group (2010) , Features of Case Management in CAI Systems

Tudor, H (2009), PIAAC – TAO Data Import and Export, General Technical Processes and File Structures, Centre de Recherche Public

6. Appendix

LaunchPiaac Script

```

#####
; Name: LaunchPIAAC
; #####
; History
; #####
; 12.04.11 : Created
; #####
; Description
; #####
; This script is executed when the 'Start PIAAC' button is selected for the
; relevant Person ID in the Home Page of the Blaise 'Programme for International
; Assessment of Adult Competencies'. It starts or resumes the current PIAAC
; virtual machine (VM) with the given person id dependent on the state of the
; interview inside the VM. When the interview is paused or finished the case
; status is written to c:\piaac\piaacstatus.txt file. Case data is dumped to
; c:\piaac\administration\persid'-dump.sql and also exported to
; c:\piaac\output\persid'.zip. Then the VM is shut down and control is
; returned to Blaise.
; #####
; PIAAC Consortium scripts used in this program:
; #####
; HandleCAPI
; GetCaseState
; DumpCase
; ExportResult
; StopVM
; #####
; Parameter: Person ID
; #####
; #####
; Executing HandleCAPI - starts VM and relevant case at relevant point of interview
; HandleCAPI calls StartCAPI/ResumeCAPI/GetCaseState
; Person ID passed as parameter from Blaise
$persid = $CmdLine[1]
$Program = "c:\piaac\handlecapi.exe "
$Arguments = '' & $persid & "
RunWait($Program & $Arguments,"", @SW_HIDE)
; #####
; bringing Virtual Machine to the front
if(WinExists("[Class:VMPlayerFrame]")) Then
    WinActivate("[Class:VMPlayerFrame]")
endif
; #####
; Executing GetCaseState - to frequently check state of relevant Case in the VM
; Person ID passed as parameter from Blaise
; PersID, current activity and state of case in the VM are copied to the clipboard
; NB in c:\piaac\piaacscriptconfig.ini Behaviour must be set to Frequently Update
$Program = "c:\piaac\GetCaseState.exe "
$Arguments = '' & $persid & "
Run($Program & $Arguments,"", @SW_HIDE)
sleep(30000);30 second delay to allow update of checked status
; #####
; Getting relevant state data from clipboard
$bak = ClipGet()
$result = StringRight($bak, 9)
$status = StringStripWS($result, 8)
; #####
; Checking for state of case in the VM, if case 'paused' or 'finished' then
; dump, export case and stop the VM. If case is 'running' keep checking state.
While $Status = "Running" ;use infinite loop since ExitLoop will get called
    $bak = ClipGet()
    $result = StringRight($bak, 8)
    $CheckStatus = StringStripWS($result, 8)
    If $CheckStatus = "Paused" Then ExitLoop
    If $CheckStatus = "nished" Then ExitLoop
WEnd
; #####
; Writing Person ID, current activity and case state from clipboard to text file

```

```

$file = FileOpen("c:\piaac\piaacstatus.txt", 1)
; Check if file opened for writing OK
If $file = -1 Then
    MsgBox(0, "Error", "Unable to open file.")
    Exit
EndIf
#include <File.au3>
_FileWriteToLine("c:\piaac\piaacstatus.txt", 1, $bak, 1)
FileClose($file)
; #####
; Stopping execution of GetCaseState
ProcessClose("getcasestate.exe")
; #####
; Executing DumpCase - dumps a case of the database in the current VM and copies
; it to the host machine
; Person ID passed as parameter from Blaise
$Program = "c:\piaac\dumpcase.exe "
$persid = $CmdLine[1]
$Arguments = ' ' & $persid & "
RunWait($Program & $Arguments)
; #####
; Executing ExportResult - all case data is copied to the host machine for
; further processing
; Person ID passed as parameter from Blaise
$Program = "c:\piaac\ExportResult.exe "
$persid = $CmdLine[1]
$Arguments = ' ' & $persid & "
RunWait($Program & $Arguments)
; #####
; Executing StopVM - powers down the PIAAC VM and stops VMWare Player
RunWait('C:\piaac\stopvm.exe')
; #####
; End LaunchPIAAC
; #####

```

Integrating Biometric Measurements in a Blaise Instrument

Sven Sjödin & Colin Miceli, NatCen Social Research

1 Background

There is increasing interest in including biometric measurements in social surveys to identify associations between socio-economic status and health and mortality. NatCen has been conducting surveys that require collection of biometric data for many years, using both nurses and, for some measures, lay interviewers. These measurements include blood pressure, height/weight, ECG, grip strength, waist/hip circumference, lung function, etc. The traditional approach has been to provide interviewers and nurses with the necessary equipment and train them to measure and enter the data in the appropriate input fields in the survey instruments. This method has some serious weaknesses, such as potential keying errors and the user having to split their attention between the laptop, the respondent and the measuring equipment.

The ideal solution is to use equipment that connects directly to the laptop and to populate the data fields in the instrument with the readings. We have achieved this for one biometric measurement, namely full lung function or spirometry. The lung function module in the survey instrument shells out of the data entry program (DEP) to a third party software that reads input parameters, controls the measurement session and returns the resulting data to be stored in the data file.

The benefits of such a method are particularly compelling for spirometry, as the outcome depends on respondent participation and feedback from the nurse. A forced blowing manoeuvre is required which many people find difficult. Reproducible blows are needed so respondent technique is crucial. The software provides information about the quality of each blow which the nurse uses to improve the respondent's technique and to spur them on to blow in a consistent way. The software can calculate predicted values based on respondent characteristics and use these to spur on the blowing in a consistent way.

2 Finding a New Solution

Our longest running survey with biometric measurements, the Health Survey for England, rotates many of these between survey years. The survey of 2010 was due to include lung function measurements of respondents aged 7 and above. This was last done in the 2001 survey. Given the need to replace our existing stock of spirometers, as the model was no longer manufactured or serviced, we had an incentive to explore the market for a better solution.

Our requirements for the equipment are somewhat different from what medical suppliers usually have to satisfy. The normal environment is a clinic or surgery, which is a fixed, stand-alone setup operated by specialist staff. We needed a portable tool that would ideally integrate with the survey instrument and be used by anyone provided with an achievable level of training.

The market research resulted in a shortlist of two suppliers, who were duly invited for demonstration and discussion. The following topics were raised at the meetings:

- Daily calibration or checking calibration
- How to prepare the machine prior to each use
- How to use the machine
- Infection control and other maintenance measures required
- Data transfer, including
 - using data from the interview (age, sex, height, ethnicity) and
 - saving flow-volume curves for each of the three best curves
- Viewing the blows in real time on the nurse's computer to give instant feedback to nurses and participants
- The ability to add children's incentive software, based on the child's predicted values, to be slightly harder than they are doing but only slightly so.
- The ability to use whatever reference values we choose, e.g. the new predictive equations
- The ability to use age-specific criteria, as the American Thoracic Society criteria are inappropriate for children under 11 years

Both systems satisfied our requirements. We did, however, find one of them superior for most aspects considered and equal for the remainder. For example, it was easier to use and had fewer and lighter components. It also appeared more open to the software changes we required.

3 The Product

The new high tech spirometer has a number of technical advantages over older spirometer models.

- The actual measurements are technically more accurate.
- It enables quality control in the field, as the machine identifies if a reading is of an acceptable quality and, if not, what problem occurred. This enables the nurse to encourage the participant to produce a better blow.
- It minimises the number of blows required by identifying whether readings are of a satisfactory quality and reproducible.
- Quality control of the field staff and data is also possible. The proportion of participants whose session quality is rated A, B, C, D or F can be assessed in the office. Any feedback and/or additional supervision and training can be provided to improve the quality of the data collected.
- It records a wider range of measurements (e.g. mid-expiratory flow variables MEF_{25-75} , MEF_{25} , MEF_{50} , MEF_{75} , FET, etc) that can be archived for future use by researchers.
- The flow-volume curve for each blow is recorded and can be archived, so the researcher can, for example, evaluate whether readings can be used, even if they have been rejected by the machine's algorithm.

- The spirometer can store data for several hundred readings which can be downloaded to the laptop, thereby preventing data transcription errors.

There are only two kinds of equipment for the nurse to carry. The **spirometer** is very light-weight and connects via a USB port. The respondent blows through a disposable tube, called a **spirette**, thus eliminating the risk of spreading infections. Some other models require sterilisation of the equipment.

After a period of close cooperation with the software developer we had a version of the spirometry software that satisfied our needs. The default behaviour of the spirometry software is to expect the operator to enter the patient characteristics, such as age, sex, height, weight and ethnicity. These are used to calculate the predicted lung function values. We already collect these variables in the survey instrument, so the software was amended to read them from a text file. We were also given an export facility to extract the session data from the spirometry database, both into a text file and a relational database.

The original software was far too open in terms of available options, such as buttons and menus. These may be useful for the specialist operator in a clinic, but confusing to the nurse working in a respondent's home, so all of those identified were locked down.

4 Making It Work with Blaise

We fortunately have some previous experience of launching third party software from within the Blaise data entry program (DEP). One example is a survey instrument that uses a VB DLL to collect life history data and populates the database through the Blaise API. This was presented at the 11th IBUC in Annapolis under the title *Retrospective Data Collection*. Another example is from the 14th IBUC in London, *WebCATI (Part of NatCen's Multimode Approach)*, which demonstrates a method of accessing Blaise IS surveys from a CATI setup.

4.1 The Interface

The requirements for the interface between the DEP and the spirometry software are different from the examples above. Each blowing session generates a vast amount of data, which are stored in a relational database. Some of these are used to construct graphs to illustrate the blows. Only a few variables are transferred to the Blaise database, such as the quality of the blows and the data items the nurses used to key in themselves when using the old equipment. The rest are sent back to the office in a separate file.

The interface needed to perform the following tasks:

- Transfer data items from the survey instrument to the spirometry software
- Launch the software
- Interrogate the spirometry database for the results of the session
- Create a return of data file
- Feed back a selection of the data to the DEP

We decided to use Manipula/Maniplus to develop the interface because we know it very well and Blaise version 4.8 allows you to launch a Manipula procedure from the data entry program (see 'Alien' in the on-line Reference Manual). The file that the spirometry software reads is of a proprietary structure, which caused us some unexpected effort to reproduce.

4.2 The Blaise Data Model

The data exchange in the data model is implemented as export and import parameters in the alien procedure that declares the Maniplus setup. It is always important to consider how the call to an alien

procedure is controlled. For example, whether or not to invoke it every time the user answers a key question and how is it affected by the selective checking mechanism.

The Blaise Reference Manual ends the section about alien procedures with a noteworthy caution:

“WARNING!

There are no restrictions for alien procedures and methods. You may develop any method for graphics, complex algorithms, sophisticated sound techniques or other advanced applications. Take notice that DLL and COM alien procedures and methods are executed outside the Blaise system and any malfunction of using them is at your own risk. We strongly advise you to test alien methods and procedures thoroughly before using them.”

We took the decision to block the call to the alien procedure as soon as control is returned to the data entry program. It would otherwise keep being triggered by the checking mechanism. This is quite easily done by routing it on a field, the value of which is changed immediately after the procedure call. The way we implemented the routing still allows the nurse to backtrack in the questionnaire and access the alien procedure again. It would, with hindsight, have been better to block this as well. It was, however, routed around in the version we used for the in-office coding and editing.

5 In Production

The success of a survey depends on much more than the quality of the technical solutions. Introducing new protocols and methods requires planning, testing and training. It all had to be put in place within less than six months, from the decision in the summer of 2009 to the start of fieldwork in January 2010 with a dress rehearsal in September. The contract with the supplier included training sessions that enabled us to train our own trainers. We purchased 100 units of software and equipment and trained a corresponding number of nurses. The training sessions could be spread out over the survey year as not all nurses work in all survey months.

5.1 Installing the New Software on Nurse Laptops

Our interviewers and nurses at the time only had dial-up connections for exchanging files with the office, so the new software couldn't be installed remotely in advance of the training. Instead, their laptops were replaced as they arrived at the training session. The replacement laptops had the software pre-installed. All our field laptops are set up through imaging, so it was added to the nurse laptop image. It also meant that we didn't have to go through a lengthy install procedure at the training venue.

5.2 Quality Control

Since the new software feeds back the quality of the blows both to the nurse and in the data file, we had the opportunity to improve the quality of the lung function measurements. We could put systems in place to monitor the nurses' performance to be able to take remedial action if required. The quality measurement was also checked in the in-office coding and editing. As mentioned above, the algorithm used by the software doesn't always reach the correct conclusion. Each case was looked at to verify the given quality value.

We were also checking the calibration data to make sure that the nurses ran the calibration routine at the start of every working day.

5.3 Reconciling the Data

As always when data are collected through different means, we needed to set up a system to reconcile the data sources and make sure they match across the data files. There was less of a risk in this case, where the case ID is passed to the spirometry database. However, the nurses may make mistakes in selecting the wrong person in the household or the wrong case ID for the address.

6 Further Use

The lung function measurement will not be included again in the Health Survey for England (now renamed to Health & Social Care) for some years to come. However, our main continuous survey, Understanding Society, introduced nurse visits including spirometry in 2010. This survey will continue until the middle of 2012. One complication is that this survey covers both England/Wales and Scotland. The Scottish nurses were never converted to the new spirometry method, so the instrument has to route the lung function module on the country in which the interview takes place.

Multiple Overlapping Waves - Challenges in Supporting Blaise Instruments Simultaneously for Four Waves of Data Collections

Lilia Filippenko, Mai Wickelgren, Venkat Yetukuri, Sridevi Sattaluri, and Preethi Jayaram, RTI International

1. Introduction

At RTI International we conduct a number of longitudinal studies where data for subsequent waves is collected after completing a previous one. In contrast, the Work, Family and Health Study (WFHS) must conduct several waves simultaneously. It is a longitudinal study collecting data from individuals in the workplace and in the home at baseline and at 6, 12, and 18 months post-baseline using both in-person and telephone computer-assisted interviews, basic health measures (height, weight, and blood pressure), blood collection, saliva collection, and Actigraphy (measuring sleep quality). Spouses or partners and selected children (between the ages of 9 and 17) of participating employees were recruited into the study during the baseline period. Interviews in the workplace are conducted on a rolling schedule; at one point 20 Blaise interviews were in production in the field for all four waves.

To deal with so many instruments, we developed for WFHS an application in .Net that uses the Blaise Component Pack (BCP) to maintain SQL Server tables for a subset of fields from the Blaise databases. The application utilizes SQL Server stored procedures to create up-to-date preload information for follow-up Blaise interviews, set statuses in a Control System for monitoring purposes, and populate reportable statuses of blood collection, saliva collection, and Actigraphy. The clients are able to view reports, download encrypted files, and to record blood spot counts, saliva receipts and tracking statuses through a secure web portal.

In addition to standard reports, about one hundred custom reports were developed to monitor data collection. A Field Management System and Control System were modified to allow reports to be viewed by wave.

The paper will describe how new and existing applications developed at RTI International help make data collection for WFHS efficient and accurate.

2 Overview of Work, Family and Health Study

The WFHS is composed of interdisciplinary research teams from the University of Minnesota, Penn State University, Harvard University, Portland State University, Michigan State University, Kaiser Permanente's Center for Health Research, and RTI International. The WFHS is studying the impact of workplace programs and policies and how they affect employees' work, health and family members' health and well-being.

2.1 Computer-Assisted Interviews

WFHS data collection includes survey and biomarker data collection led by RTI International, and Harvard University, a telephone-based Daily Diary data collection of participating families led by Penn State University, and qualitative interviews led by the University of Minnesota, Portland State University, Michigan State University.

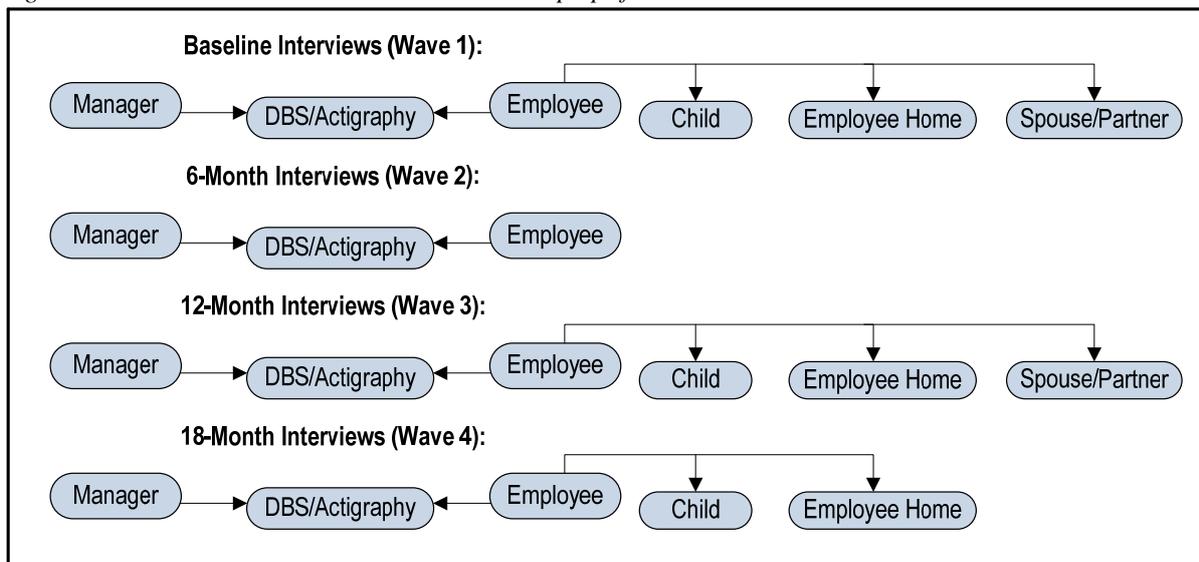
Data is collected from employees and managers within two industries and 56 sites about work experiences (including opinions about work, amount and type of work, job demands), health

(including physical health symptoms, medical care, emotional well-being, and health behaviors), and family structure and relationships.

For four waves of data collection RTI International developed 20 Blaise instruments to be conducted by Field Interviewers on laptops:

- Computer Assisted Personal Interviewing (CAPI) worksite interviews with employees and managers
- DBS (dried blood spots)/Actigraphy (collecting data via an actigraphy watch) for employees and managers
- CAPI home interviews with employee and eligible child with Daily Diary study recruitment and enrollment
- Spouse/Partner telephone interview (administered by the field interviewer)

Figure 1: Blaise Interviews on Field Interviewer Laptops for 4 Simultaneous Waves



In addition to the described above Blaise instruments, two Blaise instruments were developed to conduct interviews with a subset of employee and managers:

- Telephone Verification interview to check the quality of FI work during any in-person interviews.
- Telephone Attriter interview with employees and managers who leave the workplace as of the 6, 12, or 18 month follow-up visit.

These telephone interviews are conducted at the RTI Call Center Services (RTI-CCS) facility.

2.2 WFHS Systems

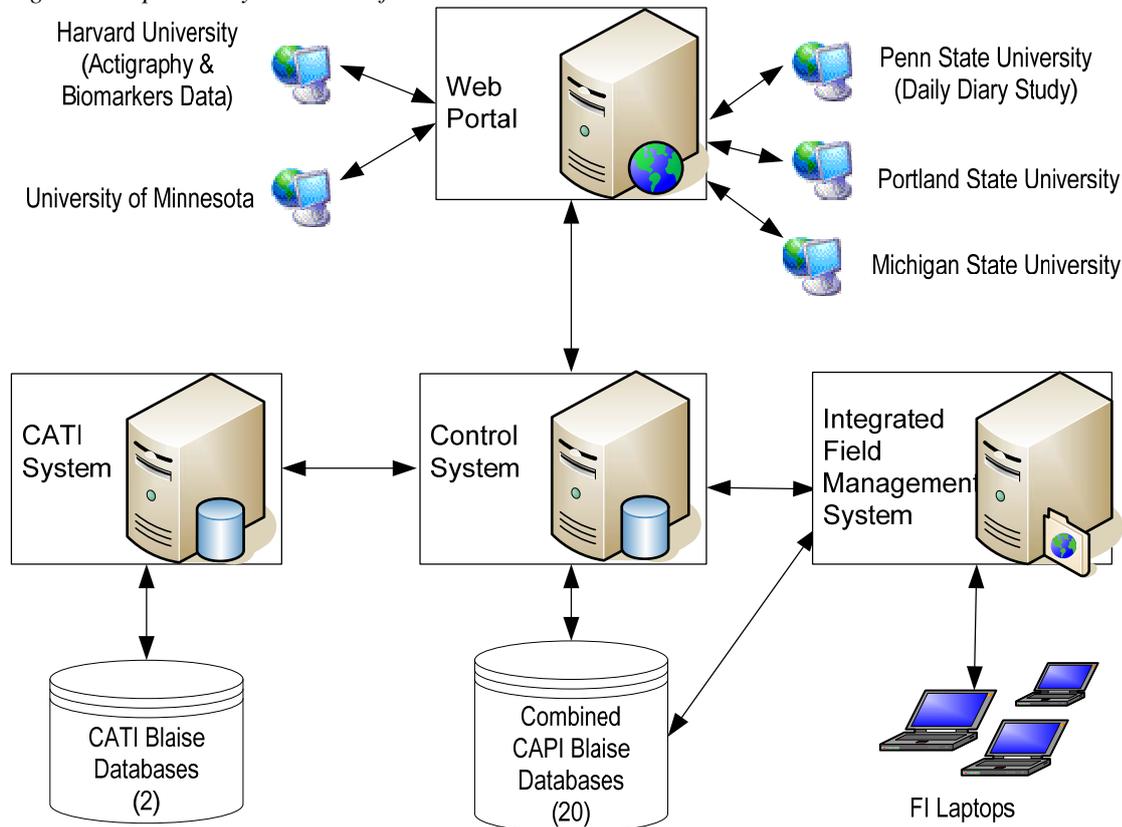
During the development stage of the study, the following standard RTI systems were adopted to satisfy the requirements of WFHS:

- Case Management System (CMS) - used on Field Interviewer (FI) laptops to allow them to update case status, enter comments, launch Blaise instruments, and synchronize the status of cases with a centralized SQL server database.
- Integrated Field Management System (IFMS) - a web application used to assign and transfer cases between FI laptops and RTI.
- General Survey Control System (GSCS) - a web application on RTI's internal secure network used by authorized staff to monitor the flow of data from the start of data collection through the creation of data files for analysis and delivery.

- Computer Assisted Telephone Interviewing (CATI) CMS - used on RTI's secure network to conduct telephone interviews.

To allow all researchers participating in WFHS to monitor data collection and receive up-to-date information from the CAPI interviews, a secure web portal was set up by RTI. During nightly processing at RTI, encrypted files are posted for the Daily Diary study and the Actigraphy study along with status reports about CAPI interviews in the field.

Figure 2: Top Level Systems Used for WFHS



Due to the complexity of the study and the potential for changes even after the start of data collection, our goal was to have all systems easily configurable so updates will be applied without interruption of data collection and in a timely manner. Significant changes were made to a GSCS and IFMS which are described below.

3 Application ImportBlaise2SQL

To accomplish our goals to simplify maintenance of so many Blaise instruments and to be able to post up-to-date collected data on the web portal, we decided to create a table for each Blaise instrument in SQL Server with a subset of fields. We considered the possibility of using the Blaise Datalink Component to create and update these tables. Although it looked like an easy solution to our needs, we soon realized that maintenance of a .BOI file for each Blaise instrument could be time consuming and would require extensive support from an experienced Blaise programmer because the list of fields to pass from the Blaise database to SQL table was not stable. We decided to develop a .Net application that will use a special MS SQL table to create all needed MS SQL tables on the fly and populate them with the most recent data from the Blaise databases.

Development of this ImportBlaise2SQL application began with the creation of a detailed list of Blaise databases and variables which are required for tracking, event coding, preloads for follow-up instruments, Data Entry in Control System and web portal. These variables are stored in an SQL table

(Tbl_Config) to allow the application to import data from these variables into the Blaise databases through the nightly job.

Each record in the Tbl_Config table consists of:

- Original Blaise database name
- Fully qualified variable name along with its size and type in the Blaise database
- Variable column name and its expected size and type in the target SQL table
- Status of the variable to process: “Active” or “Not Active”

The program uses the Blaise Component Pack (BCP) to open the Blaise database to access data collected in the Blaise interview. It then reads the variables given in the configuration table from the appropriate Blaise database and exports them into temporary tables in a SQL Server database as shown in the example below.

Figure 3: Example of Creating Temporary Table in SQL Server

```
//Open Blaise Database
BlAPI3A.Database db = dbMgr.OpenDatabase(DBPath);
db.AccessMode = BlAccessMode.blamShared;
db.Connected = true;
...
//dynamically create columns for temporary table
SQLCols = SQLCols + "[" + (string)dtrCat2["ColName"] + "],";
SQLVars = SQLVars + "@" + (string)dtrCat2["ColName"].ToString().Replace(".", "_") + ",";
command.Parameters.AddWithValue("@ " + (string)dtrCat2["ColName"].ToString().Replace(".",
"_"), FormatData(db.get_Field((string)dtrCat2["VarName"])));
...
command.CommandType = CommandType.Text;
command.CommandText = "Insert into TEMP_" + DBName + "(" + SQLCols + ") values " + "(" +
SQLVars + ")";
command.ExecuteNonQuery();
...
```

The variables and databases from Blaise are split into temporary tables, and are populated each night by a scheduled job.

After the completion of data importing steps, this program invokes an SQL stored procedure which reads the variables from the temporary tables to identify the newly completed or updated cases and computes their corresponding status codes to update the Control System. Based on these variables, the stored procedure also determines the records of DBS, Daily Diary and Saliva, and Actigraphy data which are then made available on a secure web portal for client viewing and data entry purposes.

For the baseline of WFHS we created and processed six temporary tables with about one hundred variables. When new CAPI instruments were added for the next wave, adding new tables for them in SQL Server was as simple as inserting rows into the Tbl_Config. During the period when all CAPI and CATI interviews were in progress for all four waves, twenty-two tables were updated every night, and data from more than eight hundred variables was read from the Blaise databases and written to SQL Server tables. When a wave of data collection was completed, the status in Tbl_Config was changed to “Not Active” for its corresponding Blaise variables, and the tables would then not be updated by the nightly process. The data from completed interviews is, however, still available to be used for follow-up interviews.

4 Modifications to IFMS

IFMS is used for almost all field studies conducted by RTI International and is a web application responsible for electronic assignment and transfer of cases to field staff, standard case status reports, data transmission, field monitoring, interviewer production, and laptop case management. IFMS is set up for each study and standard reports usually cover mostly all aspects of data collection.

4.1 Custom Reports

The participants in the study in question represent two disparate workplace cohorts – a white-collar, high-tech industry and a long-term care industry. Approximately 600 workplace supervisors, 3,000 employees, 1,200 spouses or partners and 1,200 children (between the ages of 9 and 17) of participating employees are recruited into the study during the baseline period. Data is collected from distinct work units on a rolling schedule across time per industry. Due to the complexity of the study, we developed different types of customized reports to oversee the data collection efforts. Below are just few examples:

- Interview Status Reports by Industry/Worksite
- Basic Health Measures Reports
- Consent Reports by Industry
- DBS/Actigraphy Reports by Industry/ Worksite

In addition, in order to monitor data collection per industry and site in each wave, a dashboard report was developed in .Net which offered a summarized simple overview of the interview counts associated with all instruments in each wave grouped by industry and worksite.

Figure 4: Partial Dashboard Report with Overall Numbers for 4 Waves and Each Site

02/07/2012	Overall		Site 1.0		Site 2.0		Site 3.0		Site 4.0	
	#	%	#	%	#	%	#	%	#	%
Manager	876		14		6		19		34	
Pending refusal	2	0.23%	0	0.00%	0	0.00%	0	0.00%	0	0.00%
Pending other	19		0		0		0		0	
Not eligible	10		0		0		2		0	
Final refusal	33	3.81%	0	0.00%	0	0.00%	1	5.88%	2	5.88%
Final other	37		0		0		0		1	
Completes	775	89.49%	14	100.00%	6	100.00%	16	94.12%	31	91.18%
Employee	3534		82		59		93		134	
Pending refusal	3	0.08%	0	0.00%	0	0.00%	0	0.00%	0	0.00%
Pending other	80		0		0		0		0	
Not eligible	79		0		2		1		6	
Final refusal	338	9.78%	10	12.20%	11	19.30%	23	25.00%	20	15.63%
Final other	204		3		1		4		12	
Completes	2830	81.91%	69	84.15%	45	78.95%	65	70.65%	96	75.00%

4.2 Changes to IFMS website

There are more than hundred reports available in IFMS for WFHS now. To make it possible to easily find a report for review, IFMS website was modified to dynamically create a list of reports for a selected wave only.

To help Field Supervisors to monitor all the paperwork in the field, a new web page was designed in the IFMS website to display all documents collected during the CAPI interview. The list includes consent forms, Daily Diary receipt forms, etc., with up to 15 forms per respondent. The number of forms and the type of forms are specific to a case and is based on the responses to the consent questions in the various Blaise instruments. The SQL tables which are populated with Blaise data using the ImportBlaise2SQL application simplified the task of generating the list of forms that are expected from a respondent. The web page allowed the Field Supervisor to select or type in a Case ID and get the list of forms associated with the case. The Field Supervisor could then verify the case folder and ensure that all the forms pertaining to the case had been received.

Figure 5: List of Forms Expected for the Case Collected in Three Blaise Instruments

Case ID	Form Name	Form Type	Instrument	Blaise Variable
312MA99E	PINK - D	DBS	Aw12m	DB_12.DB_3
312MA99E	PURPLE - E	Actigraphy	Aw12m	ACT_12.ACT_1
312MA99E	GREEN - C	Employee - HIPAA Waiver	Em12m	LE_12.LE_HIPCK
312MA99E	YELLOW - A	Employee - Workplace CAPI	Em12m	RD_12.RD_CNST1
312MA99E	N/A - Verbal	Spouse	Sp12m	SD_12.SD_CNST4

5 Modifications to Control System

The Control System is where all data collection activity is centrally tracked and monitored. The RTI General Survey Control System provides the database structure and a website that manages the events for:

- Completion status for each case in all instruments
- Incentive Mailings
- Interview Verification

While the base structure was sufficient to track the events, the system had to be modified to accomplish specific requirements for WFHS.

5.1 Custom Reports

The reporting system was enhanced to provide 18 custom reports to accommodate the study needs of monitoring the biomarkers, Daily Diary and Saliva eligibility status. Each report is provided by wave and the data could be exported out as an Excel spreadsheet for easier analysis. The following screen displays one such report developed to monitor the quality of the DBS collected for each interviewer.

Figure 6: DBS Quality Report by Field Interviewer

DBS Quality Report																				
Export To Excel																				
Wave: Wave 1																				
FIName	Completed Interviews		Refusals		Collected		Collected Not Shipped		Obtained A1C		Total 1 Blood Spot		Total 2 Blood Spots		Total 3 Blood Spots		Total 4 Blood Spots		All 5 Blood Spots	
	n	%	n	%	n	%	n	%	n	%	n	%	n	%	n	%	n	%	n	%
FI NAME A	83	30.29	2	2.41	92	110.84	2	2.17	75	81.52	17	18.48	19	20.65	17	18.48	8	8.70	10	10.87
FI NAME B	23	26.44	3	13.04	17	73.91			19	111.76	7	41.10	7	41.10	1	5.00			2	11.76

The Control System website is an internal website only accessible by RTI staff, but the reports from the Control System needed to be ported to an external web portal so all researchers participating in WFHS could review them. We developed a .Net program that would call the report URL from the Control System and save the results as an html file to be posted on the external web portal. This program is scheduled to copy the Control System reports to the secure web portal every night. As a result, there is no need to copy over the data just for reporting purposes.

5.2 Locator Database

Respondents who left their companies during the course of the study are referred to as Attriters. Attriters need to be traced so that the data can be collected for the subsequent wave. In order to trace them, we extract the contact information collected in the Blaise instruments into SQL tables using the ImportBlaise2SQL application and then move necessary data to the locator tables in the Control System. This latest contact information is exported from the Control System and passed over to the RTI Tracing Operations System (TOPS). The confirmed located information is then brought back into the Control System, so that mail-outs can be sent to the correct address and Attriter Blaise interview could be conducted over the phone.

Figure 7: Respondent's Information from Different Sources

Case Locator Data

ParticipantID: 122BE99E Add Locator to selected record

Tracing Code: 1721 - Located, confirmed phone number only

	Number	Source	First Name	Last Name	Priority	Relation	Date Updated
Select	1	CAPI-Blaise	John	Doe	0	Subject	10/22/2010 10:12:04 AM
Select	2	CAPI-Blaise	Jean	Doe	0	Mother	10/22/2010 10:12:05 AM

Phone Numbers: Add Phone

	Locator Number	PhoneID	Phone Number	Type	Priority	Source	Date Updated
Edit Delete	1	1	5558585555	H	0	CAPI-Blaise	10/22/2010 10:12:05 AM
Edit Delete	1	2	555555451	C	0	CAPI-Blaise	10/22/2010 10:12:05 AM
Edit Delete	2	1	5553525555	H	0	CAPI-Blaise	10/22/2010 10:12:05 AM

Locator detail:

Locator Number: 1

Source: CAPI-Blaise

Priority: 0

First Name: John

Middle Name:

Last Name: Doe

Relationship: Subject

Email: 55555@aol.c

Age: 26

Gender: 2

SSN: 1

Address: 555 Main St

City: AAA
Zip: 55555

MailingAddress:

TOPS Comments:
 TOPS 11/2/10. SPK W/ SUB WHO CNFRMD 2451# + JUST STARTED
 A NEW JOB & LEAVES FOR WORK @ 2PM + GAVE HER 877# &

When the CATI Attriter interview is completed, new and updated information about the respondent is used for the next Attriter interview so that interview data for all four waves will be available for researchers at the end of the data collection.

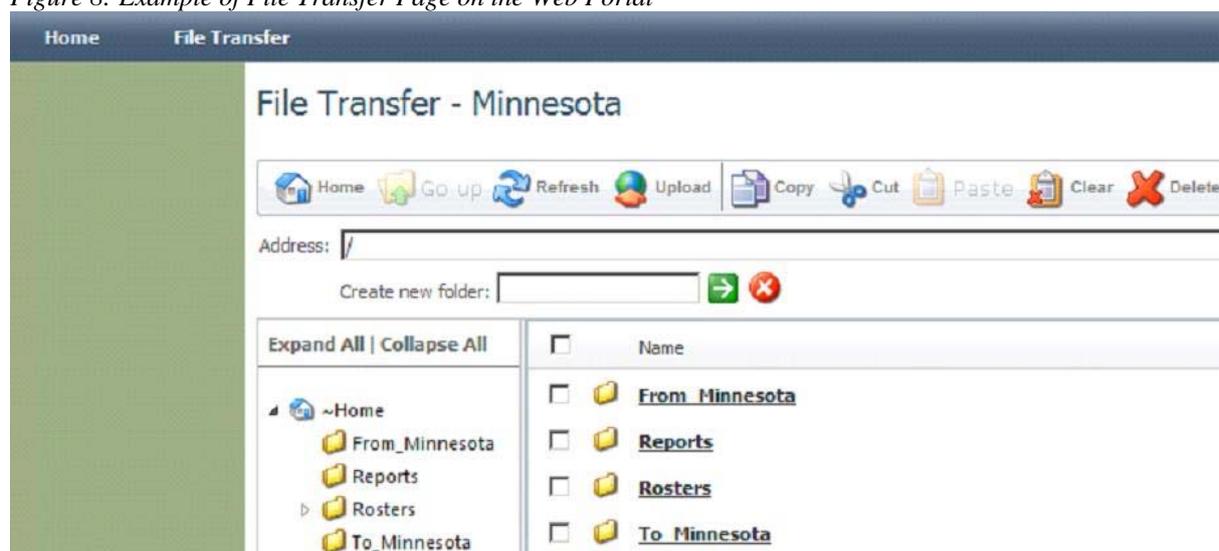
6 Web Portal

The WFHS web portal developed by RTI provides an environment that allows all participating researchers to do file and data exchanges. In addition, Harvard and Penn State have data entry web pages to record results of DBS, Actigraphy, Saliva, and Daily Diary data collections. The portal also contains various status reports to inform researchers about the status and outcomes from the CAPI interviews.

6.1 Web Portal User Access

Because the participating parties have different needs, the web portal requires user authentication for data security purposes. Access to different web portal features is controlled through roles assigned to menu items. Each menu item has certain roles assigned to it, such as "Harvard user" or "Minnesota user". When a user from University of Minnesota logs in, they see the menu specific to their user role and permissions. Thus, the role-based menu can be used to hide information based on the person's role in the study. Only RTI International, being the Data Coordinating Center for this study, has links between participants' answers in CAPI interviews and all other components of data collections. Even reports are specific to a user and only top level reports are available to everybody.

Figure 8: Example of File Transfer Page on the Web Portal



6.2 Web Portal Data Entry

As described above, when Blaise data is extracted for completed CAPI interviews into Control System tables by the ImportBlaise2SQL application, SQL server tables with fields for DBS, Actigraphy, and Daily Diary are updated and posted to an external SQL server database for processing. A “Harvard user” or “Penn user” enters laboratory results and status of their data collection into the web portal.

Another scheduled job picks up updates from the external web portal and uploads them back into Control System. At that time new statuses are applied and reports show up-to-date statuses from all sources.

7 Support and Development of Blaise Instruments

In previous sections we described some of the new development that was required for WFHS. We also used many existing tools and applications that helped us collect data efficiently and successfully.

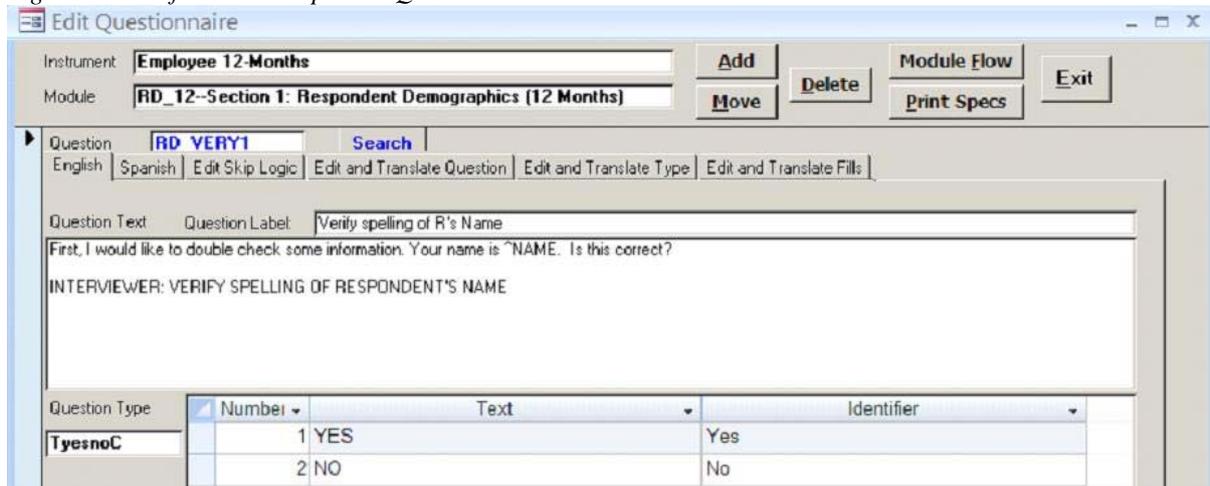
7.1 Use of Questionnaire Specification Database (QSD) to Develop Instruments

Given a very tight schedule to develop Blaise instruments, especially for the follow-up interviews that happened every 6 months, we made extensive use of our Questionnaire Specification Database (QSD) for developing all CAPI/CATI Blaise instruments. QSD consists of tiers for the user interface and business rules, and a back-end Microsoft Access database. It helps spec writers and Blaise developers streamline the process of creating Blaise instruments. A key feature of QSD is its ability to track all changes made to the specifications. This is very helpful for programmers and translators to check for updates after revisions have been made by the spec writer.

Since WFHS is a large longitudinal study utilizing 20 CAPI and 2 CATI Blaise instruments among four waves of data collection, it was and still is a challenging task to maintain and manage all facets of the instruments development process. Although many questions in follow-up interviews are repeated, text for them could be changed and new questions could be added. So for all follow-up interviews new specs were originally to be created by spec writers and then implemented by the Blaise programmers. Instead, to simplify the specification process and to quickly prepare new Blaise instruments, we used QSD to add new modules by duplicating existing modules from previous waves and to allow spec writers to enter modifications directly in QSD. Every time a change was made, information was recorded about who made the change, the date and time, and details outlining the modifications made. After spec writers were finished with the changes, programmers used QSD to

generate follow-up Blaise instruments and produce specification documents that are complete with question text, response options, fills, and routing comments. Since specifications and Blaise code are both generated from within QSD, both documentation and Blaise code are synchronized and linked throughout the development process.

Figure 9: Modifications to Specs in QSD



7.2 Updates of Blaise Instruments on Field Interviewer Laptops

When data collection for WFHS started in 2009, only 6 baseline Blaise instruments were on FI laptops. Since then all updates to existing instruments and additions of later instruments were done remotely using IFMS without interruption to data collection.

During this time, the economic situation of companies participating in the study was changing and new questions needed to be added to the instruments to reflect those changes. But for modifications that alter the structure of a Blaise data model, it is difficult to do: any existing production Blaise data files must be upgraded along with the new data model files. Although the Manipula setup for this procedure is very simple, many steps need to be taken to avoid loss of data: creation of backup folders, copying of a number of files, the preparation of Manipula setups for the upgrade and so forth. It can be an extremely complex process, especially when instruments reside on multiple computers, possibly with different versions.

This complicated task is simplified through the use of a special program, Upgrade Blaise Instrument (UBI), developed some time ago at RTI International. UBI utilizes the version number which exists for each Blaise project, data model, and data file. When a new version of the Blaise instrument is ready for the field, UBI is used to upgrade the combined CAPI database that resides at RTI and to create a package of files that are required to upgrade FI laptops. The package then is sent via transmission and executed on a laptop without FI interaction. Results of the upgrade are automatically sent back to RTI so we can monitor the status of the upgrade and the version number of the instrument. On average, every instrument in the field was upgraded twice.

8 Conclusion

Data collection for WFHS has continued successfully for more than two years, and two waves are already completed. The third and fourth are scheduled to be completed by the end of this year.

In the process of supporting so many Blaise instruments at once, the following are some of the challenges we encountered and overcame:

- Data collected in CAPI should be available to participating researchers as soon as possible on the RTI web portal.

- Reports should be split up by key variables (industry/site/etc) and on the contrary combined for all waves
- Changes to the instruments can require upgrades to Blaise data files that should very quickly be applied to FI laptops in production
- Data from different components of data collection should be merged for a respondent without disclosure of the respondent's identity
- All respondents who completed baseline interview should be tracked and data for subsequent waves should be collected - even if a respondent no longer worked at a participating company

Of course there were and still are other challenges inherent in supporting Blaise instruments for this study that are not part of this paper. The applications and systems described here addressed the requirements of the study. They help make data collection for WFHS efficient and will provide all the participating parties with accurate data for their research.

9 References

Bray, J., Almeida, D., Buxton, O., Dearing, J., Kelly, E., King, R. "An integrative, multi-level, and multi-disciplinary research approach to challenges of work, family, and health." 2012

Lilia Filippenko, Valentina Grouverman, Joseph Nofziger, "Questionnaire Specification Database for Blaise Surveys", 2007, Proceedings of 11th International Blaise Users Conference

Lilia Filippenko, Joseph Nofziger, and Gilbert Rodriguez, "Automatic Upgrade of Production Blaise Instruments", 2003, Proceedings of 8th International Blaise Users Conference

10 Acknowledgement

This research was conducted as part of the Work, Family and Health Network (www.WorkFamilyHealthNetwork.org), which is funded by a cooperative agreement through the National Institutes of Health and the Centers for Disease Control and Prevention: Eunice Kennedy Shriver National Institute of Child Health and Human Development (Grant # U01HD051217, U01HD051218, U01HD051256, U01HD051276), National Institute on Aging (Grant # U01AG027669), Office of Behavioral and Science Sciences Research, and National Institute for Occupational Safety and Health (Grant # U01OH008788, U01HD059773). Grants from the William T. Grant Foundation, Alfred P Sloan Foundation, and the Administration for Children and Families have provided additional funding. The contents of this publication are solely the responsibility of the authors and do not necessarily represent the official views of these institutes and offices. Special acknowledgement goes to Extramural Staff Science Collaborator, Rosalind Berkowitz King, Ph.D. and Lynne Casper, Ph.D. for design of the original Workplace, Family, Health and Well-Being Network Initiative.

The Labour Force Survey, a Mixed Mode Household Survey

Martha Kevers, April 2012, Design Department, Data Collection Division, Statistics Netherlands

1 Introduction

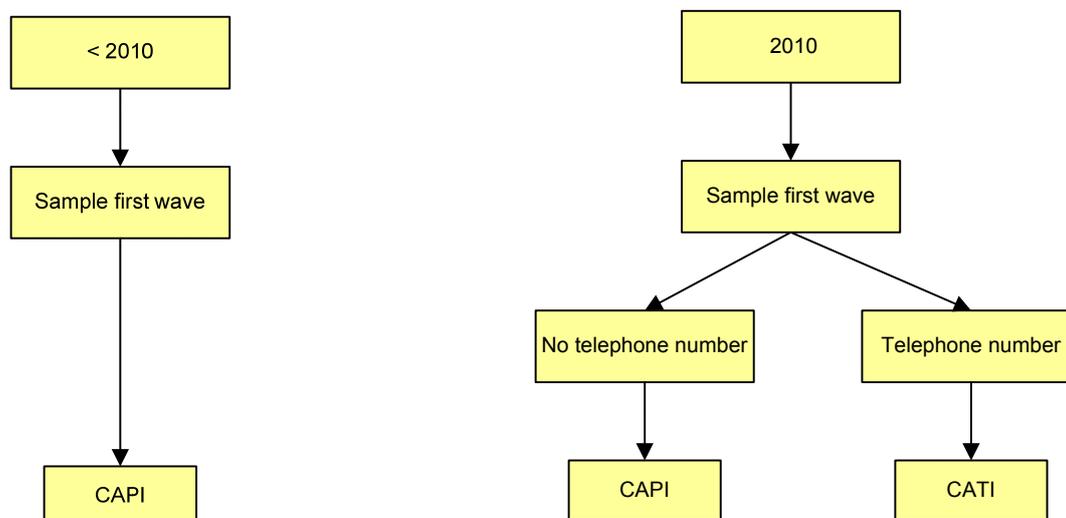
In the last few years, Statistics Netherlands has put a lot of effort in introducing web and mixed-mode data collection strategies for the social LFS household survey (*Janssen 2010. 'Web data collection for Household Surveys at Statistics Netherlands'*). Major drivers for these efforts are the rapid development of the Internet as a survey mode and the decreasing availability of telephone numbers. Also, there is a strong incentive to reduce the costs of data collection at Statistics Netherlands. In addition to this, users require more flexibility in implementing new surveys and also timeliness and coherence of statistics are important issues. To react to these developments, Statistics Netherlands initiated a programme to redesign its social surveys. The programme's main objectives relate to efficiency, quality and flexibility. These goals are met by implementing several changes in the design of the social surveys (*Cuppen, M., P. van der Laan, W. van Nunspeet 2011. 'Re-engineering Dutch social surveys: From single-purpose surveys to an integrated design'*).

One of the key elements of the new model for surveys developed in this programme is the introduction of a sequential mixed-mode data collection strategy for all social surveys. In this strategy multiple contact attempts are implemented using different modes in a prespecified order. Sequential mixed-mode strategies are particularly cost effective since they can start with the self-administered modes that have low administration costs and use interviewer-administered modes to re-approach the remaining nonrespondents. At Statistics Netherlands, in a sequential mixed-mode strategy respondents are first contacted with an advance letter inviting them to fill in a questionnaire on the Internet. For some surveys, respondents are offered the possibility to request a paper questionnaire. If respondents do not respond by using the Internet or paper questionnaire, they are either approached by telephone when a telephone number is available or else approached for a face-to-face interview. Approximately 2/3 of the respondents who did not respond by internet are approached by telephone. The remaining respondents will be approached face-to-face.

For the Labour Force Survey the introduction of the sequential mixed-mode design consisted of two steps, because in order to introduce a web questionnaire for the Labour Force Survey a number of issues had to be solved. For instance, while only one person per household is interviewed for the Health Survey, the Labour Force Survey is a household survey in which all household members (of 15 years of age or older) are to be interviewed. Interviewing large households with a web questionnaire or even by telephone raises a number of practical issues: the interview should not get too lengthy and (in cawi) it is difficult to ensure that every household member fills out the questionnaire, which could result in partial nonresponse. Also, the Dutch Labour Force Survey questionnaire has automatic coding modules integrated for educational attainment, economic activity and occupation. These modules are complex and have to be operated by interviewers who have been trained to do so. Respondents cannot be expected to classify themselves with these modules without the help of a trained and experienced interviewer. So, these modules needed to be simplified and adjusted to be available for self-administered questionnaires, which is not a trivial task and which might result in less detailed information.

Therefore, in 2010 a mixed-mode design was introduced in the first wave of the LFS using only capi and cati datacollection. Addresses with a listed telephone number were from then on contacted by telephone instead of face-to-face. In the old design, all households were contacted face-to-face for the first wave of the LFS. The introduction of cati in the first wave of the Labour Force Survey provided useful information on interviewing households by other modes than capi. Main conclusions so far are that attention should be paid to interview durations of households consisting of more than three people of 15 years of age or older. In addition, households with a known mobile phone number turned out to

be hard to reach when approached by telephone, mainly because of a relatively high number of invalid or otherwise unusable phone numbers. Therefore these households are now approached by capi instead of cati.



The cawi mode is introduced in the LFS in 2012. The new design is run in parallel to the current design for six months to be able to compare both designs and measure discontinuities directly. Before introducing the cawi mode into the Labour Force Survey design, a pilot was conducted in 2011 to provide input on how to introduce cawi in the LFS. This paper describes the goals and design of this pilot, as well as the results of the pilot and the conclusions that were subsequently incorporated in the design of the mixed-mode LFS in 2012.

2 Pilot design

The main objectives of this pilot are:

- finding out which response rates can be expected from the first and second wave of the LFS;
- finding out which response rates can be expected for specific sections of the population, especially response rates for different household sizes and for households including one or more people defined as ethnic minorities from non-western countries;
- finding out to what extent the number of partial responses (incomplete households) changes by introducing web data collection;
- finding out to what extent the number of proxy reports changes by introducing web data collection.

Sample units are approached largely in accordance with the customary approach strategy used for mixed-mode surveys involving individuals and households. The strategy for the first wave is composed of the following steps.

1. All sample units receive a letter containing the Internet address where the web questionnaire can be found and a personal login to gain access to the web questionnaire. In the letter a household residing in the address the letter was sent to, is requested to complete the questionnaire through the Internet. All household members have to use the same login to gain access to the questionnaire. Household members do not receive an individual login. The letter is sent on Thursday to ensure sample units receive it just before the weekend.
2. Reminders are sent to sample units that have not yet filled in the questionnaire. They are sent one week and two weeks after the advance letter. Again both letters are sent on Thursday.
3. One week after the second reminder the web questionnaire is closed. Nonrespondents to the web questionnaire are now approached by telephone or face-to-face. This approach is not preceded by a specific advance letter. Though, in the letters from steps 1 and 2 there is a notification that if a household does not participate in the survey via the Internet they will be requested to participate

in a telephone or face-to-face interview. In the pilot a slightly different strategy than the customary strategy is chosen. Only households for which a landline telephone number is available and households consisting of no more than two people of 15 years of age or older are approached by telephone. This means that households for which no telephone number or only a mobile phone number is available and households consisting of three people or more of 15 years of age or older are approached face-to-face.

4. At the end of the questionnaire of the first wave respondents are asked if they are willing to participate in the second wave of the LFS. Respondents that are willing to do so are asked for their telephone number.

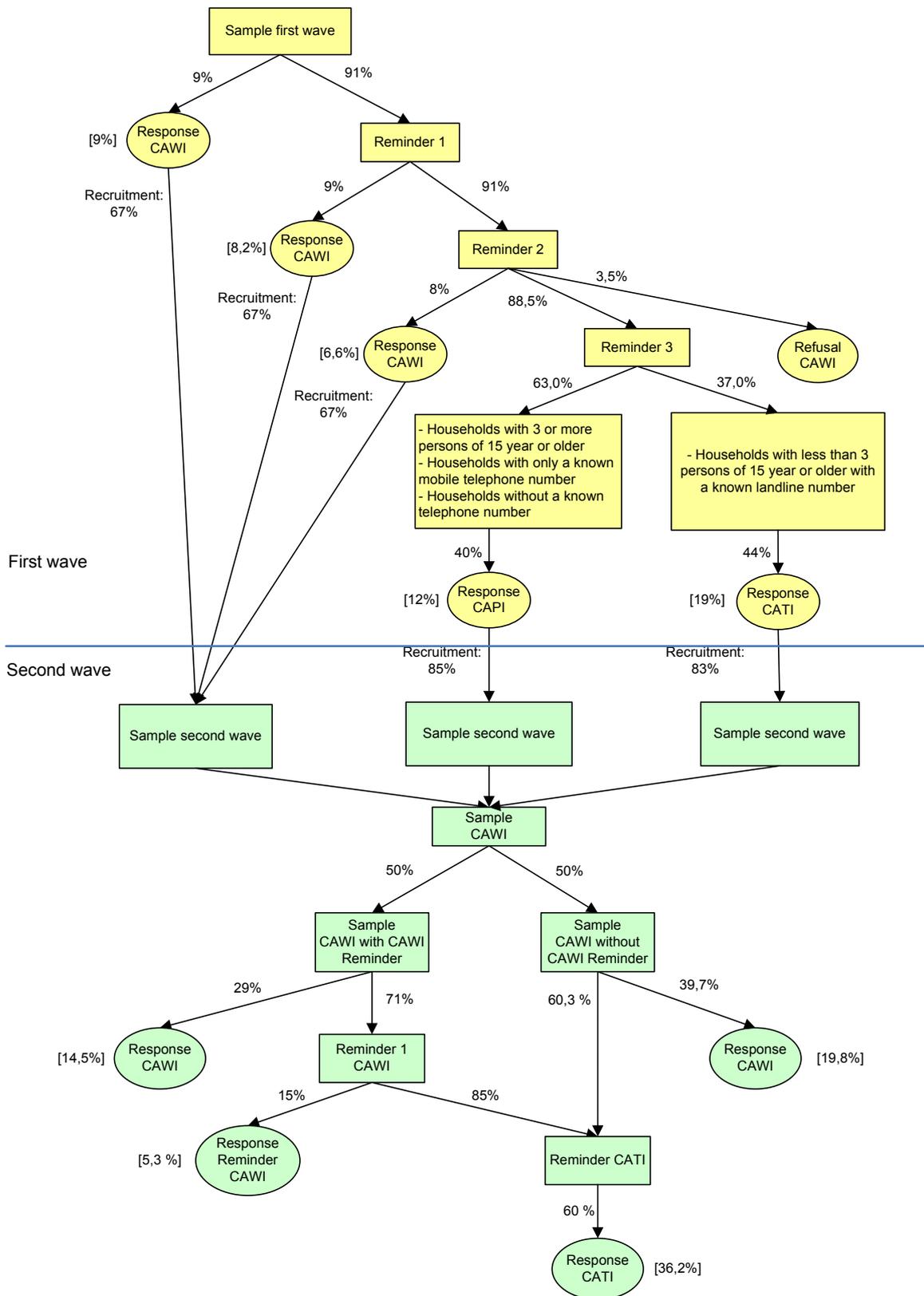
The strategy for the second wave is composed of the following steps.

1. All respondents of the first wave, regardless in which mode they participated, who have agreed to participate in the second wave, are approached by letter requesting them to complete the questionnaire of the second wave via the web. Again this letter is sent on a Thursday taking into account that respondents are approached close to three months after responding in the first wave.
2. One week after the first letter only half of the sample units that have not yet filled in the questionnaire receive a letter reminding them to do so. The other half of the nonrespondents is not reminded. This letter is also sent on a Thursday. Sample units only get the following weekend to fill in the questionnaire via the web.
3. The following Monday the web questionnaire is closed. Nonrespondents to the web questionnaire are then approached by telephone if a telephone number is available. This approach is not preceded by a specific advance letter. Though, in the letters from steps 1 and 2 there is a notification that if a household does not participate in the survey via the Internet they will be requested to participate in a telephone interview

The reason for requesting all respondents of the first wave to fill in a web questionnaire is to determine the effect of this mode change on the response rate of the respondents by telephone or face-to-face of the first wave. The reason for sending the reminder to only half of the sample units is to find out what effect is to be expected from sending the reminder close to the data collection period by telephone.

The sample size of the pilot was 4.000. An estimated 24 percent of the sample units are expected to respond via the web. The data collection by telephone and face-to-face is expected to add an estimated 31 percent as a result of which the total response rate is expected to add up to 55 percent for the first wave. For comparison, the response rate for the first wave of the regular LFS from January to March 2011 was 54 percent. The expected response rate of the second wave is not determined in advance.

Approach strategy LFS-pilot including response rates

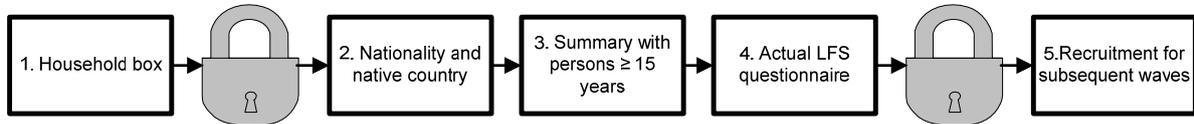


(1) The response rates indicated between square brackets are a percentage of the initial sample. The response rates not indicated between square brackets are percentages of the subportion.

3 Approach development questionnaire LFS

After the design has been established and approved by the customer, it is up to the developer to translate the design into a questionnaire and to specify the questions, routings, controls, etc.

The LFS questionnaire consists mainly of 5 chapters in following sequence:



3.1 Householdbox

The samples for the LFS are based on address data. These do not contain details about the inhabitants of a specific address but only about the address itself and a telephone number.

To collect information about the composition of the household and background information (gender, age, marital status) on the individual members, we use the so called 'householdbox'. One person completes the householdbox with the details for each occupant on the selected address, with a maximum of 8 persons.

The householdbox starts with a question on the size of the household. This information is used as input for a question with a limited range of answer-options to determine the household compositions (e.g., a household sized two can never be a household with a couple and children). When the composition is established, background information will be asked first for the couple / parent, second for their children and last for the other members of the household. The other members of the household are also asked what their relationship (e.g., mother, sibling or aunt/uncle) is to the eldest person of the couple or the parent in the household.

After the information in the householdbox is completed for all the persons (up to 8) living at the selected address, a summary appears with the details for all the inhabitants. The respondent has still the opportunity to adjust the information if necessary. If the respondent agrees with the data of the summary, he/she can check the mark in the box 'Data correct' and proceeds with the questionnaire.

We hebben nu van uw huishouden de volgende gegevens:

man, 49 jaar oud (11-11-1962), gehuwd / geregistreerd partnerschap, echtgenoot / partner
vrouw, 48 jaar oud (26-4-1963), gehuwd / geregistreerd partnerschap, echtgenote / partner
man, 18 jaar oud (9-6-1993), nooit gehuwd geweest, kind
vrouw, 16 jaar oud (22-2-1996), nooit gehuwd geweest, kind
man, 15 jaar oud (14-11-1996), nooit gehuwd geweest, kind
jongen, 14 jaar oud (2-2-1998), nooit gehuwd geweest, kind
jongen, 12 jaar oud (13-4-1999), nooit gehuwd geweest, kind
jongen, 10 jaar oud (14-12-2001), nooit gehuwd geweest, kind

Kloppen bovenstaande gegevens?
Zo niet, ga met behulp van de knop 'vorige' onderaan in het scherm terug in de vragenlijst om wijzigingen aan te brengen.

Gegevens correct

After the householdbox has been confirmed it is not possible anymore to make adjustments. The data is fixed and lays the foundation for the continuation of the questionnaire, such as routing.

3.2 Nationality and native country

The next question is about whom will answer the remaining questions of the householdbox for the members of the household.

Wie beantwoordt de vragen voor het huishouden?

- man, 49 jaar oud (11-11-1962), gehuwd / geregistreerd partnerschap
- vrouw, 48 jaar oud (26-4-1963), gehuwd / geregistreerd partnerschap
- man, 18 jaar oud (9-6-1993), nooit gehuwd geweest
- vrouw, 16 jaar oud (22-2-1996), nooit gehuwd geweest
- man, 15 jaar oud (14-11-1996), nooit gehuwd geweest
- jongen, 14 jaar oud (2-2-1998), nooit gehuwd geweest
- jongen, 12 jaar oud (13-4-1999), nooit gehuwd geweest
- jongen, 10 jaar oud (14-12-2001), nooit gehuwd geweest

Subsequently some questions follow about the nationality and the native country of the household members and the parents of the core of the household (in general the parents of the partners in the household).

3.3 Summary with persons of 15 years and older

After the householdbox has been completed, the respondent needs to enter the names of all the persons of 15 years and older in string fields. Only the persons with an age of 15 years and older appear in the summary.

In case the household contains only one person or only one person of 15 years and older, the summary will be skipped and the respondent arrives directly into the actual LFS questionnaire.

Er volgt nu voor iedere persoon in uw huishouden ouder dan 14 jaar een aantal vragen. Om duidelijk aan te kunnen geven voor wie welke vragen bestemd zijn, willen we u vragen om voor onderstaande personen een voor u herkenbare naam (of omschrijving) op te geven.

Persoon	Naam
1. Man, 49 jaar oud (11-11-1962)	John
2. Vrouw, 48 jaar oud (26-4-1963)	Anna
3. Man, 18 jaar oud (9-6-1993)	Peter
4. Vrouw, 16 jaar oud (22-2-1996)	Sophie
5. Man, 15 jaar oud (14-11-1996)	Nick

After the names have been confirmed the next screen gives the opportunity to complete for each person their part of the LFS questionnaire.

Er volgt nu voor iedere persoon in uw huishouden ouder dan 14 jaar een aantal vragen. Deze vragen kunnen door de betreffende persoon zelf worden ingevuld, maar mogen ook door iemand anders in het huishouden (ouder dan 14 jaar) ingevuld worden. Als voor alle personen de vragen zijn beantwoord, kunt u de vragenlijst terugsturen. Als u met de vragenlijst begonnen bent kunt u met de knop "overzicht" onderaan het scherm altijd weer terugkeren naar deze pagina. De antwoorden die u al heeft gegeven worden dan opgeslagen.

Start de vragenlijst voor de betreffende persoon door op de knop "Invullen" te klikken die achter de gegevens van die persoon staat.

- 1. John: Man, 49 jaar oud (11-11-1962)
- 2. Anna: Vrouw, 48 jaar oud (26-4-1963)
- 3. Peter: Man, 18 jaar oud (9-6-1993)
- 4. Sophie: Vrouw, 16 jaar oud (22-2-1996)
- 5. Nick: Man, 15 jaar oud (14-11-1996)

One of the requirements for this screen is that the respondents do not need to fill in the questionnaire in sequence of the names as shown on the screen. The respondent(s) need(s) to have the flexibility to choose their questionnaire despite in which order they appear in the summary.

Another requirement is that a respondent does not need to complete his/her questionnaire before another member of the household can start with his/her questionnaire. They are able to interrupt their questionnaire and another person can continue with his/her part.

This example shows that person 2 (Anna) and person 4 (Nick) have partly completed their questionnaire (status = busy). Other members of the household have the opportunity to complete their questionnaires, or Anna or Nick is able to continue or to finish their questionnaires. The status of the questionnaire appears behind the line(s) with the names.

Er volgt nu voor iedere persoon in uw huishouden ouder dan 14 jaar een aantal vragen. Deze vragen kunnen door de betreffende persoon zelf worden ingevuld, maar mogen ook door iemand anders in het huishouden (ouder dan 14 jaar) ingevuld worden. Als voor alle personen de vragen zijn beantwoord, kunt u de vragenlijst terugsturen. Als u met de vragenlijst begonnen bent kunt u met de knop "overzicht" onderaan het scherm altijd weer terugkeren naar deze pagina. De antwoorden die u al heeft gegeven worden dan opgeslagen.

Start de vragenlijst voor de betreffende persoon door op de knop "Invullen" te klikken die achter de gegevens van die persoon staat.

1. John: Man, 49 jaar oud (11-11-1962)	Invullen	
2. Anna: Vrouw, 48 jaar oud (26-4-1963)	Invullen	Bezig met invullen
3. Peter: Man, 18 jaar oud (9-6-1993)	Invullen	
4. Sophie: Vrouw, 16 jaar oud (22-2-1996)	Invullen	Bezig met invullen
5. Ilick: Man, 15 jaar oud (14-11-1996)	Invullen	

After the questionnaire for a particular member of the household has been completed and approved, the button to complete the questionnaire disappears and the status is updated. This means that the questionnaire for that particular person cannot be updated anymore.

Er volgt nu voor iedere persoon in uw huishouden ouder dan 14 jaar een aantal vragen. Deze vragen kunnen door de betreffende persoon zelf worden ingevuld, maar mogen ook door iemand anders in het huishouden (ouder dan 14 jaar) ingevuld worden. Als voor alle personen de vragen zijn beantwoord, kunt u de vragenlijst terugsturen. Als u met de vragenlijst begonnen bent kunt u met de knop "overzicht" onderaan het scherm altijd weer terugkeren naar deze pagina. De antwoorden die u al heeft gegeven worden dan opgeslagen.

Start de vragenlijst voor de betreffende persoon door op de knop "Invullen" te klikken die achter de gegevens van die persoon staat.

1. John: Man, 49 jaar oud (11-11-1962)	Invullen	
2. Anna: Vrouw, 48 jaar oud (26-4-1963)	Invullen	Bezig met invullen
3. Peter: Man, 18 jaar oud (9-6-1993)	Invullen	
4. Sophie: Vrouw, 16 jaar oud (22-2-1996)		Invullen voltooid
5. Ilick: Man, 15 jaar oud (14-11-1996)	Invullen	

The first question after activating the button "Fill" concerns the proxy report. Three options are given: the respondent completes the questionnaire him- or herself, another person in the household can fill in the questionnaire or no one will answer the questions.

In case the respondent will answer the questions him- or herself, the questionnaire continues with the questions of the actual LFS survey.

Wie beantwoordt de vragen voor Anna?

- Anna zelf
- Iemand anders in het huishouden
- Niemand wil de vragen beantwoorden

If someone else will answer the questions and the household consists of more than 2 people aged 15 years or older, a next question appears who in the household will complete the LFS survey for this person. As answer options, all the names of the other household members aged 15 years or older are presented.

Wie beantwoordt de vragen voor Anna?

- Anna zelf
- Iemand anders in het huishouden
- Niemand wil de vragen beantwoorden

Wie beantwoordt de vragen voor Anna?

- John
- Peter
- Sophie
- Nick

In case nobody will answer the questions for the selected person, a warning is presented. If the respondent is sure no one will answer the questions, he/she will return to the screen with the summary of the persons in the household of 15 years and older.

In that case it concerns partial response (Code 31, whereas complete response is code 30). The recruitment block will be skipped in the survey and the respondent(s) will not be approached for subsequent waves.

Wie beantwoordt de vragen voor Anna?

- Anna zelf
- Iemand anders in het huishouden
- Niemand wil de vragen beantwoorden

Weet u zeker dat niemand de vragen wil beantwoorden?
Zo ja, klik dan op knop "Akkoord" om het invullen te voltooien.

3.4 Actual LFS questionnaire

The first wave of the LFS questionnaire includes mainly blocks about economic activity, occupation and educational attainment, followed by some additional questions among others about health and religion.

After the questionnaire for a particular member of the household has been completed and approved, the data is fixed and cannot be updated anymore.

3.5 Recruitment for subsequent waves

Only after the LFS questionnaires for all the persons involved have been completed, it is possible to continue to the recruitment block. In this block Statistics Netherlands asks the respondent for permission to approach the household for the next subsequent waves.

When this block has been completed, the respondent is able to send the LFS survey to Statistics Netherlands.

3.6 Points of interest to develop the LFS questionnaire

Before a correct and complete design of the LFS survey could be handed over to program the questionnaire according to the specifications in Blaise, there were some challenges that first had to be tackled.

3.6.1 Flexible summary with persons of 15 years and older

For instance there was the dilemma about the requirement that the respondents do not need to fill in the questionnaire in sequence. This issue was solved in the “Summary with persons of 15 years and older” (See chapter 3.3)

3.6.2 Freeze data of the householdbox and the actual LFS questionnaire

In the questionnaire routes are based on data collected in the householdbox. If a considerable part of the survey has been filled in and information of the householdbox will be changed afterwards, the routing in the questionnaire can become corrupt. To avoid this situation, it was decided to freeze the information inserted in the householdbox, (See chapter 3.1, Householdbox).

After the actual LFS questionnaire has been completed for a specific person, it is not possible to update this part anymore.

3.6.3 Navigation

In case the respondent interrupts the questionnaire and returns again afterwards, he/she needs to scroll through the survey to arrive at the next question to reply. It is not possible to navigate through the questionnaire to the last or first question as the respondent will arrive in the summary with the names of the persons of 15 years and older. This is the first question to reply in case the actual questionnaires per person are not completed yet.

3.6.4 Control on respondent(s) of 15 years and older

Only persons in the household of 15 years and older are allowed to fill in the LFS survey. However, as the CAWI LFS survey is a self-administered mode there is less possibility to check if this is actually the case compared to interviewer-administered modes (CATI/CAPI).

3.6.5 Text imputations

Depending on the person who fills in the questionnaire, the imputation “he, she or you” needs to appear. This effort was rather time consuming to make sure it was on the right places in the design.

3.7 Lessons learned

From the helpdesk we received some calls from respondents about the font. For some respondents the LFS CAWI-mode is difficult to read. Besides, at the beginning from the character “W” a part does not appear on the screen. Within Statistics Netherlands a decision needs to be made which font is most appropriate to use in CAWI modes.

4 Approach programming questionnaire LFS

After the questionnaire has been developed, the Blaise programmers use the specifications of the design to program the questionnaire.

Obviously also here were some challenges to be able to execute the requirements of the design of the LFS questionnaire.

4.1 Points of interest to program the LFS questionnaire

4.1.1 Freeze details in the householdbox and the actual LFS questionnaire

Among others there was the requirement to freeze the details of the household box after it was completed for all the members of the household. This should make it impossible for the respondent to return into the householdbox afterwards.

After the details have been entered in the householdbox the respondent needs to confirm that the information completed is correct. Subsequently the respondent arrives in the next screen.



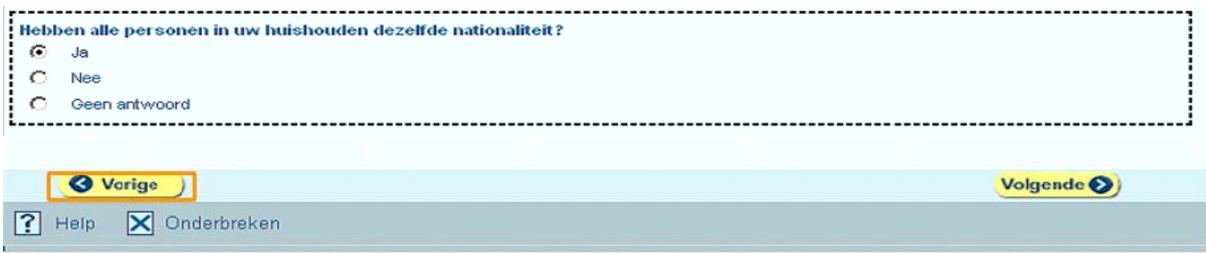
Voor het volgende deel van de vragenlijst is het belangrijk dat de persoonsgegevens niet meer gewijzigd worden. Als u geen wijzigingen in de vorige vragen meer wilt aanbrengen toets dan op akkoord om verder te gaan.

Akkoord

← Vorige Volgende →

? Help Onderbreken

If the respondent selects the button “Previous”, he/she cannot return anymore to the householdbox but arrives in the information screen. The details in the householdbox cannot be updated anymore.



Hebben alle personen in uw huishouden dezelfde nationaliteit?

Ja
 Nee
 Geen antwoord

← Vorige Volgende →

? Help Onderbreken

Welkom bij de enquête

Invullen vragenlijst

Met de knop **Help** onderaan het scherm krijgt u extra informatie over het werken met deze online vragenlijst.

Bladeren

De vragenlijst bestaat uit een aantal schermen; blader zo nodig via de knoppen  en .

Extra toelichting

Indien aanwezig verschijnt via de knop  extra toelichting bij de vraag.

Vraag niet van toepassing

Is een vraag voor u niet van toepassing, laat deze dan leeg indien deze niet verplicht is. U krijgt automatisch een melding als u per ongeluk een verplichte vraag leeg laat.

Onderbreken

U kunt het invullen halverwege onderbreken met de knop die u in de menubalk onderaan het scherm vindt. Bij het onderbreken worden de al ingevulde antwoorden automatisch opgeslagen zodat u later weer verder kunt gaan met invullen.

Verzenden van de vragenlijst

Klaar? Verzend door klikken op knop  (verschijnt rechts onderaan in beeld). Aansluitend kunt u de ontvangstbevestiging afdrucken voor uw administratie.

Stuur uw ingevulde vragenlijst vóór de retourdatum (indien van toepassing) terug naar het CBS.

Privacy

U kunt er bij onderzoeken van het CBS zeker van zijn dat uw privacy is gewaarborgd. Het is wettelijk vastgelegd dat de gegevens die het CBS verzamelt, alleen voor statistische doeleinden worden gebruikt. Geen enkele instelling kan toegang opeisen tot de gegevens die het CBS verzamelt. Bovendien zijn nooit persoonlijke gegevens te herkennen of af te leiden in de statistische informatie die het CBS naar buiten brengt.

Vragen

Bij vragen kunt u contact opnemen met het CBS Contact Center:

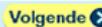
Telefoon (op werkdagen):

- (045) 570 64 00 voor inhoudelijke vragen (tussen 9-17 uur)

- (045) 570 66 27 voor technische vragen (tussen 10-12 en 14-16 uur)

E-mail: contactcenter@cbs.nl

Alvast bedankt voor uw medewerking!



 Help  Onderbreken

The same principle applies as well to the actual LFS questionnaire after it has been completed per person.

4.1.2 Buttons on command

Under certain conditions buttons need to:

- appear
- disappear
- fade (light grey), button not active
- brighten up (yellow), button active

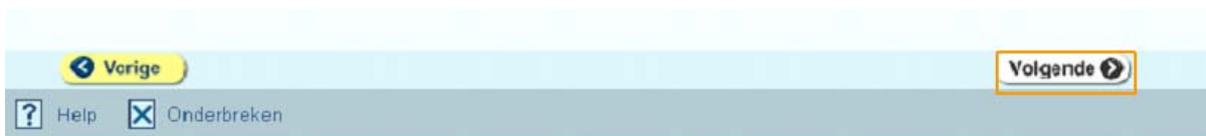
In the summary of the persons of 15 years and older in the household the button “Next” will only become active after the actual LFS questions for all the persons have been completed. As from that moment the respondents are able to continue in the LFS survey.

The actual LFS questionnaire is not yet completed for all the persons of 15 years and older in the household. The button “Next” is not yet active.

Er volgt nu voor iedere persoon in uw huishouden ouder dan 14 jaar een aantal vragen. Deze vragen kunnen door de betreffende persoon zelf worden ingevuld, maar mogen ook door iemand anders in het huishouden (ouder dan 14 jaar) ingevuld worden. Als voor alle personen de vragen zijn beantwoord, kunt u de vragenlijst terugsturen. Als u met de vragenlijst begonnen bent kunt u met de knop "overzicht" onderaan het scherm altijd weer terugkeren naar deze pagina. De antwoorden die u al heeft gegeven worden dan opgeslagen.

Start de vragenlijst voor de betreffende persoon door op de knop "Invullen" te klikken die achter de gegevens van die persoon staat.

1. John: Man, 49 jaar oud (11-11-1962)	Invullen
2. Anna: Vrouw, 48 jaar oud (26-4-1963)	Invullen
3. Peter: Man, 18 jaar oud (9-6-1993)	Invullen
4. Sophie: Vrouw, 16 jaar oud (22-2-1996)	Invullen
5. Hick: Man, 15 jaar oud (14-11-1996)	Invullen



If the actual LFS questionnaire for all the members of the household of 15 years and older has been completed, the button “next” becomes active and the respondent is able to navigate further to the recruitment block.

Er volgt nu voor iedere persoon in uw huishouden ouder dan 14 jaar een aantal vragen. Deze vragen kunnen door de betreffende persoon zelf worden ingevuld, maar mogen ook door iemand anders in het huishouden (ouder dan 14 jaar) ingevuld worden. Als voor alle personen de vragen zijn beantwoord, kunt u de vragenlijst terugsturen. Als u met de vragenlijst begonnen bent kunt u met de knop "overzicht" onderaan het scherm altijd weer terugkeren naar deze pagina. De antwoorden die u al heeft gegeven worden dan opgeslagen.

Start de vragenlijst voor de betreffende persoon door op de knop "Invullen" te klikken die achter de gegevens van die persoon staat.

1. John: Man, 49 jaar oud (11-11-1962)	Invullen voltooid
2. Anna: Vrouw, 48 jaar oud (26-4-1963)	Invullen voltooid
3. Peter: Man, 18 jaar oud (9-6-1993)	Invullen voltooid
4. Sophie: Vrouw, 16 jaar oud (22-2-1996)	Invullen voltooid
5. Hick: Man, 15 jaar oud (14-11-1996)	Invullen voltooid

Het invullen van de persoonsvragenlijsten is voltooid. Klik op de knop "Volgende" om verder te gaan.



Also in other places in the questionnaire certain buttons contain specific commands.

4.1.3 Page Up

On the screen with the proxy question it was decided not to show the button “Previous”. However, the respondent was able to return to the previous screen with the key “Page Up” on the keyboard. This issue is solved in a next version of Bismenu.

4.1.4 Proxy

The cawi question asking which person is going to answer the questions about person X is pre-filled with the answer ‘person X’.

This was due to technical reasons. Because the function of “Page Down” could not be blocked, it was decided to add first the blocks for the (maximum 8) persons of 15 years and older on the route and

then the blocks with the selection of the persons. Therefore it was necessary to pre-fill the question about person X. This issue is solved in the next version of Bismenu.

Wie beantwoordt de vragen voor John?

John zelf

Iemand anders in het huishouden

Niemand wil de vragen beantwoorden

4.1.5 Actual LFS questionnaire for maximum 8 persons

Normally, if the same questions are asked to multiple persons in a household, the questionnaire contains a loop.

Due to technical reasons relating to the summary of persons of 15 years and older, the actual LFS questionnaire contains the same questions for each person separately (maximum 8 times). This issue is solved in a next version of Bismenu.

4.1.6 CAWI and CATI/CAPI, different questionnaires

The CAWI and CATI/CAPI surveys are two different questionnaires due to deviation in text, routes, instructions, etc. This means that in case mixed mode is applied in surveys, a separate questionnaire needs to be programmed for CAWI and CATI/CAPI.

5 Results

5.1 Response and recruitment results

In the first wave of the pilot, a total response rate of 54,1 percent is achieved, which is about the same as in the first wave of the regular LFS from January to March 2011 (Table 1). Almost half of the total responses is obtained by cawi. Households with one or two members of 15 years of age or older have higher response rates than households with three or more members (Table 2). Still, one in five households with more than three members responds through the Internet. Households of a non western origin respond less via the Internet than households of a different origin. This observation also holds for the other modes.

Table 1 Response results first wave

	cawi	cati	capi	Total	cawi	cati	capi	Total
	N				%			
Total households in sample	4.000	984	1.821	4.000	100,0%	100,0%	100,0%	100,0%
<i>of which complete response 1st wave</i>	1064	404	697	2165	26,6%	41,1%	38,3%	54,1%

Source: Statistics Netherlands; Datacollection

Table 2 Response results first wave among large households and ethnic minorities

	cawi			cati			capi		
	N	response	%	N	response	%	N	response	%
Size of household									
1	1.113	279	25,1%	309	104	33,7%	495	164	33,1%
2	1.840	556	30,2%	675	300	44,4%	559	229	41,0%
3	592	133	22,5%	-	-	-	437	175	40,0%
>3	455	96	21,1%	-	-	-	326	125	38,3%
Origin of household									
non western	680	85	12,5%	122	37	30,3%	430	128	29,8%
other	3.320	979	29,5%	862	367	42,6%	1.387	565	40,7%

Source: Statistics Netherlands; Datacollection

At the end of the questionnaire of the first wave respondents were recruited for the following waves. The total recruitment rate for the subsequent waves is 78,5 percent (Table 3), which is about 10 percent points lower than in the regular design. The lower result is due to a lower recruitment rate in cawi. In addition, most cawi respondents refuse to fill in their telephone number. As it would be unethical to telephone these households, they are only approached by cawi in subsequent waves, even though the number may be listed. Note that for cawi the share of recruited households with a listed telephone number is larger than the share of recruited households with a telephone number filled in by respondents.

Table 3 Recruitment results subsequent waves

	cawi	cati	capi	Total	cawi	cati	capi	Total
	N				%			
Complete response 1st wave	1064	404	697	2165	100,0%	100,0%	100,0%	100,0%
Recruit % subsequent waves								
No objection subs. Waves	707	377	615	1699	66,4%	93,3%	88,2%	78,5%
Recruit % with available telephone number								
Recruited and telephone number filled in by respondent	284	377	613	1274	26,7%	93,3%	87,9%	58,8%
Recruited and listed telephone number	564	377	376	1317	53,0%	93,3%	53,9%	60,8%

Source: Statistics Netherlands, Datacollection

The total response rate of the second wave is 69,4 percent (Table 4). The cati response ranges between 75,0 and 78,6 percent, which is considerably lower than in the regular design (typically about 88 percent). In other words, introducing cawi again as a first mode in the second wave seems to reduce cati response.

This is problematic, because the lower cati response is not compensated by a high cawi response. As a result, total response rate is lower.

Total response for households that responded via cati or capi in the first wave is higher in comparison to households that responded via cawi in the first wave. The reason for this is, that a large part (38 percent) of the households that responded via cawi in the first wave, did not respond cawi in the second wave, but did not continue to cati follow-up either, because they did not fill in their telephone number. Cati response among the households that did fill in their telephone number in cawi is actually about the same as cati response among the households that responded capi in the first wave. Cawi response is highest among households that responded via cawi in the first wave.

It should be noted that the low cawi response for households that responded cati in the first wave may partly be attributed to problems with Internet safety certificates, which prohibited respondents from responding through the Internet. Other households did not encounter these problems, because for them fieldwork took place at a different point in time.

Table 4 Response results second wave, specified by mode in first wave

	Second wave					
	cawi	cati	Total	cawi	cati	Total
	N			%		
Total households in sample of 2nd wave	1.694	1.043	1.694	100,0%	100,0%	100,0%
<i>of which complete response in second wave</i>	378	797	1.175	22,3%	76,4%	69,4%
Households in sample of 2nd wave running through from cawi in 1st wave	707	184	707	100,0%	100,0%	100,0%
<i>of which complete response in second wave</i>	257	138	395	36,4%	75,0%	55,9%
Households in sample of 2nd wave running through from cati in 1st wave	376	350	376	100,0%	100,0%	100,0%
<i>of which complete response in second wave</i>	25	275	300	6,6%	78,6%	79,8%
Households in sample of 2nd wave running through from capi in 1st wave	611	509	611	100,0%	100,0%	100,0%
<i>of which complete response in second wave</i>	96	384	480	15,7%	75,4%	78,6%

Source: Statistics Netherlands; Datacollection

5.2 Reminder strategy

Two reminder strategies were used in the second wave. Half of the sample received a reminder by mail, the other half did not (random assignment in advance). Cawi response is higher among households that received a reminder, but cati response is lower (Table 5). Overall, response is slightly (but not significantly) higher among households that received a reminder.

Table 5 Response results second wave, specified by reminder strategy

	Second wave					
	cawi	cati	Total	cawi	cati	Total
	N			%		
Total households in sample with reminder	817	476	817	100,0%	100,0%	100,0%
<i>of which complete response</i>	212	360	572	25,9%	75,6%	70,0%
Total households in sample without reminder	877	567	877	100,0%	100,0%	100,0%
<i>of which complete response</i>	166	437	603	18,9%	77,1%	68,8%

Source: Statistics Netherlands; Datacollection

5.3 Partial response

Partial response (when not all household members respond) is 3,1 percent in cawi and 3,5 percent in total in the first wave. This is more than in the regular design (1,5 percent), but well below 5 percent. At the start of the pilot, it was agreed that a partial response rate below 5 percent would be considered acceptable.

5.4 Proxy reports

The number of proxy reports appears to be the same in cawi as in cati (both lower than cati). In households with two members, for example, about one in three reports are proxy. The results should be treated with some caution, however, because the cawi question asking which person is going to answer the questions about person X is pre-filled with the answer 'person X' (for technical reasons). Respondents might have overlooked that they have to change this actively in case of proxy report.

5.5 New questions about educational attainment

Parallel to the pilot, a separate field test was conducted to test new questions about educational attainment that do not require the use of complex automatic coding modules. The field test included the modes cawi and cati. During this test Cati interviewers wrote down comments and questions of respondents and themselves. Cawi respondents were contacted afterwards by telephone to share their experiences with the new questions. Based on the reactions from respondents and interviewers, the new questions about education appear to function well. Respondents who answered the questions proxy, indicate that they had no difficulty. In addition, the new questions take less time to complete than the old questions. Subsequently the new questions are implemented as well in the CAWI questionnaire as in the CATI/CAPI questionnaires.

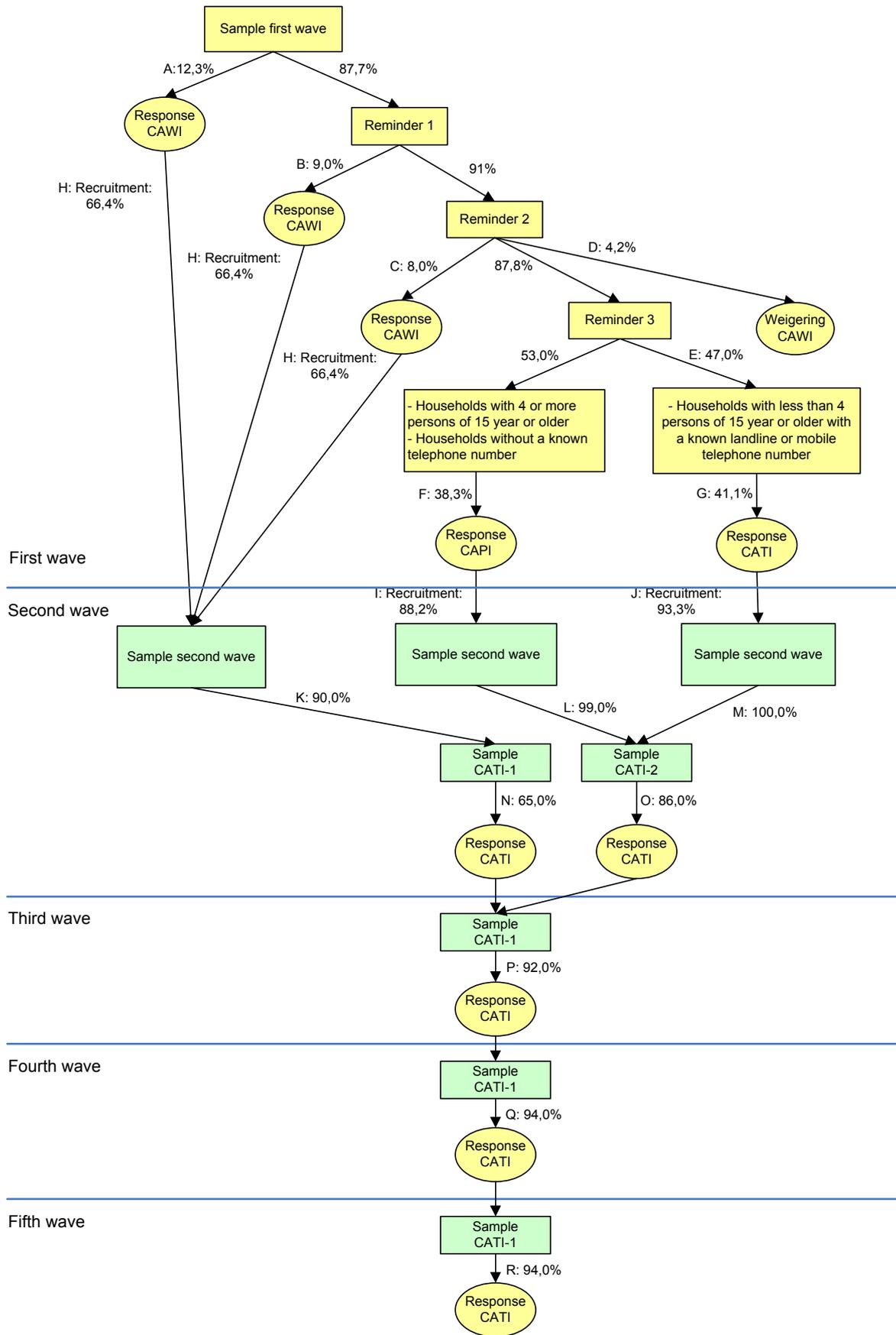
6 Design changes

On the basis of the results of the pilot it was concluded that the approach strategy of the pilot could not be used for the parallel run of the LFS without any adjustment. In the parallel run a slightly different approach strategy will be used. The first wave of the LFS will be almost identical to the first wave of the pilot. There are two differences. Firstly, in order to generate a higher cost reduction a larger portion of the cawi nonrespondents will be approached by telephone: all households consisting of up to three people of 15 years of age or older for which a landline telephone number is available will be approached by telephone. Secondly, cawi-respondents of the first wave are only asked for their telephone number if this is not yet available.

The approach strategy for the second wave is adjusted fundamentally. The CAWI mode will not be used at all in the second wave. If a telephone number is available, the second wave is carried out by telephone. From table 4 in paragraph 5.1 it was concluded that of all cawi respondents of the first wave who indicated that they were willing to participate in the second, only 36,4 percent actually did so via cawi. This is rather disappointing. Moreover it was concluded that the follow-up approach by telephone of web nonrespondents did not raise the response level to the customary response level for the second wave of the LFS. Also the response by web in the second wave for cati or capi respondents of the first wave is very low and the total response level of these groups is actually lower than in the regular design. Therefore it was concluded that web data collection does more harm than good for the second wave of the LFS.

Cawi respondents of the first wave for which no telephone number is available are excluded from the following waves. It is assumed that the majority of respondents of the first wave will be prepared to give a telephone number if none is available. The remaining portion of respondents, meaning the ones that are not prepared to give a telephone number, is assumed to be so small that the number of responses generated from this portion in the second and following waves is outweighed by the amount of work needed to approach them. The future will show if these assumptions are justly made. The third to fifth wave were not carried out in the pilot, so no empirical facts are available. Since in the second wave only data collection by telephone is carried out, it is logical to go ahead with this strategy in the following waves.

Approach strategy LFS 2012 including response rates



(1) The response rates indicated between square brackets are a percentage of the initial sample. The response rates not indicated between square brackets are percentages of the subportion.

7 Conclusions and further research

The most important conclusion of the pilot is that moving to the Internet is very well possible and reduces costs in the first wave of the LFS, but also results in higher attrition rates in the panel of the LFS. So, when implementing cawi or any other non-interviewer supported mode in a panel design, attention should be paid to the willingness of respondents to participate in following waves. In the new mixed-mode design of the LFS that is introduced in 2012 at Statistics Netherlands, this problem is addressed by approaching all respondents only by telephone and not through the Internet in the second and following waves. Also research on the design of the LFS now focuses on improving recruitment rates of cawi respondents.

From the Blaise programming point of view, most of the requirements in the design, such as the introduction of the flexible summary with persons of 15 years and older, have been met. The programmers succeeded to build a good working mixed mode LFS household questionnaire within the imposed deadline.

References

Cuppen, M., P. van der Laan, W. van Nunspeet 2011. 'Re-engineering Dutch social surveys: From single-purpose surveys to an integrated design', Paper presented at the World Statistics Congress of the International Statistical Institute (ISI), Dublin, Ireland, 21-26 August 2011

Janssen, B. 2010. 'Web data collection for Household Surveys at Statistics Netherlands', English internal paper, Statistics Netherlands, Heerlen, November 2010.

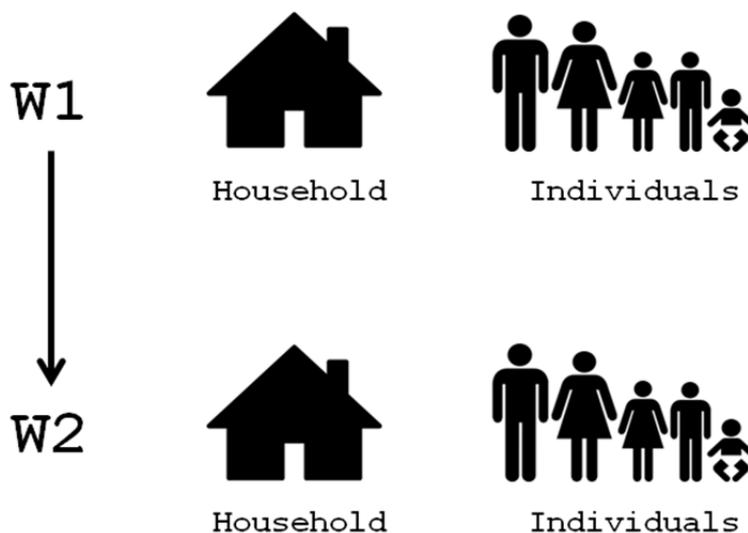
Godelief Mars, Rokaya Chadli, Björn Janssen, Menno Cuppen 2011. 'Introducing web interviewing in the Labour Force Survey at Statistics Netherlands: a pilot', English internal paper, Statistics Netherlands, December 2011

Developing and Testing A Complex Mixed Mode Questionnaire Instrument

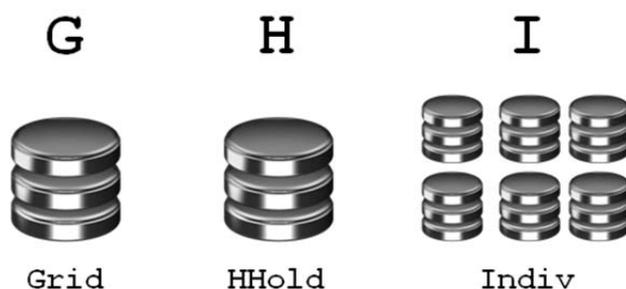
Richard Boreham, NatCen Social Research

Structure

Understanding Society is a longitudinal household survey. At Wave 1 households were randomly selected and interviewed, with every adult aged 16+ eligible for an individual interview. Anyone enumerated in an interviewed household at Wave 1 becomes an original sample member and they are included in the issued sample at all subsequent waves, and an attempt is made to interview them and all the people that they live with.

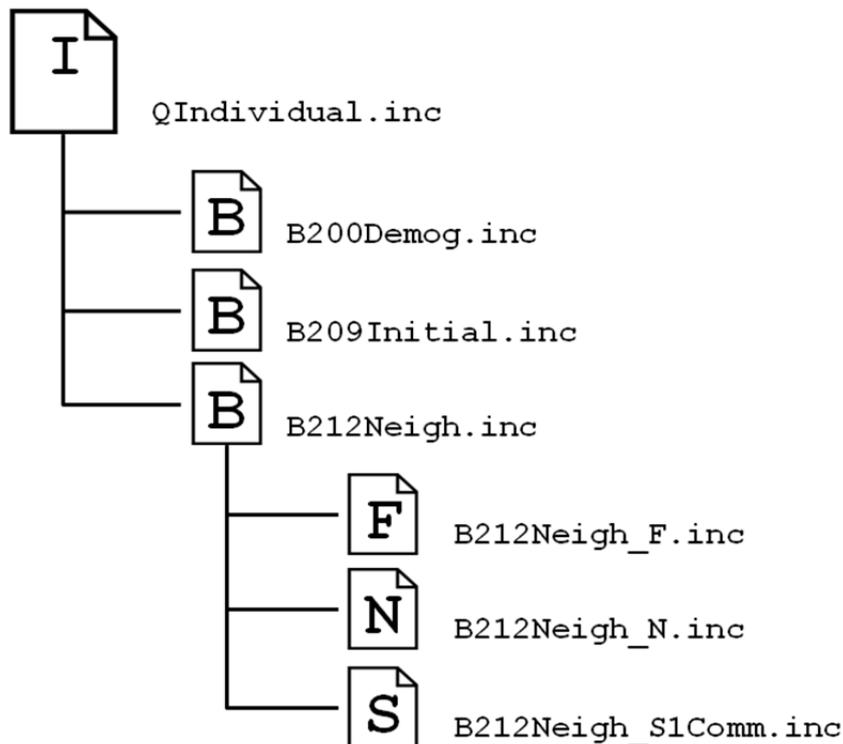


The Understanding Society questionnaire has 3 main components, a household grid, a household questionnaire, and then individual questionnaires for each adult aged 16+. These are programmed within one CAPI instrument as separate parallel blocks.



The household and individual instruments on Understanding Society consist of modules which are rotated on a wave by wave basis. Each module has its own separate block within the household or individual instrument, and each block is contained within a separate source file – all of which are named starting with a 'B', for example 'B212Neigh.inc' refers to the module on neighbourhood cohesion.

Within each block source file, the fields are specified within a separate field source file referenced from within the block, so B212Neigh_F.inc is called from the B212Neigh source file. At NatCen, researchers tend to program the fields, whereas programmers program the rules – therefore it makes sense to separate the fields from the rest of the block so that researchers can make amendments to question wording at the same time as programmers are working on the rules for a module. If there is a loop within a module, then the block containing the looped fields and rules is stored in a separate subfile and called from the main block. As an additional complication, the Understanding Society questionnaire is translated into 9 languages (other than English), so a distinction is made between translated and non-translated questions, where non-translated questions are either derived variables or questions which consist purely of interviewer instructions and with no question wording.



At Wave 1, the fieldwork was all face to face, so the questionnaire was programmed as a CAPI instrument. At Wave 2, Understanding Society incorporated the old British Household Panel Survey respondents, some of whom would only take part via a telephone interview, and so the questionnaire needed to work in both CAPI and CATI modes. For Wave 2, the questionnaire was developed initially in CAPI, then once the CAPI version was signed off as correct, it was frozen and a CATI version was developed using the CAPI version as the base. The situation was complicated, by having a separate version for the Northern Ireland sample as their fieldwork was carried out by NISRA (Northern Ireland Statistics and Research Agency), who use Blaise for their interviewing, but have a different sample management system to NatCen, which therefore required additional fields within the Blaise instrument. An edit version of the instrument was created to pull all the data from the separate instruments together.

The drawback of the approach taken at Wave 2 was that we had to maintain four separate versions of the Blaise instrument. Although the CAPI questionnaire had been signed off and frozen prior to creating any of the additional instruments, once fieldwork started, minor amendments had to be made to the CAPI and hence to all the other three versions, which made version control a nightmare. Therefore for Wave 3, one single instrument was created that used common code, primarily by using textfills to distinguish CAPI and CATI wording.

Understanding Society consists of a main survey wave and an innovation panel (IP) for methodological testing (such as the effect of different monetary incentives on response). The main methodological focus of IP5 was to test the use of mixed mode interviewing on overall response and to look at the mode effects associated with people’s answers to individual questions. Therefore the Understanding Society interviewing instrument needed to work in CAWI as well as CAPI and CATI.

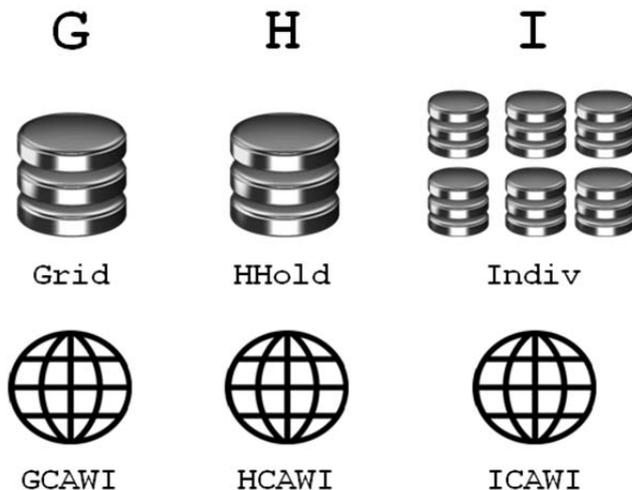
Our initial approach to re-programming the combined CAPI/CATI instrument was to continue to program using textfills for CAWI, and to use a ModeType variable to control routing of the different modes. However we quickly found that this was extremely time-consuming to program and was not a practical solution. The main problem we found was that although question wording could be similar for CAPI, CATI and CAWI, the Understanding Society has help screen for a large number of questions, and it was impossible to write help text that would make sense to both an interviewer and a respondent reading it, so we were having to create multiple textfills (as we are restricted to 255 characters for each textfill) for each question that had long help text. Textfills were also time consuming to create as they have to be referred to in 3 separate locations, namely specified under an Auxfield command, referred to in Fields and set in Rules. We realized that the best solution was to use Prepare Directives to identify the code that needed to be run in different modes.

We identified that we needed to generate five different survey instruments. One instrument was needed for each mode, CAPI, CATI and CAWI. NISRA needed a separate instrument for Northern Ireland which was essentially the same as the CAPI instrument with some extra code, and a final Edit version was needed to bring all the different component versions together. We worked out that we needed 7 different Prepare Directives to be able to create the 5 different datamodels that were required.

Prepare Directives	 CAPI	 CATI	 CAWI	 NIreland	 Edit
CAPI	●			●	●
CATI		●		●	●
CAPI_CATI	●	●		●	●
CAWI			●		
CAPI_CATI_CAWI	●	●	●		
NISRA				●	
EDIT					●

The Understanding Society instruments for IP5 were a 5 minute household grid, a 10 minute household questionnaire and a 32.5 minute individual questionnaire. Some routing within the household questionnaire and the individual questionnaire depends on variables within the household grid – for example parents were asked about childcare for each child within the household. We allow for up to 16 people to be enumerated in a household so there are numerous arrays within the household grid and individual questionnaire with 16 sets of the appropriate questions.

Due to the complexity of the existing CAPI/CATI instrument and concerns about navigating between parallel blocks in CAWI and possible performance issues, it was decided to have 3 separate CAWI instruments (Household Grid, Household Questionnaire and Individual Questionnaire) rather than one complete instrument.



As a result, separate Prepare Directives were needed for GCAWI, HCAWI and ICAWI, which meant that we needed 11 different Prepare Directives to generate 7 different datamodels.

Code within the body of the questionnaire

The W3 questionnaire has been developed as a joint CAPI/CATI instrument, so there were already some textfills in place to cope with differences between modes – mostly to do with the presence or lack of showcards in CAPI and CATI respectively.

```

FIELDS

NBRCOH1_A
    ^TF_SHOWCARD212A I am going to read out a set of statements
    that could be true about your neighbourhood. Please tell me
    how much you agree or disagree that each statement describes
    your neighbourhood.
    @/@/First, this is a close-knit neighbourhood. ^TF_READOUT"
    /"AGREE OR DISAGREE THAT NEIGHBOURHOOD IS CLOSE-KNIT" :TAgree
  
```

In this code there are textfills ^TF_SHOWCARD212A, and a global textfill ^TF_READOUT which are then set in the rules as follows.

```

RULES

TF_READOUT:=' '
TF_SHOWCARD212A:=' '
{ $IFDEF CAPI }
    TF_SHOWCARD212A:='SHOWCARD 212A@/@/'
{ $ENDIF }
{ $IFDEF CATI }
    TF_READOUT:='@/@/INTERVIEWER: READ OUT'
{ $ENDIF }
  
```

Previously the textfills had been set using routing based on a ModeType variable, but we altered the code so that every mode change was programmed using Prepare Directives.

The question NBRCOH1_A would actually work in CAWI as it stands because the textfills are set up correctly for CAWI. However, this question is actually the first question in a grid, therefore we need to split the question text up into the initial introductory sentence and sentence that is the question itself.

FIELDS

```
NBRCOH_CAWIIntro
"Here are a set of statements that could be true about your
neighbourhood. Please tell us how much you agree or disagree that
each statement describes your neighbourhood.": TCont

NBRCOH1_A (NC_NBRCOH1_A)
{$IFDEF CAPI_CATI}
  "^TF_SHOWCARD212A I am going to read out a set of statements
  that could be true about your neighbourhood. Please tell me
  how much you agree or disagree that each statement describes
  your neighbourhood.
  @/@/First, this is a close-knit neighbourhood. ^TF_READOUT"
{$ELSE}
  "@/@>This is a close-knit neighbourhood@>@/"
{$ENDIF}
/"AGREE OR DISAGREE THAT NEIGHBOURHOOD IS CLOSE-KNIT" :TAgree
```

We now have a NBRCOH_CAWIINTRO question variant, and because the NBRCOH1_A is part of a grid, we right-align the question text, and hence we do need to use Prepare Directives to create a CAWI version of the question enclosed with right align tags @> and with hard returns before and after the statement to space it out from other statements.

Throughout the questionnaire instrument we generally use the routing convention

```
{ $IFDEF CAPI_CATI }
{ $ELSE }
{ $ENDIF }
```

Within the rules section we also use Prepare Directives to make sure that all datamodels are compatible.

RULES

```
{ $IFDEF CAPI_CATI }
  NBRCOH_CAWIIntro.KEEP
{ $ELSE }
  NBRCOH_CAWIIntro :=Cont
  NBRCOH_CAWIIntro
{ $ENDIF }
NBRCOH1_A
NBRCOH2_A
NBRCOH3_A
NBRCOH4_A
```

So in the CAPI_CATI version NBRCOH_CAWIINTRO is kept, whereas in CAWI it is set to 'cont' (this is because it is displayed with just question text and no answer text so respondent can't code an answer) and asked (effectively just the question text is displayed). The same effect could have been achieved by allowing NBRCOH_CAWIINTRO to be Empty.

Code at the Datamodel level

The Prepare Directives are set at the very top of the DataModel. The code below shows how the Prepare Directives have been set for the CAPI and ICAWI datamodels.

```
DATAMODEL USIP5CAPI "IP5 CAPI"
```

```
{ $DEFINE CAPI }  
{ $DEFINE x-CATI }  
{ $DEFINE CAPI_CATI }  
{ $DEFINE CAPI_CATI_GCAWI }  
{ $DEFINE CAPI_CATI_HCAWI }  
{ $DEFINE CAPI_CATI_ICAWI }  
{ $DEFINE x-GCAWI }  
{ $DEFINE x-HCAWI }  
{ $DEFINE x=ICAWI }  
{ $DEFINE x-EDIT }  
{ $DEFINE x-NISRA }
```

```
DATAMODEL USIP5CAWI "IP5 CAWI"
```

```
{ $DEFINE x-CAPI }  
{ $DEFINE x-CATI }  
{ $DEFINE x-CAPI_CATI }  
{ $DEFINE x-CAPI_CATI_GCAWI }  
{ $DEFINE x-CAPI_CATI_HCAWI }  
{ $DEFINE CAPI_CATI_ICAWI }  
{ $DEFINE x-GCAWI }  
{ $DEFINE x-HCAWI }  
{ $DEFINE ICAWI }  
{ $DEFINE x-EDIT }  
{ $DEFINE x-NISRA }
```

The code highlighted in red shows which Prepare Directives are set to be active for each Datamodel. If a Prepare Directive is not active in a Datamodel then it is prefixed with 'x-' to make it inactive. This works because for example there is no code in the entire questionnaire defined by {\$IFDEF x-CATI}.

Once the different Datamodels have been defined, then it is simply a case of using Prepare Directives throughout the rest of the instrument. So different Modelibs can be set using the code below.

```

{$IFDEF CAPI}
    {$MODELIB CAPI.bml}
{$ELSE}
    {$IFDEF CATI}
        {$MODELIB CATI.bml}
    {$ELSE}
        {$MODELIB CAWI.bml}
    {$ENDIF}
{$ENDIF}

```

Different numbers of Primary Keys are needed for different Datamodels. There is a separate Datamodel for the ICAWI instrument, so it needs a unique identifier for each person, whereas for all the other Datamodels the unique identifier is at the household level.

```

{$IFDEF ICAWI}
    PRIMARY
        QID.Area,
        QID.Address,
        QID.HHold,
        QID.PNo
{$ELSE}
    PRIMARY
        QID.Area,
        QID.Address,
        QID.HHold
{$ENDIF}

```

The ICAWI instrument needs to look up variables defined at the household level and so needs to define the HHGrid as an external file whereas other Datamodels don't need to do this.

```

{$IFDEF ICAWI}
    USES
        HGModel 'USIP5GCAWI'

    EXTERNALS
        GData : HGModel ('USIP5GCAWI.BOI', OLEDB)
{$ENDIF}

```

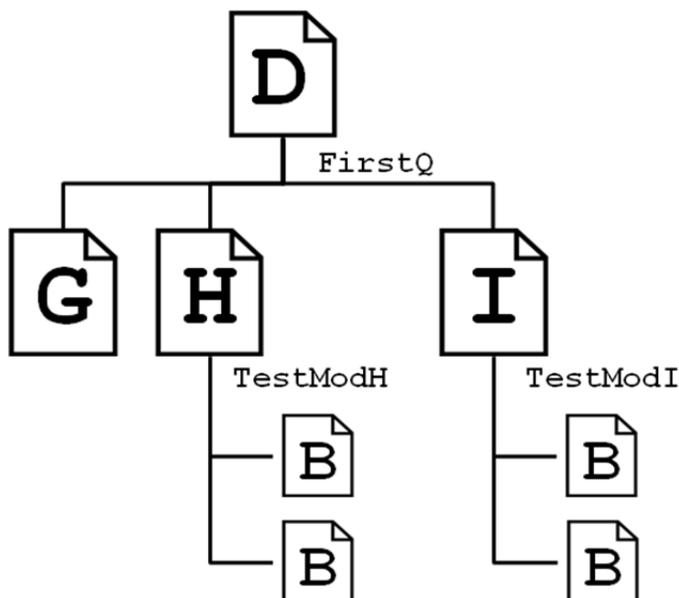
Ideally we would have liked to have put all the additional top level commands like defining the Modelib, Externals and Primary Keys into a top level include file, but Blaise didn't like this, so we have had create a top level .BLA file with all these commands in. However, once the code has been set with routing using Prepare Directives, then the only code that need to be altered to create a different Datamodel is the initial definition of the different Prepare Directives.

Testing

One of the difficulties with testing a complex and long questionnaire is that in order to test a module of questions at the end of a questionnaire, the tester has to answer all the questions preceding this module to get to it in the first place. Even though testers can become familiar with a questionnaire and know the quickest route through, it can still be time consuming to even get to a module at the end of a questionnaire before even starting the testing. Testing is made even more time consuming if the routing in a module depends on the answers to earlier questions, because the tester has to move backwards and forwards through the questionnaire (sometimes having to go down a different route) to be able to check the routing properly.

We decided that we wanted to design the Understanding Society instrument in a way that would allow testers to skip huge chunks of the questionnaire and get directly to the module that they wanted to test. BUT we also did not want to create separate test and live versions of questionnaires, so we needed a mechanism to allow us to create one instrument which could be used as a testing and live version without any modifications.

The diagram below illustrates how we decided to implement a combined test/live instrument.



At the Datamodel level the first question that interviewers are asked is called FirstQ which on standard NatCen surveys is just a “Press 1 to continue” question.

FIELDS

FirstQ

```
"INTERVIEWER: You are in the questionnaire for ^NCSerial
@/~/To update admin details press ^CtrlEnter"
:(Cont (1) "Interview Mode",
  Test (7) "Test Mode"), NODK, NORF
```

```
Password "Enter Test Mode PassCode":0..9999
```

We amended the standard FirstQ question to distinguish between interviewing and testing modes and had a follow up question for testers to enter a 4 digit passcode. The reason for this is that interviewers are inquisitive and someone would choose the test option when they were supposed to be doing live

interviewing, so we needed a mechanism to stop them progressing beyond this point. We chose a memorable date as the passcode (note that we didn't actually choose 1939).

RULES

```
IF Password<>1939 THEN
  CHECK ERROR "Incorrect PassCode
  @/@/INTERVIEWER: PLEASE ENTER <1> AT <FirstQ> TO CONTINUE"
ENDIF
```

Then at the household and individual level we had questions TestModH and TestModI which listed the modules within the questionnaire. TestModI at the individual level is shown below.

FIELDS

```
TestModI
"Which modules do you want to test?":SET OF
  (Demog (200) "Demographics",
  Initial (209) "Initial Conditions",
  Neigh (212) "Neighbourhood Cohesion",
  ...
```

TestModI is only asked in test mode, and then each module within the questionnaire is on route if either FirstQ was the live version or if it has been marked in TestModI.

RULES

```
IF (FirstQ=Test) THEN
  TestModI
ENDIF

IF (Demog IN TestModI) OR (Neigh IN TestModI) OR (FirstQ=Cont) THEN
  Q200Demog
ENDIF

IF (Initial IN TestModI) OR (FirstQ=Cont) THEN
  Q209Initial
ENDIF

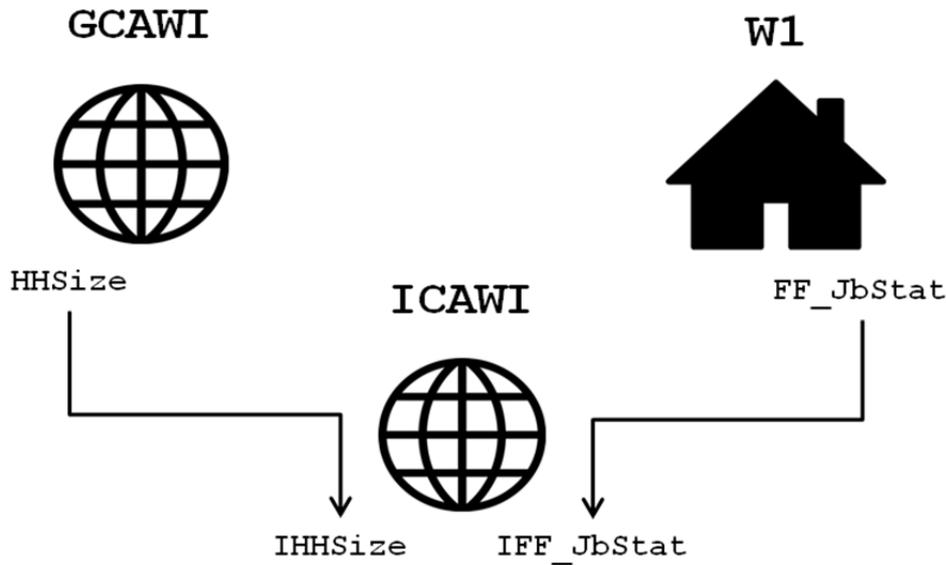
IF (Neigh IN TestModI) OR (FirstQ=Cont) THEN
  Q212Neigh
ENDIF
```

Where the routing in a later module depends on answers to a question in an earlier module, it is possible to force the earlier module to be on route if the later module is selected. In the example shown above, if module 212 Neighbourhood Cohesion is the only module selected at TestModI, then both 200Demog and 212Neigh modules are asked (because routing in 212Neigh depends on the answers to a question in 200Demog).

PROGRAMMING TIP: When naming modules use the same names to refer to the block and the module in TestMod. So the Block name is B200Demog (and is called using Q200Demog) and 'Demog' is the answer label in TestModI.

One final complication with the Understanding Society instrument is that routing within modules in the Individual questionnaire can depend on variables defined at the household grid level and on the

feedforward data from a previous wave. The best mechanism we found to test dependent routing was to impute a copy of any variable that affected routing into the Individual level block.



In the diagram above household size is defined at the household grid level as HHSize and a copy is imputed into the individual levels as IHHSize. Similarly fed-forward data about the job status was stored in ff_JbStat and imputed into the individual level as IFF_JbStat.

We then needed to set up a mechanism so that the imputed feed forward or household grid variables could be altered at the individual level to test the individual level routing, shown below for FF_JbStat

RULES

```

IFF_JbStat.KEEP

IF (ManFF=No) THEN
  IFF_JbStat:=FF_JbStat
  ManFF:=Yes
ENDIF

IF (Demog IN TestMod) THEN
  IFF_JbStat
ENDIF

```

So IFF_JbStat is initially kept, and then set within routing which ensures that it is only set once and not recalculated. Then if the 200Demog module is selected IFF_JbStat is asked so that it can be amended.

This code then needed to be altered for ICAWI as the ICAWI instrument is separate from the household grid and needed to look-up the values via the external BOI file.

RULES

```

IF (ManFF=No) THEN
  {$IFDEF CAPI_CATI}
    IFF_JbStat:=FF_JbStat
  {$ELSE}

```

```
        IF HGData.Search(QID.Area,QID.Address,QID.Hhold) THEN
            HGData.READ
            IFF_JbStat:=HGData.QSignIn.FF_JbStat
        ENDIF
    {$ENDIF}
    ManFF:=Yes
ENDIF
```

We used this approach to test the IP5 mixed mode questionnaire and found it extremely useful as it enables testers to concentrate on the routing and question wording variants for individual modules, and minimises the time that they spend having to record answers to additional questions.

Summary

We used Prepare Directives to create different datamodels for each interviewing mode, and therefore could use common source code to ensure that the data collected in different modes is compatible. The advantage of this approach is that version control is much easier than having to update source in multiple places, and it minimises the resources needed to develop instruments for different modes as the same code is being re-used.

The questionnaire blocks are structured so that the fields are in a separate file from the rest of the block, which allows researchers to make minor working changes while programmers are working on rules and allows parallel working which is more efficient than serial working.

The key improvement to the development process was the introduction of a modular approach to development and testing. This meant that testing could start as soon as the first module has been completed, and that development and testing processes could progress in parallel which greatly reduces the total elapsed time to develop and test a complex multi-mode questionnaire instrument.

ONS Web Development Project: Tackling the Difficulties of Social Survey Data Collection

Lucy Fletcher, Office for National Statistics

1 Introduction

In the context of challenging UK public sector efficiency targets and the increasing cost of traditional survey data collection methods (such as face-to-face interviewing), there is considerable interest in developing a capability for internet data collection on the social surveys run by the Office for National Statistics (ONS).

ONS aims to be an innovative, cost effective and considerate (in terms of respondent burden) producer of official statistics. As such, it is placing increasing emphasis on developing mixed mode survey designs, including exploring the use of the internet for data collection, to improve efficiency and reduce the burden on households who respond to its surveys.

2 Background

The Labour Force Survey (LFS) is one of the UK's largest and most important social surveys and a key ONS statistical source for economic and labour market measures. Through face-to-face and telephone interviewing, it collects a wide variety of questions ranging from employment to health and income. In terms of time and complexity, it is probably one of the most burdensome surveys for respondents. However, the data collection process is also one of the most costly within the ONS Social Survey portfolio.

Due to the high costs and complexity, plus also increasing pressure on the office to be innovative with its approach to collecting and disseminating statistics, the LFS was an ideal candidate for a web survey pilot.

In 2010, Business Statistics Division (BSD) of ONS obtained a corporate licence for Conformat Professional, the web survey authoring software aimed at the market research sector. BSD collects data from businesses through paper questionnaires and telephone data entry, and was put under pressure to trial web based collection.

While BSD piloted an online version of their Capital Expenditure Survey, Social Survey Division (SSD) was able to share the licence, and also start piloting work. Although Blaise would have been the preferred software, as this is what is used for all ONS's social surveys, at the time there was no funding for setting up the necessary hardware for running live questionnaires. SSD carried out two LFS web pilots using Conformat, and further work is in progress using Blaise IS.

This paper will outline some of the piloting work already conducted, and discuss the current project to test the feasibility of adding a web collection mode to one of ONS's most complex surveys.

3 Completed LFS web pilot work

3.1 First LFS web pilot - 2011

In 2010 the Blaise Development and Support Team (BDSS) in Social Survey Division developed a questionnaire using Conformat for the first Labour Force Survey web pilot. With around 200 questions, this was a shortened version of the LFS questionnaire that is used in face-to-face and

telephone modes. Only the key topics were programmed, such as employment and earnings, and some questions and onscreen guidance were modified to make them clearer to understand.

In addition, the online version was only administered to one member of the household, whereas the normal LFS is administered to all household members aged 16 or over (in person or by proxy). We didn't manage to produce a workable household questionnaire in Confirmat where members could record their own details in sequence, so we proceeded with an individual-level questionnaire. The questionnaire did however include a short household section, collecting basic socio-demographic information about each household member (see screenshot 1), and their relationships to each other.

Office for National Statistics

Labour Force Survey
2010-11

HOUSEHOLD INFORMATION

Please fill in the Title, First and Last names of all members of your household.

If you do not want to enter a name please enter an initial.

	Title	First name	Last name	Male/Female
You	Mr	Bob	Smith	Male
Person 2	Mrs	Anne	Smith	Female
Person 3	Miss	Emma	Smith	Female
Person 4	Miss	Rosie	Smith	Female

Back | Next

Screenshot 1: Collecting demographic data in a Confirmat web interview

The main aims of the pilot were to examine the following:

- Respondents' ability to answer LFS questions, and complete a whole interview;
- Respondents' attitudes to responding to the online survey;
- Internet survey design and usability, including presentation of questions and answer lists.

1,424 respondents who had previously taken part in the LFS were invited to take participate. The fieldwork took place in early 2011, and a quarter of respondents completed the full survey. On average, the online questionnaire was completed in 18 minutes. A further cognitive exercise was carried out with 21 participants (from a different sampling frame) to gather further feedback about the online facility. These respondents were observed completing the questionnaire to assess how well they were able to navigate through it, understand the questions and provide meaningful answers.

In general, we found that both sets of respondents had no problems completing the online questionnaire, and said they would prefer this method in future to a face to face or telephone interview. They did struggle however with some existing LFS questions which they found confusing, despite being familiar with the LFS already. This demonstrates that we face a big challenge in adapting the questionnaire to suit a mixed mode environment, but without losing the meaning of questions and compromising data quality.

Blaise 4.8 Data Entry - D:\JM12 Version 8\LF0112

Forms Answer Navigate Options Help Show Watch Window

LF0112 Information_Sheet Household_Information Person[1] Person[2] Person[3] Person[4] Person[5] Person[6] Person[7] Person[8] Household_Reference_Person Benefit_unit Information

I would now like to ask how all the people in your household are related to each other.
 Code relationship of DAMIEN JONES to JAMES SMITH.
 Treat relatives of Civil Partners as though the Civil Partners were married.
 Also, treat cohabiting members of the household as though the cohabiting couple were married, including some sex couples.

1. Spouse
 2. Cohabiting partner
 3. Son/daughter (incl. adopted)
 4. Step-son/daughter
 5. Foster child
 6. Son-in-law/daughter-in-law
 7. Parent / Guardian
 8. Step-parent
 10. Foster parent
 11. Parent-in-law
 12. Brother/sister (incl. adopted)
 13. Step-brother/sister
 14. Foster brother/sister
 15. Brother/sister-in-law
 16. Grand-child
 17. Grand-parent
 18. Other relative
 19. Other non-relative
 20. Civil Partner

	Name	RelTxt	R[1]	R[2]	R[3]	R[4]	R[5]	R[6]	R[7]	R[8]	R[9]	R[10]	R[11]	R[12]	R[13]	R[14]	R[15]	R[16]
QRg[1]	JOHN SMITH	p1																
QRg[2]	ANNE SMITH	wife	1															
QRg[3]	JAMES SMITH	son	3	3														
QRg[4]	ROBERT SMITH	son	3	3	12													
QRg[5]	EMMA SMITH	daughter	3	3	12	12												
QRg[6]	ROSIE SMITH	daughter	3	3	12	12	12											
QRg[7]	DAMIEN JONES	nonr	19	19	19	19	19	19										
QRg[8]	VICTOR ROBERTS	nonr	19	19	19	19	19	19	19									

Screenshot 2: The household relationship grid is one of the challenges that lies ahead when adapting the interviewer-led instrument for a web mode

In order to develop a full online LFS, further investigation needed to be carried out to assess acceptable and optimal questionnaire lengths. As we know from this research that respondents are willing to fill out a 20 minute LFS questionnaire, how would they feel about 30 minutes or even 40 minutes? As the Conformat licence was still available, this requirement led to the development of a second pilot in 2012.

3.2 Second LFS web pilot - 2012

The main aim of the second LFS web pilot was to look at whether length of the questionnaire affected respondents' willingness to complete the questionnaire, and also investigate whether people would complete questionnaires for other members of the household.

The questionnaire instrument for the first pilot was replicated, and then extra sections of questions were added to increase the approximate length to 40 minutes. In terms of content it was almost as long as the interviewer-led LFS but without some of the ad-hoc modules.

30 respondents were assigned to three different questionnaire lengths – 20 minutes, 30 minutes, and 40 minutes. Those in the 20 minute group were asked the same questions from the original pilot, while for the other two groups, routing was added to the extra questions to provide an approximate 30 minute interview, and a 40 minute interview with the full set of questions.

All were familiar with ONS's social survey questionnaires having agreed to take part in future work, but none had taken part in the LFS. Interviews took place in the respondents' homes using their own computers and the respondents were required to complete the survey on their own with no guidance

from the interviewer. The interviewer observed how the respondent completed the survey and a cognitive interview was conducted once the respondent had finished.

There was a mixed reaction when respondents were asked to provide feedback regarding the length of the questionnaire they had completed. Whilst respondents were generally positive in terms of the 30 minute interview, they did at the same time suggest that 30 minutes was probably the maximum length of interview that they would be willing to complete. For the 40 minute interview the reaction was much more varied, however with many of the participants stating that they thought the questionnaire was too long.

There were also mixed feelings when respondents were asked whether they would be happy to complete the information on behalf of other household members. Those who were willing to spend time completing this information said they wanted to help where possible, and were confident that they could answer the questions in reference to family members. Those who were not willing to provide information suggested reasons of confidentiality, time spent completing the questionnaire and insufficient knowledge about other household members.

Both pilots provided many useful recommendations to inform future work, particularly in terms of questionnaire development and length. Conformat Professional was useful in providing a fairly quick and easy way to set up and host a web survey, but the conclusion drawn was that it didn't fully meet the needs of Social Survey Division for complex household surveys and integrating with current systems. The best way forward was to spend time developing a prototype LFS household questionnaire in Blaise IS which would be ready to go live when the hardware was put in place.

4 Current work

4.1 LFS household web prototype (Blaise IS)

In 2011, the BDSS team started to program a web version of the LFS using Blaise IS 4.8.1.

This time we made it more similar to the interviewer-led LFS in comparison to the Conformat pilot by programming it as a household survey. A tab will appear for each household member, which will jump to his or her personal questionnaire. Each respondent can fill in their own information, or one respondent can do this for themselves and then the others in the household. As with the face-to-face and telephone modes, it is possible to state that the questionnaire is being completed by someone else (by proxy; see screenshot 3) and appropriate textfills will then be used in the question text.

Labour Force Survey
Blaise Pilot 2011

Main Person 1 Person 2 Person 3 Person 4

Questions for person 1: Bob Smith

Please click the person who is answering the questions for Bob Smith.

Children under 16 are not listed, so please choose another person to fill out the answers.

- Bob Smith
- Anne Smith
- Emma Smith
- Rosie Smith
- Somebody outside the household

Office for National Statistics

Next >>

CONTENTS

- Background information
- Work schemes
- Employment
- Place of work
- Hours worked
- Working patterns
- Benefits
- Earnings

Screenshot 3: Collecting data from all members of the household

Once each person has attempted their questions (either in person or by proxy) the household selection screen (screenshot 4) displays the status of each person and allows the selection of another person, or submission of the whole questionnaire.

Labour Force Survey
Blaise Pilot 2011

Main Person 1 Person 2 Person 3 Person 4

Household overview

Click on a name to enter the data for that person.

(1) [BOB SMITH](#) Completed ✓

(2) [ANNE SMITH](#) Not completed

(3) [EMMA SMITH](#)

(4) [ROSIE SMITH](#)

Office for National Statistics

<< Back Next >>

Screenshot 4: Household member selection screen with status

We have not yet been able to pilot this work as we do not have the hardware in place to host externally, so the questionnaire is currently hosted on the ONS intranet for internal testing. Until we obtain a hosting solution, we are still working on the questionnaire so it will be ready to go live in the future.

As the LFS is a very lengthy and complex questionnaire, there is still a lot more work to do to make it user-friendly and engaging for respondents. As there will be no interviewer to guide respondents through the questionnaire, the wording of questions, placement and wording of checks and onscreen guidance will need to be revised. Also, the LFS uses many coding frames, and so far we have only tested out the shorter, simpler ones such as the country coding list for country of birth.

As well as visual improvements, we are concerned with ethical considerations such as the situation where more than one household member is interviewed within the same questionnaire, and individual responses may be visible to other household members. Also, following a household's first interview, a lot of its data will be 'rotated' into the next wave and used in question text, response options and questionnaire checks to cut down on interview time and decrease the burden on the respondent. As well as the ethical issues this may cause, it also raises security concerns as personal details are being stored on an external server.

The earlier internet piloting work focussed on the survey stages up to and including collecting data from respondents. The data processing stage was outside the scope of the pilots. Work is taking place to identify the requirements and agree the procedures to process data from an online mode and integrate it with data from the face-to-face and telephone modes. This is further complicated by the inflexible, aging systems in place that currently collect and process LFS data.

4.2 Electronic Data Collection project

4.2.1 Background

Towards the end of 2011, a project was set up in Business Statistics Division (BSD) of ONS to look at modernising ways of collecting data from businesses. Currently, this division collects data from businesses largely from paper questionnaires, or for the simpler questionnaires, telephone data entry (TDE). The systems that have been built up to support this method are antiquated and expensive to maintain. Paper questionnaires place an unnecessary burden on respondents, and take time to dispatch, collect and process. Responding to ONS's business surveys is a legal requirement under the Statistics of Trade Act (1947), so the office is facing increasing pressure from businesses to modernise the data collection process.

The Electronic Data Collection project (EDC) was set up to address these concerns, with the specific aim of delivering web based business data collection through a web-based portal. The definition of portal for the purposes of this work is:

“a web site that functions as a point of access to information in the World Wide Web. A portal presents information from diverse sources in a unified way...Portals provide a way for enterprises to provide a consistent look and feel with access control and procedures for multiple applications and databases, which otherwise would have been different entities altogether.” (Wikipedia, 2012).

It became clear early on that the aims of BSD and the EDC project were similar to those of Social Survey Division, and so both divisions started to work together with the help of specialist contractors to ensure that the project delivered a solution that met the needs of the two areas.

The vision is to provide a common and secure web data collection platform, extendable to delivering a variety of secure data collection and messaging services to ONS's business and household respondents (and potentially for consulting users of statistics), that integrates securely with ONS's back-end systems where much of the complexity lies.

4.2.2 Liferay and Blaise: 'Proof of concept' phase

The first phase of the project started in November 2011 and was completed in March 2012. The aim was to provide a 'proof of concept' to demonstrate how web data collection might operate within ONS, while meeting the list of requirements from both work areas. This phase did not include actual data collection, though the aim is to demonstrate this in the next phase.

Various software packages were assessed which provide a web-based portal, and Liferay 6.1 (Community Edition) was selected by the contractors as the package they would use to demonstrate the proof of concept.

The Proof of Concept Report (2012) describes the features of Liferay:

“Liferay Portal is a free and open source enterprise portal written in Java and distributed under a GNU Lesser General Public License and a number of proprietary licenses. Liferay has been design for delivery of intranets and extranets, for organisations of any size. Liferay Portal allows users to set up features common to most websites without requiring large amounts of custom development. Liferay is constructed from functional units called portlets assembled in a content management framework and web application framework. Liferay has achieved a solid track record across a number of industries, providing real world performance with for intranets and extranets across verticals. In the context of the electronic data collection project Liferay offers a range of building blocks with the capability to accelerate development and deployment to meet the objectives of the programme in both the business and social contexts.”

The main purpose of the portal was to deliver these functions:

- **Registration** – Allow a user to register using a unique identifier and password. Following successful authentication, a user will be able to confirm his or her company/personal details and create a unique password and user id as well as entering required information such as email address. This email address will be used to activate a user’s account.
- **Login** – Handle the day to day authentication and authorisation of registered users.
- **Portfolio management** - Allow a user to view all completed, in progress and ‘not started’ surveys for that company/user.
- **Web survey integration** - On selection of a questionnaire or following a link to a questionnaire, the portal will call a specific web survey application.

Although both divisions used Confirmit Professional for early pilot work, Blaise IS 4.8.2 (and later, 4.8.3) was chosen as the web survey application to integrate with the Liferay portal. The main reasons for this were functionality, as we know Blaise IS can produce the complex surveys required; financial, in that ONS already has Blaise licences; and support, with many expert Blaise users plus the available help from Statistics Netherlands. For Social Survey Division, it makes consistency across modes as we already produce Blaise datasets for telephone and face-to-face. The portal can however work with a variety of applications, so in the future, BSD could use a different web survey package, and SSD could make the transition to Blaise 5.

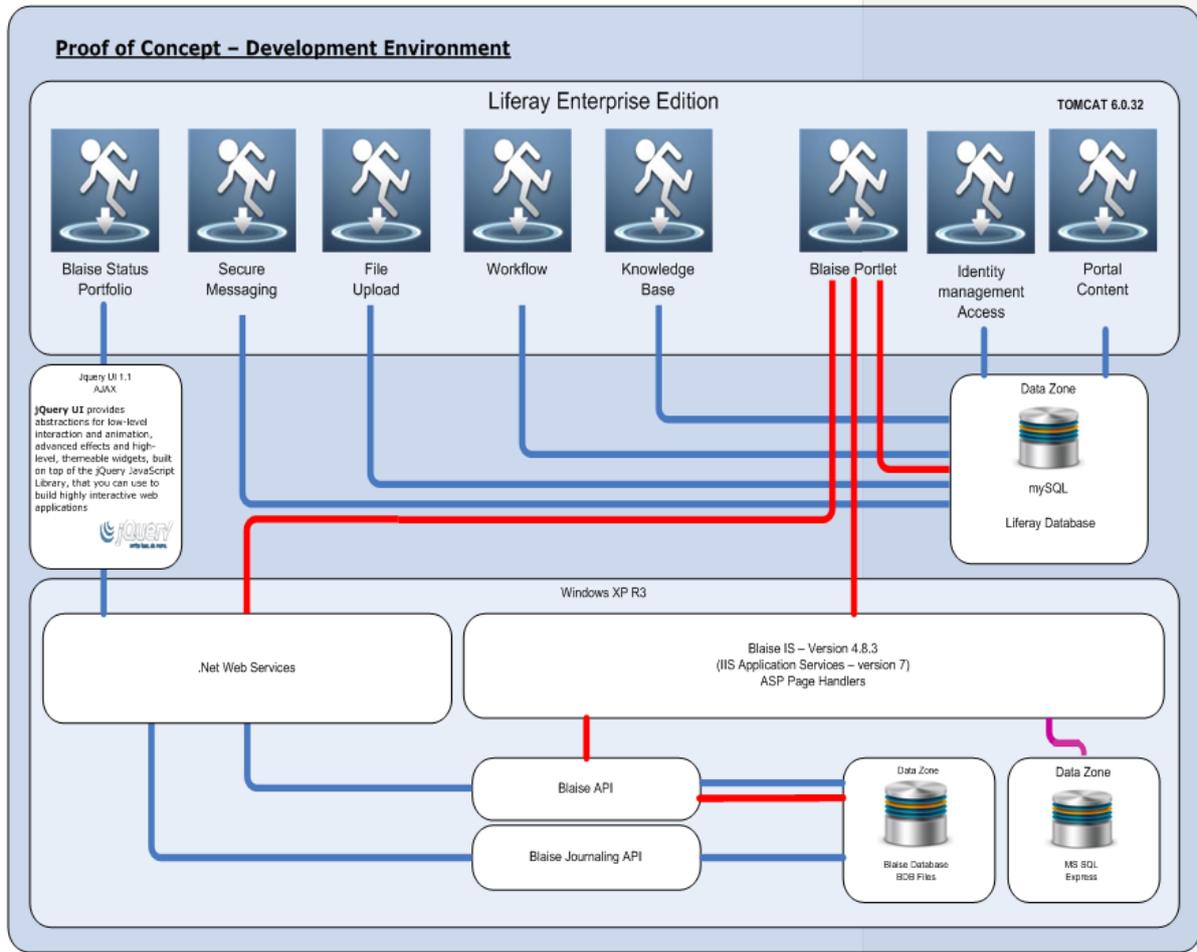
The Blaise team in SSD produced an online version of BSD’s Capital Expenditure Survey (Capex), which was the first attempt to program a business survey using Blaise. The team also provided the code for the LFS prototype web questionnaire. The two surveys were then integrated into Liferay.

Some work was necessary to try and pass external values into Blaise. To test this, we amended the BiInterviewStarter.asp page using hard coded values. We input a variety of different values such as language, currency and business reference number into Blaise (see screenshot 5), and this allowed us to control who viewed certain questions in the routing, ensuring that respondents were viewing relevant questions and information. Eventually the API was used instead to dynamically pass values between Liferay and Blaise. Both methods allowed us to change the values within Blaise, so for example a respondent could select Welsh as their language of choice when they register an account on Liferay, but change it to English part way through the questionnaire. This preference would pass back to Liferay and then be remembered next time they log in.

```
Call AddField(xmlDoc, FieldsNode, "QBParameters.Lang", "ENG", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.Currency", "GBP", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QID.RefNum", "49900123456", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.Respondent", "1", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].Q1Aquvalid", "Yes", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].Q2Aquvalid", "Yes", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].Q3aAquvalid", "Yes", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].Q3bAquvalid", "Yes", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].Q4Aquvalid", "Yes", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].TotAquvalid", "Yes", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].Q2Disvalid", "No", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].Q3aDisvalid", "No", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].Q3bDisvalid", "No", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].Q4Disvalid", "No", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [1].TotDisvalid", "No", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [2].Q1Aquvalid", "No", biResponse, "")
Call AddField(xmlDoc, FieldsNode, "QBParameters.QvalidQues [2].Q2Aquvalid", "No", biResponse, "")
```

Screenshot 5: Modification to BiInterviewStarter.asp with hard-coded values

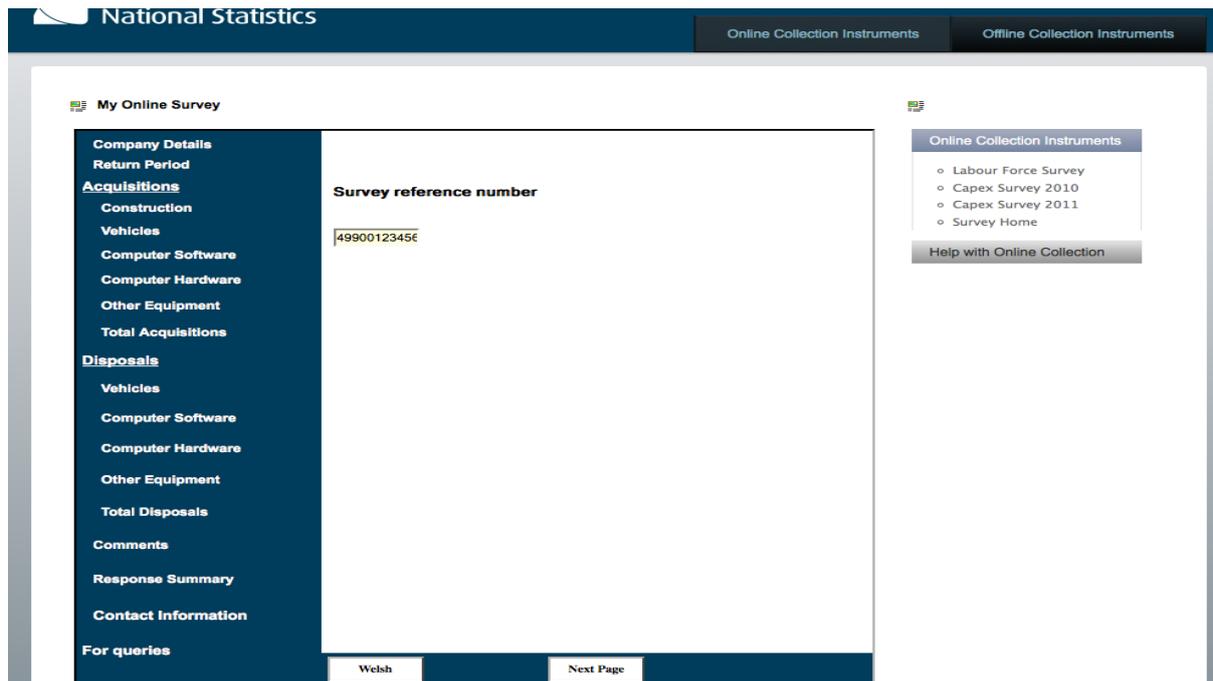
Screenshot 6 shows the development environment which was set up for the proof of concept:



Screenshot 6: Proof of concept development environment

Following the setup of the Liferay portal, a series of stakeholder demonstrations were held, showing how the proposed solution would work for both business and social surveys. Using two different scenarios for a business survey respondent and a social survey respondent, the demonstrations showed the process of registration through Liferay (where the user can create a username and password), then logging in to a survey, and completing the survey.

Screenshot 7 demonstrates the Blaise IS Capex survey running within the portal:



Screenshot 7: Blaise IS running the Capex survey through the web portal

The proof of concept exercise proved that Liferay integrates well with Blaise IS to produce a solution that will help ONS make a step forward in modernising its data collection processes. A series of performance test are being conducted (results not available at the time of writing).

Although Blaise IS appears to function well with Liferay, the portal allows different packages to be slotted in alongside or instead of Blaise if necessary. It can be updated and programmed without any coding knowledge, making it easy to maintain.

The portal has other features that will be useful to engage with respondents such as news feeds and secure messaging, plus the ability to show relevant information based on respondent id such as when they need to complete surveys and relevant news items. These features may also benefit other areas of ONS, and not just the social and business data collection areas.

5 Thoughts about using Blaise IS

Before designing the LFS prototype questionnaire, the BDSS team has little experience in using Blaise IS. The team drew upon the samples in the Blaise examples folder and training course materials which were helpful. Some aspects of the work were more challenging, in particular layouts and journaling.

5.1 Layouts

We found that the layout options were not user-friendly. The questionnaire must look professional, and enable the respondent to fill in their details accurately and swiftly, maximising response and minimising error. While it was fairly straightforward (although time-consuming) to transfer the LFS questions into the web version, the layouts were fiddly, and the results were sometimes down to trial and error and much recompiling! We struggled in particular with custom edits and grouping.

5.2 Journaling

One of the tasks for the EDC project was to set up a system of collecting paradata. Blaise IS has a journaling feature which allows paradata to be collected, such as browser type, length of time spent on each page, interview status, and so on. In the short amount of time we had to look at this feature, we managed to demonstrate that we could collect some basic information, but not as much as much as what was specified by the project. This would have involved tweaking ASP pages and style sheets, and there was no technical expertise in the team to enable us to amend these in any depth. Liferay can also collect paradata and it looks likely that a final solution would involve either just using Liferay, or using a combination of the two.

6 Future plans

Much of the future work around electronic data collection is reliant on funding, and obtaining funding is becoming increasingly difficult due to government efficiency targets, despite the obvious gains that would be made in the long term. Assuming that ONS secures funding to make further progress with electronic data collection, then there are a few areas that we plan to explore.

6.1 Develop the Blaise LFS prototype

BDSS plans to develop the Blaise LFS prototype based on the recommendations from the two LFS pilots. Ideally, this would be integrated into Liferay for a full-scale pilot.

6.2 Offline electronic data collection

ONS hopes to be able to offer an offline form of electronic data collection, for example spreadsheets or e-questionnaires, which respondents can complete offline, but are still received and sent via the web portal. BDSS have just started looking at the BASIL component of Blaise as a possible solution.

6.3 Blaise training needs

At the moment, there is only one Blaise support team in ONS, and this sits within Social Survey Division providing support to social survey researchers. If Blaise IS becomes the future tool for electronic data collection, then it will be vital to consider the training needs within both divisions. The support team will face the challenge of not only helping to set up many different surveys, but also training one division in how to use Blaise, and another (who already have Blaise skills) in how to use the IS component.

6.4 Develop standards for web surveys

ONS social surveys currently have a series of standards in place to ensure consistency across surveys. These standards cover both programming and screen standards, for both telephone and face-to-face modes. A similar set of standards (particularly relating to display of questions, answer lists, checks and guidance) will need to be devised for web surveys to ensure consistency.

7 References

Aubrey-Smith, S.A. and Thatcher, B (2012). *Labour Force Survey Online Pilot 2012: Usability/Cognitive testing report*. Office for National Statistics, Newport (unpublished internal document).

Burgis, A. and Hamblin, J. (2012). *Proof of Concept Report: Electronic Business Data Collection and Social Surveys Internet Data Collection*. Office for National Statistics, Newport (unpublished internal document).

Portanti, M (2011). *Results and recommendations from the 2010/11 internet pilots: Summary report*. Office for National Statistics, Newport (unpublished internal document).

Wikipedia (2012). *Web portal*. http://en.wikipedia.org/wiki/Web_portal. [Accessed 9 Mar 2012].

Web CATI (Part of NatCen’s Multi-Mode Approach)

Peyman Damestani and Maya Agur, NatCen Social Research

1 Introduction

NatCen Social Research has been developing its internal capabilities in offering a multi-mode solution to its clients. The rationale for using multi-mode solutions in surveys is to maintain high response rates while maximising cost-effectiveness. This paper focuses on the combination of CATI and CAWI using both Blaise and BlaiseIS survey processing platforms. We will focus on one case study and discuss the reasons to why we have adapted this method of data collection. We also demonstrate a method of switching from Blaise Data Entry Program (DEP) to a BlaiseIS Web survey during an interview session.

This paper is a reflection of the experience of conducting NatCen’s first WebCATI multi-mode project – The Evaluation of Children’s Centres in England (ECCE).

1.1 What is WebCATI at NatCen?

NatCen’s WebCATI is a dynamic and interactive survey processing environment that utilizes Blaise CATI system for sample management and BlaiseIS for data collection.

1.2 How does it work?

In our WebCATI environment, Blaise DEP calls Internet explorer and passes control of the survey content to BlaiseIS. Once the web survey data collection process is completed by the interviewer, the process returns the control to DEP for completion. During the data collection process in BlaiseIS, DEP locks the active interview to prevent access for other interviewers.

1.3 WebCATI Server Environment

NatCen has a Multi-Mode Unit that consists of 40 telephone workstations which are connected to a dedicated CATI server running Blaise services over a Local Area Network (LAN). Our web surveys are hosted on a remote server. The main benefit of acquiring a remote web server facility is that it provides cost-effective around the clock support.

2 The Evaluation of Children’s Centres in England (ECCE) – WebCATI case study

NatCen Social Research in collaboration with the University of Oxford and Frontier Economics has been commissioned by the Department for Education (DfE) to carry out a six year evaluation of children’s centres. Children’s centres are intended to be accessible, welcoming places where children under five years old and their families can receive integrated, good quality services and find any information or advice they might need. Since 2010 every community has a Children’s Centre. The aim of the children’s centres programme is to improve developmental outcomes for all children and to reduce inequalities in outcomes between the most disadvantaged children and the rest.

The ECCE study will provide an in-depth understanding of the effectiveness of different children’s centre models and will offer powerful and wide ranging evidence on the best ways to support families and children. The evaluation includes a set of surveys:

- A survey of children’s centres
- A panel survey of families using the centres
- A survey of families who live in the catchment area of the centres
- In-depth interviews and visits at centres

- A cost-benefit analysis

The survey of children's centres was the first survey to be carried out as part of the ECCE, and was designed as a WebCATI survey.

2.1 Why WebCATI Multi Mode Approach?

The discussion around benefits and effects of different modes in research has always been central in the process of designing the methodology of surveys. For NatCen, the rationale for using a multi-mode design of combining web and CATI tools in surveys is fourfold:

- Maximise cost-effectiveness (since web surveys are relatively cheap to administer)
- Maintain high response rates (since telephone interviews tend to have higher response rates than web self-completion);
- Ensure population coverage (since a substantial proportion of the population may not be able to complete a web survey but will be approachable by telephone); and
- Offer flexibility to respondents in engaging with the survey.

The ECCE led with a web self-completion questionnaire and offered a telephone option for non-respondents. In addition, a telephone interviewer contacted respondents to the web questionnaire to follow-up on unanswered questions.

An important early design decision was to use the same tool for both the web and the telephone survey, using BlaiseIS. The rationale behind this was to ensure that the same questions were asked with both modes so as to ensure consistency in content. Furthermore, the use of a single instrument helped to facilitate the production of a single dataset, as opposed to separate ones for each mode. Having just one dataset saved on time and costs of data management as there was no need of editing and reconciliation between datasets.

The questions in the interview were designed so that they could be delivered orally as well as visually. Special focus was given at the interviewer briefing to the questions that were most complicated for oral administration. Interviewers were encouraged to practice reading these questions aloud, and practice the suggested adjustments, so that all questions were conveyed in a consistent and clear way over the telephone.

Given that the questionnaire was primarily intended for self-completion, it was decided to build the program in such a way that the respondent could choose to skip questions (rather than requiring them to complete each question before being allowed to move on). It was also possible to complete the questionnaire on more than one sitting and navigation bars enabled respondents to move around the questionnaire. The progress of the respondents in completing the questionnaire was monitored automatically. Key cut-off points throughout the questionnaire were programmed so that when a respondent was allocated to telephone, the interviewer would be able to see the level of completion.

Access to the web instrument for the telephone interviewers was through the CATI management system, to allow the interviewers to manage their assignments as they would normally do in a CATI survey. Before calling up the web instrument, the interviewer was taken through a short CATI questionnaire to ascertain whether the respondent wished to complete the questionnaire online or as a telephone interview and whether an appointment was needed to continue filling in the questionnaire. When contact was made by telephone interviewers, the introductory text in the short CATI questionnaire was tailored to the respondent's level of completion of the survey.

3 WebCATI Questionnaire Development

The questionnaire for the Evaluation of Children's Centres in England was initially developed as a CATI questionnaire and tested with the Blaise DEP to enable a faster method of testing routings and textfills. It was then further developed as a web instrument in BlaiseIS where we made sure that

layout and behaviour of the survey functioned correctly for both web and CATI modes. Our focus was mainly on questionnaire usability and accessibility for both respondents and interviewers.

3.1 The WebCATI System Process

A Blaise CATI instrument was implemented to regularly update sample from the web database. This information was then presented to the interviewer via the CATI dial screen. The interviewer could then check the web status outcome code prior to calling the respondent.

3.2 Passing parameters from Blaise DEP to BlaiseIS

Once the respondent agreed to participate in the survey, a procedure was executed from DEP to call the Internet browser and pass parameters such as login details from the CATI sample to the BlaiseIS instrument based on indicative value (CATIX) (Figure 1).

Figure 1: Procedure for passing parameters from DEP to BlaiseIS

```
PROCEDURE RunWeb
PARAMETERS
  IMPORT pStart : STRING
  ALIEN('WebProcs.msu', 'doAll')
ENDPROCEDURE

FIELDS
StartWeb "Do you want to start " : STRING[100]
ShowWeb "INTERVIEWER: DO YOU WANT TO DO THE WEB SURVEY NOW
StartWeb "CODE 'Yes' IF YOU WANT TO ACCESS THE WEB QUESTIONNAIRE": (Yes, No, Done "")
RULES
StartWeb.Show
ShowWeb
IF (ShowWeb = Yes) THEN
StartWeb := 'http://www.myserver.co.uk:80/project/survey.asp?KeyValue=' + {passw} + '&CATIX=1'
  Runweb(StartWeb)
  Showweb := Done
ENDIF
```

The key value (passw) and the field representing the mode (CATIX) were passed to the web questionnaire in a query string in a command line constructed in the CATI questionnaire. The command line was then passed to a Manipula script which we called from the CATI questionnaire as an alien procedure. The Manipula script ran the browser and navigated to the web questionnaire.

The interview starter (BiInterviewStarter.asp) for the web questionnaire retrieved the value from the query string and stored this into the XML stream so it could be used for routing. The value in the field was used to determine whether the questionnaire was being completed by the respondent or by a telephone unit interviewer.

Figure 2 shows the interviewer view of the screen in CATI, before calling on the web survey.

Figure 2: Interviewer view of DEP screen



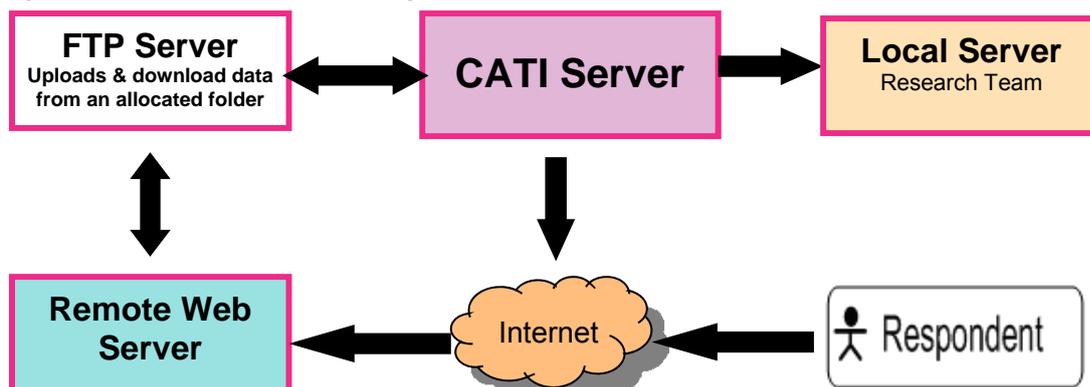
3.3 DEP and BlaiseIS locking mechanism issues

Early on at the start of the telephone fieldwork, we came across a problem relating to interview management. After the web questionnaire was opened we noticed that the CATI form was released and became available to the call scheduler within the current daybatch after 10 minutes. This was because we called the browser with the wait parameter. The solution was to omit the wait parameter. The disadvantage of this was that it was possible to tab out of internet explorer, back into the DEP to complete the administration on a serial number before completing the interview in IE, however this was outweighed by the advantage that the record lock persisted throughout interviews longer than 10 minutes. This risk could be tackled with interviewer training and instructions at briefings.

3.4 Automating WebCATI Sample Process

The ECCE had its own folder structure on the remote web server. This structure contained a VB script that executed a Blaise Manipula program to convert the Blaise data to ASCII, and upon completion encrypted the data into a zip file and transferred it to the FTP directory. Web survey data was transferred to the CATI Server using a secure FTP server/system (FIPS 140-2 compliant). A Manipula script updated the current web status of each case in the CATI sample. This script was scheduled to update CATI sample hourly and then other processes were employed to generate progress reports and interim datasets for MMU.

Figure 3: Servers and Data Flow Diagram



4 Challenges

Teams at NatCen were faced with different challenges in different surveys working with NatCen's WebCATI.

4.1 Post Launch Program Changes

Although quite rare, post launch changes to a program may be sometimes required. While these sorts of amendments are a fairly simple and quick task in a CATI environment, these can be quite complicated in a web survey. When a survey is launched and respondents are made aware that the survey is available to them, the only option to make amendments is to block access to respondents while the survey is taken offline. It is difficult to anticipate when a respondent might be accessing the questionnaire, as we would not want to cut off people in the middle of filling in the questionnaire, and risk losing the information they had entered. In this respect, a web survey is less flexible than CATI.

4.2 Handling Question Text for CATI and Web

Using the same instrument for both modes posed a challenge in terms of question wording and layout design. In the ECCE we used textfills in the body of the question when wording tweaks were made between the self-administered and interviewer-administered mode. However, those textfills had an impact on the question layout, especially when interviewer instructions were added.

In later surveys we decided to handle wording tweaks using the Blaise Language mapping feature.

Intro

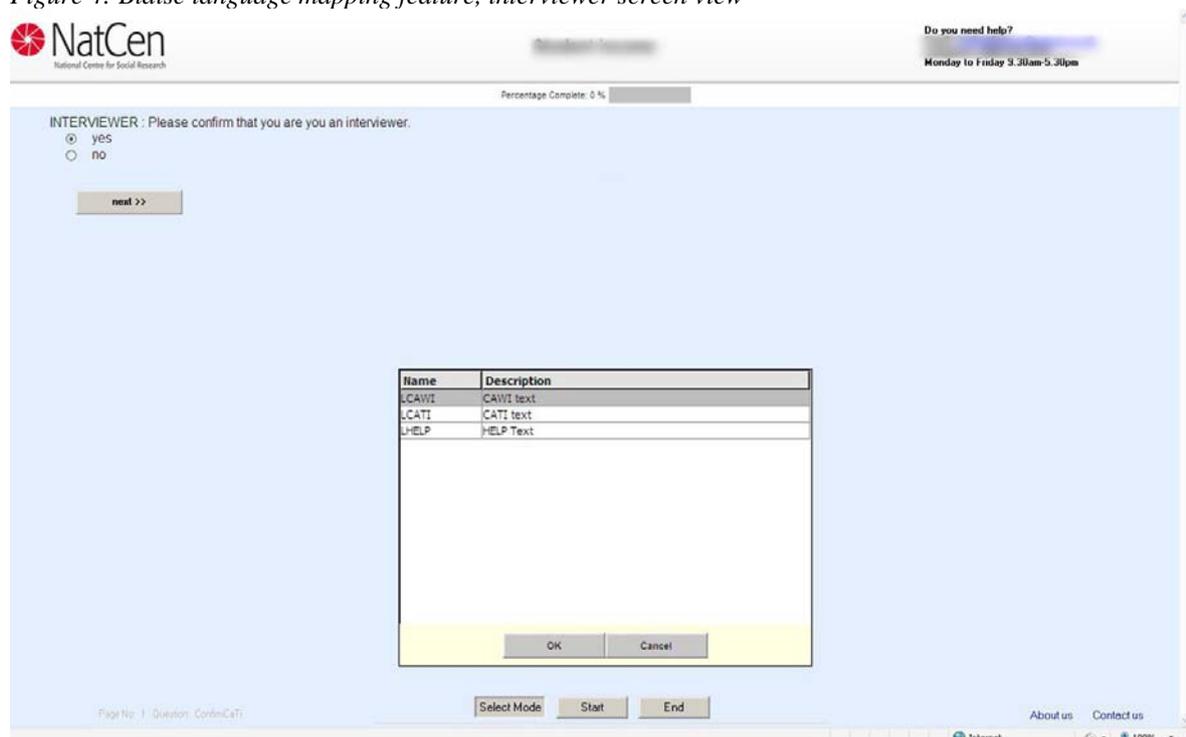
"Now follows some questions about your local area."

"Now I'd like to ask you about your local area."

: TQNoAnswer

This option was only made available for the interviewer, so they could select 'CATI' option to change the wording to read more fluently for an interviewer-administered interview. This ensured consistency across interviewers and simplified the process of adapting the wording of the questionnaire to the different modes (Figure 4).

Figure 4: Blaise language mapping feature, interviewer screen view



4.3 Issues working with a remote Web Server

As the data was held on a remote server it was not instantly available to interviewers. We therefore needed to schedule a process to download data on an hourly basis. This delay was acceptable in the projects we have done so far.

We also discovered that a live server environment was not ideal for testing other surveys. When carrying out the testing we sometimes had errors installing or removing the server manager. Our IT infrastructure team has addressed these issues and configured a local test web server.

4.4 Data Security Issues

Blaise database (.bdb) is not encrypted and secured and so standard FTP was not a safe option for transferring Blaise data to our internal network. NatCen has strict data security rules for sending and receiving data to the remote web server. Currently the following are the only acceptable methods:

- PGP encryption (a higher standard of encryption required by some clients)
- File transfer over secure electronic connections

4.5 CATI Management Summary Screens

Supervisors at the multi-mode unit monitored WebCATI progress through Blaise CATI case Management (btmana.exe) screens. We updated CATI sample with data received from the Web surveys on an hourly basis. Supervisors and interviewers could see updated information via the dial screen or the overview screens. However, we had to address the issue with stats in the CATI summary reports, as they were recording CATI call outcomes only.

4.6 Collecting Paradata information (Using Journal instead of Audit trail)

BlaiseIS does not produce the same audit trail information as normal CATI. There is some information that is captured in a journal file such as time spent on a page. Our aim would be to capture more paradata in the future.

4.7 Training and usability issues for CATI interviewers

When designing a WebCATI survey the need for interviewer training in carrying out the survey must be carefully considered. In addition, the program needs to be easy to use by the interviewers so that they can manage their assignment effectively and access the questionnaire without difficulties. The approach of using BlaiseIS to set up the program for WebCATI surveys proved to be the best solution for both interviewer training and usability.

NatCen telephone interviewers are very experienced using the Blaise CATI system, so choosing BlaiseIS meant that after some initial training they became familiar with the web interface. This lessened the burden of learning new systems or ways of working and enabled continuation of regular practices. The Blaise CATI frontend enabled cases to be managed within the CATI Management system in the same way that any other CATI sample was managed. It also enabled interviewers to move seamlessly into the web questionnaire when conducting an interview over the phone. For the telephone interviewers this meant that it was easy to adapt to the new system.

5 Conclusion

The WebCATI multi-mode instrument is a relatively new development in NatCen, which has been in use in a number of surveys over the past year. Although it is still a learning and development process for all involved, it is agreed that the solution of WebCATI multimode instrument in surveys, particularly in this age of tighter budgets for research, provides an effective method of data collection. We plan to continue to develop it in future surveys.

There were some challenges in working with the WebCATI methodology and some disadvantages. One disadvantage was longer development time. It was necessary for the questionnaire to work as both self-completion and interviewer administered survey, so much consideration needed to be given

to question wording (to account for the different modes) and the question layout. Therefore, question design, programming and testing took longer than usual. In addition, due to the survey being available online for the respondents to fill in at any time at their on convenience, there was a need for around-the-clock technical support to the survey. At the moment this is not something that is manageable for NatCen to provide in-house and so an external solution was adapted. Finally, since data is stored in a remote server in the first instance, monitoring fieldwork progress was challenging and required a third party application to produce daily progress reports.

However, it is agreed by all involved that the advantages of the WebCATI approach outweigh the disadvantages. One of the main benefits of the WebCATI approach is the cost effectiveness of the tool. It allows respondents who are able and willing to complete the survey online to do so, thus saving the cost of telephone interviewing. In addition, respondents to the surveys benefited from having the choice of mode. Quite a few respondents preferred filling in the surveys online, having the flexibility to stop and continue at their convenience, while others expressed preference to complete the survey over the phone. We also believe it is likely that the provision of this choice of mode increased overall response rates. Finally using BlaiseIS in WebCATI surveys mean that there was a need to develop only one tool, rather than two (one for each mode) and that the data was collected in just one dataset, so there was not need to edit or reconcile data between modes.

Acknowledgements

We would like to thank our colleagues at NatCen Social Research who have contributed to the writing of this paper and helped us with their comments and insights on the development of WebCATI in NatCen: Alessio Fiacco, Colin Miceli, Richard Hall, Emma Drever, Chloe Robinson, Chris Massett, Emily Tanner and Steven Finch.

Blaise Multiple Contact Interface for Telephone Interviewing on a Complex Household Survey

Nafiis Boodhumeah, NatCen Social Research

1 Introduction.

This paper will describe the challenges that we faced in developing a Blaise multiple contact interface in order to deliver the Understanding Society study in NatCen Social Research's Multi Mode Unit (MMU) and the solutions we implemented. It will start with some background information about the Understanding Society study and NatCen's computer-assisted telephone interviewing (CATI) resources. It will then discuss the challenges faced in developing this multiple contact interface and describe the dial screen solution we arrived at. Finally, it will reflect on the implications of the solution for interviewer training and assess its success in delivering this multiple contact requirement.

1.1 Background to Understanding Society.

Understanding Society is the world's largest longitudinal social science study, interviewing around 100,000 respondents each year. It was launched in 2009, and is already delivering high-quality, world-leading data on the social and economic circumstances of households from across the UK. The study is funded by the Economic and Social Research Council, with scientific leadership provided by the Institute of Social and Economic Research (ISER) at the University of Essex.

A nationally representative sample of 40,000 households was selected in 2009. Interviewers now contact each of these households every year and interview all members of the household. Adults are interviewed either face-to-face or over the phone. Respondents aged 10-15 fill in a paper self-completion questionnaire. It is arguably one of the most technically demanding social science studies in the world today.

In 2010 approximately 500 households were issued for CATI interview. These were households which had previously indicated that they would prefer to be contacted by telephone.

Understanding Society requires the whole sequence of both successful and unsuccessful telephone contact attempts with all household members to be recorded, capturing a complete call history for each household. The outcome of the attempt, who was spoken to and which telephone number was attempted, had to be recorded for each call. In addition, where sample members could not be contacted, attempts at tracing them, including contacting the stable address contact, were documented.

1.2 The Multi Mode Unit (MMU) and Project Computing at NatCen

The Multi Mode Unit (MMU) is situated in our Brentwood office and carries out CATI interviewing. It consists of 5 staff members (a Manager and 4 Project Supervisors) and a panel of 64 freelancers interviewers.

The CATI dial screen design and implementation was carried out by the Project Computing department at NatCen. The Project Computing department are responsible for the smooth running of surveys and also for any project specific survey systems needed in order to deliver the survey. Blaise 4.8.2 is used in the MMU. After evaluating the survey requirements, taking into account time constraints, interviewer training and cost we decided to keep the Blaise CATI call scheduler framework and develop our own dial screen to meet the survey requirements.

2 Design challenges we faced

2.1 Displaying and updating contact information.

The client's survey requirements were for us to contact and obtain an interview with up to 16 people in the household. Each person could have up to 5 phone number associated with them. The standard Blaise CATI dial screen would require a long list of contact fields to be displayed and although these could be updated this would become very difficult for interviewers to work with in a live interviewing situation. Furthermore, as interviewers needed to contact and interview all people in household in many cases it would not be possible to achieve all the interviewing through a single call. So dynamic updating of the dial screen to inform interviewers of current status of the case was needed.

Provision also needed to be made for following up individuals who had moved from the household since last interview, splitting the case, and transferring details from the original household to a new household created for those who had moved. To do this effectively we used Manipula to create the new households and updated contact information.

2.2 Keeping a complete call history of each phone number dialed and who answered the call.

The client also required us to keep a complete call history of each number dialed (including any new numbers discovered by interviewers) and to record the person that answered any call (this would in most cases be a household member but could also be a "new" person). This information could also then be displayed to interviewers so that they can review the dial history of numbers. We also needed to record when the phone had been transferred from one household member to another to do the interview.

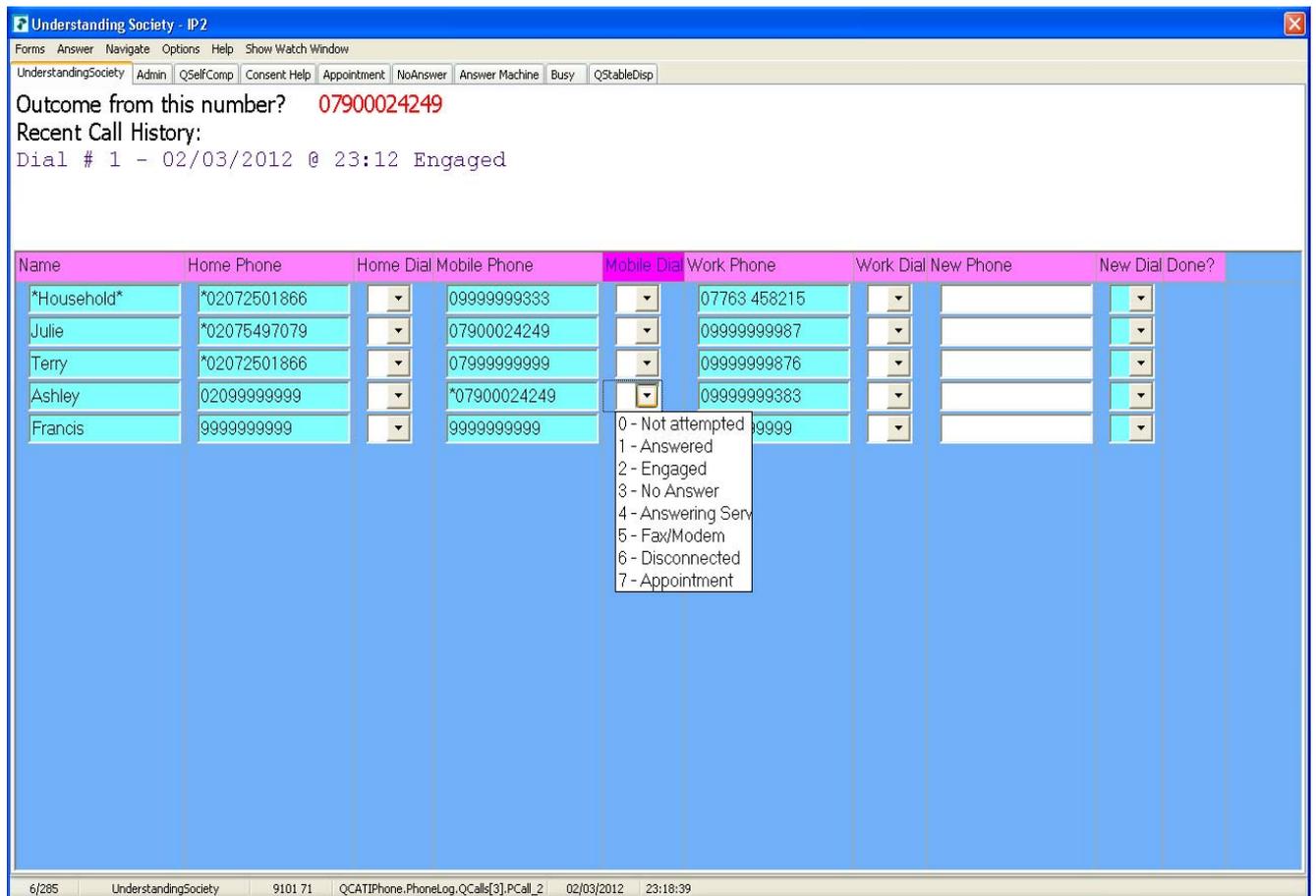
2.3 Ensuring usability for interviewers.

It was important to provide interviewers with information so that they could have a quick and accurate snapshot of the current status of the household. As it was possible to contact more than one person in the household at a time interviewers could also potentially make appointments for different people in the household. These appointments needed to be ordered by date and time and displayed so that interviewers could then make the next appointment via the standard CATI appointment function.

2.4 Initial design and lessons learned.

The design challenges described in above led us to the conclusion that we would need a multiple contact dial screen block at the start of the Blaise instrument.

We initially developed a front-end grid-type dial screen block in which all the household members names and contact phone numbers were displayed in a grid as the example below (which uses dummy sample information) shows.



Interviewers could dial numbers from the grid and record the dial result. The time, date and dial result would then be stored and displayed for future reference when the number was clicked on in the grid. Once contact was made with a household member the interviewer could enter the main questionnaire and start the house hold grid enumeration. We found that this method was unsuccessful as:

- When dialing a number from the dial screen grid if the person answering the phone was not the person who the phone number was associated with, it was not possible to record this on the grid, i.e. to link the phone number back to who actually answered the call rather than who the phone number was associated with.
- Interviewers found the grid could sometimes become quite cluttered with phone numbers. They generally found it difficult to keep track of which numbers had been dialed.

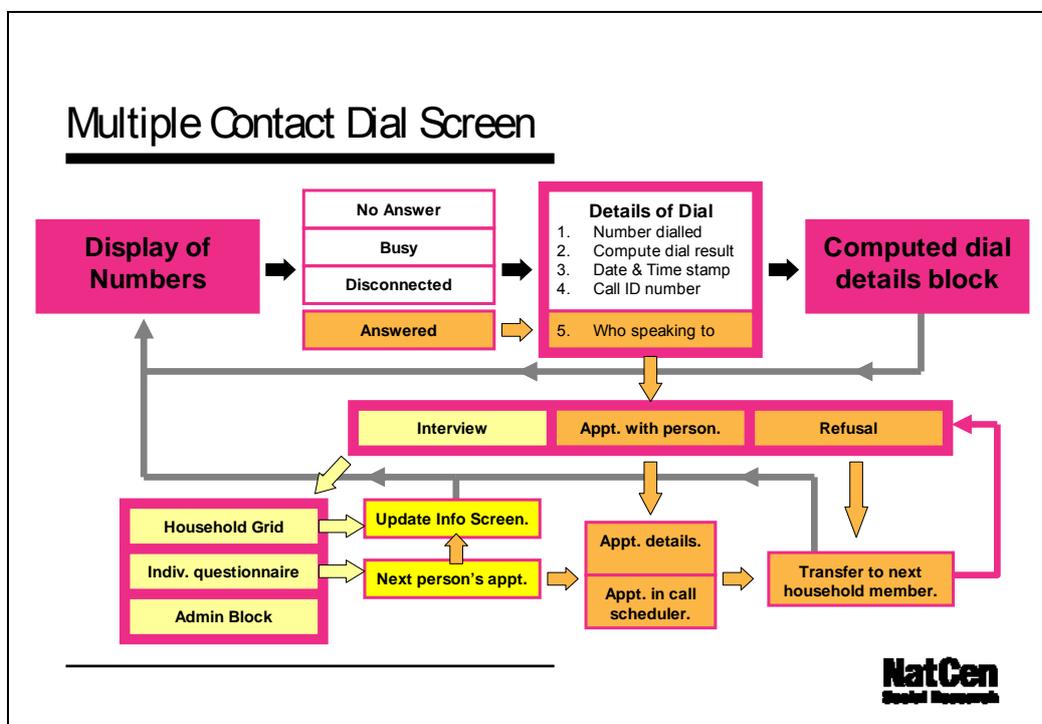
3 Implementation of the Multiple Contact Dial Screen block.

We needed to ensure that all the numbers interviewers were dialing were recorded as well as being able to record who actually answered the call. With this in mind, we moved away from the free-navigation grid-type approach and instead developed a dial screen auxiliary block at the start of the instrument. The purpose of this block was to:

- Display all household information (names, phone numbers, previous status, current status) and allow for this display to be updated at the start of each new call.
- Ask a series of questions which would establish which phone number had been dialed, the dial result and which person answered the phone and store the corresponding answers.

- Allow interviewers to try (up to 10) different phone numbers without having to exit the form. When exiting the form via a busy or no answer dial result, ensuring that this result was passed to the CATI call scheduler for appropriate dial treatment.
- Record the dial history and display this to interviewers.
- Record when the phone had been transferred to another household member.
- Record and display appointments for one or more household member.
- Record and display interviewer comments.

We produced a flow diagram of the whole survey processes needed for Understanding Society to help us focus on how the multiple contact dial screen should work to meet the survey needs.



We used the standard Blaise CATI dial screen to display only the interviewer comments and the main home phone number. Then interviewers had to select “interview” to go to the multiple contact dial screen block. As mentioned we made this an auxiliary array block so that it could be used for collecting call information for up to 10 dials before having to record a dial result for the call. The majority of households contained 10 or fewer unique phone numbers so looping the block 10 times and also allowing for the phone to be transferred 5 times within each dial was considered sufficient when balanced against performance of an already large interviewing model.

The information in the block was copied and stored in a call history block, an appointment history block and an interviewer comments block and used to provide interviewers with information on the current status of the household. Throughout the implementation process the MMU were involved in testing and ensuring that the multiple contact dial screen met their interviewing aims.

3.1 Details of each dial made.

The following set of core questions were asked to establish which phone number had been dialed, the dial result and which person answered the phone.

- Number dialled: listing all known numbers in an enumerated type list.
- Record new numbers: option of recording a previously-unreported number.
- Outcome of call: enumerated as 'Answered', 'Busy', 'No Answer', 'Disconnected'.
- Person spoken to: listing all known persons in an enumerated type list, with the option of recording a previously-unreported person.
- Date and time stamps: recording date and time dial was made.

3.2 Updating household information in display screen.

The initial display screen for interviewers contained household member information before the household grid was filled. Once contact was made with a household member and the grid filled the information was updated to reflect any movers into and out of the household.

For example household member's names were updated as follow:

FIELDS

PersonTXT "Forename of each household member": ARRAY[1..20] OF STRING[40]

SPersonTXT "Surname of each household member": ARRAY[1..20] OF STRING[40]

RULES

```
FOR ipp:=1 TO 20 DO    {initialising name}
    Persontxt[ipp]:=EMPTY
```

```

    Persontxt[ipp].KEEP
    SPersontxt[ipp]:=EMPTY
    SPersontxt[ipp].KEEP
  ENDDO

```

```

FOR ipp := 1 TO 16 DO
  PersonTxt[ipp]:=QPFFW.QP[ipp].QA.ff_forname {setting forename from sample
                                             name}
  SPersonTxt[ipp]:=QPFFW.QP[ipp].QA.ff_surname {setting surname from sample
                                             name}

  IF AName[ipp]=RESPONSE THEN
    PersonTxt[ipp]:=AName[ipp] {setting forename from household grid name}
    SPersonTXT[ipp]:=ASName[ipp] {setting forename from household grid name}
  ENDIF
ENDDO

```

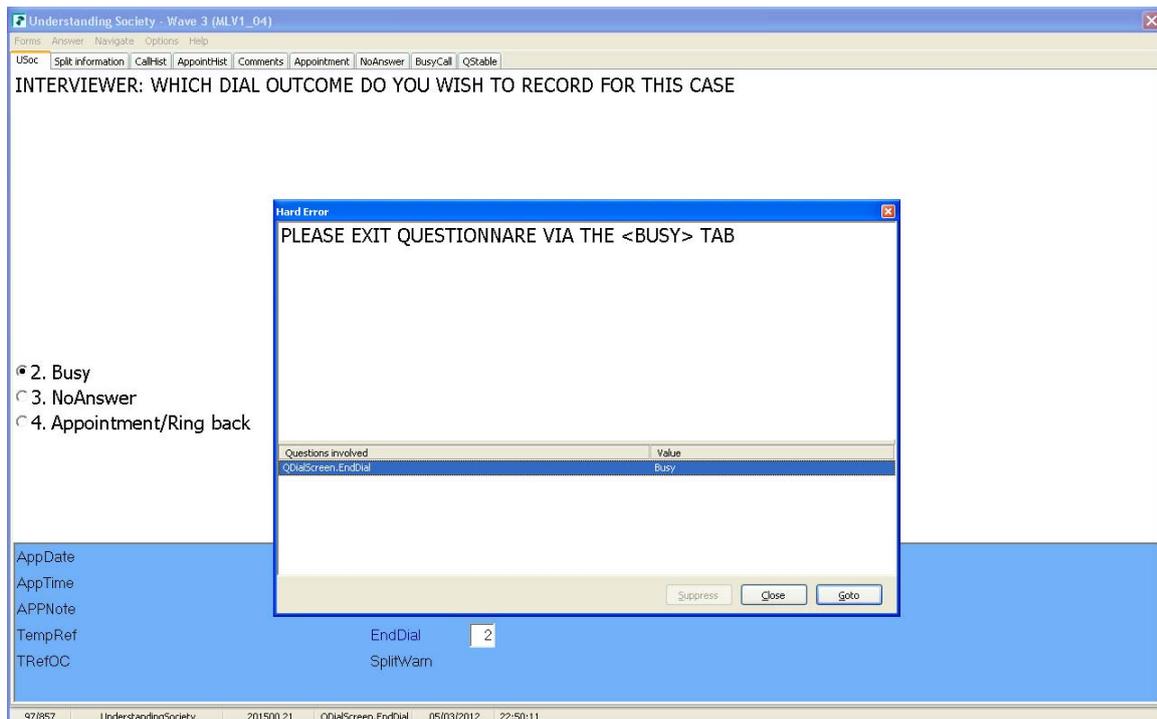
```

IF (DMAge[ipp] = RESPONSE AND DMAge[ipp] < 16) OR
   (AGE(QPFFW.QP[ipp].QA.ff_birth,QSignIn.StartDat) < 16) OR
   (QPFFW.QP[ipp].QA.ff_Age < 16) THEN
  PersonTxt[ipp]:=EMPTY {if household member in grid is aged <16}
  SPersonTXT[ipp]:=EMPTY {if household member in grid is aged <16}
ENDIF

```

3.3 Busy and No answer calls.

Dial results such as busy, no answer and disconnected were handled in the multiple contact dial screen by use of parallel blocks. When interviewers entered a busy, no answer or disconnected dial result relevant display questions and hard checks ensured that interviewers exited the case using the appropriate parallel block so that the appropriate dial treatment would be recorded in the call scheduler.



3.4 Call history display details.

The details of each dial made (as described in 3.1 above) were stored in another parallel block we named the 'Call History' block. Details of each dial were written to this block once the interviewer had established a dial result. If the dial result was busy, no-answer or disconnected interviewers were given the option to try another phone number.

```
More "@/DO YOU WANT TO TRY ANOTHER NUMBER FOR THIS CASE?"  
      : (Yes,No), NODK, NORF
```

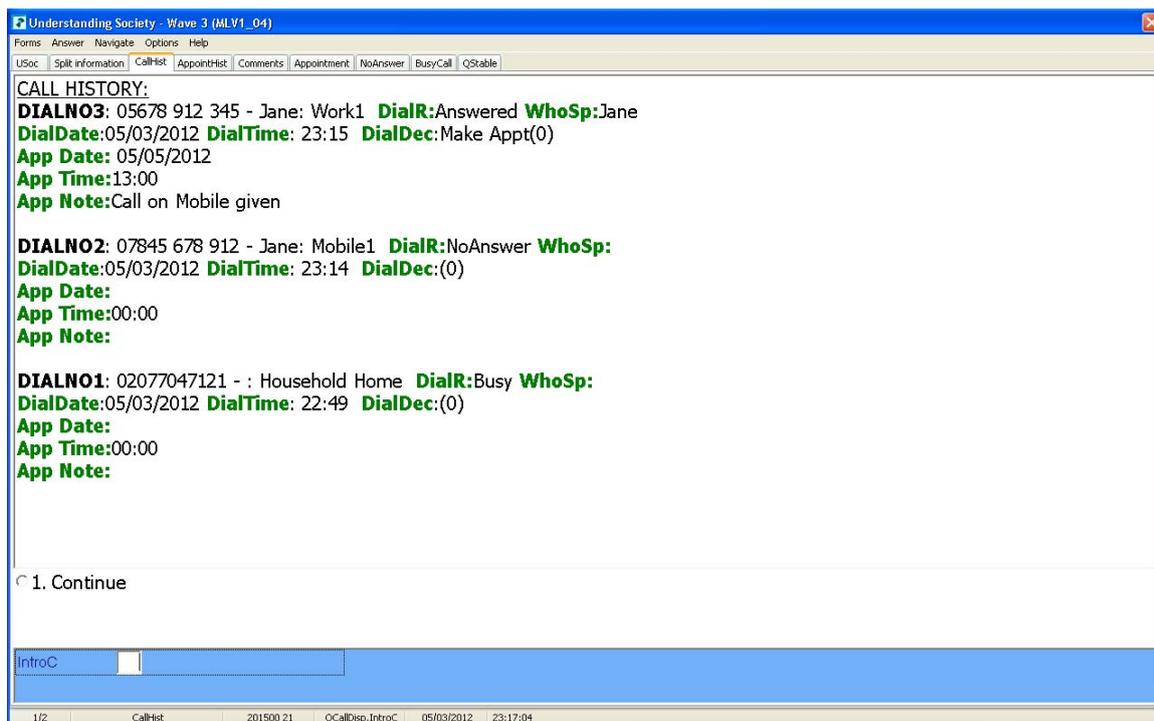
If the interviewer had other phone numbers available to try they could select the "Yes" option here and they would then be prompted with a soft check to confirm that the current dial had been concluded. The information was then computed, stored and displayed in the Call History block and the next element of the dial screen array came on the route ready for the next dial. The soft check was considered necessary in order to ensure interviewers could confirm or change the dial information they had recorded before it was stored.

RULES

```
Interview  
IF (Interview <> EMPTY) THEN  
    FlagInt:=Yes {used for copying to Call History block}  
ENDIF  
IF (Interview <> EMPTY) THEN  
    SIGNAL  
    (Interview = EMPTY)  
    "INTERVIEWER: ARE YOU SURE? YOU WILL NOT BE ABLE TO GO  
    BACK AND CHANGE YOUR ANSWERS AFTER THIS QUESTION"  
ENDIF
```

```
{Computing dial details to Call History block}  
IF (More<>EMPTY OR FlagInt and datastored <> yes THEN  
    QCallHist[NextRow].ListNum := ListNum  
    QCallHist[NextRow].OtherNum := OtherNum  
    QCallHist[NextRow].OtherNam := OtherNam  
    QCallHist[NextRow].DialRes := DialRes  
    QCallHist[NextRow].Date := DialDate  
    QCallHist[NextRow].Time := DialTime  
    QCallHist[NextRow].whospoke := whospoke  
    QCallHist[NextRow].OthName := OthName  
    QCallHist[NextRow].DialDec := DialDec  
    QCallHist[NextRow].AppDate := AppDate  
    QCallHist[NextRow].AppTime := AppTime  
    QCallHist[NextRow].AppNote := AppNote  
    NextRow := NextRow + 1  
    Datastored := Yes  
ENDIF
```

The Call History block was made readily available to interviewers via a tab control on the multiple contact dial screen. The block displayed the result of the latest dial at the top of the page so that the most up to date information was available without having to scroll.



3.5 Transferring the phone and appointments.

For any dial that resulted in the phone being answered interviewers were presented with three options to proceed

DialDec "PLEASE SELECT APPROPRIATE OPTION BELOW"

::(Inter "Interview ^name now",
 Appt "Make appointment to call back to speak to ^name",
 Refu " ^name refuses"), NODK, NORF

If the household member wished to make an appointment for a call back interviewers were prompted to provide details of the appointment date (AppDate), appointment time (AppTime) and any useful comments regarding the appointment (AppNote).

Once the appointment details were recorded interviewers were then asked if the phone could be transferred to another household member.

Transfer "INTERVIEWER: CAN THE PHONE NOW BE TRANSFERRED TO SPEAK TO ANOTHER HOUSEHOLD MEMBER?":(Yes,No)

If a phone transfer did take place interviewers were routed to a sub block of the multiple contact dial screen block which we call the "Transfer" block. Here interviewers were presented with the same 3 options as in DialDec above so a further appointment could be recorded. The Transfer block was looped 5 times thus allowing the phone to be transferred 5 times for each dial made on the multiple contact dial screen. The result of the phone transfer was also computed, stored and displayed in the Call History block.

```
FOR J:=1 TO 5 DO
  QCallHist[NextRow].QTransHist[j].Twhospoke:=QTransfer[j].Twhospoke
  QCallHist[NextRow].QTransHist[j].TDialDec:= QTransfer[j].DialDec
```

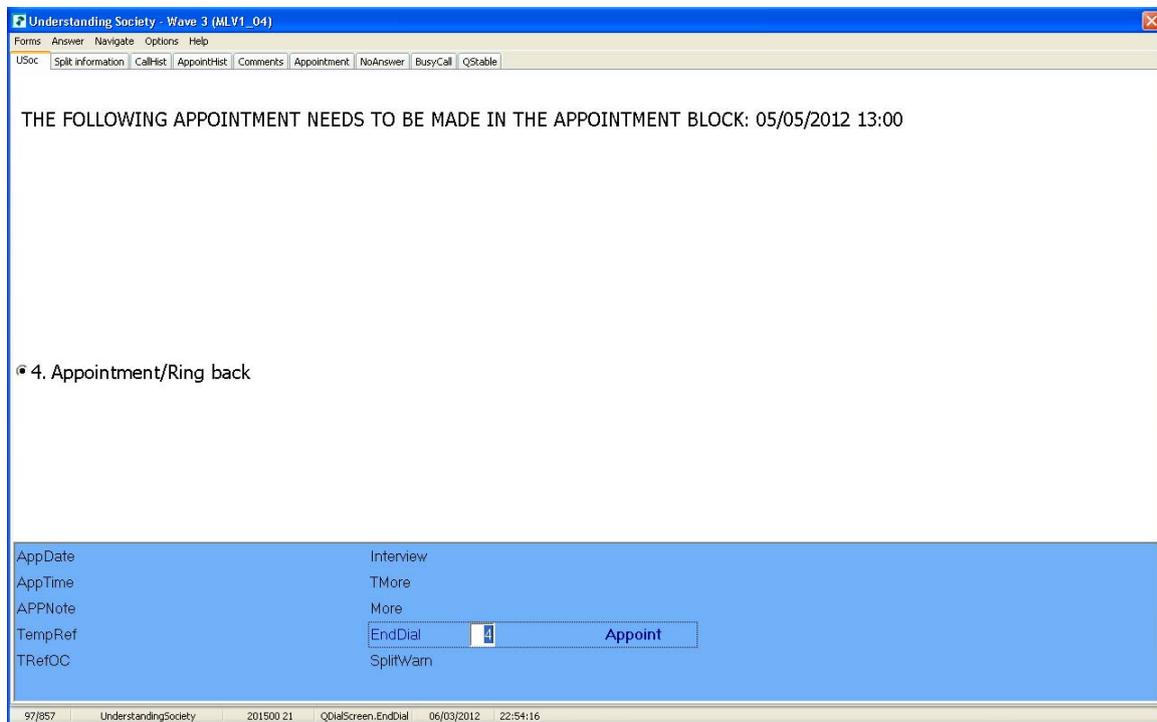
```

QCallHist[NextRow].QTransHist[j].TAppDate:= QTransfer[j].AppDate
QCallHist[NextRow].QTransHist[j].TAppTime:= QTransfer[j].AppTime
QCallHist[NextRow].QTransHist[j].TAppNote:= QTransfer[j].AppNote
QCallHist[NextRow].QTransHist[j].TDate:= QTransfer[j].DialDate
QCallHist[NextRow].QTransHist[j].TTime:= QTransfer[j].DialTime
ENDDO

```

If a household member refused at DialDec (above) interviewers were asked to code if the refusal was for the whole household or just the household member they were speaking to. If the refusal was for the household member only the interviewer was routed to the Transfer question (above) and subsequently onto the Transfer block.

As it was possible for interviewers to make appointments for several household members on ending the call all the appointments recorded in the multiple contact dial screen were ordered and interviewers were prompted to record details of the next one in date order via the standard CATI appointment function. A hard check here also ensured they left the form via the appointment block.



Details of all appointments were also made readily available to interviewers via an “AppointmentHist” tab control on the multiple contact dial screen.

4 Interviewer training and usability.

The multiple contact dial screen required complete re-training on the way our interviewers worked. Initially we decided to select the more experienced interviewers to work on Understanding Society as it was felt that they would be able to deal with the change of working more easily. Interviewers were used to working with the standard Blaise dial screen, however this new approach meant that interviewers had to skip past this dial screen and instead use the multiple contact dial screen to make calls.

Usually on a given survey interviewers would be briefed on the CATI program but not on the dial screen and call scheduler as this was part of the standard training they would have received. For

Understanding Society a set of specific training sessions was developed to take interviewers through the functionality of the multiple contact dial screen and the logic behind it. The initial training sessions included feedback from interviewers so that shared concerns could be addressed and focused on during subsequent sessions.

We also built in a significant practice period prior to survey launch so that interviewers could use dummy sample cases to gain familiarity with the various screens.

MMU supervisors were also trained to check and inspect the various information tabs at the start of each shift so that they could check all appointments had been made and to help plan the work of interviewers to ensure the smooth running of the survey.

5 Conclusions.

Understanding Society was unlike any other CATI project that we had run in the MMU. Interviewing multiple household members with multiple phone numbers meant that we had to adapt the way we used the standard Blaise CATI call scheduler. The design of the multiple contact dial screen block had to meet our clients needs to capture all dial information as well facilitate successful interviewing. We have tried to take into account dynamic updating of the dial screen, calling multiple phone numbers, dealing with phone transfers and multiple appointments and recording and displaying all dial information. To this end we have looked to meet the clients survey needs as well as attempting to make the multiple contact dial screen as user-friendly and practical as possible.

We feel we have achieved a broadly successful solution to a complex problem. Interviewers have been using the multiple contact dial screen now for the past 2 years. Key to its success has been the careful monitoring of households that MMU supervisors do at the beginning of each interviewing shift using the dial and appointment information available.

A system allowing sequential interviewing of household¹ members, where the household members are individual cases, within the Blaise CATI-framework

Trond Båshus, Statistics Norway

Introduction

The Norwegian labour force survey (LFS) uses a household sample where each household member is an individual case in the Blaise-database. The previously used case management system at Statistics Norway was mainly oriented towards CAPI. Multi-person household was easy to manage as every member of a household was routed to the same interviewer, who was able to interview the whole household one after the other during the same session, if possible.

The new case management system (Sivadm) now in use at Statistics Norway is more oriented towards CATI (with possibilities for CAPI and CAWI), and it was therefore necessary to develop a system within the Blaise CATI-framework to allow the CATI-interviewers to interview an entire multi-person household consisting of individual cases in one session. The system offers the interviewer updated interview statuses of all household members, both in the dial screen and within each form, and an option to open a form for another household member at the end of the interview.

Background

The previous case management system used at Statistics Norway was primarily an offline CAPI system (CAI), where the interviewers connected to the central database at regular intervals to return and download cases. The system was used as a list based CATI-system (CATI-L), since true CAPI was (and is) not used for the LFS in Norway. All cases belonging to a certain household were transferred to a single local interviewer, and an entire household, if all present, could be interviewed in one session. The system made it relatively simple for the interviewer to keep track of the household, with regard to appointments, and also whether it would be beneficial to do a proxy interviews for absent household members.

A new case management system (Sivadm) which together Blaise forms the new interview management system at Statistics Norway (SIV) has long been under development. It is centred around a database driven CATI-system (CATI-D) to better utilise Statistics Norway's two call centres, while also allowing the ca. 100 interviewers working from their homes to connect to the system from home and work online. SIV also provides an integrated offline CAPI/CATI-L solution.

To allow for sequential interviewing of household members on CATI-D, two approaches have been considered:

1. All cases of a household contained within one single form.
2. Each case of a household at single form, with a functionality to move between the cases within a household.

The first scenario would allow household appointments, and also shared address and telephone numbers for the household. But it would not allow for individual treatment for each case, and it would also significantly complicate administration.

¹ In reality a family sample, due to limitations in the register of residents.

The second scenario would allow individual treatment of each case, but new functionality had to be developed: Interview status for each household member must be displayed in the Dial menu and within the forms, and it must be possible to start an interview with at another household member upon finishing an interview.

In addition, it was a requirement that cases can be moved to the offline system for follow up by the local interviewers (CATI-L) after a certain time period has passed. This is quite simple to accomplish with the second scenario, but would probably require changes in the offline system if we went for the first option. After careful consideration it was decided to develop the necessary functionality to make option 2 possible.

Technical description

The new solution builds on and extends on the existing integration between Sivadm and Blaise, in particular mechanisms which synchronises events between the two systems. The main mechanisms are Manipula procedures which are run OnDialBegin and OnDialEnd, as specified in the CATI-specification.

Blaise and Sivadm integration

Synchronisation and integration between the Blaise and Sivadm is based on that each case, irrespective of which survey they belong to, has a unique identification number (io_idnr). Events from Blaise are written to a special table in Oracle which Sivadm reads at regular intervals, the events are mainly changes in the status of the cases, and also information about changes in address and/or telephone numbers. If there is an address or telephone number change in the Blaise-database(s), Sivadm will read this information through the Blaise-API. Events from Sivadm to Blaise are always written directly the Blaise-database through the Blaise-API.

Overview of the system

The system developed for LFS is an extension of mechanisms already present.

1. Status information about all respondents is stored in a separate Blaise-database: status.bdb.
2. Io_idnr for all household members are included for all respondents. This is used to read and write status information to status.bdb.
3. A procedure runs OnDialBegin which reads status information for the household members.
4. Status information about the household members is presented in the dial menu, in addition to age and telephone numbers.
5. The same status information is also presented at the beginning and end of the form. The interviewer can select which household member to open next at the end of the form. An option to let the CATI-scheduler select a form is also available.
6. A procedure running OnDialEnd calls an external Manipula setup to write status information (skriv_status2) about the current respondent to status.bdb, in addition to an Oracle table used by the Sivadm application.
7. Depending on the choice made by the interviewer at the end of the form, the Dial menu of a selected household member or a form selected by the CATI scheduler appears.

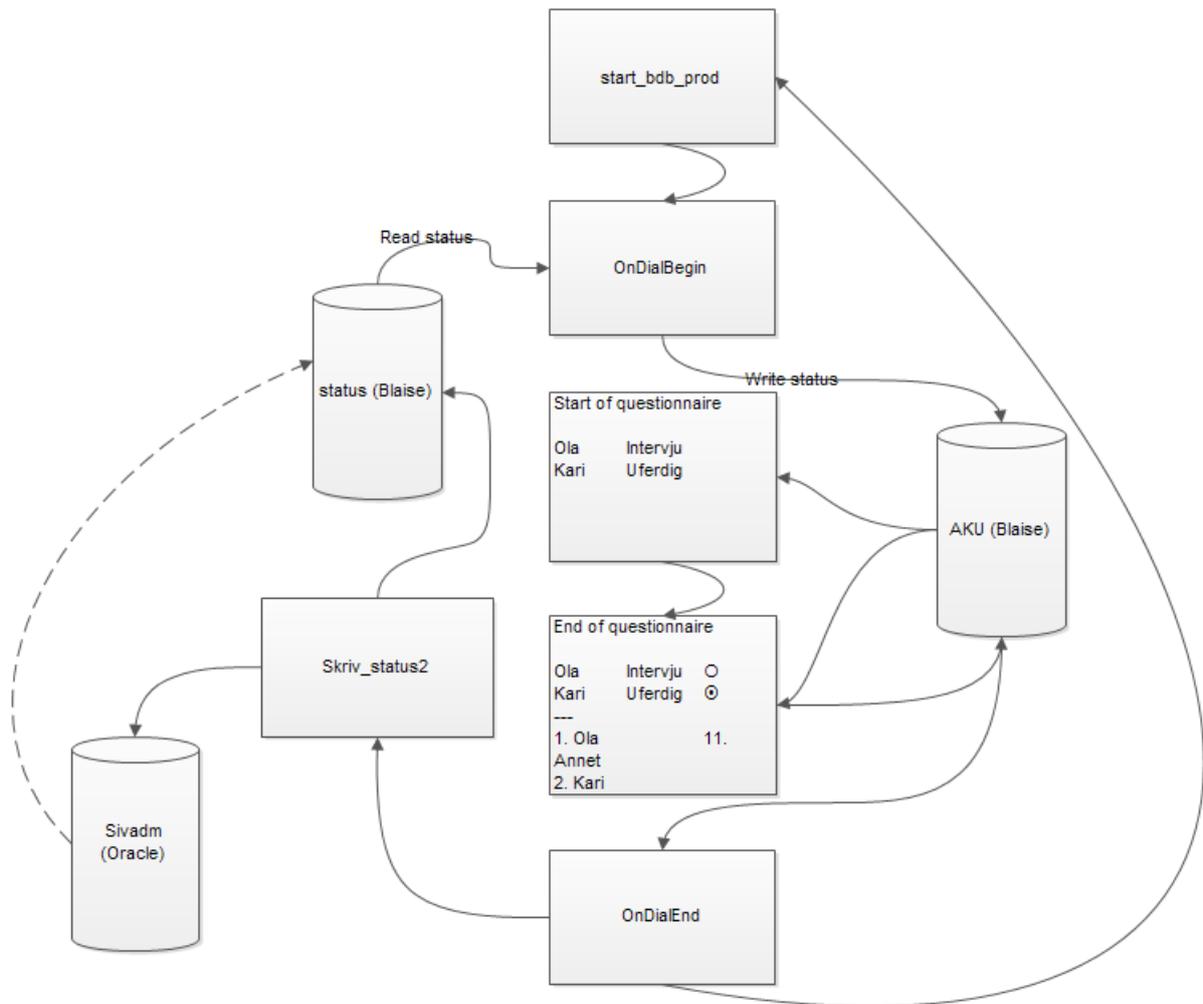


Figure 1: Overview of the system

Prefilled case information and status database

Prefilled case information, including io_idnr, name, age, gender, etc. of every household member, is included in every form. This is used to read status information for the household members from the status-database (status.bdb).

This is the datamodel for the status-database:

```

datamodel status
  primary
    io_idnr
  fields
    io_idnr : integer[8]
    status_case : string[2], empty
    intervju_status : string[2], empty
    merke : string[1], empty
    telefon1 : string[9], dontknow ,empty
    telefon2 : string[9], dontknow ,empty
    telefon3 : string[9], dontknow ,empty
endmodel

```

The "status_case" field can have the following meanings:

00: interview

01: non-eligible respondent

02: non-response

The telephone fields will be used to display phone numbers of the household members in the Dial menu. This is currently being tested, and is not yet a part of the production system.

CATI-Specification

In the CATI-specification it is possible to define procedures which should be run before the Dial menu and at the end for a form specified using OnDialBegin and OnDialEnd under “Events”. Code to read and write to the status-database is included in these procedures.

Procedures (registrerkontakt2)

OnDialBegin

The relevant code loops through an array containing io_idnr for all household members in the current form, and calls another procedure to retrieve the interview status of the household members from the status database.

OnDialEnd

The procedure used OnDialEnd is responsible for calling an external manipula setup (skriv_status2), in order to write statuses to Sivadm. The procedure checks if the survey in question is the LFS and passes a parameter to skriv_status2 to tell it to write status information to the status-database described earlier.

Dial menu

The interviewer is presented with information about name, age, status and telephone numbers of all household members.

Ring! [X]

Ringemeny

Skjema
 Avtale
 Endre Kontaktopplysninger
 Opptatt

Ikke svar
 Telefonsvarer
 Send til sporing

OK

Avbryt

Hjelp

Spørsmåls data:

IO_nummer	70901
VisTlf1	[REDACTED]
VisTlf2	
VisTlf3	
F_nummer	[REDACTED]
Navn_IO	[REDACTED], Alder: 44, Kjønn: Kvinne
Kommune	[REDACTED]
Adressa	
p_Adressa	[REDACTED]
MedHvem	
Avtmeld	
Ringtlf	
tfforsok	
SpesRing	
hinfo[1]	[REDACTED], Alder: 48, Status: Uferdig, Tlf: [REDACTED]
hinfo[2]	[REDACTED], Alder: 18, Status: Uferdig, Tlf: [REDACTED]
hinfo[3]	
hinfo[4]	
hinfo[5]	
hinfo[6]	
hinfo[7]	
hinfo[8]	
hinfo[9]	
Resu1	Int
Resu2	Int
Resu3	Int
Resu4	Int
Resu5	Int
Resu6	Int
Resu7	
PeriodeNr	4

Mer Info

Ring

Rediger...

Figure 2: Dial menu

Beginning of form

The form starts with a presentation of name, age and status of household members.

The screenshot shows a software window titled 'aku2012k1 | Endre kontaktinformasjon'. The main content area is yellow and displays the following text:

70901 [redacted] tlf 67734491, Alder: 44 år.

Familien består av:

IO-nummer: 70900, Navn: [redacted], Alder: 48, Status: Uferdig
IO-nummer: 70902, Navn: [redacted], Alder: 18, Status: Uferdig

Trykk <Enter> for å gå videre med intervju av IO

Below the yellow area is a grey control panel with the following variables and their values:

fhoversikt	<input type="checkbox"/>	direkInt	
Innled	<input type="checkbox"/>	HvemGav	
Frafgr		FlgInnl	
Dvifgr		IlGng2	
Avggr		IlGng3	
FrafSpes	<input type="text"/>		
Starttid	10:34		
Samtykke1	<input type="checkbox"/>		
KommNavn	<input type="text"/>		
NyKomm			

At the bottom of the window, there is a status bar with the following text: Gammel 1/33 Endret Med feil Naviger aku2012k1 Intervjuing

Figure 3: Beginning of form

End of form, and selection of next case.

At the end of the form, name, age and status are again presented for the interviewer, and in addition there is an option to start interview with any of the household members without a current status, or alternatively start an arbitrary respondent selected by the CATI-scheduler.

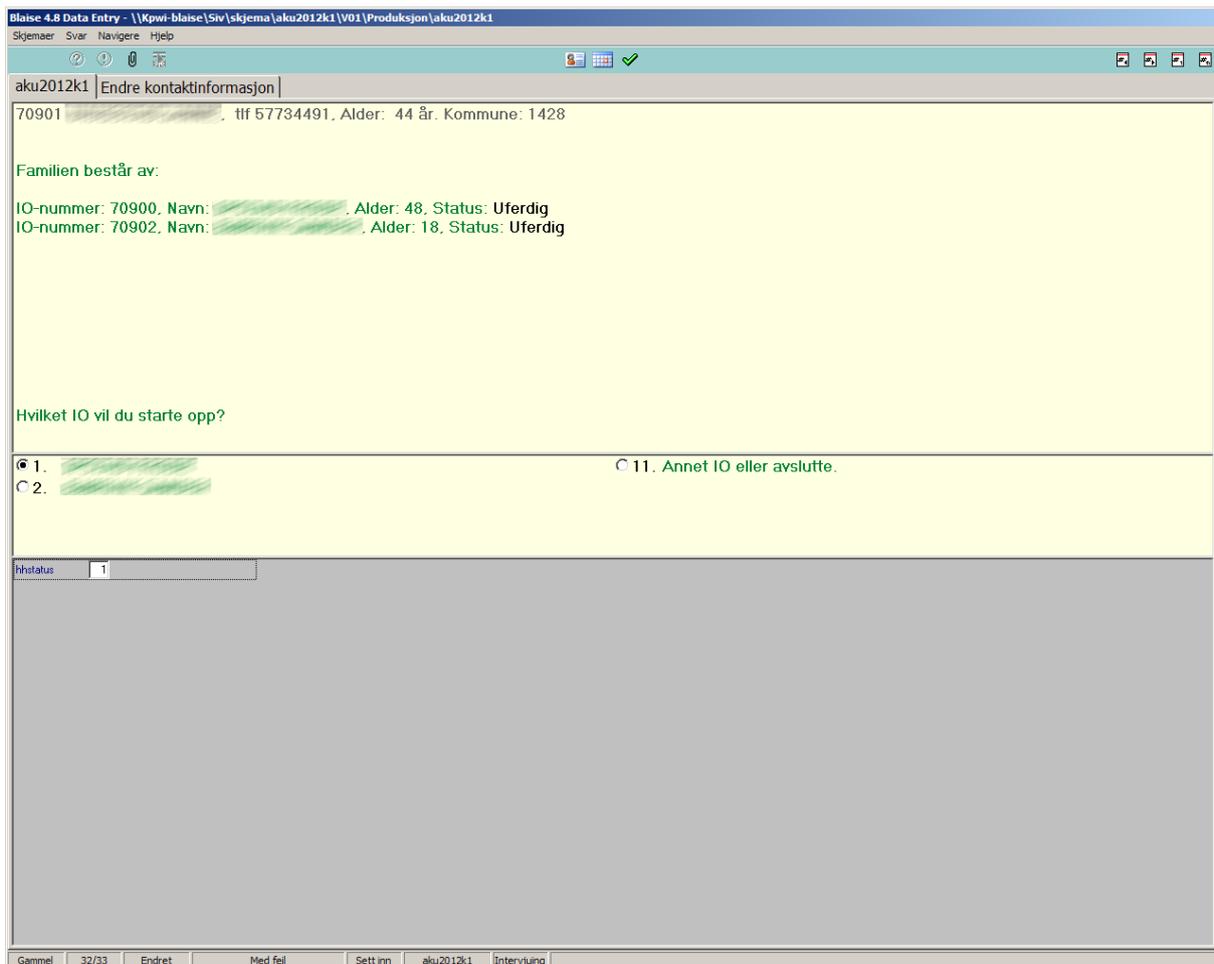


Figure 4: End of form

From pilot to production

Two pilots were planned before LFS was put into a production environment. The first was run for one reference week in the 3rd quarter 2011. One important bug was identified and fixed and the pilot also helped to establish new routines for the call centre interviewers.

The second pilot was run in the 4th quarter to ensure that the routines for preparing 2-8 wave respondents worked as intended, since some changes to the SAS-programs used for sample preparations were needed due of the transition from CAI to SIV.

The LFS was transferred to production the 4th quarter 2011, with no significant technical problems identified.

Production lifecycle

A reference week starts first on the CATI-D system, and runs there for about one week. All non-contact non-response cases are then transferred to the offline CATI-L system for individual follow up by local interviewers. Other types of non-response are handled continually.

Future developments and problem areas

The system will continually be under development, and will take advantage of improvements made specifically for the LFS, but also general improvement to SIV. This involves development of new features in both the Blaise and Sivadm part of SIV.

Improvements of the CATI-system

The interviewers have asked for more information about the household to be presented in the Dial menu. Most important is the inclusion of telephone numbers for all household members. This feature is currently being tested.

Currently it is not possible to open a Dial screen for another household member without actually completing a form or making an appointment. A future improvement will be to open a form for any household member (more) directly from the dial menu.

Other possible improvement includes appointments for the entire household, and also some kind of locking of the forms of the other household members when an interview with a household member has started, to prevent the Blaise CATI scheduler to deliver the cases belonging to the household to other interviewers. This last problem has to a certain degree been eased by sorting the daybatch in a way that minimizes the probability of this occurring.

Online CATI-L

Today we are using the offline system provided by SIV for follow up (CATI-L). This makes it very difficult to monitor progress for cases that are sent to the offline system. A future improvement to SIV, allowing online list based CATI interviewing (CATI-LD), can be used to great advantage to the LFS. Using this system, the local interviewers will work on a list of cases within the web based Sivadm application. This will allow real time monitoring of progress, since the all interviewers will work using the same central Blaise-database, and also ease movement of cases between CATI-D and CATI-LD. An option for the interviewer to move cases to the offline system is also planned.

CAWI

A probable future extension to the LFS will be to integrate a CAWI solution, but this is a project in a very early planning stage.

Conclusion

The transition of the LFS from the previous case management system, CAI, to SIV has in most respects gone as planned. The system for sequential interviewing of household members within the Blaise CATI frameworks has also worked well, although there is potential for improvement. Especially when it comes to the handling of appointments, information presented to the interviewers and more flexibility in how and when a CATI interviewer can move between the forms of household members.

Literature

Gravem, Dag. F. *Towards an integrated mixed-mode case management system for the Norwegian LFS*. Paper presented at the Workshop on Labour Force Survey Methodology, Wiesbaden 12-13 May 2011.

Extending Blaise Capabilities in Complex Data Collections

Paul Segel and Kathleen O'Reagan, Westat

International Blaise Users Conference, April 2012, London, UK

Summary: *Westat Visual Survey (WVS) was developed for use on studies with deeply nested hierarchical data models. We discuss the use of WVS in CAPI studies, benefits and issues, and lessons learned.*

Introduction

Westat Visual Survey (WVS) was developed to support longitudinal CAPI studies with very deeply nested hierarchical data models. It uses a standard Blaise data model with an additional language field text to combine the Blaise rules engine with Blaise Component Pack (BCP) API calls in order to navigate through the complex rostering. The benefits and issues with that approach are summarized. As a web application, WVS was able to separate the presentation from the rules engine, and backend SQL database storage. Extending navigation outside the rules engine poses some challenges. Lessons learned are discussed including managing the development life cycle with portions of the product in multiple platforms, maintaining compatibility with Blaise versions, and developing automated testing procedures.

Design Considerations

CAPI instruments with deeply nested data structures such as those collecting information about multiple families, multiple persons in each family, and each person's specific history across various event topics present programming challenges. With complex navigation rules, Blaise is the logical choice of platform, but situations occurred where the data model was too large to prepare due to the iterative rostering of entities (persons, families, events, etc.), collection of details about the rostered items and maintenance of relationships between the rostered entities.

Design features for the WVS development included: retaining the Blaise metadata and rules engine, supporting variable length rosters, representing and navigating the deeply hierarchical data structure, and providing a multi-tiered architecture with the Blaise engine providing the business rules, a native Web presentation layer and a data access layer to a SQL Server database. Version control for all components of the system and project were important as well. WVS also includes the ability to search large directory databases, perform CARI recording for CAPI instruments, fine tune performance and support automated testing. Another design goal was to be able to integrate the Blaise and WVS development process and utilize the standard utilities available to support and manage a CAPI project. These design considerations are addressed in more detail below and the overall WVS design is shown in Figure 1.

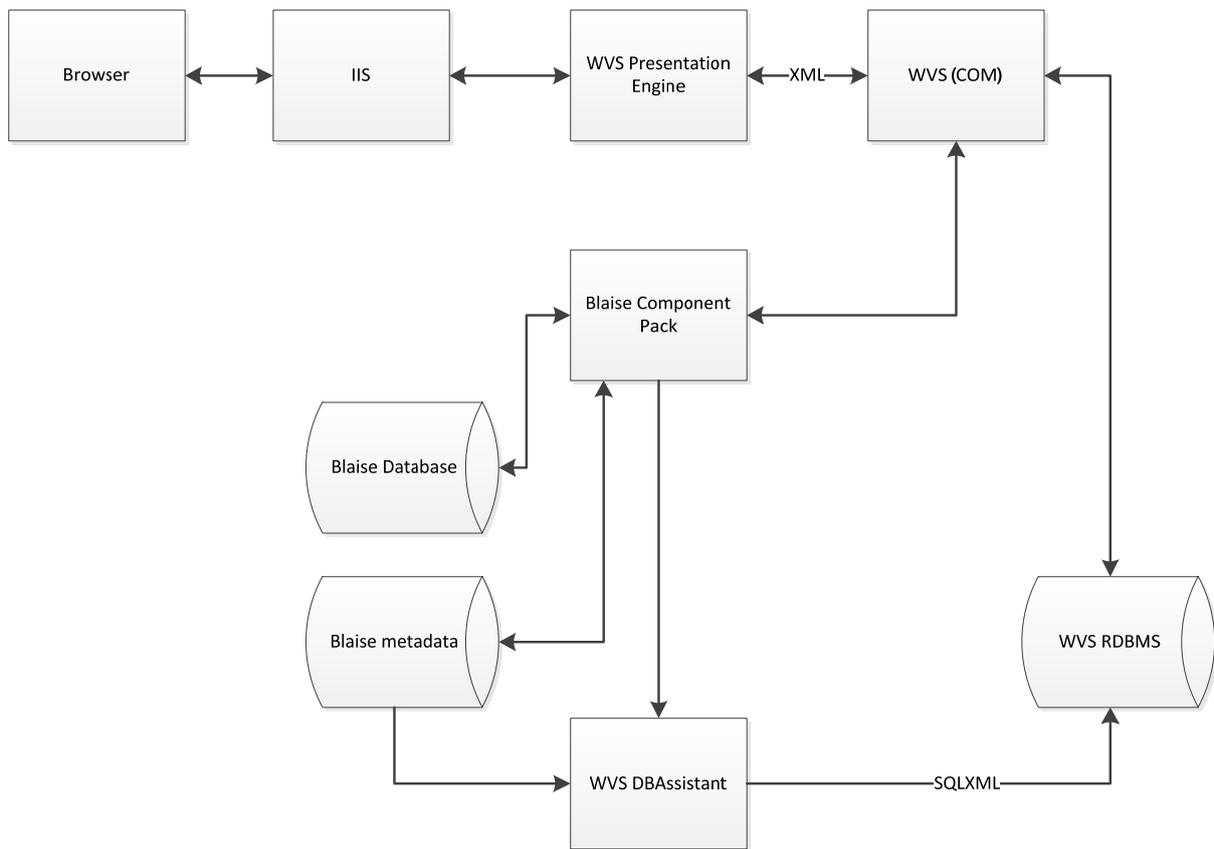


Figure 1. Blaise WVS Design

Retain the Blaise metadata and rules engine

Our instrument development processes are built to support Blaise instruments including specification development, program coding, testing, and documentation. Apart from the complex rostering, Blaise supports the structure and flow of the instrument and retaining the metadata and rules engine works well. Blaise is also used within the detail items and loops, even within the complex question sequences for navigating around non-response.

In order to support the additional features of WVS, we developed a WVS language that was added to the metadata items. The WVS language included such items as definition for roster creation, navigation, and any special presentation specification, similar to the MML multi-media language.

Support for variable length rosters

In order to support roster lists of any length WVS requires language support, a data structure, and a link that remains consistent with the Blaise database. WVS maintains a normal Blaise bdb database file for the rules engine. Although this database would not contain all the data associated with a case, it would be an accurate snapshot to support asking the rules engine what item was next on the route at any given time.

In addition, WVS maintains a SQL Server database with the full case data. Since all the data are collected in normal Blaise blocks, the database structure can be automatically derived from the Blaise data model. The SQL Server schema does, however, try to reflect the entity relationships, and contains additional references to the block location in the Blaise instrument. Much of the action in the data access layer is maintained through SQL bulk loads and driven by SQL XML schema files.

WVS maintains a distinction between defining a roster entity and collecting details about that entity. From an instrument design perspective that distinction helps especially if roster details are collected in multiple locations in the instrument.

Represent and navigate the deeply hierarchical structure

WVS supports two ways of collecting details about a rostered entity or roster navigation: normal Blaise looping and an “Instance Navigator” approach. If the roster data collection is mapped to an array of Blaise blocks, normal Blaise loops can be used to navigate the roster collection. WVS adds some constructs to control the sequencing of the loop, but the routing does not require any action from the interviewer.

Some of the rosters, deeply nested in a hierarchy, would not fit within the Blaise data model size constraints, as large as they are. For these rosters, WVS re-uses a single Blaise block (including any sub-blocks) in the data model, retrieving and storing the appropriate roster entity from the SQL database as the flow enters and exits the block. Although, the roster entities can be sequenced automatically for collection, usability studies suggested that it was preferable to allow interviewers to direct the sequence.

To support that selection, WVS creates an “Instance Navigator” screen that displays each roster entity as shown in Figure 2 with a description and status of the collection, typically labeled “complete”, “partial”, “not started”. An option can prevent re-entry into completed instances, if needed to manage complexity. The ability to select the instance facilitates backing up into previous instances quickly, for example.



Figure 2. Instance Navigator Screen Example

When a new roster collection instance is created, WVS has to maintain the completion status across all instances. WVS performs several activities to do this:

- Write the current instance to the SQL Server database as instances themselves can be nested. The writing requires concordance between the Blaise rules engine and a WVS declaration specifying the levels of the data structure.
- Assess all instances against the rules engine for completion. Since instances in WVS are allowed to interact, the process of assessing the completion status is dynamic, although another WVS declaration can allow previous status to be reported. In the full dynamic version, any current instance is cleared in the Blaise data model (fields reset to an unanswered status), current instance values are retrieved from the database for the entire block structure, and then the block structure is traversed looking for unanswered items.

- Select a new instance. When an instance is selected by the interviewer, the clearing and loading is performed, if needed, for the selected instance.

This process is repeated whenever the routing enters the instance block from either the forward or backward direction.

Deleting a roster item in this scenario also introduces complexity. The same roster, e.g. people in the household, can be used as the basis of collection in multiple locations in the instrument. When a roster entity is deleted, checks are made to assure that the deletion is handled appropriately in all locations.

Provide a multi-tiered architecture

WVS provides a multi-tiered architecture with the Blaise engine providing the business rules, a native Web presentation layer and a data access layer to a SQL Server database. Since WVS separates the rules engine (business rules) from the SQL data access, it further separates the presentation layer. The presentation layer is a dynamically created Web page, using a version of IIS on the CAPI laptop. The Web page performs local field syntax and range checking, and routing within the objects on the page.

A central master WVS component processes the Blaise rules and any associated data to create an XML document describing a screen type. The original version of WVS built on classic ASP used XSLT to transform the XML document into the web page, and the current version uses ASP.Net .

The most common screen types include:

- Data Entry screen – supporting multiple fields, possibly with internal routing. In addition to multi-field forms, examples could include an “Other-Specify” screen (the Specify field appears only if an enumerated field has an value of “other”), Quantity-Unit forms where the quantity depends on a previously selected unit.

110118 CP11 | EDetails.EDetail.CP11 | English | GO | About Wvs

Report Bug | Comments

CESAR MARTINEZ | GENERAL HOSPITAL | HS | Oct 09 2011

How much of the \$600.00 did anyone in the family pay for (PERSON)'s visit to (PROVIDER) on (VISIT DATE)? Please include all amounts paid 'out-of-pocket', that is, amounts paid before any reimbursements.

IF AMOUNT PAID IS NOTHING, DK, OR RF, SELECT DOLLARS, THEN RESPONSE.

IS ANSWER IN DOLLARS OR PERCENT?

DOLLARS

PERCENT

ENTER DOLLARS: \$

SELECT HELP FOR INFORMATION ON AMOUNTS TO INCLUDE.

Previous Page | Next Page

Figure 3. Entry Screen Example

- Grid screen – built from Blaise tables. The cells can be traversed in any order. An Ajax implementation allows the Blaise rules engine to update selected fields without redrawing the entire page.

110118 C/P Source 1 EDetails.EDetail.Payments English GO About Wvs

Report Bug Comments

TOTAL CHARGE: \$ 600

IF REPORTED IN DOLLARS: What was the amount Person/Family paid?

ENTER AMOUNT PAID TO COLUMN 2 OR COLUMN 3.

#	1. Sources of Payment	Payment	
		2. Amount \$ Paid	3. Amount % Paid
1	Person/Family	\$ 40.00	7 %
2	Blue Cross	\$ 0.00	0 %
3	Supplemental Policy	\$ 0.00	0 %
Unpaid Balance \$		580	

TO ADD MORE SOURCES OF PAYMENT, SELECT PREVIOUS PAGE.

Previous Page Next Page

Figure 4. Grid Screen Example

Version control for all components of the project

A project's WVS implementation includes several components in addition to the Blaise instrument. WVS deployments use text versions of these components to facilitate version control, and maintain the ability to compare versions. The components include:

- SQL Server schema (as scripts)
- WVS system configuration items stored in the SQL Server database (as scripts). These configuration items include, for example, tables that manage the roster instances, including their structure and their status.
- SQL XML schemas for the data access

Provide search capability in large directory databases

WVS projects also needed the ability to search large external databases using a variety of search strategies. In the current version, the external databases contains data in fields and WVS flexible specifications for search strategies are defined based on the number of known fields and field values.

Enable CARI recording for CAPI instruments

As projects requested CARI recording to assess CAPI interviewer performance and the quality of the question items, WVS integrates a CARI specification and recording capability. Although historically developed independent of the current Blaise CARI features integrated with Blaise DEP, it uses a configuration similar to the CARI settings file.

Support automated testing

Automated testing provided a means of regression testing new versions of the WVS system. We developed a suite of test scripts to exercise critical features of the system. To retain the ability to test within the range of data structures, script maintenance involved modifying existing scripts to exercise new features. Test data for SQL scripts is maintained and version controlled to match the script path execution. The automated testing tool that we use requires a unique identification for the objects on the web page. Also, in order to maintain the testing scripts more easily with small instrument changes, the object identification has to be consistent for a page. The WVS page rendering builds a tag based on the field name that basically satisfies these requirements. Audit trails are also useful in helping to identify updates needed to some of the testing scripts.

Ability to tune performance

Depending on the size of the instrument and its structure, some of the WVS background activities can require substantial execution time, longer than is optimal for data collection. For example, the process of clearing, loading, and traversing roster instances to assess their completeness can require extensive use of the Blaise rules engine. To tune some of these performance issues, WVS audit trails can be mined to locate bottlenecks, and fine tune performance so that execution speed is at normal levels. WVS has evolved specifications at the item level to choose among various techniques for maximizing performance. Of particular benefit was managing the transition between Editing mode (to load data without rules) and Interviewing mode (that applied the rules) in a way that did not impact performance.

Utilities to manage a CAPI project

As a Blaise instrument at the core, WVS instruments can take advantage of existing utilities developed to support the Blaise project life cycle. Some activities, like extracting case data, require WVS extensions to deal with the SQL Server case data.

Integrated Blaise and WVS instrument development process

Instrument development for the WVS system builds on the normal process for other Blaise instruments. The specification for Blaise fields remains the same and the instrument development starts with a normal Blaise instrument. To insure compatibility with the Blaise rule engine, the full WVS instrument can be run with DEP. Additional steps are required to author the WVS-specific portions of the instrument.

Although there is a default presentation for Blaise field types in WVS, additional WVS commands in the WVS language can be added to control the presentation including page layout, within page routing instructions, and item help references. The web page performs field level validation, primarily from the Blaise metadata specification, although additional validation can be added such as date ranges based on the fields on the page. More complex validation uses the normal Blaise rules when the page is sent back to Blaise.

In addition, WVS commands are required to control the loading of data from the WVS database into the Blaise database. The WVS DBAssistant tool creates XML maps between the Blaise block

structure and the WVS database entities. The WVS commands and front and back covers control the retrieval and storage of the WVS database items.

All the components of Blaise WVS instrument development have text specifications. The text files can be maintained directly with source code control applications. Version stamping is used to provide information both during development and testing, but also for problem reporting from the field. The integration with automated testing allows testing new WVS versions against existing instruments. Although more script maintenance may be required, it also allows the testing of instrument changes.

Lessons learned

WVS projects have been successfully fielded and the system has evolved over time. We learned many lessons during this process as noted below.

The Blaise rules engine is remarkably robust. WVS treats it as a black box. We have empirically derived a sense of what activities are expensive and experimented to find techniques to manage the resources.

The Blaise API has provided all the hooks that were needed to build WVS around the Blaise core. It exposed all the objects needed with great efficiency.

There are subtle differences in the Blaise API that can have considerable impact on performance. WVS relies heavily on DEP behavior modes to manage performance as it manipulates the Blaise data model. The timing of when field values are available has proven tricky.

The ability to integrate automated testing has proven invaluable for regression testing new WVS versions. The ability to consistently and uniquely name screen objects is highly desirable.

We think there are many elements of the Blaise 5 design that would simplify or replace many of the WVS implemented features, and we look forward to exploring Blaise 5 further.

Programming and Use of Blaise to SAS Transformation Tool for Streamlining Processing in the Panel Study of Income Dynamics

*April Beaulé and Mohammad Mushtaq,
Survey Research Center, University of Michigan*

1 Introduction

The Panel Study of Income Dynamics (PSID) is a nationally representative longitudinal study of approximately 9000 U.S. families. Since 2001, the PSID has used Blaise as its main software for the data collection instrument. Due to size and complexity of the instrument, the PSID staff developed a tool to assist with extraction and transformation of initial files. This tool, referred to as Data Extraction, Transformation and Loading (DETL), streamlines first step processing of Blaise files.

This paper will discuss the development and use of the DETL tool in the processing cycle, specifically providing an overview of the development and how we improve efficiency and save time setting up first step files for further processing.

2 Background

In the early years of the study, the data collection tool was a paper and pencil questionnaire. In 1992, the PSID automated the survey instrument using SurveyCraft software. By 2001, the Survey Research Center at the University of Michigan began using Blaise and the PSID was re-programmed at that time in 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 2009, the interview length increased to an average of 89 minutes and we have released over 5012 variables in this last wave.

In order to make the processing of first step files more efficient, the DETL tool was developed. This tool allows the user to take advantage of the Blaise meta information including question text, code frames, loop levels and frequencies in order to bundle sets of variables together in a logical way.

The end product from DETL are SAS programs that the user produces by grouping together sets of variables from the basic output tables (extracted at the block level), and sub-setting for certain types of individuals when necessary.

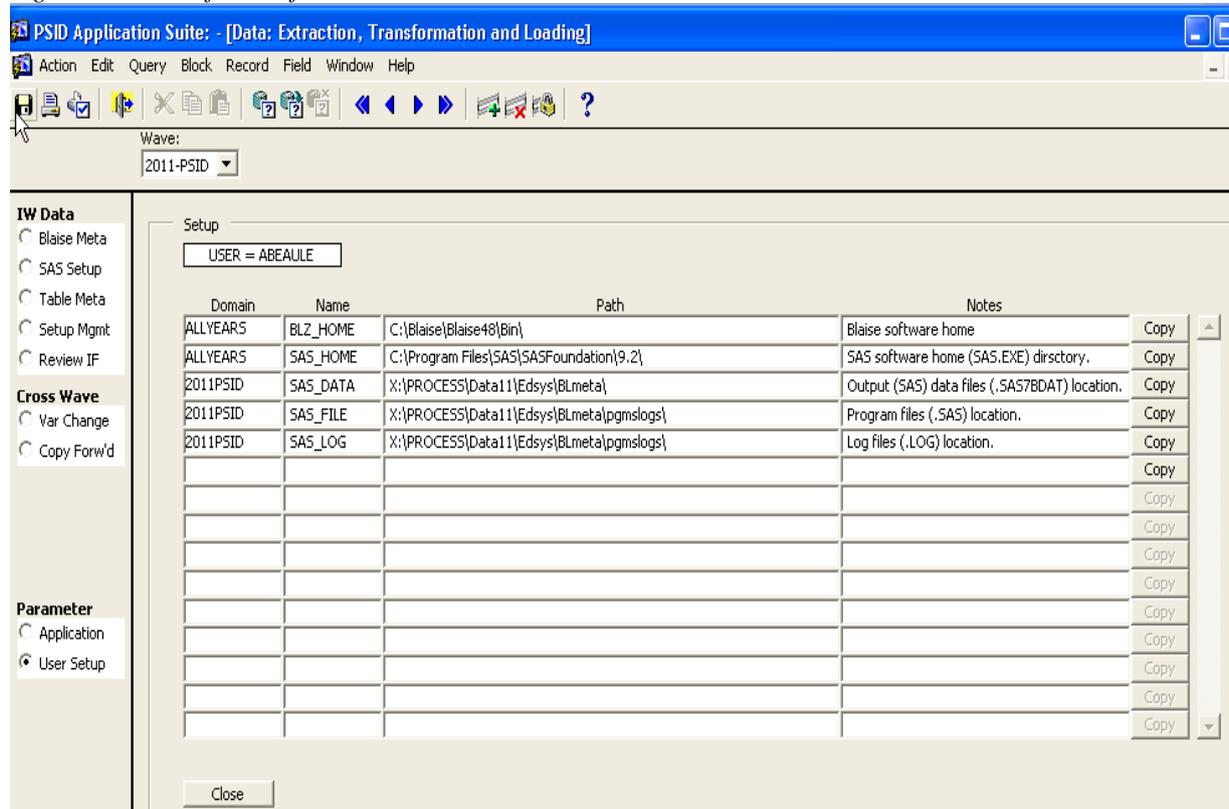
3 Display of Meta Information

The main purpose of first step files is to aggregate variables together at the content level and at the person level. The PSID sample is at the family level but within the instrument we ask more questions about certain people in the family, we label those 'special' people Head and Wife (if there is one). We ask fewer sets of questions about the rest of the family members and we label those individuals Other Family Unit Members (OFUM's). For Blaise programming those questions which are the same for Head and Wife and sometimes OFUM's are looped. However, for processing we have to break them apart. Using DETL we can subset the types of records we want to focus on and label the SAS setups and output resulting from the sub-setting using 'Head' or 'Wife' labels.

The basic extraction from Blaise produces tables at the block level. For Blaise programming purposes often to maximize run time speed, questions are grouped together in blocks. For processing and editing however, we need to group variables in alternative ways.

The first step for the user is to define the directories for a given year. The user defines the location where he or she wants to copy the output data as well as where the .SAS programs and .LOG files are to be written (Figure 1).

Figure 1: User Definition for DETL



Once the user has defined the parameters, they begin by reviewing the meta information about the variables and begin selecting variables for setup. The user clicks the radio button to the left under IW Data, 'Blaise Meta' (Figure 2.1). Displayed under Blaise Meta is a complete list of all the SAS files extracted by block for this BMI. The name of the output file is the name of the block as defined by the Blaise programmer. For PSID in 2009 for the main BMI, there were one hundred and twenty-three tables extracted. Variables from all these tables need to be merged together and sub-set in a coherent format for processing.

To assist the user with choosing the correct variables for each setup, the Blaise Meta screen allows the user to click on a block in the upper portion of the window (Figure 2.1). Down in the bottom pane, all the variables in this block are displayed along with their meta information. Helpful information such as the type of variable (character or numeric), the width, the number of loops and whether 'Don't Know' and 'Refused' were allowed for this variable (Figure 2.2). In the check boxes in the center of the lower pane, there are options to view the question text (Figure 2.1), the code frame, frequencies and the universe for every variable in the block selected.

Figure 2.1: Blaise Meta Display – Question Text

The screenshot shows the SAS Blaise Meta Display interface. On the left, there are navigation options for 'IW Data', 'Crd Wave', and 'Parameter'. The main area displays a list of variables with columns for 'varId', 'Name', 'Label', 'Type', 'Width', 'Dec', 'Loops', 'Min - Max', 'DK', 'RF', 'Qtext', 'Frame', and 'Universe'. Variable 003, 'G33a_1', is highlighted. Below this is a 'Variables' table with columns for 'varId', 'Name', 'Label', 'Type', 'Width', 'Dec', 'Loops', 'Min - Max', 'DK', 'RF', 'Qtext', 'Frame', and 'Universe'. Variable 010, 'G34', is highlighted. To the right, there is a 'Question Text' window for variable G34, showing the question text: '[F1] - HELP How much was the total amount from Social Security? <<G34> w ENTER amount here, then ENTER unit of time on next screen (Month, Year)'. Annotations include: 'Click on the output table SocSec - block defined by the Blaise programmer' pointing to row 059 in the variable list; 'Click to view question text' pointing to the 'Question Text' window; and 'Meta information - question text appears here' pointing to the question text itself.

Figure 2.2: Blaise Meta Display – Frequencies

The screenshot shows the SAS Blaise Meta Display interface with the 'G34 [Freqs]' window open. The 'Variables' table is visible on the left, with variable 010, 'G34', highlighted. The 'G34 [Freqs]' window displays a table of frequencies for variable G34. The table has columns for 'Value', 'Count', and 'Percent'. The first row shows a value of 1 with a count of 2430 and a percent of 93.21. The second row shows a value of 999998 with a count of 90 and a percent of 3.45. The third row shows a value of 999999 with a count of 87 and a percent of 3.34. Annotations include: 'Helpful information displayed including width, loops, whether DK/RF allowed' pointing to the 'Width', 'Loops', and 'DK/RF' columns in the 'Variables' table; 'Displays the frequencies' pointing to the frequency table; and 'User clicks here to see the freqs' pointing to the 'G34 [Freqs]' window.

4 SAS Setups

Once the user has decided on the tables and variables they need based on the meta information display, they turn their focus to defining the SAS steps. There are three basic types of SAS setups available: (a) Long, (b) Wide, and (c) Merge. The Long format setup takes a file that is collected in a wide format and flips it vertically (Figure 3). We end up with a more compact file especially when the data we collect is formatted using our standard 24 person array but we only ever have a couple of mentions.

The Wide format set up transforms in the opposite way. It takes data collected in a looped format and widens it out so that there is one record per case. For example, we use this format for mortgages. We collect a maximum of two, but we release it flattened with mortgage one variables followed by mortgage two variables. The SAS syntax that is required to produce these two transformations is lengthy. DETL is able to produce SAS code quickly, accurately with just a few clicks from the user.

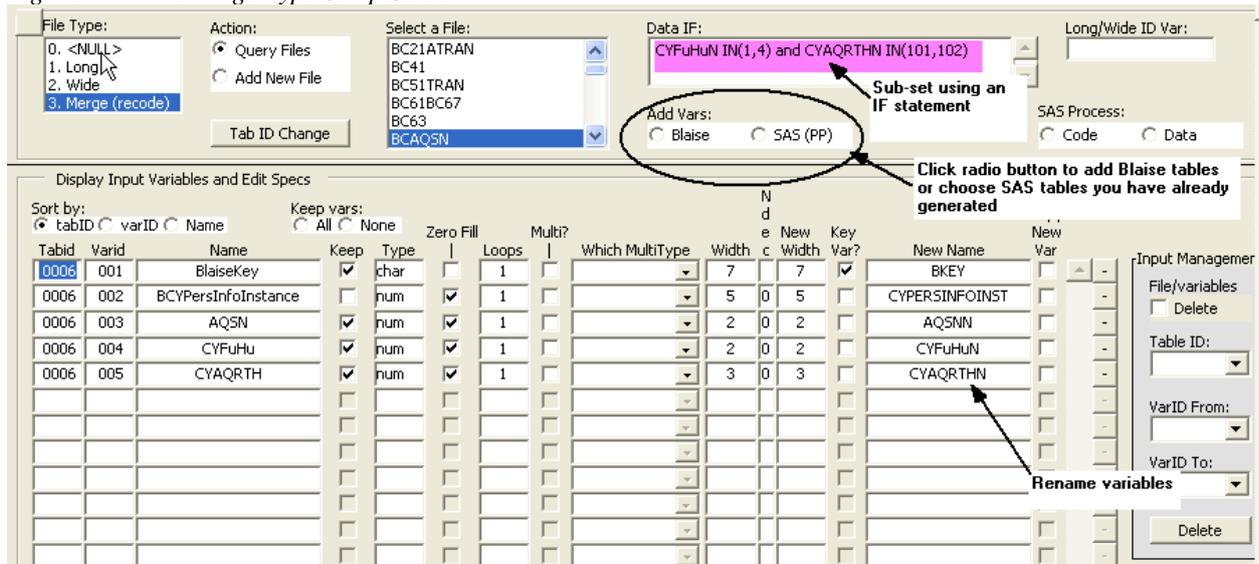
Figure 3: Example of SAS Table Transformation – Wide to Long

VIEWTABLE: Blio.Bdeceased								
	BlaiseKey	BDECEASEDInstance	D1	D2_01	D2_02	D2_03	D2_04	D2_05
1	0004211		1	97	1	.	.	.
2	0041134		1	1	1	.	.	.
3	0079001		1	97	1	.	.	.
4	0088003		1	97	1	2	.	.
5	0117001		1	97	1	.	.	.
6	0129004		1	1	1	.	.	.
7	0165002		1	97	1	.	.	.
8	0173002		1	97	1	.	.	.
9	0297131		1	1	1	.	.	.

VIEWTABLE: B111.D1d3					
	BKEY	DECINST	D1	i1	AQSNN
1	0004211		1	97	1
25	0041134		1	1	1
49	0079001		1	97	1
73	0088003		1	97	1
74	0088003		1	97	2
97	0117001		1	97	1
121	0129004		1	1	1
145	0165002		1	97	1
169	0173002		1	97	1
193	0297131		1	1	1
217	0301003		1	2	1
241	0393001		1	1	1
265	0418001		1	97	1
289	0419002		1	1	1

The most commonly used set up is the Merge type. The user is selecting tables extracted at the block level and merging them with other similar content variables. The Merge type set up is flexible. The user can merge together raw tables directly from Blaise or newly created DETL tables can be merged with Blaise tables (Figure 4). The user must define the merge keys between the tables. He or she can also rename variables, prefill system missing data with zeros, and subset for specific record types. One of the examples in the PSID using a merge setup is the creation of the housing variables. We ask about mortgages and foreclosures up to a maximum of two in a looped format. First we widen the loop and then we merge this wide table back with the anchor table so that we end up with one record for every case with the housing variables even if the family had neither a mortgage nor a foreclosure.

Figure 4: DETL Merge Type Setup Screen



5 Handling Multi-Mention Variables

The PSID has many multi-mention type variables. DETL allows them to be transformed into the output type needed quickly and easily. The first type output we may need is the standard mention order type. The question asks ‘How is your home heated – with gas, electricity, oil or what?’

Figure 5: Multi-mention question

How is your home heated--with gas, electricity, oil or what?

- ENTER all that apply
- PROBE: Any others?
- For multiple response, use space bar or dash to separate responses

<input type="checkbox"/> 1. Gas	<input type="checkbox"/> 6. Solar
<input checked="" type="checkbox"/> 2. Electricity	<input checked="" type="checkbox"/> 10. Bottled gas; propane
<input type="checkbox"/> 3. Oil	<input type="checkbox"/> 11. Kerosene
<input type="checkbox"/> 4. Wood	<input type="checkbox"/> 97. Other - specify
<input type="checkbox"/> 5. Coal	

Heating Method

In DETL, if we like to have the output as mentions, we simply mark the multi-mention type to ‘OK’ meaning ‘as is’ in mention order. But more frequently analysts do not want to see mentions displayed in order of how they were chosen but rather, they want to see if the respondent said yes to any of the options. In DETL, there is 2nd option for multi-mentions ‘Dummy’. If one chooses to transform to a dummy indicator, the output for each mention is displayed as a True/False type (Figure 6).

Figure 7: Multi-mention question – Month String

During which months of 2010 did you get this income?

- ♦ ENTER all that apply
- ♦ For multiple response, use space bar or dash to separate responses

<input type="checkbox"/> 1. Jan	<input type="checkbox"/> 7. July	<input checked="" type="checkbox"/> 13. All
<input type="checkbox"/> 2. Feb	<input type="checkbox"/> 8. Aug	
<input type="checkbox"/> 3. March	<input type="checkbox"/> 9. Sept	
<input type="checkbox"/> 4. April	<input type="checkbox"/> 10. Oct	
<input type="checkbox"/> 5. May	<input type="checkbox"/> 11. Nov	
<input type="checkbox"/> 6. June	<input type="checkbox"/> 12. Dec	

Months in 2010 Rec'd SS Ben

Here is what the SAS data looks like – transformation to a month string.

G35_M
111111111111
111111111111
111111111111
111111111111
111111111111
111111111111
111111111111
000001100000
111111111111
111111111111
000011111111
111111111111
111111111111
111111111111
111111111111
111111111111
111111111111
111111111111
000111111111
111111111111

6 Producing SAS Code, Log and Tables

Once the data manager is satisfied with their setups, they simply press a radio button to either generate the SAS program (so that it may be run from SAS itself) or they can choose to both generate the SAS program, run it producing the SAS table and save the log all at the same time. The program, output table and logs are all saved to the location determined in the parameter setup.

7 Pulling Setups Forward for the Next Wave

As a panel study, the PSID continues to have a need to be able to take advantage of all the data preparatory work in the prior wave. Because a great deal of effort goes into the DETL setups, our programmer was able to bring setups forward into the next wave; the caveat being that variables that were dropped disappear from the setups for the current wave and variables that are new need to be added to the appropriate existing setups.

DETL provides the user with critical between wave reports. The first report indicates which variables have been dropped and which variables have been added. The change report can be generated at the study, bmi or block level. A total of 580 variables were changed from main instrument (PSID_Main.bmi) between 2009 and 2011 waves of data collection, and 687 variables were changed from all bmi's. Figure 8 below lists variables added and dropped from the foreclosure (BA37FOR) module/block. By using this detail report, the setups for the current wave can be carefully edited (Figure 8).

Similarly a second report alerts the user to width changes so that adjustments can be made in the current year setups where the variable name has remained the same but the width has been changed.

Figure 8: Variable change report – between 2009 and 2011

PSID Blaise Variable Name Change Report
(PY = 2009, CY = 2011)
Count = 580

Table	Bmi	Year	Variable	CY/PY Status
BA37FOR	PSID09	2009	A37FOR2Mo	Dropped in PY.
			A37FOR2Yr	Dropped in PY.
			A37FOR3	Dropped in PY.
			A37FOR4	Dropped in PY.
			A37FOR5	Dropped in PY.
			A37FOR5SPEC	Dropped in PY.
			A37FOR6	Dropped in PY.
BA37FOR	PSID11	2011	A37B	New in CY.
			A37C	New in CY.
			A37D	New in CY.
			A37E	New in CY.
			A37F	New in CY.
			A37FSPEC	New in CY.
			A37G	New in CY.

8 Table Number Changes between Pretest and Production

The use of DETL starts immediately when data are delivered from test or pretest stages of data collection instrument or questionnaire. After uploading metadata information into the Oracle database, SAS setups and data files are created to analyze data from various development cycles. Based on this analysis, the Blaise data model changes significantly during an instrument development cycle: for example to add new blocks or variables or update a code frame. These structural adjustments modify the table order and thus void the prior SAS setups created by the data user (Figure 9).

Figure 9: Table number changes between stages of instrument development

If the setup is based on a single table and it has few variables, then it can be updated manually. However if the setup consists of several variables from many tables, manual editing is time consuming and is prone to error. The “Tab ID Change” function of DETL is an elegant way of making such changes. As shown in Figure 9, the “Tab ID Change” function pops up a small window

with three columns. Using Tabid to New Tabid map created from old and new bmi's, the table numbers can be changed by providing values to the New Tabid column. After applying changes, users can create new SAS setups and data files.

9 Miscellaneous Features

DETL collects metadata information about the setup created by the users. As mentioned earlier, users can create setups only or setups and data. The metadata information is displayed in a master-detail relationship as shown in Figure 10 below.

Figure 10: Metadata of SAS setup files

The screenshot shows the 'SAS Setup File Info' window. At the top, there is a 'Wave:' dropdown menu set to '2011-PSID'. Below this, there are radio buttons for 'File Type' with options 'Long', 'Wide', and 'Merge' (selected). The main area is divided into two panels. The left panel, titled 'SAS Setup File Info:', contains a table with columns 'Table name', 'Nvar', 'Nobs', and 'Select'. The right panel, titled 'Variable Info:', shows 'Table Name' as 'A37_B' and a table with columns 'Varid', 'Name', 'Type', 'Width', 'Ndec', and 'Key var'.

Table name	Nvar	Nobs	Select
A37_B	15	8941	<input type="checkbox"/>
BC14A14F	18	8941	<input type="checkbox"/>
BC14CTRAN	2	262	<input type="checkbox"/>
BC14FTRAN	2	262	<input type="checkbox"/>
BC18JOBTYPE	5	11487	<input type="checkbox"/>
BC21ATRAN	3	186	<input type="checkbox"/>
BC41	5	5404	<input type="checkbox"/>
BC51TRAN	3	100	<input type="checkbox"/>
BC61BC67	18	8941	<input type="checkbox"/>
BC63	6	8941	<input type="checkbox"/>

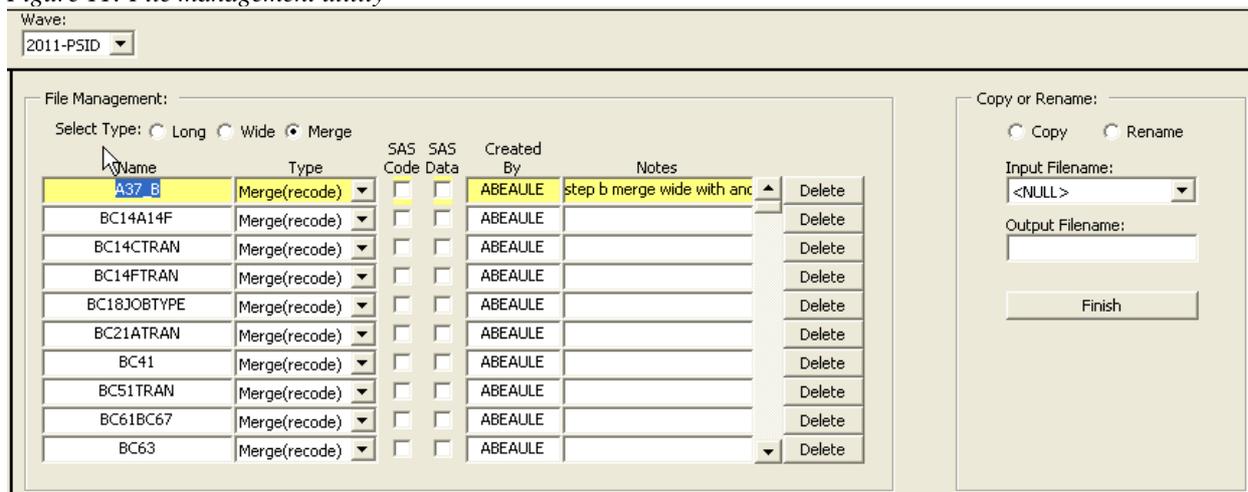
Varid	Name	Type	Width	Ndec	Key var
1	BKEY	C	7		<input checked="" type="checkbox"/>
2	A37BMO_1N	N	2	0	<input type="checkbox"/>
3	A37CYR_1N	N	4	0	<input type="checkbox"/>
4	A37D_1N	N	1	0	<input type="checkbox"/>
5	A37E_1N	N	1	0	<input type="checkbox"/>
6	A37F_1N	N	1	0	<input type="checkbox"/>
7	A37FSPEC1	C	255		<input type="checkbox"/>
8	A37G_1N	N	7	0	<input type="checkbox"/>
9	A37BMO_2N	N	2	0	<input type="checkbox"/>
10	A37CYR_2N	N	4	0	<input type="checkbox"/>

For PSID 2011, there are 203 files created by DETL, out of which 19 are long, 7 are wide and 177 are merge type/format. The number of variables (nvar) and number of observations (nobs) are displayed in the left hand side block of figure 10. The variable information, (order, name, type, width, ndec, etc.), from the selected file is displayed in the right hand side panel. This information is helpful to users as they are able to see the structure of output data tables from the same interface and make changes to the setups if needed.

DETL also offers file management utility, where file setup can be renamed or copied to a new name. User can add processing notes to a file or can delete files that are no longer needed. For further details about this interface. See Figure 11 on the next page.

Another useful feature of DETL is the collection of “data if statements” in one tab together where users can review them with proper context (i.e., along with other files) and make changes if needed. When satisfied, then one can proceed to creating SAS setup and data files again. A modified ‘if statement’ may change the number of observations. The process will update file metadata and users should check meta information before moving to the next steps.

Figure 11: File management utility



10 Data Model and Interface Development (Programming Details)

DETL was developed using Oracle Forms/Reports 6i and Oracle Database 9.2. The development process started in 2003 and the application went into production in early 2004. Later on, it was customized for other studies and additional features were added as needed by the data processing team.

To develop this tool, the first task is to extract metadata from Blaise data model (bmi) and merge it with metadata extracted from the dataout tables (SAS tables). During this process, the looped variables need special attention because there is a one-to-many relationship between the bmi variables and the dataout variables. Then, the next task is to develop relational data model or schema that includes tables, views, functions and procedures.

Figure 12.1 describes the relationship among Blaise-to-SAS tables. A new row is inserted into blz_domain for each wave of data collection. Then blz_tables, blz_variables and blz_fmt_map are populated. Blz_freqs table is updated during data extraction process and information from this table is used to create SAS setups as discussed earlier in Sections 3 and 4. Blz_questions and blz_universe are part of the relationship. The information from these tables is joined with blz_variables using a one-to-many relationship between PVNAME and VNAME. Similarly, SAS format names are extracted from SAS data tables and then information is linked back to Blaise TCodes using blz_fmt_map and blz_formats. It should be noted that the data model is not normalized. Social science data are problematic to model into a normalized relational database and we hope to achieve this goal during next upgrade.

Figure 12.2 describes the relationship among SAS setup tables, specified in Section 4 and further explained in Sections 5 and 6 of this paper. The SAS setup module is using an Oracle database for storage and interview data are in SAS tables. The next step is to integrate the user interface with SAS, so that users are able to send tasks to SAS and get the results back to the calling environment. The bridge between SAS and Oracle is created by a SAS/Access interface to Oracle.

As explained in previous sections, this tool creates a SAS program per user specifications and it is submitted to SAS for data processing. The application uses SAS/Macros in the background to generate SAS code and calls SAS to execute the program.

Figure 12.1: ER diagram of Blaise and dataout tables

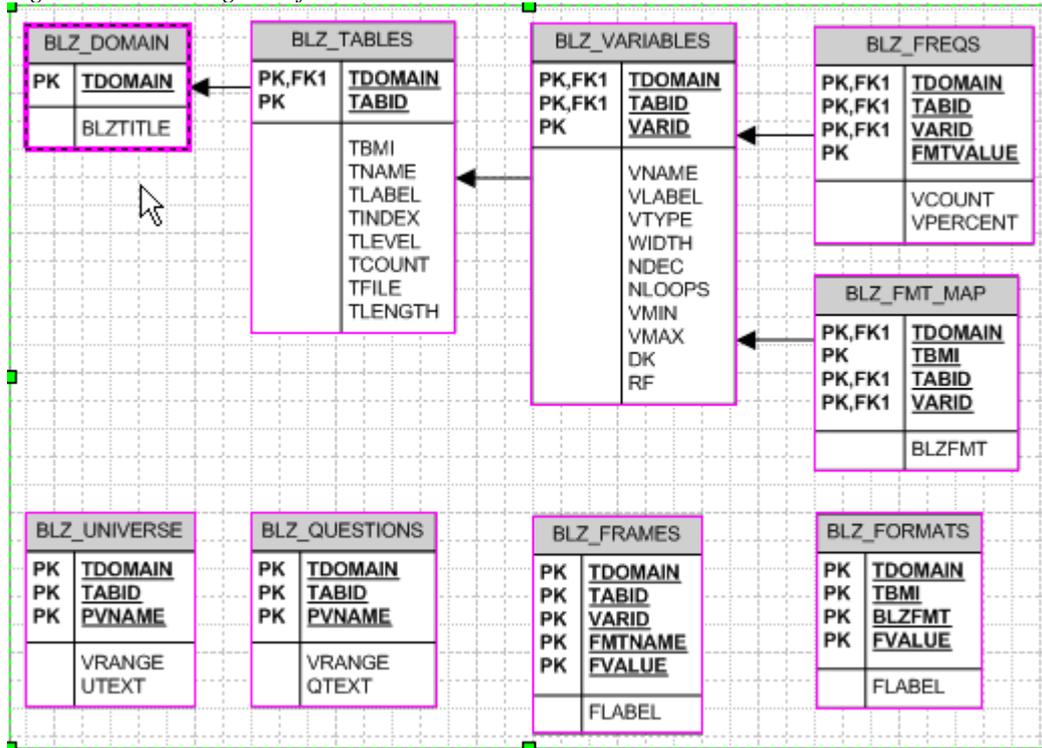
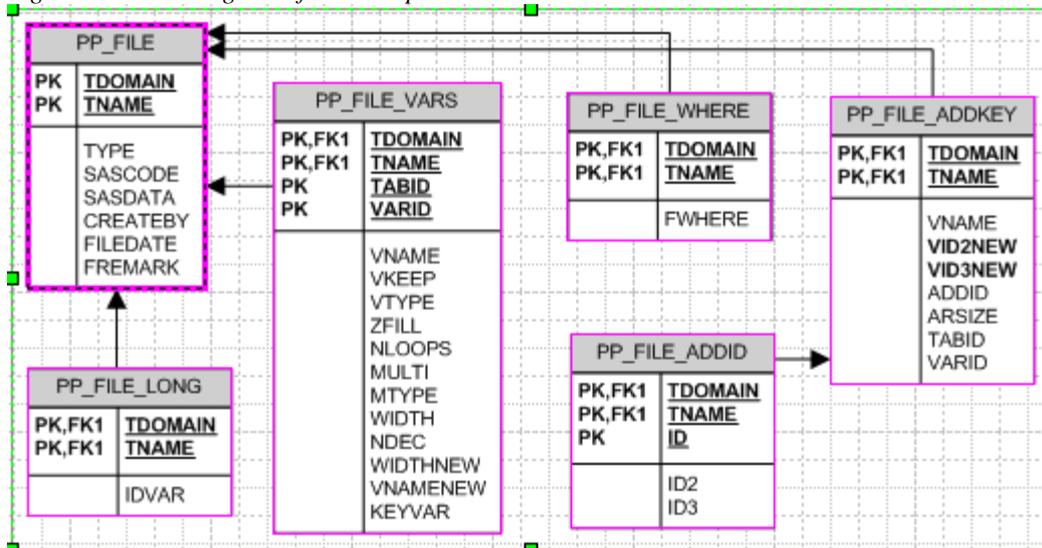


Figure 12.2: ER diagram of SAS setup tables



The SAS data, log and output are written to the folder specified in Section 3. DETL also has a function to check for errors in the log file and report them back to the calling interface.

11 Conclusion and Summary

The DETL tool developed by the PSID programming staff has enabled the processing staff to methodically see all the changes that were made from one wave to the next and update first pass setups accordingly. DETL informs the user about the meta information about each variable including the loop level, the question text, the code frame and even the raw frequencies. Using this information, the user can create first step files and do basic transformations and variable renaming without having to write out the long SAS syntax to complete these.

The SAS programs generated from DETL can be run while also saving a copy of the SAS program and the log file with one click. DETL is an important tool for the processing staff that has saved time and reduced errors and confusion by providing all the Blaise meta information to the user to make informed decisions about variables, where they should be placed, and at what level.

New Tricks with Old Tools

Peter Spark

Survey Research Center, University of Michigan

Blaise has a wealth of utilities, tools and features packaged within its system, and multiple ways of extracting and using information from datamodels, databases, modelib files; e.g., datamodel properties and so forth. Such tools as Manipula/Maniplus, Cameleon, Registry Editor, Delta, Data Centre, BCP, “watch” window, and more allow authors and managers to get the in-depth information they need to effectively perform their job.

However, the Blaise environment allows even richer capabilities beyond typical methods to produce a quality survey. Such features as alien routers and procedures (using BCP and/or Manipula), datamodel and data XML, Modelib, native and BOI databases, menu files, etcetera, give even more control. With such information, methods can be used to analyze current programming and provide needed details that help make decisions in the hectic survey programming environment.

This paper examines the capabilities of existing tools in the Blaise suite and explores practical new ways to make use of them, as well as ways of extracting and using information in ways that are relevant to both authors and managers in the Blaise 4.8 environment. Crosswalks, counts, finding unused code and questions, complexity calculations, and more are the sort of tricks that can be added to the survey tool bag that reduces errors and decreases turn-around time.

1. Blaise Datamodel Information

Before analysis of Blaise datamodels can begin, it is useful to review the list of current tools and their practical applications to extract datamodel information. The Blaise datamodel, for the most part, is the most useful source of information. It has the advantage of being syntactically correct according to the Blaise parser. However, certain information is lost, such as comments and source code formatting, and other information is more difficult to retrieve, such as the code within procedures, and rules text.

The source code more easily provides all this other details, but requires a custom parser that mimics the behavior of the Blaise parser. In many cases the source code is not available, or the proper version does not prepare to a datamodel, or some other vital file could be missing. This paper concentrates mainly on existing utilities that read the datamodel, although it is possible to read the source code with custom parsers written in a high level language or Manipula.

The goal is not to reinvent the existing utilities but to take advantage of them. Otherwise it would be just a matter of writing a new utility using the BCP and a high level language to extract all the needed information. The disadvantage of such an approach is the need for in-depth programming skills. Intermediate methods, such as Manipula and Cameleon, may also require programming skills, while other utilities provided by Blaise just need a skilled user.

The table below shows a few of methods that can be used to retrieve data and metadata from the Blaise datamodel.

Method	Data	Meta	Notes
BCP	Full case data, comments	Full meta information	Original comments and formatting lost, requires advanced programming skills, essentially creating a new utility.
Blaise Data Centre	Filtered/full case data	Can export data selected on fields/row filters to text, BDB,	Can also get Blaise-generated source code for the dictionary. ¹

Method	Data	Meta	Notes
		or XML	
Blaise Registry Editor	Related to settings for various files	Settings information and some metadata (i.e., BXI: All defined types, INI format, version and copyright info)	Supports reading BOI, BTR, BIC, BIS, BFS, DBI, BIN, IDM, BMF, MSX, BXML, BCI file types. Use File – Export Registry to get a BRF (INI format).
Cameleon	No data	Defined fields, blocks	May require additional routines to create a report. TechDesc.cif - Gives technical information about the datamodel. Such as number and length of fields, number of parameters for each block.
Datamodel properties	No data	All defined types, INI format	Use – Types – Export. Can be used as part of an input for other utilities. Does not include codes, ranges
Delta	No data	Partial metadata (fields, rules, used types, calls) Use special stylesheet to export.	Need to rename html files to xml, and change utf-16 to utf-8. May be a Delta bug to always produce separate files. XSLT could merge all these files together with document().
Manipula	Full case data, comments	Partial meta (defined and used fields, blocks, used types)	Not a “utility” but provides an easier-to-use method of retrieving both data and meta outside of the BCP. This can also be used to read Blaise source code and parse for information.
XSD Schema Wizard	No data	Partial metadata (defined fields, used types, blocks)	Matches XML from Blaise Data Centre.

¹Note: escaped quotes (two double quotes to make one double quote upon export) within question text are not handled correctly – they are not escaped in the generated source code (4.8.4.1737). It can be manually fixed, and once the datamodel is prepared it can be used for the XML export.

1.1 BCP

The Blaise Component Pack is the API to the Blaise datamodel and database. It gives full access to the Meta information with (Field) and without (FieldDef) a connection to the data. It is most useful in building new utilities and is the core of the Blaise system, but by itself cannot be considered a utility. Its flexibility and inherent richness also mean that it is technically much more challenging to program.

1.2 Blaise Data Centre

The Data Centre is excellent for selecting data records and columns from Blaise data, filtering the results, and exporting the information in a variety of formats. It can create a new datamodel based upon the selected fields, export the data in XML, and create BOI files to the data. It can be thought of as a Blaise to Data extraction tool.

The XML file can be used as an input when looking at the data and Meta information, but it is not very rich in details, and so would have limited use for certain applications. The data is only stored per element if there’s something to store. That is, skipped questions don’t appear. The data is also in the order of the fields defined in the datamodel. Essentially, this is the Blaise to XML export run by the Data Centre interface.

1.3 Blaise Registry Editor

The Editor is a very handy utility that examines a variety of formats:

BXI	Blaise Extended Meta (datamodel properties)
BOI	Blaise OLEDB Interface
BTR	Blaise Cati Specification
BIC, BIS, BFS	Blaise IS Specifications
DBI	Blaise Database Info
BIN	Binary files
IDM	Blaise IS ID Mapping
BMF	Blaise Menu Files

would remove the Delta-related elements, such as <RichText> and <htmlproperties> elements, and remove the html formatting of the <cat_rules> element.

The information can be stored from Delta by selecting File – Save – Save all details. This would store an html header file, and then a separate html-named file for each node of the tree. These files are actually xml files, and need a small adjustment. First, the filename extension needs to be changed from .htm to .xml. Secondly, the statement

```
<?xml version="1.0" encoding="utf-16"?>
```

should become

```
<?xml version="1.0" encoding="utf-8"?>
```

Another stylesheet, or Manipula script, or other utility should be sufficient to combine all the separate files into one XML file and change the encoding to utf-8.

Delta has the advantage of also exporting the rules text for each block in the datamodel (but not for the datamodel itself). This information is stored in the <cat_rules> element for blocks, but is html-based.

For example, here is a question from a small questionnaire:

```
DoWalking (B1)
  "Do you do any appreciable amount of walking?"

  TYesNo
```

And here is an excerpt of Delta XML for the same question:

```
<qnode caption="DoWalking" position="2.1" positionf="2_1" captiontype="QUESTION"
  imgsrc="quest.gif" imagepath=".">
  <cat_fielddef fieldnamestripped="SubQs.DoWalking" fieldname="SubQs.DoWalking"
    fieldkind="DataField" fieldtype="Enumeration" typename="TYesNo" fieldsize="1">
    <field_descriptives dummy="neverhide"/>
    <field_specifications dummy="neverhide"/>
    <question_field tag="B1">
      <fieldattributes empty="false" dontknow="true" refusal="true"/>
      <field_description/>
      <field_text>
        <langtext langid="ENG">
          <origrichtext>Do you do any appreciable amount of walking?</origrichtext>
          <RichText FontFamily="Arial" FontSize="12" FontColor="rgb(0,0,0)"
            FontBold="false" FontItalic="false" FontUnderline="false">
            <RichTextLine Align="Left">
              <RichTextElement Text="Do you do any appreciable amount of walking?"
                />
            </RichTextLine>
          </RichText>
        </langtext>
      </field_text>
    </question_field>
  </cat_fielddef>
</qnode>
```

```

<field_answers>
  <answers_closed>
    <answer>
      <name>Yes</name>
      <code>1</code>
      <langtext langid="ENG">
        <origrichtext>Yes</origrichtext>
        <RichText FontFamily="Arial" FontSize="12" FontColor="rgb(0,0,0)"
          FontBold="false" FontItalic="false" FontUnderline="false">
          <RichTextLine Align="Left">
            <RichTextElement Text="Yes"/>
          </RichTextLine>
        </RichText>
      </langtext>
    </answer>
    <answer>
      <name>No</name>
      <code>5</code>
      <langtext langid="ENG">
        <origrichtext>No</origrichtext>
        <RichText FontFamily="Arial" FontSize="12" FontColor="rgb(0,0,0)"
          FontBold="false" FontItalic="false" FontUnderline="false">
          <RichTextLine Align="Left">
            <RichTextElement Text="No"/>
          </RichTextLine>
        </RichText>
      </langtext>
    </answer>
  </answers_closed>
</field_answers>
</question_field>
</cat_fielddef>
<cat_statement_text>DoWalking.ASK</cat_statement_text>

```

1.7 Manipula

Manipula is capable of pulling an extensive list of data and Meta items from the datamodel, but not the rules text and some information on external files. It can, however, retrieve fields in the order of the rules as well as fields in order of their definitions.

Writing manipula scripts to accomplish the work is not as complex as writing a program for BCP, but still involves good working knowledge of how to navigate through the structure of the datamodel. It also has the advantage of being flexible, easily updated, and current with new builds of Blaise by simply preparing the script again.

The capabilities of Manipula have been expanded for some time, and `Getfieldinfo()`, `Getmetainfo()`, and `Gettypeinfo()` are invaluable for retrieving the datamodel meta information.

1.8 XSD Schema Wizard

The XSD Schema Wizard exports an XML schema for a Blaise datamodel. The field information is based upon the list of defined fields rather than the fields actually used within the datamodel, and are missing some pieces that may be useful: field and block sizes, description texts, and parameters. However, it appears the schema generated is compatible with the Manipula Blaise to XML file format.

2. Crosswalk

Crosswalks are a listing of all variables within a program and how they're used: within functions, assignments, questions asked, shown, kept, referenced, fills, and so on. They are essential tools for authors by identifying variables that are no longer being used, or ones that potentially could cause confusion, such as the same named variable used within different blocks. They help pinpoint old code that should be removed, or new code that has not been fully implemented.

Crosswalks can be used in different environments, but for this discussion the DEP datamodel will be used.

Let's assume that we have the Meta information in a format that is usable from one or more of methods described earlier, and the information is complete. Then the type of crosswalk that could be created from them could be classified according to the type of metric, such as fields, assignments, or parameters. Below is a table summarizing the sort of crosswalk items that should be possible to extract and use for documentation.

Item	Crosswalk metric (references)	Counts Metric (#)	Notes	Metric can be found from
Alien Procedure	Reference in RULES	# references	Independent of fields/auxfields	BCP Delta
Alien Router	Reference in RULES	# references	Always associated with a block	BCP Delta
Arrays	ASK KEEP SHOW Assigned Calculations	Total # # ASK # KEEP # SHOW # Assigned # calculations	Usually requires loops. A structure used on blocks, fields, auxfields, and locals.	BCP Cameleon Manipula Delta XSD Data Centre
Auxfields	ASK KEEP SHOW Assigned Fill reference Calculations Languages Question text Description Tag DK RF EMPTY	Total # defined Total # used # ASK # KEEP # SHOW # Assigned # used in fills # calculations Length of all question texts # descriptions Length of all description texts # codes # tags # DK # RF # EMPTY	Temporary field, does not store data	BCP Cameleon Manipula Delta XSD Data Centre
Blocks	KEEP ASK SHOW Assignments	Total # # ASK # KEEP # SHOW # assigned # parameters Deepest level # embedded	Complex type structure. Can have block assignments	BCP Cameleon Data Centre Delta Manipula XSD
Checks		# CHECK keyword # SIGNAL keyword # checks # signals	Compare # keyword occurrences vs. check/signal to look for inadvertent type	BCP Cameleon (checks/signals) Delta

Item	Crosswalk metric (references)	Counts Metric (#)	Notes	Metric can be found from
Datamodel		Languages Parallel Blocks Key fields Embedded blocks		BCP Cameleon Delta Manipula
Externals	Declaration to external database	# references	Needed before using SEARCH, LOOKUP statements in the rules	BCP Cameleon (external parameters) Manipula Delta (lookup and external field kind).
Fields	ASK KEEP SHOW Assigned Fill reference Calculations Languages Question text Description Tag DK RF EMPTY	Total # defined Total # used # ASK # KEEP # SHOW # Assigned # used in fills # calculations Length of all question texts # descriptions Length of all description texts # codes Length of all code texts # tags # DK # RF # EMPTY	Stores data	BCP Cameleon Manipula Delta XSD Data Centre
Layouts		# layout sets # page breaks # infopane refs # fieldpane refs # grid refs		BCP Source code
Locals	Assigned Used in fills, calculations	Total # # assigned # calculations # used in fills	Loop counters, fills, calculations	BCP Manipula Delta (used)
Parameters	Assigned Used in fills, calculations	Total # # IMPORT # EXPORT # TRANSIT # generated # assigned # calculations	Treat like locals. Can be explicit or generated	BCP Cameleon Manipula Delta
Procedures	Declarations	# definitions # parameters	Can call other procedures, be used as alien routers/procedures	BCP Source code Delta (used)
Rules	N/A	# lines of codes File references # conditions # functions # methods # procedure calls # loops # assignments		BCP Delta (not for datamodel level)

Item	Crosswalk metric (references)	Counts Metric (#)	Notes	Metric can be found from
Types	Defined Fields Auxfields	# defined # in Fields # in Auxfields Per kind (Named, Block, Date, Classification, Enumerated, Open, String, Set, Integer, Real, Time, Dummy)	Can be declared but never used	BCP Datamodel properties Cameleon Manipula Delta XSD
Uses	Declaration to external datamodel	# references	Needed before using EXTERNALS statement	BCP Source code

Item refers to the crosswalk metric, the references column to how the metric is used, the counts column to the number of times the metric is encountered (discussed more below), a few notes, and finally the places the metric could be obtained.

For example, if we were to look at the Locals item and its crosswalk metrics we could get some output from a utility that might look like this:

Locals	Assigned (line)	Referenced (line)	Code
Counter	119 121	129	Counter := 0 Counter := Counter + 1 IF Counter > 5 THEN
I	120	122 130 132	FOR I := 1 TO 10 DO IF HHL[I].Name = EMPTY THEN xFill := HHL[I].Name xFill2 := HHL[I].Name + ','
J	140	135	IF J > 2 THEN FOR J := 1 TO 10 DO
K	--	--	--

The line number would ideally refer back to the source code, but potentially could also reference a generated document that contains “new” source code.

Then information in the results table could be interpreted, with potential problems flagged:

Variable Counter is assigned twice, used once
Variable I is assigned once and referenced three times
Variable J is referenced once, and assigned only once. ****reference before assignment****
Variable K. **** Never assigned, never referenced****

Common loop variables potentially could be used in many places and be used before they are initialized, such as for variable J. An automated report could indicate suspect situations that may be hard to catch otherwise.

Another view of the features of each method and the Meta information they contain are shown below.

Meta	BCP	Data Centre ¹	Datamodel Properties	Delta	Cameleon	Manipula	XSD
Arrays	X	X		X	X	X	X
Block	X	X		X	X	X	X
Block size	X			X	X	X	
CHECK, checks	X			X		X	
Codes (text, value)	X			X	X	X	X

Meta	BCP	Data Centre ¹	Datamodel Properties	Delta	Cameleon	Manipula	XSD
Conditions	X			X			
Datamodel attributes	X			X		X	
Datamodel languages	X		X	X	X	X	X
Datamodel name	X			X	X	X	X
Datamodel title	X			X	X	X	X
Dictionaries used	X			X			
Externals	X			X			
Field attributes	X	(data)		X	X	X	
Field (name, text, ...)	X	(name)		X	X	X	X
Field kind	X			X	X	X	
Field long name	X			X	X	X	X
Field size	X			X	X	X	
Field/Block type	X			X	X	X	X
Field tag	X			X	X	X	X
Parameters	X			X	X	X	
Parallels	X		X				
Primary, secondary keys	X			X	X	X	X
Rules	X			(no root) ²			
SIGNAL, signals	X			X		X	
Status bar	X		X				
Types	X		X	X	X	X	X

¹Data Centre only exports data or creates a datamodel based upon the data to be exported. The meta information is sufficient for this need. Hence, only the basic field name within the block structure can be retrieved. Only when DK/RF values entered on a field will it show in the data.

²Delta does provide a version of the rules, but the export using a simple dump of the information has already been formatted into html. A slightly more complex stylesheet to convert this to xml would make it easier to use. Delta currently does not export the top-level rules of a datamodel (build 4.8.4.1737).

3. Metric Counts

It's easy enough to find the width of the data record via the structure browser, but what's the answer to the total number of variables and number of each type? How many assignments are made within the code, how many questions are asked, how many arrays are there, the number of blocks, etc.?

Metric counts refer to the number of times a metric is encountered within the datamodel. Blaise comes packaged with an existing Cameleon script, TechDesc.cif that reports a number of these metrics within the data structure of a datamodel. However, there is more information that can be retrieved in a variety of ways. The table for the crosswalk also has a column for the counts metric.

3.1 How to use the counts

Counts by themselves don't mean anything, but can be used as a complexity measure, or could be used to compare two datamodels to see how much effort (as a percent) it would take to bring one datamodel up to another, or how a datamodel has changed over time. Certain measures, such as generated parameters, could help indicate areas to improve to decrease hidden overhead.

Counts could be used to locate metrics that are overused as well as those underused. The former would be useful to point out areas that may need a reduction of complexity, while the latter may be areas that could be removed.

4. Finding unused items

During the course of development, and when working with succeeding years of the same survey, changed over time, there is a potential for having old fills, auxfields, locals, and types remain when

they should be removed. And for new development, fields often can be declared but can be missed in testing. The question is how can we find these items?

4.1 Unused items: fills, auxfields, locals, types

An unused fill occurs when a variables (such as an auxfield or local) has been declared, and perhaps even is assigned but is never used within question text or the construction of another fill, or used as a parameter to a block or procedure.

For example, it's not too hard within the Blaise Control Centre to locate unused fills manually (enhanced search on each fill declaration and look to see if there are any references in the rules or displays in texts), but an automatic routine could locate these unused variables more quickly. The unused variables take up space in both the source code and in the datamodel when running, and removing unneeded variables would improve the load time (probably just marginally) for a survey. More practically, it makes the code tighter and easier to maintain.

Once the metrics and counts have been created for a datamodel, it would be a simple thing to look for variables that are assigned but never used as a fill in question text, assigned and never referenced, or never assigned or referenced.

Likewise, an overabundance of unused types makes the code more complex than necessary. (Types are linked to the datamodel properties, and removing a type from the program also removes the entry from the BXI file.) However, Blaise does allow libraries with the express purpose of providing standard types across a number of surveys, and so removing unused types may not fit the organization's methods.

To find the unused types is a two-step process. First, all the defined types can be examined, and then all fields, auxfields, and parameters are examined for their types. The two lists are then compared. If a type has been declared but isn't used it should appear in the first list but not the second. A report could then be created showing these discrepancies, with either the line number or at the block where the unused types are defined.

4.2 Unused fields

Questions that are defined but never used could be a problem in the making. Either they were purposely removed from the logic and never removed from the FIELDS declaration, or they were accidentally removed. And what could be worse than to field a study without collecting data on critical questions? Finding unused questions can be done manually by the same process as finding unused fills by searching for references on each question. This is a time consuming process.

Unused questions can be found by running in a similar manner as other unused code: run a metrics and counts report and then look for questions that have a zero count for being ASKed, SHOWed, KEEPed, or used in a reference.

Blaise has generated statements at the end of each block for fields/auxfields that have been defined but have not explicitly been ASKed/SHOWed in the rules (mostly for assigned fields/auxfields). This is another area to examine in order to look for potential problems.

5. Code complexity

Another use of the metric counts is to help calculate the complexity of a Blaise program. By using a series of measures, the datamodel can be analyzed and comparisons made between authors, instances of a survey, and similar or different surveys.

For example, it would be safe to say that if a survey has few IF..THEN statements it will be relatively simple, and one that has 50% IF..THEN statements versus asked questions it is more complex, and

one that has 200% IF..THEN statements would be even more complex, and one that has in addition many levels of conditional embedding would be very complex.

There are a variety of measures that can be used to measure Blaise datamodel complexity. The practical application of knowing the complexity of a datamodel is to be able to relate that to the amount of effort to modify, add, or remove code, or to develop similar surveys.

The complexity of a datamodel is determined by a number of metrics. They can include, but are not limited to:

- Number of questions
- Number of fills
- Lines of code (rules)
- Number of preload variables
- Number of decision points
- Number of procedure calls
- Number of languages
- Number of blocks
- Depth of blocks
- Number of operands in an expression

Furthermore, some metrics are linear, such as a series of ASKed questions, and other metrics should be exponential, such as the depth of decision points in the rules. Therefore, a program with a series of linear decision points is simple, but a series of decision points that are embedded become complex very quickly.

A weight should be applied to each metric. In general, linear items, such as the number of questions, should have a weight of one. Decision points should be exponential, and such things as languages should be factored as a multiplier of complexity, as shown in the table below.

Metric	Weight	Symbol
Number of fields asked/showed	1	Q
Fills displayed	1	F
Number of assignments	1	A
Lines of code (rules)	1	R
Number of preload variables (used/displayed)	1	P
Depth of each decision point	2	2^D
Number of procedure calls	1	C
Number of fields in blocks	1	B
Depth of blocks	1.5	1.5^K
Number of languages	1	L

Therefore, one possible measure of complexity for a survey would become

$\text{Complexity score} = (Q + F + A + R + P + (2^D) \dots + C - B + (1.5^K) \dots) * L / Q$

This reads as the summation of (the number of questions, number of fills, lines of code, preload variables, each decision point's depth, procedure calls, each block's depth, and operands) multiplied by the number of languages defined in the datamodel.

Blocks at the top level of the datamodel would have a score of 1 (1.5^0), a block within a block then has a score of 1.5 (1.5^1), and another block within that would have a score of 2.25 (1.5^2).

A condition, such as IF A = 1 THEN, would have a score of $2^1 = 2$, and another condition within that would have a score of $2^2 = 4$, and so forth. The ELSE portion of an IF would count as a condition.

For example, suppose we had a very simple datamodel with no fills, blocks, preload...

```
A
B
C
```

Score is 2: $(Q=3 + F=0 + A=0 + R=3 + P=0 + D=0 + C=0 - B=0 + K=0) * 1 / 3$

A slightly more complex datamodel:

```
A
IF A THEN
  B
ELSE
  C
ENDIF
```

Score is 4.3: $(Q=3 + F=0 + A=0 + R=6 + P=0 + (D=2^1 + 2^1) + C=0 - B=0 + K=0) * 1 / 3$

And even more complex:

```
A
IF A THEN
  B
  IF B THEN
    D
    SomeVal := 1
  ELSE
    E
  ENDIF
ELSE
  C
ENDIF
```

Score 6: $(Q=5 + F=0 + A=1 + R=12 + P=0 + (D=2^1 + 2^2 + 2^2 + 2^1) + C=0 - B=0 + K=0) * 1 / 5$

Using these examples implies that the third datamodel is three times as complex as the first, and about 50% more complex than the second.

The number of fields in blocks has been expressed as a positive element by subtracting these values from the score. That's because breaking down the datamodel into parts is beneficial, and although embedding blocks within each other is useful it does raise complexity.

Of course, this oversimplifies a number of things about producing a survey and the environment it is intended to be in. It leaves out other measures that are worthy of consideration: number of lines used

in layouts, web or CATI, external lookups, alien routers, procedures via DLLs and Manipula, and so forth. This measure is intended as a starting point for further exploration and discussion.

Although the complexity score as discussed is at the datamodel level, this can be applied against a single block or procedure. Then scores could be computed in parts and those areas of programming that get a high complexity score are areas that should be reviewed.

6. Reducing errors

Even the best authors and the most careful checking by an expert team of testers will miss mistakes and errors. Not only will these have an impact for data collection, but when it comes time for making documentation, creating codebooks, running analysis, and so forth there will be problems found that would have been corrected if only they had been caught. Being able to detect and correct these sorts of errors is beneficial all around. Below are a few examples that can be created once a metrics and counts analysis has been generated.

6.1 Inadvertent checks and signals

These are situations where an assignment was intended, but instead a check was created by a typo. For example, $A = 1$ is a check while $A := 1$ is an assignment. These can be located by the counts and metrics by looking for all the checks with defined error text.

6.2 Ill-defined lengths

Blaise checks all simple string assignments and numeric assignments against the width of the variable being assigned when the datamodel is prepared. However, the parser does not catch situations where the operand is made of a series of constants, such as $x\text{Fill} := \text{'this'} + x\text{SomeOtherFill} + \text{'that'}$ for a fill, or $x\text{Number} := 1 + 5$ for a numeric expression. It is possible to calculate the length of each operand of an assignment and therefore the overall length needed for a fill, or the possible ranges needed for a numeric assignment. Catching these during the development period would reduce the number of errors that could occur from truncated fills or “imputation errors” that happen during data collection.

6.3 Missing Parts

Different parts of the source code can be missing by accident or on purpose, and while not critical to making a working survey, these are things that could have impact to authors, interviewers, researchers, analysts, and others before, during, and after data collection. Making sure everything is completed is time consuming up front but could reap huge savings afterwards.

Some areas that can be reported by the metrics and counts report that would help pinpoint areas that need work include:

- Texts for each language (Questions, Descriptions, Blocks, Checks/signals, Codes)
- Datamodel text (header)
- Explicit numbering in code frames
- Language identifier for each text
- Layouts for each language
- User-defined types for each field
- Detection of generated parameters (implied parameter)

Needless to say, programming time available for projects is rarely in excess. However, if these areas and other like them are addressed then time spent in debugging should decrease. And during analysis trying to figure out what happened should also decrease because the datamodel will be well-documented.

7. Conclusion

The large variety of existing tools and utilities within the Blaise system makes it possible to retrieve a number of metrics about the Blaise datamodel. Using these metrics then allows authors to produce a higher quality datamodel that fulfills immediate and future needs, reduces errors, and should decrease development time. They would also facilitate estimating programming efforts and costs for future Blaise surveys of similar types.

References

Rhonda Ash and Danilo Gutierrez, Longitudinal Survey Data – “Move It Forward”, IBUC 2010 Proceedings pp.195-200.

Barbara S. Bibb, Lillie Barber, Ansu Koshy, Coding Tricks To Save Resources, IBUC 2010 Proceedings pp.188-194.

Rebecca Gatward, An evaluation of Delta, a documentation tool, IBUC2004 Proceedings pp.389-410

Jean-Pierre Kent, Performance and Design, IBUC1995 Proceedings pp.73-102.

Peter Sparks, A Systematic Approach to Debugging in the Blaise Environment: An Author's Perspective, IBUC2009 Proceedings pp.185-195.

Margaret Tang and Daniel Collison, Blaise Testing , IBUC 2007 Proceedings pp. 109-116.

Using metadata in Manipula and Maniplus

G J Boris Allan, Richard Frey, Jim O'Reilly

Westat

International Blaise Users Conference, April 2012, London, UK

***Summary:** Traditionally in Blaise Manipula and Maniplus have been used to examine data and Cameleon to examine metadata. We discuss here several ways of integrating and extending these tools to address challenging operational issues related to capturing and reusing metadata, manipulating data across rounds of longitudinal studies, and identifying key parameters of partially completed cases.*

Introduction

Since metadata information can now be accessed from Manipula, it is possible to more fully use the power of this capability in Blaise. In the past we used one type of tool to look at data (Manipula/Maniplus) and another tool to look at metadata (Cameleon). As Cameleon is able to produce text files as output, one can use Cameleon to generate a Manipula program (which is text) to use the metadata extracted by Cameleon to examine a database. We illustrate this technique in three applications:

- Removing remarks in rollover programs
- Finding where instruments stop
- Automatic conversion of Blaise data to a SAS dataset

Example code for using Cameleon to create a file with Manipula source code is given in the appendix.

Removing remarks in rollover programs

Longitudinal surveys typically require that data from one or more previous rounds be carried over to the current round. When data are assigned from one field to another, any remarks associated with the field are also copied into the current round. This means that fields in the current round are already unintentionally supplied with comments from the prior round, and thus it is difficult to distinguish current remarks from previous round remarks. One solution is to set the remarks for all fields in the current database to empty as part of the data assignment process for the new round. An example of the Manipula procedure used to clear out the prior round remarks (called in the MANIPULATE section by `RemoveRemarksBlockProc('')`) is as follows:

```
PROCEDURE RemoveRemarksBlockProc
PARAMETERS
  BlockName : STRING

AUXFIELDS
  FieldName : STRING
  SectionName : STRING

INSTRUCTIONS
  FieldName := Present.GETFIRSTFIELDNAME(BlockName)
  REPEAT
    IF (Present.GETFIELDINFO(FieldName, 'BASEFIELDTYPENAME') = 'BLOCK') THEN
      RemoveRemarksBlockProc(FieldName)
    ELSE
      Present.PUTREMARK(FieldName, '')
    ENDIF
    FieldName := Present.GETNEXTFIELDNAME(FieldName)
  UNTIL (FieldName = '')
ENDPROCEDURE
```

This code is set up to attempt to emulate Cameleon metadata loops. Note that `Present` is the current update/output file. The code performs the following actions:

- Identifies the first field name for the named block (the initial call specifies an empty block, taken to be the top level).
- Repeats the looping until there is no next field name (end of block).
- Calls the procedure with the new block name, if the field is a block.
- Makes the remark empty if the field is not a block.
- Gets the next field name.

We could also use a similar procedure to output remarks as shown in the Cameleon and Manipula combination shown in the appendix.

Finding where instruments stop

Studies, particularly those with long and complex instrumentation, often define a partially completed case status. For example, the interview has been mostly completed, but not totally completed. In this study the partially completed were given an operational status between 80 and 89. Since post-collection processes need to identify where exactly the data collection ended, the last question in the interview has to be located. The following program identifies this information for all partially completed cases in this study.

```

SETDESCRIPTION(ParentID)
IF (SUBSTRING(CAPIStatus,1,1) = '8') THEN
  InputFile1.CHECKRULES
  FieldName := InputFile1.GETNEXTROUTEFIELDNAME('', 'ASK')
  REPEAT
    OutputFile1.EndFieldName := FieldName
    FieldName := InputFile1.GETNEXTROUTEFIELDNAME(FieldName, 'ASK')
  UNTIL ((FieldName = '') OR ((InputFile1.GETVALUE(FieldName) = '' ) AND
    (InputFile1.GETVALUE(InputFile1.GETNEXTROUTEFIELDNAME(FieldName, 'ASK')) = '' )))
  OutputFile1.WRITE
ENDIF

```

The steps are:

- Select only partial completes based on the operational definition.
- Check the rules for that case.
- Get the name of the next field on the route that is an ASK field, starting from the beginning.
- Repeat the loop for the next section of code.
- Store that the current field name in the output file buffer (as EndFieldName).
- Get the name of the next field on the route that is an ASK field.
- Repeat the code loop unless there are no more fields on the route with data.
- Write the contents of the output file buffer.

An example of the output is as follows:

```

XXXXXXXX|CFQ.CFQ310_F
XXXXXXXX|HEQ.HEQ570c_F
XXXXXXXX|NRQ.NRQ266_F
XXXXXXXX|Fsq.OriginsKeyParent[2].WhereBorn
XXXXXXXX|PPQ.PPQ270_F
XXXXXXXX|HEQ.HEQ400_F
XXXXXXXX|HEQ.HEQ210_F
XXXXXXXX|SSQ.SSQ010o_F
XXXXXXXX|CHQ.CHQ345n_F
XXXXXXXX|DWQ.DWQ060_F
XXXXXXXX|HEQ.HEQ393_F
XXXXXXXX|SSQ.SSQ010x_F
XXXXXXXX|Fsq.RelTable.Relations[3].RelationToChild
XXXXXXXX|Fsq.OriginsKeyParent[2].HowOld
XXXXXXXX|NRQ.Questions[2].NRQ122_F
XXXXXXXX|Fsq.OriginsKeyParent[2].WhereBorn
XXXXXXXX|SSQ.SSQ010x_F
XXXXXXXX|Fsq.EducationsKeyParent[2].FSQ221_F
XXXXXXXX|HEQ.HEQ370_F

```

This information is also being used to provide QC reports to field supervisors, so that they know how far field interviewers have progressed in the assigned interviews. Data from interviewed cases is captured whenever interviewers contact Home Office so that information is kept as current as possible about the last question completed during the interview. The program could be easily modified to capture other data about the interview as well.

Automatic conversion of Blaise data to a SAS dataset

Many studies have the need to convert a Blaise database with its datamodel into a SAS dataset with variable descriptions, formats, and a matching text input file. We have developed a WesBlaisetoSAS process consisting of two Maniplus programs: the first program asks for file names and folder locations, and the second (called by the first) uses a variable datamodel, and uses many temporary files during the creation of the SAS data definitions:

Users install WesBlaisetoSAS by running an executable file `InstallWesBlaiseSAS.exe`, accepting `c:\` as the default location to unzip files. The program creates a folder with a `README.TXT` file that lists the other files in the folder as follows:

- `Manipula.exe` (used by `msu` files to read and write Blaise information)
- `RunSAS32.msu` (used to collect details about files and folder locations)
- `SAS32BlaiseDriver.msu` (uses Blaise datamodel descriptions and Blaise data to produce a SAS program, which opens in SAS)
- `WesBlaiseSAS.lnk` (used to execute `RunSAS32.msu` to collect file and folder information and then produce the SAS program)
- `westat.bmp` (the Westat logo)

For users without a complete Blaise installation, the programs can be run by providing `Manipula.exe` and the prepared `Manipula` files. Such users start the program by use of the `WesBlaiseSAS.lnk` shortcut, or – if they have Blaise installed – start the prepared `Manipula` file `RunSAS32.msu`. The code for this file is as follows:

```
PROCESS CreateSASInstructionsFromBlaise
SETTINGS
  DESCRIPTION = 'WESTAT: Creating SAS dataset instructions for a Blaise database'

AUXFIELDS
  DatamodelLocation, DatabaseLocation, SASFolderFileLocation, SASFolderLocation : STRING
  SASFileName "What is the name of the SAS program?" / "Dataset name": STRING[50], EMPTY
  DescriptionOrQuestion "Do you want to use the Blaise description or the Blaise question text for
    labels?"
    / "Type of SAS labels" : (Description "Blaise description text", Question "Blaise question
    text")
  AttachFormats "Do you want to attach SAS formats to variables?"
    / "Attach SAS formats" : (Yes "Attach formats to variables", No "Do not attach formats to
    variables")
  DataOnly "Do you want create SAS descriptions?"
    / "Create SAS descriptions" : (Yes "Create SAS descriptions and output data", No "Output data
    only")

  AcceptName "Accept this file name" : (Yes, No)
  DoExit "Exit application" : (Yes, No)
  Rslt : INTEGER

DIALOGBOX TheFileName "Dataset name"
  FONTNAME = "CANDARA"
  FONTSIZE = 12
  SIZE = (500, 300)
  DEFAULTBUTTON = NO
  CONTROL SASFileName
    POSITION = (200,30)
    LABEL = Yes

  CONTROL DescriptionOrQuestion
    POSITION = (200,60)
    LABEL = Yes
```

```

CONTROL AttachFormats
  POSITION = (200,120)
  LABEL = Yes

CONTROL DataOnly
  POSITION = (200,180)
  LABEL = Yes

BUTTON AcceptName
  CAPTION 'Accept'
  STORE = YES

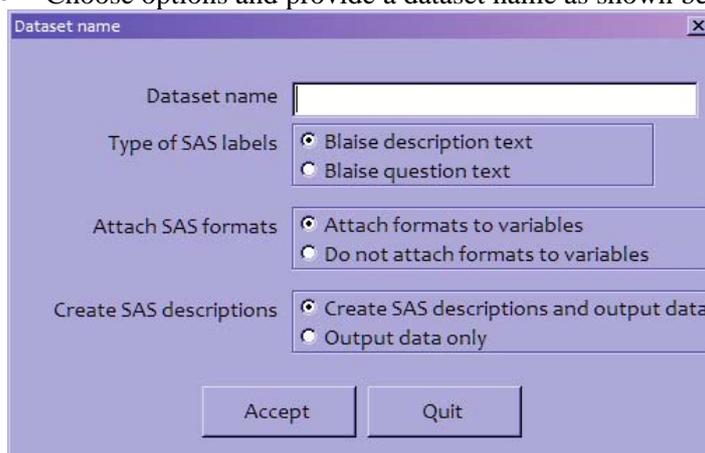
BUTTON DoExit
  CAPTION 'Quit'
  VALUE = Yes
  STORE = YES

MANIPULATE
  SETLOGO('westat.bmp')
  DatamodelLocation := '' + SELECTFILE ('Select Blaise datamodel', '*.bmi', 'Blaise datamodels') + ''
  DatabaseLocation := '' + SELECTFILE ('Select Blaise database', '*.bdb', 'Blaise databases') + ''
  SASFolderLocation := SELECTFOLDER('Select SAS data folder')
  IF (LEN(SASFolderLocation) > 0) THEN
    SASFolderLocation := '' + SASFolderLocation + '\'
  ELSE
    SASFolderLocation := ''
  ENDIF
  TheFileName
  IF (DoExit <> Yes) THEN
    IF (POSITION(SASFileName + '==', '.SAS==') < 1) AND (LEN(SASFileName) > 0) THEN
      SASFileName := SASFileName + '.SAS'
    ENDIF
    SASFileName := '' + SASFileName + ''
    Rslt := CALL('SAS32BlaiseDriver.msu /KInputMeta=' + DatamodelLocation + ' /I' + DatabaseLocation +
      ' /P' + SASFolderLocation + ';' + SASFileName + ';' + STR(DescriptionOrQuestion) + ';' +
      STR(AttachFormats))
  ENDIF

```

The sequence of processing and information that users provide when running the program is:

- Select the Blaise datamodel
- Select the Blaise database
- Select the folder for the SAS program
- Choose options and provide a dataset name as shown below.



The options are:

- SAS labels
 - Use the Blaise descriptions for labels
 - Use Blaise question text for labels
- SAS formats
 - Attach formats to variables

- Make a dataset where formats are not part of the variable description
- Creating SAS descriptions and outputting data
 - Both create SAS descriptions and create an output data text file
 - Create an output data text file (assumes the SAS descriptions are already available)

A second Manipula program runs using information collected by the first Manipula program. This second program uses the Blaise metadata to produce SAS descriptions describing the data, and uses Blaise data to produce a text file that matches the descriptions already produced. It starts:

```
PROCESS SASDataDescriptions
SETTINGS
  DESCRIPTION = 'WESTAT: Create SAS datamodel descriptions and extract data'
  ACCESS = SHARED
  CONNECT = YES

USES
  InputMeta (VAR)

  DATAMODEL UniqueMeta
    PRIMARY UniqueObject
    FIELDS
      UniqueObject : STRING[250]
      ObjectCode : STRING[8]
    ENDMODEL

  DATAMODEL OutputMeta
    FIELDS
      OutStr : STRING[1023]
    ENDMODEL

INPUTFILE BlaiseData : InputMeta('', BLAISE)
SETTINGS
  CHECKRULES = YES

TEMPORARYFILE UniqueFields : UniqueMeta
TEMPORARYFILE UniqueFormats : UniqueMeta
TEMPORARYFILE FormatDefsOutput : OutputMeta
TEMPORARYFILE VarPosnsOutput : OutputMeta
TEMPORARYFILE VarDescriptionsOutput : OutputMeta
TEMPORARYFILE VarFormatsOutput : OutputMeta

OUTPUTFILE SASOutput: OutputMeta ('', ASCII)
SETTINGS
  OPEN = NO
  TRAILINGSPACES = NO

OUTPUTFILE TextFile: InputMeta ('', ASCII)
SETTINGS
  OPEN = NO
```

Some of the intermediate code includes:

```
PROCEDURE ListUniqueObjects
PARAMETERS
  IMPORT BlockName : STRING
AUXFIELDS
  ObjectName, LocalName, UniqueObject, TypeName, BaseType, CategoryDescription, FieldDescription,
  StrSign : STRING
  UniqueID, Categories : INTEGER
INSTRUCTIONS
  ObjectName := BlaiseData.GETFIRSTFIELDNAME(BlockName)
  StartPosn := EndPosn + 1
  REPEAT
    LocalName := BlaiseData.GETFIELDINFO(ObjectName, 'LOCALNAME')
    UniqueID := 1
    UniqueObject := ConvertToASCII127(LocalName,31 ,2, 1)
    SETDESCRIPTION('WESTAT: Create SAS datamodel descriptions -- ' + ObjectName)
```

```

FieldDescription := REPLACE(FieldDescription, '''', '')
VarDescriptionsOutput.OutStr := ' ' + UniqueObject + ' = ' + FieldDescription + ''
VarDescriptionsOutput.WRITE
IF (UPPERCASE(BaseType) = 'ENUMERATION') THEN
  IF (UPPERCASE(TypeName) = '') THEN
    TypeName := REPLACE(ObjectName, '.', '_')
  ENDIF
  TypeName := ConvertToASCII127(TypeName + '_W', 31, 2, 2)
  UniqueFormats.GET(TypeName)
  IF NOT (UniqueFormats.RESULTOK) THEN
    UniqueFormats.UniqueObject := TypeName
    UniqueFormats.ObjectCode := 'W' + REPLACE(STR(FormatNum,6), ' ', '0') + 'W'
    UniqueFormats.WRITE
    FormatDefsOutput.OutStr := ' VALUE ' + UniqueFormats.ObjectCode
    FormatDefsOutput.WRITE
    Formatnum := FormatNum + 1
    Categories := VAL(BlaiseData.GETFIELDINFO(ObjectName, 'CATEGORIES.COUNT'))
    FOR I := 1 TO Categories DO
      IF (BlaiseData.GETFIELDINFO(ObjectName, 'CATEGORIES[' + STR(I) + '].DEFINEDTEXT') <> '')
      THEN
        CategoryDescription := BlaiseData.GETFIELDINFO(ObjectName, 'CATEGORIES[' + STR(I) +
          '].DEFINEDTEXT')
        ELSE
          CategoryDescription := BlaiseData.GETFIELDINFO(ObjectName, 'CATEGORIES[' + STR(I) +
            '].NAME')
        ENDIF
        CategoryDescription := ConvertToASCII127(CategoryDescription, 31, 2, 2)
        FormatDefsOutput.OutStr := ' ' + BlaiseData.GETFIELDINFO(ObjectName, 'CATEGORIES[' +
          STR(I) + '].CODE') + ' = ' + ''' + CategoryDescription + '''
        FormatDefsOutput.WRITE
      ENDDO
      FormatDefsOutput.OutStr := ' ;'
      FormatDefsOutput.WRITE
      FormatDefsOutput.OutStr := ''
      FormatDefsOutput.WRITE
    ENDIF
    VarFormatsOutput.OutStr := ' ' + UniqueObject + ' ' + UniqueFormats.ObjectCode + '.'
    VarFormatsOutput.WRITE
  ENDIF
ENDIF
ObjectName := BlaiseData.GETNEXTFIELDNAME(ObjectName)
UNTIL (ObjectName = '')
ENDPROCEDURE

```

After the completion of the program, the SAS screen is displayed with the appropriate output.

Example outputs are shown below.

```

DATAMODEL Stopping
TYPE
  tYesNo = (Yes, No)
  tReason = (Health (1) "@UHEALTH OR FUNCTIONING@U",
            Other (2) "@UOTHER REASON@U")
FIELDS
  One "Why do you want to stop?" / "Reason to stop" : tReason
  Two "Are you sure?" / "Confirm stopping": tYesNo
RULES
  One
  Two
ENDMODEL

```

SAS program for the 'Stopping' datamodel:

```
TITLE 'StopInterview_Default';
```

```

PROC FORMAT;
  VALUE W000001W
    1 = 'HEALTH OR FUNCTIONING'
    2 = 'OTHER REASON'
  ;

  VALUE W000002W
    1 = 'Yes'
    2 = 'No'

```

```

;
RUN;

LIBNAME SAVEFILE "H:\MY DOCUMENTS\Blaise\";
DATA SAVEFILE.StopInterview_Default;

INFILE 'H:\MY DOCUMENTS\Blaise\StopInterview_Default.ASC' LRECL = 2;

INPUT
  One      1-1
  Two      2-2
;

LABEL

  One = 'Reason to stop'
  Two = 'Confirm stopping'
;

FORMAT
  One  W000001W.
  Two  W000002W.
;

RUN;

```

SAS program for question text labels:

```

TITLE 'StopInterview_QuestionText';

PROC FORMAT;
  VALUE W000001W
    1 = 'HEALTH OR FUNCTIONING'
    2 = 'OTHER REASON'
  ;

  VALUE W000002W
    1 = 'Yes'
    2 = 'No'
  ;

RUN;

LIBNAME SAVEFILE "H:\MY DOCUMENTS\Blaise\";
DATA SAVEFILE.StopInterview_QuestionText;

INFILE 'H:\MY DOCUMENTS\Blaise\StopInterview_QuestionText.ASC' LRECL = 2;

INPUT
  One      1-1
  Two      2-2
;

LABEL

  One = 'Why do you want to stop?'
  Two = 'Are you sure?'
;

FORMAT
  One  W000001W.
  Two  W000002W.
;

RUN;

```

SAS program for formats not attached to variables:

```
TITLE 'StopInterview_NoAttachedFormats';

PROC FORMAT;
  VALUE W000001W
    1 = 'HEALTH OR FUNCTIONING'
    2 = 'OTHER REASON'
  ;

  VALUE W000002W
    1 = 'Yes'
    2 = 'No'
  ;

RUN;

LIBNAME SAVEFILE "H:\MY DOCUMENTS\Blaise\";
DATA SAVEFILE.StopInterview_NoAttachedFormats;

INFILE 'H:\MY DOCUMENTS\Blaise\StopInterview_NoAttachedFormats.ASC' LRECL = 2;

INPUT
  One      1-1
  Two      2-2
;

LABEL

  One = 'Reason to stop'
  Two = 'Confirm stopping'
;

/*

FORMAT
  One  W000001W.
  Two  W000002W.
;

*/

RUN;
```

Appendix: Using Cameleon to create a file with Manipula source code

The idea is to extract all remarks, for every case in a database, using Manipula. This Cameleon script creates Manipula source code to do that task:

```
[* Writes Manipula code to extract all remarks for each case in a database]
```

```
[VAR
  PRIMARYFIELD : STRING,
  LabelText    : STRING,
  CharToken    : STRING,
  Quote        : STRING,
  I            : REAL
]
[MAXNAMELENGTH := 127]
[OUTFILE := DATAMODELNAME+'_Remarks.MAN']
[[]USES
[[] InMeta '[DATAMODELNAME]'

[[] DATAMODEL PrintMeta
[[] FIELDS
[[] AuxOut : STRING[[1023]]
[[] ENDMODEL

[[]INPUTFILE InFile : InMeta ('[DATAMODELNAME]', BLAISE)
[[]OUTPUTFILE PrintFile : PrintMeta('[DATAMODELNAME]_Remarks.txt', PRINT)

[[]MANIPULATE

[PROCEDURE RemoveQuotes]
```

```

[Quote := '']
[LabelText := '']
[FOR I :=1 TO LENGTH(FieldLabel) DO]
  [IF (I < 64) THEN]
    [CharToken := COPY(FieldLabel,I,1)]
    [IF (CharToken <> Quote) THEN]
      [LabelText := LabelText + CharToken]
    [ENDIF]
  [ENDIF]
[ENDDO]
[ENDPROCEDURE]

[BLOCKPROC]
[FIELDSLOOP]
  [IF PRIMKEYFIELD THEN]
    [PRIMARYFIELD := FIELDNAME]
  [ENDIF]
[ENDFIELDSLOOP]
[FIELDSLOOP]
[ARRAYLOOP]
  [IF TYPE <> BLOCK THEN]
    [IF TYPE <> OPEN THEN]
      [SETLOOP]
        [IF ([FIELDPATH] = REMARKED) THEN]
          [RemoveQuotes]
          [AuxOut := [PRIMARYFIELD] + '|' + [FIELDPATH] | [LabelText] |' +
REPLACE(REPLACE([FIELDPATH].REMARK,CHAR(13),' '),CHAR(10),'')]
          [PrintFile.WRITE]
          [ENDIF]
        [ENDSETLOOP]
      [ENDIF]
    [ELSE]
      [BLOCKCALL]
    [ENDIF]
  [ENDARRAYLOOP]
[ENDFIELDSLOOP]
[ENDBLOCKPROC]

```

The following Manipula source is for datamodel HH01:

```

USES
  InMeta 'HH01'

DATAMODEL PrintMeta
  FIELDS
    AuxOut : STRING[1023]
  ENDMODEL

INPUTFILE InFile : InMeta ('HH01', BLAISE)
OUTPUTFILE PrintFile : PrintMeta('HH01_Remarks.txt', PRINT)

MANIPULATE

IF (Identifier = REMARKED) THEN
  AuxOut := Identifier + '|Identifier|What is the household identifier?|' +
    REPLACE(REPLACE(Identifier.REMARK,CHAR(13),' '),CHAR(10),'')
  PrintFile.WRITE
ENDIF
IF (ZipCode = REMARKED) THEN
  AuxOut := Identifier + '|ZipCode|What is the zip code for this household?|' +
    REPLACE(REPLACE(ZipCode.REMARK,CHAR(13),' '),CHAR(10),'')
  PrintFile.WRITE
ENDIF
IF (Household.Demographics[01].HHIndex = REMARKED) THEN
  AuxOut := Identifier + '|Household.Demographics[01].HHIndex|HH position|' +
    REPLACE(REPLACE(Household.Demographics[01].HHIndex.REMARK,CHAR(13),' '),CHAR(10),'')
  PrintFile.WRITE
ENDIF
IF (Household.Demographics[01].Name = REMARKED) THEN
  AuxOut := Identifier + '|Household.Demographics[01].Name|What is your name?|' +
    REPLACE(REPLACE(Household.Demographics[01].Name.REMARK,CHAR(13),' '),CHAR(10),'')
  PrintFile.WRITE
ENDIF

```

Optimal Ways of Developing Blaise IS Instruments

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

1 Blaise IS: Capable, Powerful, Open, but Under-Documented

The Blaise IS system for web surveys has been around for more than 10 years and has undergone several major changes in that time. In response to feedback from major users the Blaise Team has improved it into a powerful system. To update Blaise IS documentation, Mark M Pierzchala of MMP Survey Services, LLC wrote five documents. Two main goals of this project were to (1) explain in detail how the system works, and (2) to make Blaise IS development easier by providing a complete Blaise IS Mode Library, model tools such as the menu, suggested standards, and source code examples.

1.1 Five Documents and Supporting Examples

The five documents are summarized in the following table.

Table 1. Five Blaise IS Documents

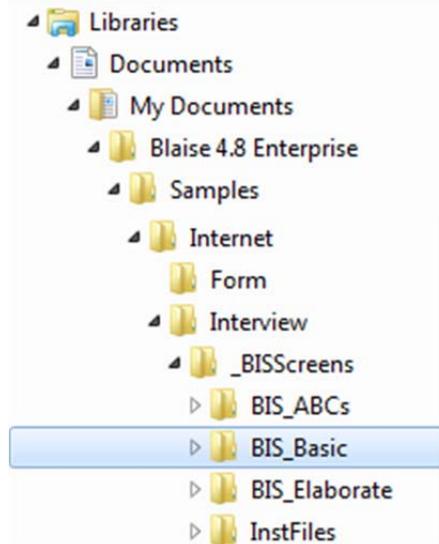
Name	Pages / Figures (Est.)	Notes	Supporting Examples
Blaise IS Samples	51 / 65	Summarizes the Blaise IS samples that come with the Blaise 4.8 distribution from <i>Ajax</i> through <i>Working Database</i> samples. It lists field pane definitions and .ASP pages that are used in each sample. It summarizes the characteristics of each sample in several different ways so that you can quickly find essential information and samples.	
Blaise IS ABCs	44 / 63	Covers the very basics of Blaise IS screens from the menu through the mode library and simple screens. It also discusses the Help language and alternate spoken languages. It covers major uses of the Blaise Internet Workshop.	BIS_ABCs, menu, mode library, datamodel properties
Blaise IS Basics	91 / 121	Demonstrates how to put together multi-question screens and edits. It explains four Blaise IS Groups including (1) Other-Specify, (2) Multi-Column; (3) Matrix, and (4) Group Table. It gives details on the use of the Group Layout Editor.	BIS_Basic, menu, mode library, datamodel properties
Blaise IS Elaborate	49 / 75	Illustrates the use of the Custom Group, and shows how to implement some miscellaneous screens. Shows an advanced use of a menu panel to implement parallel blocks.	BIS_Elaborate, menu, mode library, datamodel properties
Blaise IS Journal	40 / 51	Covers the implementation of the Blaise IS Journal including basic and advanced capabilities. Covers the new (with Blaise 4.8.3) client-side paradata capability. Illustrates how to interpret and handle the paradata that are output during data collection.	TimingTest, Travel, 3 versions of Journal, Paradata, and 2 Manipula scripts
Totals	275 / 375		

Figures are primarily screen images but there are also many diagrams. Further, there are tables and source code snippets inserted throughout the documents. In order to cover this visual subject matter, text is kept to a minimum and the material is explained as visually as possible.

1.2 Three New Blaise IS Datamodels and Supporting Configuration Files

Three datamodels *BIS_ABCs*, *BIS_Basic*, and *BIS_Elaborate* were developed in order to give a thorough coverage of Blaise IS screen development. Figure 1 below shows the location of these examples.

Figure 1. Three Additional Blaise IS Samples



These examples are placed near the traditional Blaise IS samples. There is a main folder called *_BIScreens*. Subfolders *BIS_ABCs*, *BIS_Basic*, and *BIS_Elaborate* hold these datamodels while the subfolder *InstFiles* holds a common Mode Library, Menu File, and other supporting files. The *InstFile* folder emulates an institute’s setup which is to share configuration files across projects.

1.3 Analysis of Web-Survey Browser Screens and Blaise IS Samples

In order to write Blaise IS documentation, it was necessary to fully understand its capabilities vis-à-vis industry practices and expectations. The author had already reviewed web-survey screen examples as a basic research task for Blaise 5 documentation. Additionally he led a team at Mathematica Policy Research, Inc. (MPR) that produced an internal guide of web-survey standards (Pierzchala et. al., 2009). Finally, he conducted an extensive review of the traditional Blaise IS samples that are part of the system distribution. All of this was in addition to the author’s experiences in adapting Blaise IS to diverse client needs while at MPR. Many of the examples used in this document are based on those experiences and the author thanks MPR for allowing background reference to their standards document.

The three new Blaise IS datamodels were designed to illustrate various screen styles and how to achieve them with Blaise IS capabilities. This approach helps to guarantee the relevance of this documentation. It also validates Blaise ID evolution over the years as many of its modern features are the result of user experiences and feedback.

1.3.1 Web-Screen Standards

All three datamodels were designed according to a strict set of web-screen standards. These standards include font size and type, header, the use of logos and survey title, the placement of buttons, and the use of a bottom bar that holds links to FAQ and Help. They are documented in the *BIS_ABCs* document.

1.3.2 Model Mode Library

Previous experiences with Blaise IS indicated that screens with vertical arrangements of fields were easy to produce while those with horizontal layouts could be unexpectedly difficult. A particular problem with adapting Blaise IS had been in understanding how Mode Library fieldpane definitions influenced rendering of screens especially with Blaise IS groups (horizontal displays).

A major goal of this documentation was to produce an industrial-strength Mode Library file that met display standards and objectives. This would give an institute a ready-made set of fieldpane and grid definitions that would enable it to accomplish the vast majority of its work.

The resulting Mode Library *BIS_Screens.bml* contains 26 fieldpane and 9 grid definitions. These are used throughout the three sample datamodels. A feature of this Mode Library is that it has group-specific fieldpane definitions. There are 4 fieldpane definitions for the *Custom* group, 2 intended for the *Group Table* group, 6 to be used with the *Multi-Column* group, 2 that work with the *Other Specify* group, and 1 used with the *Matrix* group.

Eight of the fieldpane definitions are variations on the *Default Field Pane* definition. These allow for a range of adaptations to the vertical display of fields. An example is the display of a large number of radio buttons in 2 columns. Finally, there are a few other miscellaneous fieldpane definitions such as *Scaling*.

Fieldpanes in this model Mode Library are documented in *Appendix Delta* of the *BIS_Basic* document.

1.3.3 Model Menu File

A model Menu File called *BIS_Screens.bmf* is also used by the three new sample datamodels. It has illustrative panes and panels that show how to implement a variety of Blaise IS menu-related features. These include the header, the footer, the buttons, and elements of these items as well as references to expressions and field names in the datamodel. The menu forms the skeleton of the Blaise IS web screen.

2 Blaise IS Samples

The first document is *Blaise IS Samples*. While on one hand the traditional Blaise IS examples are diverse, on the other hand, it can take a very long time to find the information you want. The document solves this problem. It provides an extensive summary of these 18 samples. This summary takes the form of cross reference tables and images. Through either, you can find the information you need.

2.1 Cross-Reference Tables of Characteristics of the Traditional Blaise IS Samples

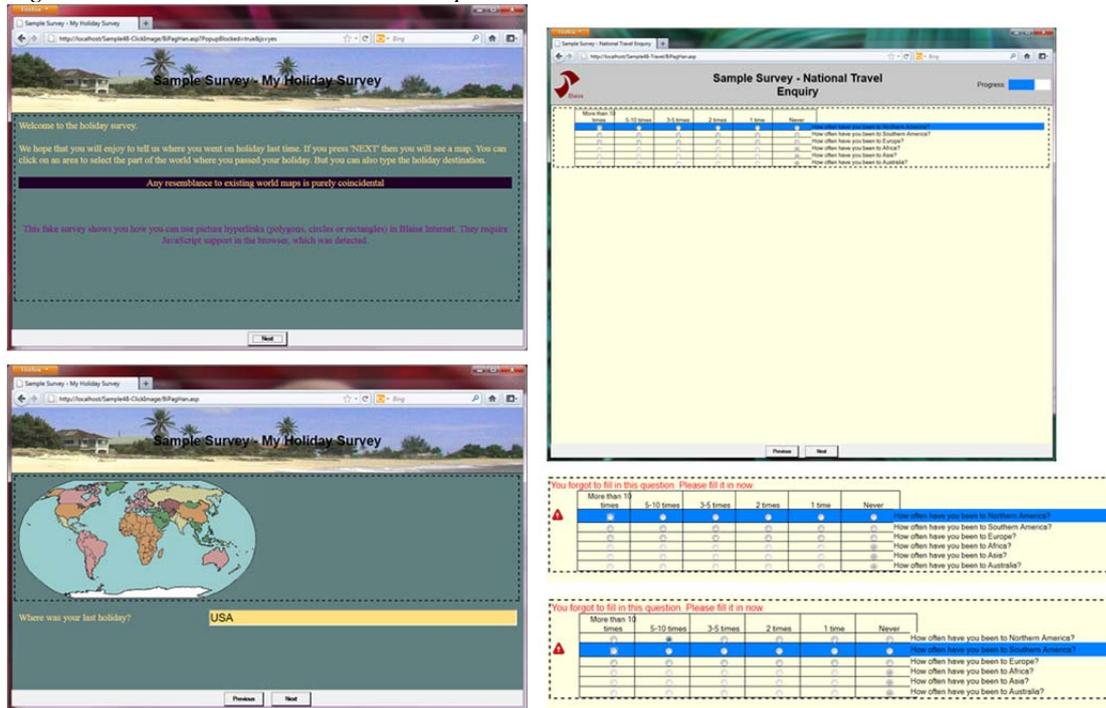
There are 9 tables that characterize the features of these samples in various ways.

- Blaise IS Samples and their Purposes
- Use of .ASP Files
- Summary of .XML Files
- Blaise IS Samples and the Fieldpanes Available to Each
- Blaise IS Samples and the Fieldpanes Used in Each
- Blaise IS Samples and the Groupings Used in Each
- Blaise IS Samples vs. Type of Grouping vs. Fieldpanes
- Fieldpanes vs. Groupings
- Groupings vs. Fieldpanes

2.2 Screen Images of the Traditional Blaise IS Samples

There are 41 pages of many screen images from these datamodels. These are provided so that you can quickly determine which datamodel example is suited for your needs. Following is a sample of some of these screen images.

Figure 2. Some Traditional Blaise IS Sample Screens



3 Blaise IS ABCs

Blaise IS ABCs is the first of a trilogy of documents that explain how to generate Blaise IS browser screens. It explains the fundamentals of screen design. Blaise IS does not have a drag-and-drop interface design. It does have a variety of features that speed instrument development.

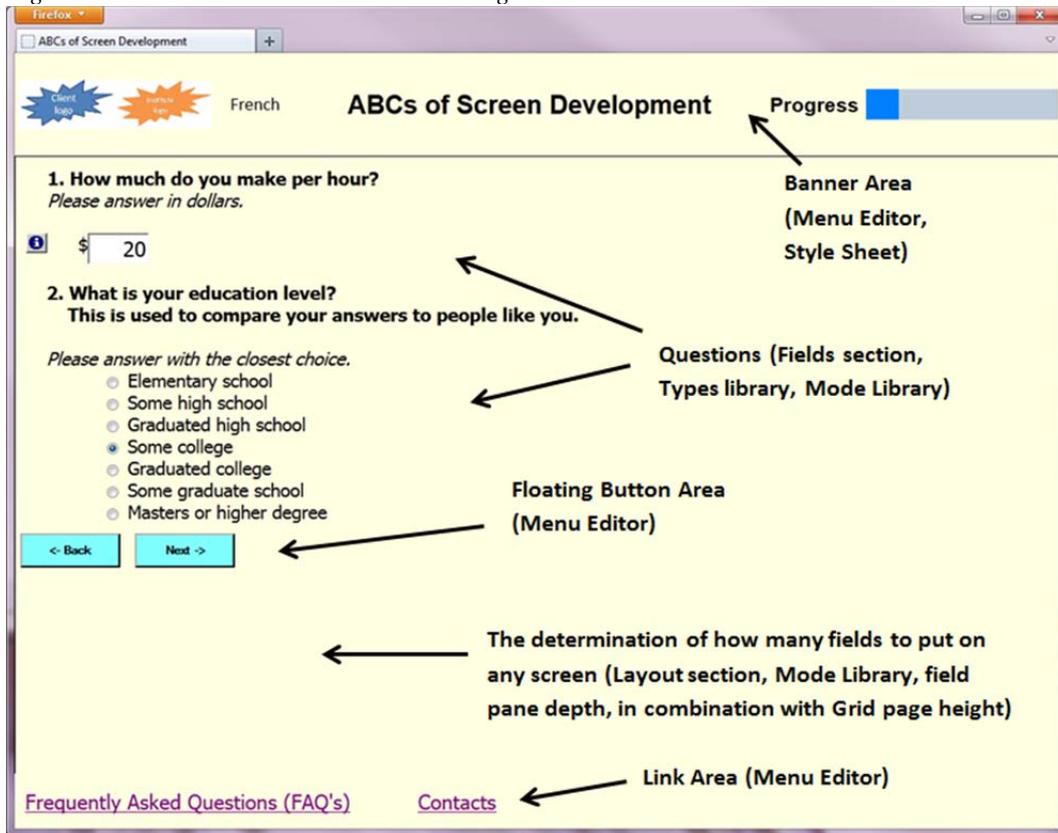
3.1 Blaise Modules and their Impact on Screen Design

One of the challenges of Blaise IS Screen design is determining which parts of the Blaise system influence the interface. This document, and the others, identifies these modules which include:

- Mode Library Font Settings, Grid and Fieldpane definitions, languages, and more
- Menu
- Datamodel Properties
- Style sheets
- Blaise Internet Workshop, especially the Group Layout Designer
- Blaise source code syntax and datamodel settings
- Expressions

How these items come together to create the overall screen is not always obvious. Figure 3 below is an example of the kind of diagram in the document that helps to explain the underpinnings of the system.

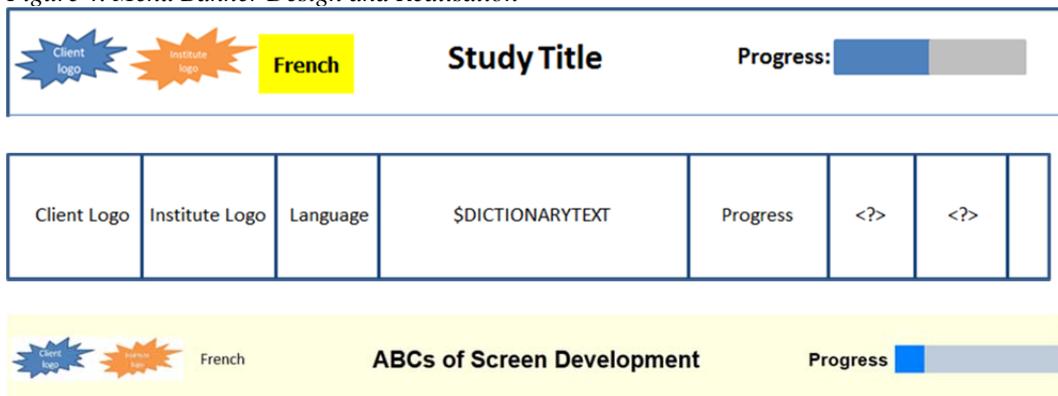
Figure 3. Browser Screen Parts and their Origins



3.2 The Menu as the Browser Screen Skeleton

A great deal of the *Blaise IS ABCs* document is taken with the explanation of the menu and how it comprises the header, buttons, and footer of a browser screen. It covers some advance topics such as setting up language switching in the menu, and it explains how to put in a simple progress bar. Figure 4 shows some images used to explain the design of the banner.

Figure 4. Menu Banner Design and Realisation



3.3 Design Fundamentals

Through the conceptual use of diagrams, and screen shots of parts of the Mode Library, there is thorough coverage of grids and other underlying concepts that impact screen layout.

3.4 Miscellaneous Topics

A few miscellaneous topics are covered in this first document. These include handling languages (English and French), use of Help, links to HTML documents, image files, and the basics of writing effective edits.

3.5 Optimal Use of Blaise IS Tools

This document starts to explain how to optimally use the Blaise IS Internet Workshop and the best way to integrate it with the Blaise Control Centre. It covers the relative uses of the Preview versus using a browser screen to evaluate layout.

4 Blaise IS Basic

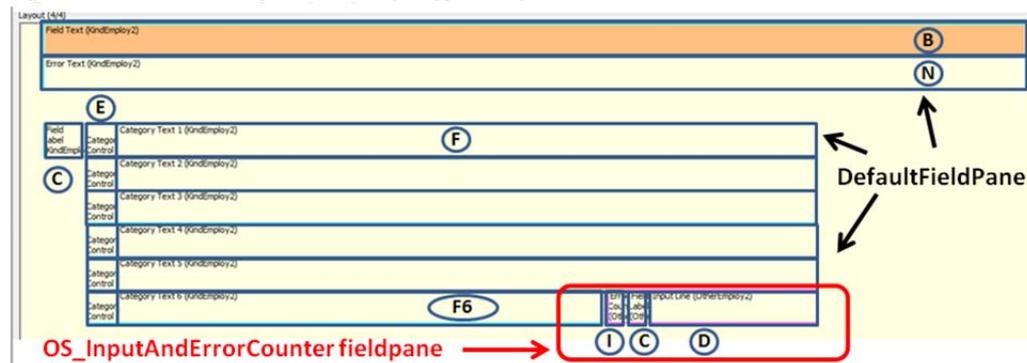
Blaise IS Basic is the second of the trilogy of documents that explain how to generate Blaise IS browser displays. It explains the design goals of bringing multiple fields onto the same page. There can be multiple fields vertically arranged, or horizontally arranged, or both. With multi-field displays, there are specification issues, such as how you handle skip patterns and edits.

The horizontal display of fields can be achieved through a group. Blaise IS supplies four oft-used groups: (1) Other Specify, (2) Multi Column, (3) Group Table, and (4) Matrix. A fifth group, the catch-all Custom Group is saved for the third document.

4.1 Groups

Diagrams and screen shots are used to explain how the groups work. For example, the Other Specify Group makes use of two fields using two different field panes as shown in Figure 5.

Figure 5. A Schematic of an Other Specify Group



The letter labels in Figure 5 refer to parts of the fieldpane definitions in the Mode Library.

4.2 Group Layout Editor

The Group Layout Editor is a tool you can use to make the final adjustments to a group's screen layout. There is an appendix that explains how to use this tool easily ('with alacrity'). Another set of labeled images help to explain the concepts as shown in Figure 6.

Figure 6. Labelled Image of the Group Layout Editor



4.3 Appendices

The last third of the *Blaise IS Basic* document is given to 5 appendices. They are:

- Alpha: Parts of the Screen
- Beta: Using the Group Layout Editor with Alacrity
- Gamma: Taking Care of Special Circumstances
- Delta: Table of Fieldpane Properties
- Epsilon: Page-by-Page Summary of the *BIS_Basic* Datamodel

5 Blaise IS Elaborate

Blaise IS Elaborate is the third member of the trilogy. This document starts with the rendering of more miscellaneous examples. For example, it covers alternate methods of implementing hierarchical coding. It then explains some of the uses of the catch-all Custom Group.

5.1 Custom Group

Significant space is given to the philosophy and use of Custom Group and the use of the Group Layout Editor to achieve desired displays. The Custom Group is available in case the four standard groups are not sufficient. It has considerable flexibility and this flexibility is usually achieved in the Group Layout Editor. Figures 7 give starting-and-after views of an address display.

Figures 7. Starting and Final Layouts of an Address Collection Screen

Starting

The starting layout shows a vertical stack of input fields and error messages. At the top is a large orange box for 'StreetLabel' with an associated error text. Below it are several smaller input fields for 'Street', 'Apartment', 'City', 'AState', 'Zip5', and 'Zip4', each with its own error text. The fields are arranged in a column, with error messages appearing to the left of the input lines.

Final

The final layout shows a more compact and organized arrangement of input fields and error messages. The fields are arranged in a grid-like structure, with error messages appearing to the left of the input lines. The overall layout is more efficient and easier to read.

5.2 Applying a Group to an Arrayed Block

While the optimal use of the Group Layout Editor is an acquired skill, it has a number of features that speed the design of a Blaise IS instrument. One of the very nice surprises that came in writing this document is the way a group can be applied to an array of a block. Figures 8 and 9 give an idea of this capability. Figure 8 shows part of the Grouping Dialog while Figure 9 shows screen captures of the 5 resulting grouped-pages.

Figure 8. Grouping Dialog Showing 5 Generated Group Pages

Role	Index	Begin Question	End Question
✓ Custom (Principal Em...	1	PrincipalEmployer_	Header4
✓ Custom (Name)	2	NameLabel	Suffix
✓ Custom (Address)	3	StreetLabel	Zip4
✓ Custom	4	Names[1].NameLabel	Names[1].Suffix
✓ Custom	5	Names[2].NameLabel	Names[2].Suffix
✓ Custom	6	Names[3].NameLabel	Names[3].Suffix
✓ Custom	7	Names[4].NameLabel	Names[4].Suffix
✓ Custom	8	Names[5].NameLabel	Names[5].Suffix

Figure 9. Five Identical Generated Group Pages

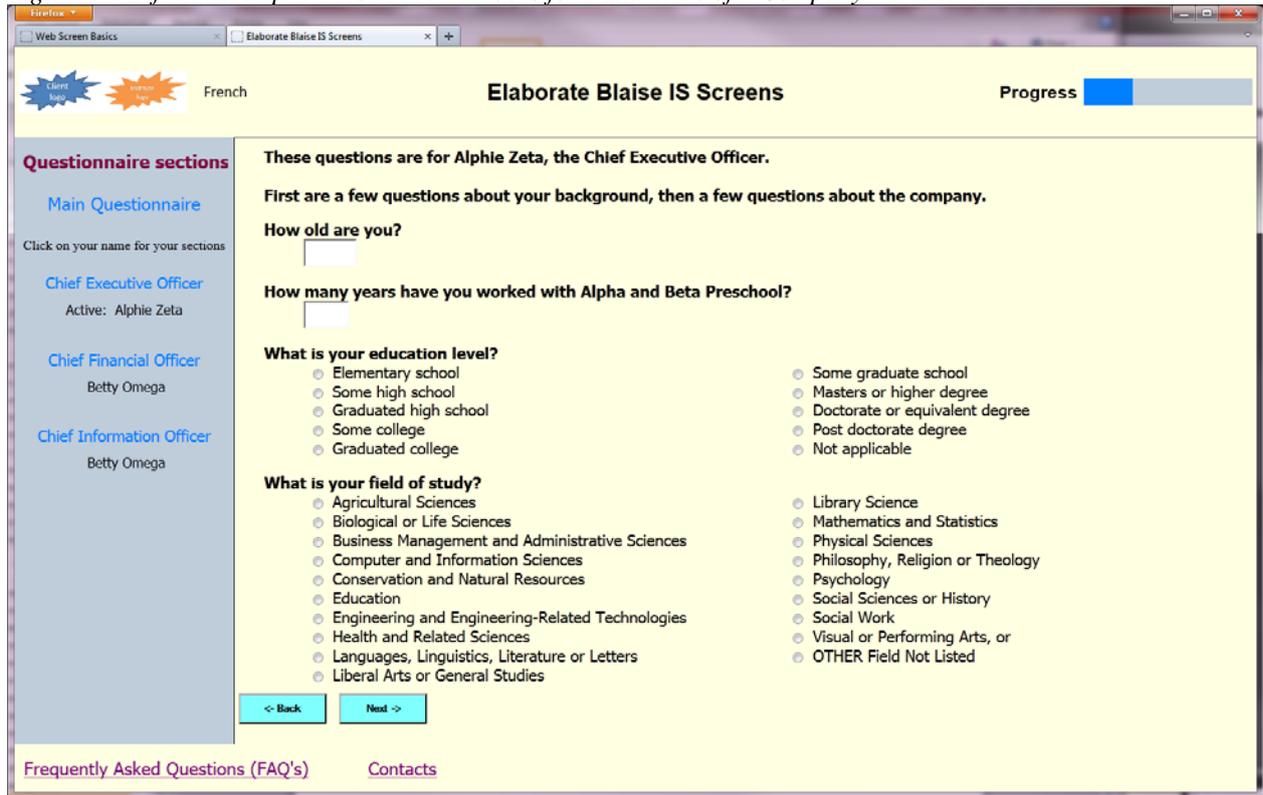
The figure shows five identical generated group pages. Each page contains a form with the following fields: 'Title', 'First name', 'Middle name', 'Last name', and 'Suffix'. The form is designed to be user-friendly and consistent across all five pages.

The generated group pages were incredibly easy to generate. In the instrument, they apply to up to 5 corporate officers in succession.

5.3 Handling Parallel Blocks

A powerful way to handle parallel blocks is explained in this document. In Blaise IS you have to do a bit of work to handle parallel blocks in a way that a methodologist is apt to agree with. This topic represents a return to the menu file. The handling of parallel blocks is done through a panel in the menu. In Figure 10, the light blue panel on the left suddenly appears part-way through the instrument.

Figure 10. Left Panel Implements Parallel Blocks for Individuals of a Company



The panel is dynamic and keeps track of the completion status of the requisite individuals. It is achieved in part through the clever use of expressions in the menu file. Expressions are covered to a strategic degree in this document.

5.4 Choices for DK and RF and Other Configuration Settings

Blaise IS is highly configurable. In other words, you have many choices. As an example, a few pages are spent on display options for DK and RF. Additionally, Blaise IS has settings for style sheets, file handling, layout, languages, and many other aspects. The document displays several dialogs in order to give you an idea of these options. You have to experiment to see what these settings really do for you.

6 Blaise IS Journal

The document *Blaise IS Journal* takes on an entirely different aspect of Blaise IS. This concerns the implementation, use, interpretation, and summary of paradata¹ that are generated through *Journal*. Paradata give information about the survey taking process.

6.1 Implementation of Journal and Paradata Capabilities

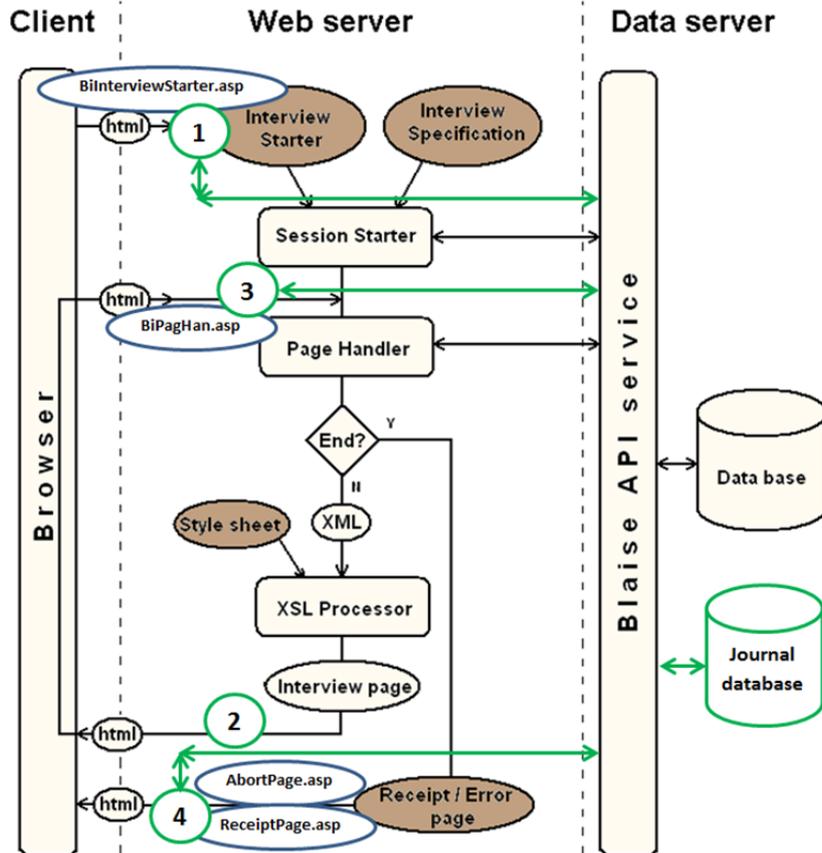
The document starts by covering the traditional *Journal* that accesses web server information. It continues with a new capability (Blaise 4.8.3, January 2012) that can access client-side paradata. This latter manifestation of *Journal* is implemented with a datamodel called *Paradata*. This capability was developed and motivated by work done by the University of Michigan Survey Research Center (Ostergren and Liu, 2010). The implementation of Blaise IS Journal ranges from fairly easy to very easy.

¹ Paradata is information about the survey process itself. According to Dirk Heerwegh, this term was coined by Mick Couper in 2000 (see references).

6.2 Open Implementation of Blaise IS Journal

You can access several different .ASP pages in the implementation of *Journal*. This is easy to do, yet it gives you great flexibility. Figure 11 shows places where you can gather web-survey paradata. The diagram is adapted from one found in the on-line Blaise IS documentation.

Figure 11. Blaise IS Paradata Diagram of ASP Pages



Schematic overview of how Interview Mode web surveys work

6.3 Interpretation and Summary of Journal Paradata

The interpretation and summary of paradata is given considerable space. Connections are drawn between what the respondent sees and does on a browser screen and what the paradata show.

Figure 12a. A Succession of Blaise IS Fields on the Same Browser Page

<p>How much did you spend on your last trip?</p> <input type="text" value="2000"/> <input type="text" value="2000"/>	<p>How much did you spend on your last trip?</p> <input type="text" value="2000"/> <p><input type="radio"/> Don't know</p>
<p>Currency</p> <p>Select <input type="button" value="v"/></p> <ul style="list-style-type: none"> Select US Dollar Canadian Dollar Euro Yuan Yen <li style="background-color: #0070C0; color: white;">Other 	<p>Other currency</p> <input type="text"/>

Figure 12b. Paradata Display Showing Entries from Multi-Field Screens

SessionID	PrimaryK	Action	SubmitStat	PrevPag	CurrentPag	PageStartTimeSta	PrevPageTimeSta	PrevPageLength
8164747		Start			1	20120131105820		0
8164747	1001000	Other		1	1	20120131105829	20120131105820	9
8164747	1001000	Other		1	1	20120131105855	20120131105829	26
8164747	1001000	Next		1	2	20120131105904	20120131105855	9
8164747	1001000	Next		2	3	20120131105944	20120131105904	40
8164747	1001000	Next		3	4	20120131110012	20120131105944	28
8164747	1001000	Next		4	5	20120131110022	20120131110012	10
8164747	1001000	Other		5	5	20120131110035	20120131110022	13
8164747	1001000	Other		5	5	20120131110036	20120131110035	1
8164747	1001000	Other		5	5	20120131110049	20120131110036	13
8164747	1001000	Next		5	6	20120131110100	20120131110049	11
8164747	1001000	Submit	Completed	6		20120131110103	20120131110100	3

6.4 Sorting, Parsing, Extracting and Summarizing Web-Survey Paradata

Many thousands of lines of paradata can be generated for a web survey. In fact, the quantity of Blaise IS paradata can overwhelm the amount of survey data. All paradata are stored in a Blaise database, and from there relevant summaries or uses can be devised. Two Manipula programs in particular give you a start in devising your own summary software. They are:

- AnswersVsTime.man: This program generates a spreadsheet (.csv file) of the answers of a question and the time it took to answer each question. This is a highly specific program, but you can adapt it to your own needs for other surveys.
- AuditFileOut.man: This program generates audit-like files of the kind you might see for CATI or CAPI. This is a more general program, but you would still probably have to adapt it.

7 References

Couper, M. "Usability Evaluation of Computer-Assisted Survey Instruments" Social Science Computer Review (vol. 18, No. 4, 2000, pp. 384-396)

Heerwegh, D. "The CSP Project Webpage"
[\[https://perswww.kuleuven.be/~u0034437/public/csp.htm\]](https://perswww.kuleuven.be/~u0034437/public/csp.htm)

Ostergren, J. and Liu, Y. "Blaise IS Paradata", 13th International Blaise Users Conference (Westat), Baltimore, MD, October, 2010. [\[http://www.blaiseusers.org/2010/papers/5a.pdf\]](http://www.blaiseusers.org/2010/papers/5a.pdf)

Pierzchala, Mark M. Blaise IS Samples, 2012, Blaise Documentation, Statistics Netherlands

Pierzchala, Mark M. Blaise IS ABCs, 2012, Blaise Documentation, Statistics Netherlands

Pierzchala, Mark M. Blaise IS Basic, 2012, Blaise Documentation, Statistics Netherlands

Pierzchala, Mark M. Blaise IS Elaborate, 2012, Blaise Documentation, Statistics Netherlands

Pierzchala, Mark M. Blaise IS Journal, 2012, Blaise Documentation, Statistics Netherlands

Pierzchala, M., Sonnenfeld, K., Brinkley, M., and Wright, D., MPR Web Survey Guidelines (March 2009). Internal MPR Document, used as a background reference by permission.

Performance and security enhancements on the Blaise IS standard stylesheet

Arnaud Wijnant, Edwin de Vet – CentERdata, Tilburg University, The Netherlands

1 Abstract

CentERdata has several years of experience in administering Blaise surveys over the Internet. In 2010 we migrated to Blaise IS, the standard CAWI server included in the Blaise installation.

The Blaise IS package consists of 3 parts that cooperate to fulfill the task of hosting a web questionnaire. These parts are the web server which creates the pages for the respondents, the data server that manages the answers given by the respondents and the rules server that applies the rules defined in the Blaise questionnaire. They can be replicated on several servers to increase the capacity of a questionnaire.

For some surveys that contain large tables and groupings of questions we found that the web server component required a high amount of system resources per respondent. This was mainly caused by the transformation of the XML output of the questionnaire engine to the HTML output of the web server. This transformation is done via a XSLT style sheet that is shipped in the Blaise installation. We customized this style sheet to obtain the look and feel we wanted, but the style sheet contained a lot of functionality and overhead that we do not need. To resolve this issue, we developed a smaller and simpler style sheet from scratch. This style sheet contains only the needed functionality, is easier to maintain and produces simpler HTML output. Load tests show that we can serve more respondents per server with our self created style sheet.

We also implemented extra security features in Blaise IS. In our setup, respondents provide the primary key as a KeyValue parameter (either GET or POST) at the start of the interview. We developed a simple mechanism to verify the identity of the respondent using a hash value based on this KeyValue.

2 Introduction

At our institute we run web questionnaires using Blaise IS for internet panels consisting of several thousands of people. At peak times, approximately 10 respondents per minute start a questionnaire and answer 5 questions per second on average. During several of these peak times, we noticed a strong drop in performance: respondents sometimes had to wait for over 10 seconds to get a new page or could not access the server at all. This drop in performance was especially problematic with a particular questionnaire that contained lot of tables.

3 Current situation

To be able to solve this problem, we first had to identify the cause of the problem. For this we first looked into our server park. Our Blaise IS server park consists of 2 combined web and rule servers and one central data server. During peak hours we could identify that the problem was not on the data server, since the CPU load on this server was negligible.

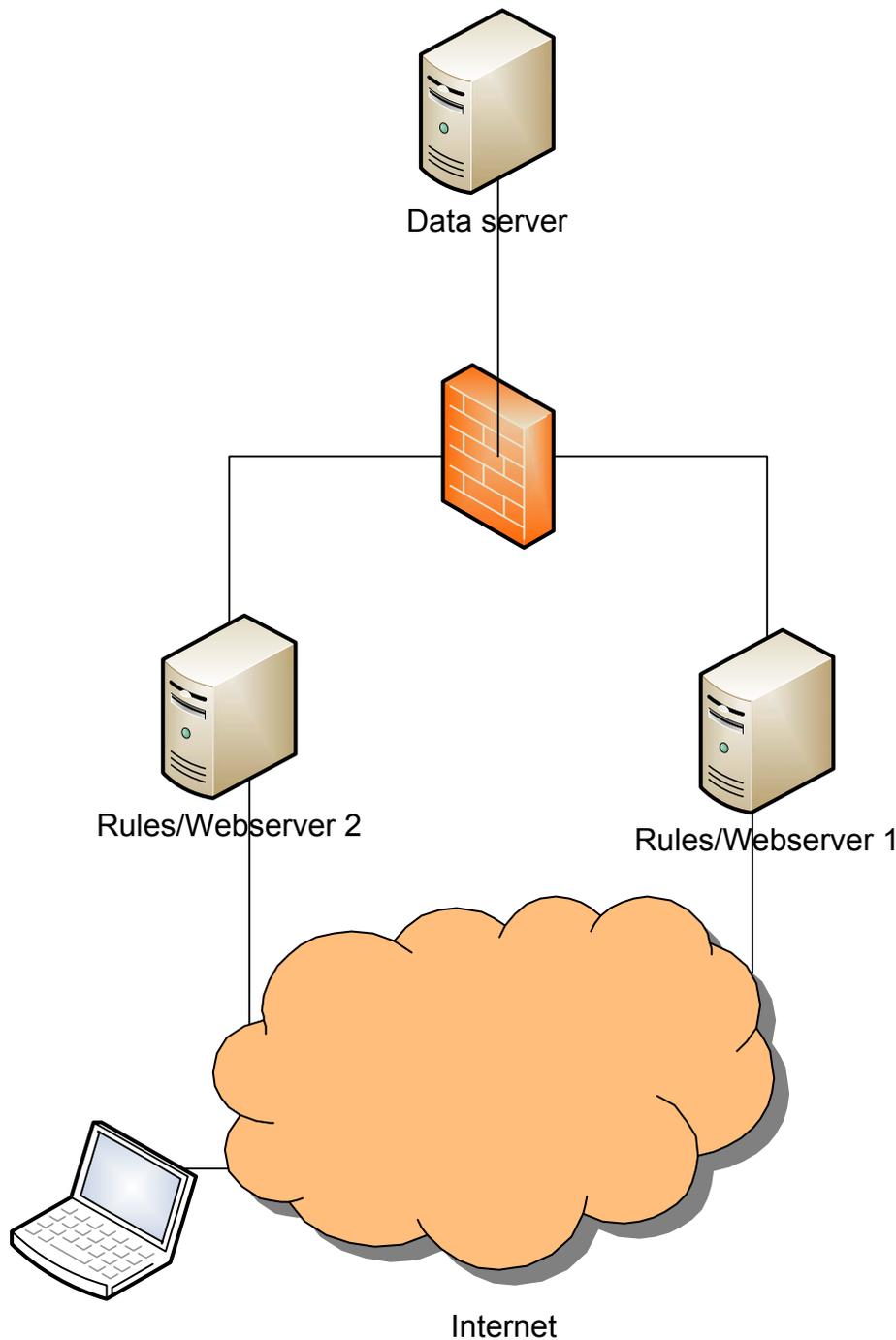


Figure 1 The Blaise IS server park

The problem had to be on the combined data and web-services. In our setup we balance the load between the webservers by assigning questionnaires to a specific web server. This means that every questionnaire is only visited through one of our 2 web servers. Since the performance problems were only on one of our servers, it was likely that the problems were questionnaire specific. It was also clear that the problems were related to the web server and not the rules server, because there was only one process that had a heavy load on the system and this was the process of the web server. The next step in the process was to find out which of the questionnaires was causing the problem. We observed this by isolating the questionnaires on a separate server. The high CPU load was only observed on a questionnaire with pages that contain large tables of radio buttons. We therefore suspected that the XSL transformation was the bottleneck in the generation of the output. The Blaise IS XSL style sheet included in Blaise is rather big (around 6000 lines) and the output is in our opinion too large and complex. We often observe highly nested tables and div elements.

4 Solution

In order to investigate whether we could improve the performance and simplify our HTML output, we wrote a simplified style sheet from scratch with only the required functionality. This simple style sheet is roughly 6 times smaller than the original style sheet. For this reason the style sheet is easier to read and modify and the HTML output is much simpler (hardly any use of nested html tables and div elements). This simplicity is accomplished by removing unneeded functionality. However there are also new features present, for instance, we use the label tag for text belonging to checkboxes and radio buttons, which makes the text clickable in most modern browsers. Also, the generated question pages do not rely on JavaScript. The settings in the .bml and .bmf files are often ignored. Changes to the Look and Feel should be done directly in the style sheet.

5 Analysis of solution

To test our style sheet in a systematic way, we used a tool developed by MicroSoft and called Web Capacity Analysis Tool (WCAT). "Web Capacity Analysis Tool is a lightweight HTTP load generation tool primarily designed to measure the performance of a web server within a controlled environment. WCAT can simulate thousands of concurrent users making requests to a single web site or multiple web sites. The WCAT engine uses a simple script to define the set of HTTP requests to be played back to the web server. Extensibility is provided through plug-in DLLs and a standard, simple API." [1]

For our setup we wrote a script which perfectly mimics a respondent that fills in the questionnaire with which we had the performance problems. The simulated respondent answers a page with questions every 5 seconds. The script is loaded into the WCAT system and simulates the visit of several respondents. For our test we did test with 150 and 200 simultaneous respondents.

The server that was used to host the questionnaire for this load test is the following:

Processor: Quad-Core AMD Opteron(tm) processor 2369 HE 2.40 GHz

Installed memory: 4.00 GB

Operating system: Windows Server 2008 R2

No other questionnaires were in use at this server during the load test.

5.1 Results

The performance can be measured in several performance parameters. For our load test we present two of them: the number of requests/second being handled by the server and response times.

The number of requests per second is the number of pages that were generated for respondents during the load test. For a test with 150 simultaneous respondents that ask for a page every 5 seconds the ideal number for this parameter will be $150/5 = 30$ requests/second. This number is only achieved if the generation of the page is done in 0 seconds. For the test with 200 simultaneous respondents the ideal number is $200/5 = 40$ requests/second.

The table below shows the outcomes for the load tests that we did with both the original and simple style sheet.

	Standard style sheet	Simple style sheet
150 sim. respondents	26.12 req./sec.	28.81 req./sec.
200 sim. respondents	5.51 req./sec.	33.88 req./sec.

For 150 simultaneous respondents we see that the simple style sheet handles a bit more requests per second. Both numbers are close to the optimal number of 30 requests per second, so no major problems occur. However for the 200 simultaneous respondents we see a huge difference in the number of handled requests per second. Because the server with the standard style sheet is overloaded with requests, the time it takes to process one requests increased significantly.

Another parameter to look at is the response time of the server. The following table shows the maximum response time and the average response time of every request.

Style sheet	Sim. resp.	Max RT (s)	Average RT (s)
Standard	150	20.654	0.740
Standard	200	300.427	28.503
Simple	150	4.024	0.183
Simple	200	4.555	0.888

Here again we see significant better results for the Simple Style Sheet, especially when the number of respondents is 200. We can also have a look at the distribution of how fast requests are processed. The table below shows the maximum waiting times in ms for a percentage of the requests when the times are in an ascending order.

Stylesheet	Sim. Resp.	1%	5%	25%	50%	75%	95%	99%	100%
Standard	150	15	78	125	216	512	3584	8064	20654
Standard	200	1008	2288	7488	17792	35968	96896	159360	300427
Simple	150	15	62	78	110	172	608	1456	4024
Simple	200	31	63	156	624	1440	2528	3328	4555

Acceptable waiting times are around 2 seconds. For the simple style sheet these are accomplished for almost 95% of the requests when the server is tested with 200 respondents. For the standard style sheet this is only achieved for 5% of the requests.

6 Security enhancement

In our setup, respondents log in into a web portal with their own username and password. Once logged in, respondents are directed to the Blaise IS questionnaires with an integer primary key as Key/Value parameter (either GET or POST). This setup is unsafe, since the respondent can alter their own primary key value in the portal and would thus be able to fill in the questionnaire under a different identity and/or potentially see the answers of other respondents.

We came up with a simple and robust solution. In the portal, a hash value (we use the sha1 algorithm) is calculated from the concatenation of a salt string (normally bigger than 20 random characters) and the Key/Value. This hash value is submitted together with the Key/Value to a modified start page of BlaiseIS (biInterviewStarter.asp). In this start page, we added code to recalculate the hash value from the Key/Value using the same salt string and compare this value with the submitted hash. Respondents only get an active interview session if the recalculated hash matches the submitted hash.

7 Conclusion

We discovered that the XSL transformation can be a bottle neck in the performance of the system when the server load is high. In summary we can say that the simple style sheet can improve the server capacity. More respondents can be served by the same server and these respondents have shorter waiting times for a page to be generated.

The modifications to the Blaise IS system described in this paper make the system better suited for web questionnaires for large groups of respondents using limited hardware resources. Especially for large projects with relative high sample sizes, it pays off to invest in optimization of the style sheet.

8 References

[1] WCAT 6.3 – MicroSoft Corporation
<http://www.iis.net/community/default.aspx?tabid=34&g=6&i=1466>

Blaise On-the-Go: Using Blaise IS With Mobile Devices

Alerk Amin (CentERdata, Tilburg, The Netherlands)

Arnaud Wijnant (CentERdata, Tilburg, The Netherlands)

1 Introduction

Blaise IS provides a mechanism for respondents to complete their questionnaires over the web. The interface is designed for respondents with a mouse, using a web browser running on full-sized screen. However, people are increasingly accessing the internet with touchscreen mobile devices, such as smartphones and tablets. For mobile respondents, answering Blaise questionnaires should be as natural and comfortable as any other app on their device. The unique characteristics of mobile devices require a different type of user interface, which the standard Blaise IS style sheets do not offer.

Figure 1 shows a questionnaire on a mobile screen, using the default Blaise style sheets. The text and buttons are sized correctly for a desktop screen, but they clearly are too small for the mobile screen. Respondents must zoom in to read the question text, and then either zoom out or scroll the page down to reach the next button. This makes the questionnaire difficult to use with a mobile device, and greatly increases the respondent burden.

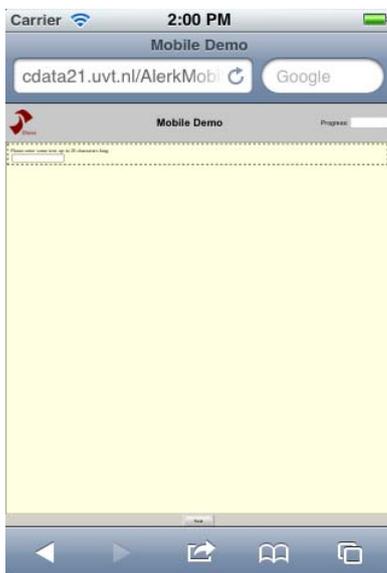


Figure 1

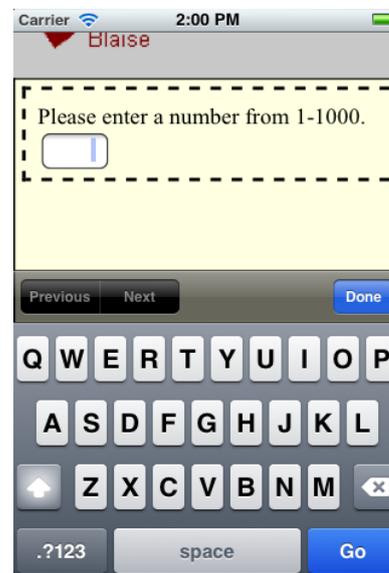


Figure 2

Another issue concerns on-screen keyboards. Most touchscreen mobile devices utilize on-screen keyboards. Because the size of the screen is limited, the on-screen keyboards do not display all of the keys at one time. Figure 2 shows a problematic case. The question asks the respondent to enter a number, but the on-screen keyboard shows letters. The respondent must take the additional step to click the number button in the bottom left corner, to switch the keyboard to the numeric keyboard.

To address these issues and others, a new style sheet was created. This mobile style sheet generates questionnaire pages that are adapted for mobile devices. Addressing issues such as screen size, on-screen keyboards, touch gestures, network bandwidth, and more, it allows respondents to more easily complete a Blaise IS questionnaire on a mobile device.

2 C-Moto Mobile Style Sheet

The new style sheet and associated files constitute the CentERdata Mobile Touchscreen Style Sheet, or C-Moto. The following sections describe the design principles behind the development of C-Moto, and how it fits into the Blaise IS architecture.

2.1 Separation of Presentation and Content

A principle of good web design is the separation of presentation and content. For questionnaires, Blaise accomplishes this through the separation between the BLA file (content) and the BML ModeLib file (presentation). While the BLA file contains the actual questions, the BML defines the various styles and the presentation of the questions. When the system is run with Blaise IS, web pages are generated that contain the combine the question and style, so they are displayed properly in a web browser.

For many organizations, this separation works well. But for organizations with experience in building websites, this mechanism creates problems.

The separation of presentation and content can be extended to web pages. Good web design involves using HTML to describe the content of a page, and CSS (Cascading Style Sheet) to describe the presentation of the content. The web pages generated by Blaise IS have presentation information mixed into the HTML, making it difficult to adjust the layout via CSS.

The C-Moto solution consists of a new style sheet, which replaces the biHTMLWebPage.xsl and biSimpleHTMLWebPage.xsl that are included with Blaise IS. The new style sheet creates “clean” HTML, without any presentation information. This HTML is combined with JavaScript and CSS files to make the web browser render the page properly.

2.2 Mobile Framework

With C-Moto, the HTML generated by Blaise IS can be completely controlled. This opens up the possibility to leverage a mobile framework to generate pages that work well on mobile devices.

There are many mobile frameworks available to aid the development of mobile apps and websites. Based on the requirements of mobile questionnaires, and to create an environment that works well with Blaise IS, the C-Moto was developed with jQuery Mobile.

jQuery Mobile is very easy to use and implement. Most functionality is accomplished via HTML markup, simply by adding extra attributes to the HTML elements. As the new style sheet gives total control over the generated HTML, it is straightforward to integrate the appropriate jQuery Mobile markup. When the page is loaded in the browser, the jQuery Mobile JavaScript transforms the HTML elements into appropriately styled elements for mobile devices. jQuery Mobile also has a very broad platform support, including all major desktop and mobile browsers. It also degrades gracefully, in that some features may not work in all browsers, but the pages in those browsers will still look nice and be functional for respondents.

2.3 Mobile Style Sheet Architecture

The architecture of C-Moto works seamlessly with the normal Blaise IS architecture. The key to the solution is the new mobile style sheet, which replaces the style sheets provided with Blaise IS. The new mobile style sheet can easily be added in the .BIS file, along with the supporting JavaScript, CSS and image files.

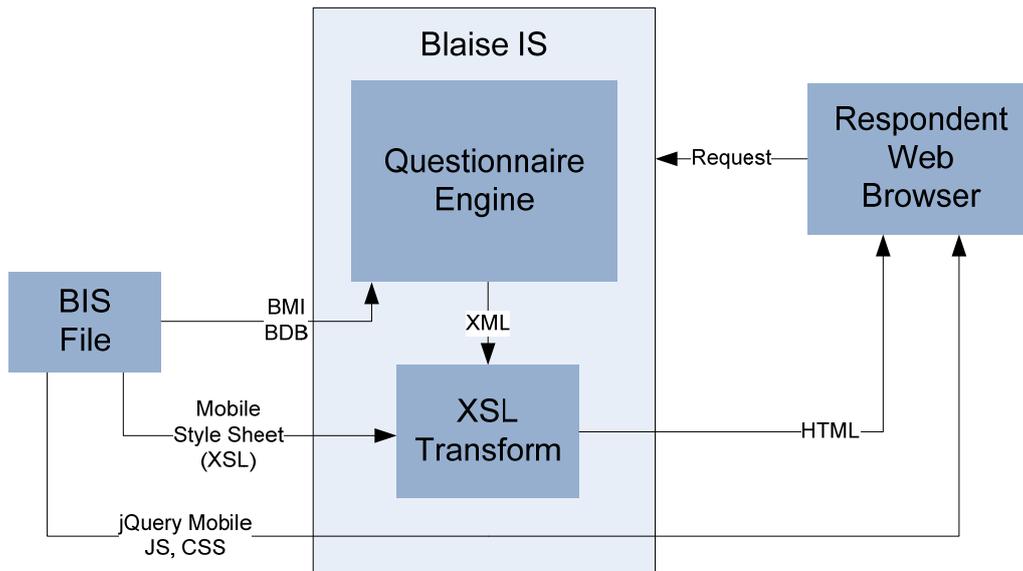


Figure 3

Figure 3 shows the process flow of a web questionnaire that uses C-Moto. When the respondent requests a page, the normal Blaise questionnaire engine (a combination of the web server, rules server and data server) creates an XML page. The mobile style sheet is used to convert this XML into an HTML page. The HTML page uses the JavaScript and CSS files required for jQuery Mobile, instead of those provided with Blaise IS.

The HTML page created with the Blaise style sheets includes a form. Blaise expects certain parameters to be submitted with this form, so the questionnaire engine can process the user input correctly. The form generated by C-Moto is designed to submit the exact same parameters as the normal Blaise form, so the Blaise questionnaire engine can process it without any other modifications.

3 Functionality

C-Moto contains a great deal of extra functionality for mobile and touch screen devices, compared to the standard Blaise style sheets. This section describes the various issues of questionnaires on mobile devices, and how C-Moto addresses them.

3.1 Screen Size

Compared to traditional web interviewing, mobile web interviewing is most often done on smaller screens. For this reason, C-Moto ensures that the text is large enough to be easily read on the screen, and buttons are large enough to be clicked with a finger.

To accomplish this, the design of the questionnaires is very compact. The aim is to minimize the unused screen area as much as possible. The screen is split into 3 areas. The header and footer areas have a fixed height and contain the basic functionality of a questionnaire (like next, back and help buttons) and one area is flexible in size and if necessary also scrollable.

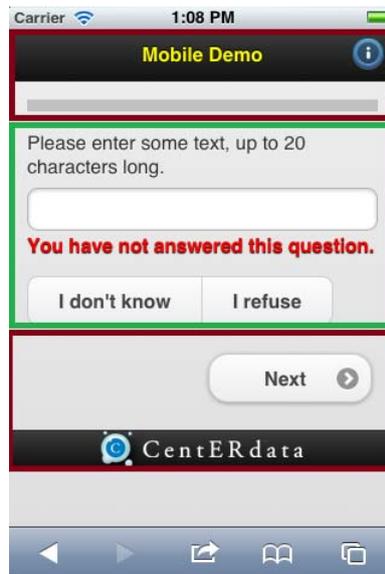


Figure 4

Figure 4 shows the 3 areas. The two areas with a red line are the fixed areas. These areas contain more static functionality that is more questionnaire-oriented than question-oriented. The green area in the middle represents the question itself and is more flexible.

The text, input areas, and buttons on the page are all sized appropriately for a mobile device. The text is large enough to be easily read, while the input areas and buttons are large enough to be comfortably clicked with a finger.

A benefit of C-Moto is that it also works well on desktop devices. Figure 5 shows the same question on a desktop or tablet screen. The page expands to fill the available width, but otherwise most aspects are the same. This allows for both consistency and usability across a wide range of screen sizes.

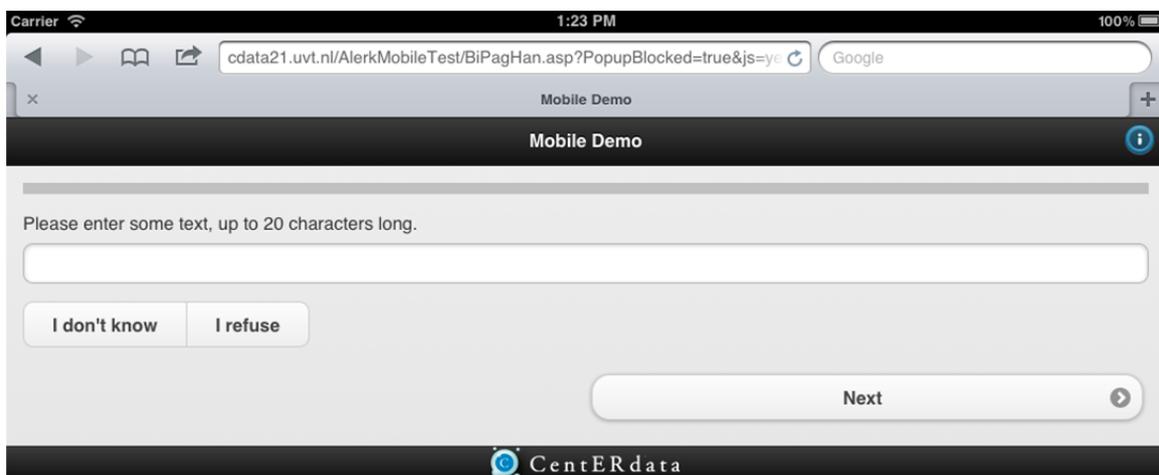


Figure 5

This adjustment of size is especially important for questions that have an answer type of a list or set. Figure 6 shows a question with a list using the standard Blaise IS style sheets. The list is rendered with radio buttons that are too small for respondents to click. If the question was a set instead of a list, the radio buttons would be replaced with checkboxes, but the size would be the same. Figure 7 shows the same question with C-Moto. The list is still rendered as radio buttons, but the style is much more usable for mobile respondents. The radio buttons are grouped together, in a “container box” with

rounded corners, to show that these options all go together. The box shading is slightly different than the page background, to emphasize this point. Each option is rendered with a box that contains both the radio button, as well as the label. This makes it clear to respondents that they can click anywhere on the row (in the radio button or on the label) to select an option. Increasing the size of the click target area makes it easier for respondents to click an option on a small screen.



Figure 6

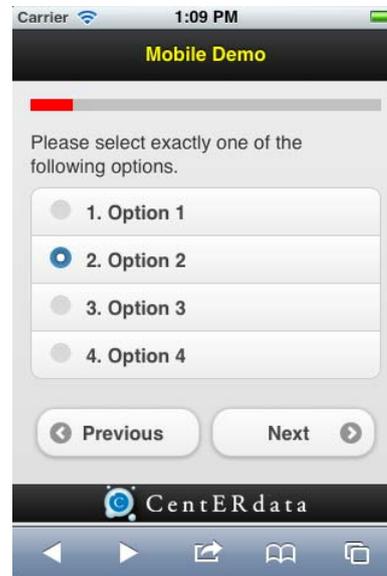


Figure 7

3.2 Tables

Blaise supports the ability to display multiple questions on a screen, either via tables or blocks. In either case, the result is a list of questions that can be converted into rows through configuration in the ModeLib and/or the Internet Workshop.

In most cases, tables are displayed with a question on each row. Each row contains the question text in the leftmost column, and then the answer box (for string or number fields) in the next column, or a set of options (for lists) in subsequent columns. There can also be columns added for error texts and other items.

On a mobile screen, this type of layout usually does not work, as the screen is just too small. In this case, it may be better to display the various questions in the table as individual questions. Each question looks just like it would without a table, but multiple questions are displayed one-on-top-of-the-other on the same page. Figure 8 shows an example of this solution.

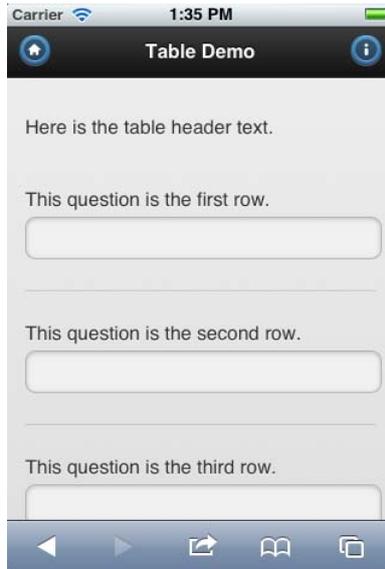


Figure 8

This type of layout works well for small screen, such as smartphones, but is not always the best solution for tablets. Tablets have screens that are much larger, and can support a more traditional table layout. A possible solution is to have the question text above the answers for small smartphone screens, but to the left of the answers on larger screens.

Fortunately, this can be accomplished through the use of CSS media queries. The mobile style sheets enforce the separation of content and presentation, with the presentation being controlled through CSS. CSS media queries allow for different layouts, based on characteristics of the device. The mobile style sheet adjusts tables based on a screen width of 450px, the default value in jQuery Mobile. For screens with a width smaller than 450px, tables are displayed with the question text above the answers. For screens with a width of 450px or greater, tables are displayed with the question text to the left of the answers. Figure 9 shows the same table as above, but on a larger screen.



Figure 9

CSS Media Queries are extremely powerful and flexible way of adjusting the same content to have a different presentation on different devices. In addition to screen sizes, presentation can be adjusted based on the type of device, or even the orientation of the device. For instance, if a respondent rotates their device from portrait to landscape mode, CSS Media Queries can adjust the style for the new layout.

3.3 Help Text

Assistance is sometimes important for respondents while filling in a questionnaire. Respondents should be able to get help during the interview. In the mobile style sheet, support for explanatory help texts is provided. If more information is available for a specific question, an “i” button pops-up in the upper right corner of the screen. If the respondent clicks this button, a status box appears that contains the help-item of this specific question. No additional requests to the server are required for this help, which makes the action as fast as a local application. Figure 10 shows how this help would pop up if the “i” button in the right upper corner was clicked.

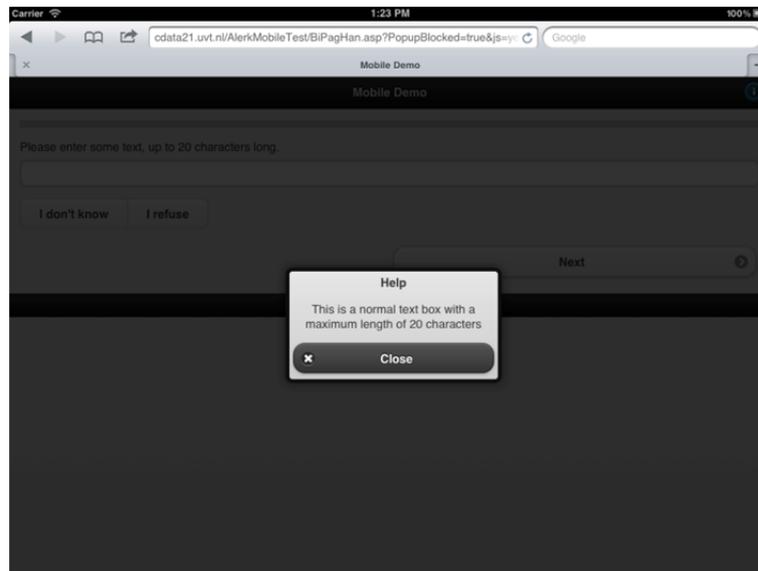


Figure 10

On the Blaise side, the help texts are programmed as an additional language and defined in the ModeLib, as normal. C-Moto automatically detects whether there is help text for a question. If there is help text, it shows the help icon and manages the dialog box with the help text. No additional programming or configuration is required in the ModeLib for this functionality.

3.4 Touch Screen Keyboards

The most common type of keyboard on a smartphone is a touch screen keyboard. Many smartphones lack a physical keyboard, and instead have a virtual keyboard that pops up on the screen when it is necessary. Since this keyboard is not fixed, it creates the possibility to adapt the keyboard layout to the situation.

In questionnaires, this is really useful as keyboard layout can match the type of question. For normal text inputs, the default (QWERTY) keyboard can be displayed. But when a question requires a number to be filled in, the keyboard can only contain the number keys, which makes it more comfortable and easy to type in the correct number. The same can be done for special data types like telephone numbers, web addresses or email addresses.



Figure 11



Figure 12

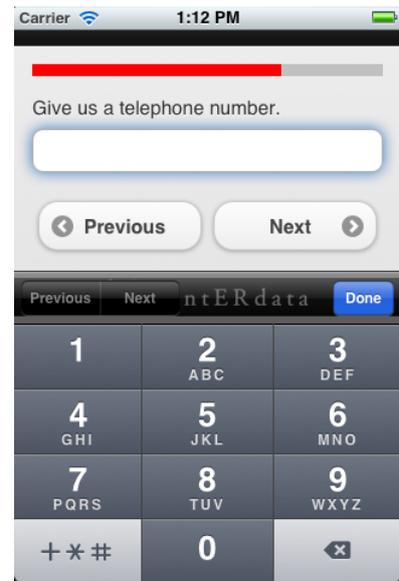


Figure 13

Figure 11, figure 12 and figure 13 show several examples of questions with different keyboard layouts. C-Moto uses HTML5 input types to control the keyboard. Support for HTML5 input types varies across devices and browsers. A table that shows support is available at <http://wufoo.com/html5/>. But even on browsers that do not support all of the input types, the functionality degrades gracefully. In these cases, the default QWERTY keyboard is displayed, giving the respondent the ability to still answer the questions.

3.5 Date and Time Pickers

Date and time pickers are a special case of on-screen keyboards. Many devices have native controls for date and time pickers. Figure 14 and figure 15 show two examples. However, support for these is not as extensive as for keyboards, and the look and feel of the controls vary greatly between devices. While using native controls is supported by C-Moto, it also supports an alternative method.



Figure 14

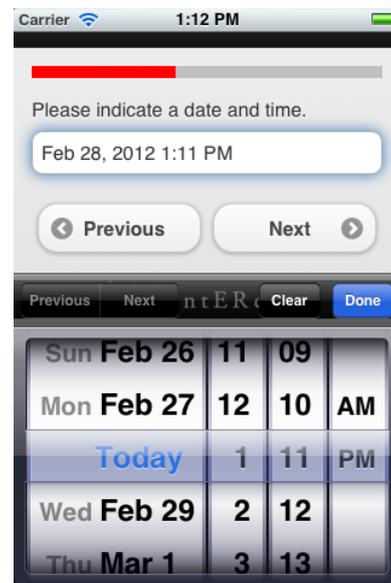


Figure 15

Instead of using native controls for date and time pickers, C-Moto supports the jQuery Mobile Datebox library. This library is an extension to the standard jQuery Mobile framework, and adds support for many different date and time pickers. In contrast to native controls, jQuery Mobile Datebox implements the controls using JavaScript and CSS, to create a consistent look and feel across different mobile devices.

Figure 16 shows a date picker using spinner controls, similar to those found on many Android phones, while figure 17 uses slider controls similar to those found on iPhones. By using these controls, researchers can choose what type of control respondents should use.



Figure 16



Figure 17



Figure 18

Another option would be to use a classical calendar-style date selection box, as used in many other websites. Figure 18 shows an example of a calendar control.

3.6 Navigation with Buttons and Gestures

The forms in the Blaise style sheet include a Next and Previous button to navigate through the questionnaire. In the HTML page, these buttons are input elements with `type=submit`, to submit the form. They each have a different name, so the Blaise Questionnaire engine can tell which button was pressed, to navigate in the appropriate direction.

The mobile style sheet also includes Next and Previous buttons. However, the placement of these buttons may vary. For example, on some devices, it is customary to put a Back or Previous button in the upper left corner of the screen. This puts the button in the header area of the page, outside of the form element. Additionally, many mobile users are accustomed to using gestures to navigation. Simply swiping a figure from left to right across the screen should move to the previous page, and swiping from right to left should move to the next page.

To make the layout more flexible, the mobile style sheet uses JavaScript to manage the form. The Next and Previous buttons can be placed anywhere on the screen, as input element with `type=button`. When they are clicked, JavaScript code manipulates the form to insert the appropriate values, so that the Blaise questionnaire engine knows whether the Next or Previous button was clicked. This JavaScript code is also used to process swipe events, to allow gesture-based navigation in the Blaise questionnaire.

3.7 Animation

When navigating from page to page in a mobile app, it is customary to use animation. For example, when going to the next page, the old page slides to the left, while the new page slides in from the right. When moving to the previous page, the animation is displayed in the opposite direction. Animation is an extremely powerful tool to provide visual cues to the respondent.

On a normal Blaise page, submitting a form causes a request to be sent to the server, which delivers the next page. The browser simply replaces the previous page with the new page, with no animation. The mobile style sheets make use of AJAX, a widely used technology that allows for asynchronous requests. When the user submits a form (via a button click or a gesture), the JavaScript code on the web page submits the form data, but the current page stays on the screen. When the server returns the HTML for the next page, the browser does not replace the page, as with the normal form. Instead, because the request was an AJAX request, the contents of the new HTML page are sent to the JavaScript code in the old page. This code then prepares the new page, and runs an animation. This gives the effect of the old page sliding out and the new page sliding in.

Because all of the form submission and animation are controlled by JavaScript code, more advanced options are also possible. The direction of the animation should be right-to-left when moving forward, and left-to-right when moving backward. Because the JavaScript code differentiates between a click of the Next or Previous button (or a gesture), it can display the animation in the appropriate direction. A future possibility is to improve the handling of errors. If Blaise detects an error on the page, it will not move to the next question, but rather display the same page, with error texts on this. Ideally, this type of transition would not use a right-to-left animation, to make it clear to the respondent that they have not moved to the next question, because of errors on the page.

3.8 Rating Scales

Rating scales, such as a Likert scale, are a common method of measuring how respondents rate something. For example, a 5-point scale that goes from “Completely Disagree” to “Completely Agree” is a common answer type to measure how a respondent feels about a statement. In Blaise, rating scales are usually implemented as a field of type List.

In Blaise IS, lists are converted to a set of options, which are usually displayed in the browser as a set of radio buttons. Clicking one button selects it, and deselects any other button that was previously

selected. On a desktop browser, the buttons are small, but a mouse can control the cursor, so the desired button is easy to click.

However, a mobile device has a much smaller screen, and respondents usually select items by touching them with their finger. This is much less accurate than a mouse/cursor, so the radio button areas need to be much bigger. While a mobile device may have room for 5 radio buttons arranged horizontally, a larger scale (such as a 11-point scale) may be too difficult to use.

A better option is to use a slider, as shown in Figure 19. The slider is displayed as a horizontal bar, and there is a small “handle”, that shows the selected value. The key difference between a slider and a set of radio buttons is that sliders also include “drag” functionality. A respondent can touch the handle and drag it left or right, to the appropriate value. In practice, this is much more accurate than small radio buttons, and allows for larger scales on smaller screens.



Figure 19

The mobile style sheet contains support for sliders. This is accomplished in the Blaise source via the tag. A questionnaire can be programmed with fields of type list, as normal. However, if the field has a tag “slider”, the style sheet will display a slider instead of a set of radio buttons.

3.9 Images and Video

Images and videos need special attention in the mobile style sheet. One aspect is that the size of the images should be optimized to the device it is being sent to. There is no need to send pictures that are larger than the actual screen size to the mobile device, as this causes unnecessary data traffic. To be able to know what image needs to be included, CSS Media Queries can be used to detect the screen size and include a picture that is suitable for the size of the screen.

For videos, most mobile devices support different video formats. Managing multiple formats to target different devices requires a great deal of effort. A possible solution is to use YouTube, as most mobile devices have support for displaying YouTube videos. The video can be “embedded” into a question, by inserting the embed code (from YouTube) into the question text. On desktop browsers, the video will play in this embedded box on the screen. But on most mobile devices, the video will play on the full screen, making better use of the screen space. Additionally, YouTube will manage the video formats, converting as necessary to the appropriate size and format for the device. This functionality works extremely well, and requires nothing from C-Moto.

3.10 Performance and Bandwidth

Besides the user friendliness of the C-Moto style sheet, performance is also an important issue for mobile device web pages. Pages should load fast enough over a mobile connection. For this reason, the output HTML of the C-Moto style sheet is much more compact than the traditional Blaise IS style sheets.

One part of this optimization is achieved by moving the presentation/ mark-up data from the HTML files to separate CSS files. These CSS files need to be loaded one time per questionnaire or even one time per questionnaire server (if all questionnaires use the same style). Additional optimization occurred by creating simpler, smaller HTML, which gets customized by the JavaScript and CSS files.

A comparison of the C-Moto and standard Blaise style sheet shows us that the data per page in C-Moto is half the size of the same page in the traditional Blaise web pages. The initial files that need to be loaded in both systems are comparable. In the traditional Blaise style sheet, these are mostly JavaScript files, whereas in the mobile style sheet most of the initial files are CSS files. In the table below some page sizes are mentioned.

	Blaise IS style sheet	C-Moto
Initial files (CSS, JS)	172 kB	188 kB
Question page (HTML)	25-35 kB	12-18 kB

4 Conclusion

Running questionnaires on mobile devices presents challenges, but also a unique opportunity to customize the questionnaire for mobile respondents. C-Moto helps bridge the gap between desktop and mobile devices, by ensuring that the same BLA questionnaire displays in a usable form on different devices. By separating the content and presentation, the display becomes easy to customize and conforms to modern web standards. Better usage of mobile input options, including virtual keyboards and gestures, allows for easier and more accurate data entry. Taken together, these features implement a better mobile interface, creating a better respondent experience.

Blaise Translation Challenges: Versioning, Multimode and Exporting

M.G.J. Martens MSc, CentERdata

Abstract

CentERdata has developed a system for managing translations for CAPI Blaise questionnaires for several international panel studies. This was done with an online tool called the Language Management Utility (LMU). It is highly configurable and support includes various element types, assignments, workflows, states, versions, users, modules and languages.

Recently the LMU has been extended. The backend was redesigned to provide new functionality. Rather than fixed versions, every translation is saved, which makes it possible to revert and fully analyse the translation process. Several methods of CAWI integration were tested and implemented. Blaise questionnaires can now be uploaded and changes since the last upload are detected and automatically updated and flagged in LMU. Questionnaire routing is stored in the LMU.

Several new exports were added: each type of questionnaire can be reviewed online in any language with texts imported from the LMU database. Multilingual Blaise source code can be generated from the LMU directly. It is possible to browse through the questionnaire in all stages of development in track changes mode. Several new exports in Excel are available. Questionnaires can be exported in DDI3-compatible format.

The new LMU features provide the means to support a larger part of the questionnaire design process. They can be regarded as the first steps towards an online Questionnaire Management Environment for international studies.

1 Introduction

Since 2002 CentERdata has been developing a system to guide the translation process for multilingual Blaise questionnaires. Initially developed for the SHARE project. **The Survey of Health, Ageing and Retirement in Europe (SHARE) is a multidisciplinary and cross-national panel database of micro data on health, socio-economic status and social and family networks of more than 55,000 individuals from 20 European countries aged 50 or over.** After a few waves of SHARE it was fine-tuned and also opened up to other parties

This paper discusses some projects we started but not yet finished, some inspiration we picked up along the way and how this all cumulated in a complete redesign of the Language Management Utility (LMU).

Originally this system consisted of a (web-) frontend and a backend. The frontend allowed translators to add their translation through a website, LMU. Texts from a structured Blaise questionnaire were manually copied over into the LMU. After an iterative process of translating, redesigning, testing this delivered us a final version that could be used in the field. On the backend two tools were implemented to manage these translated texts;

- A 'Blaise Generator' developed in Visual Basic, that pasted translated texts into a generic version,
- A 'Paperversion Generator' that exported a textual overview of the questionnaire and routing.

This process and the tools involved are described more in depth in the 2009 IBUC paper “Managing Translations for Blaise Questionnaires” (Martens, et al, 2009).

In following waves of SHARE the LMU expanded in many dimensions. The functionality of the web-frontend grew; it was also made possible to translate texts for the Share Sample Management Systems, help files and an implementation of the Event History Calendar.

For the Understanding Society study a management layer was build around the LMU to better manage the translation process. Problems were encountered with the Blaise DEP not supporting Unicode and a Unicode Enabled data entry program Unitip was created. This tool and process are described in the 2009 IBUC paper “An End-to-End Solution for Using Unicode with Blaise to Support Any Language” (Amin et al., 2009).

Early 2011 CentERdata was asked to develop and host a multilingual web questionnaire for a dozen European countries. Although the LMU tools were originally designed for CAPI questionnaires only, it was decided to try to adapt the code to get it working for web questionnaires as well. The experiences of rebuilding the LMU for this project could be used to design a new system that would fully support multi-mode. In section 2 our findings and some ideas we got in the process are described.

Over the years the LMU database got filled with question texts, answers, fills, labels and their translations in various stages of the questionnaire design over multiple studies over multiple waves in over 40 languages with various character sets. We decided to see if it was feasible to build alternative interfaces on this data; interfaces to analyze the translation process and questionnaire development or a system that could function as a question bank. To see what exports could be made. Maybe even create new questionnaires directly from the questions in the database. In section 3 some experiments with new interfaces on the data, some ideas on exports and linkage to a data dissemination system is discussed.

Based on the findings from rebuilding the LMU for web mode and the discussion and experiments that were done for question banks some architectural flaws were discovered. The code base and database design needed to be completely rewritten to be ready for challenges in the near future. In section 4 the new design and interface will be discussed.

2 Translating for web mode

For a study on sustainability CentERdata was asked to create a web questionnaire for 12 European countries. (Austria, Denmark, Finland, France, Germany, Hungary, Italy, Netherlands, Poland, Spain, Sweden, UK). It would seem logical to use the LMU to accommodate the translation process. But it was originally built to support the translation process for Blaise Questionnaires in CAPI mode only. This meant the Blaise source code was structured in a manner that could easily communicate with the LMU. For example the assumption was made a question existed of the following 6 components, making it easier to parse the source code:

```
QuestionName (QuestionTag)
  "QuestionText
  InterviewerInstruction"
  /"Description" :
Answer
```

Other texts were loaded using fills, either attached to a question or defined on a global level. One of the key thoughts behind the LMU is that translators should be presented the questions in the order and with the texts of the questionnaire that will be available during the field. A translator should translate questions not text strings without context.

In our experience when programming for CAWI the structure of the source code can become quite different than for CAPI. For our CAPI environments we normally show 1 question on screen. In CAWI more tables are used; more questions are on the screen simultaneous, the layout tends to have impact on the way the questionnaire is programmed. Therefore the strict definition of a question we used in LMU earlier didn't fit this mode. Several approaches were tried to adapt the LMU to support more complex structured questionnaires.

First we attempted to extend the parsing of the source code. We tried to catch tables, blocks and more difficult structures and stored them in the databases and displayed them in the LMU web environment. These more complex structures were stored in xml structures in the original LMU tables. This approach felt a bit stupid. Trying to parse source code looking for structures seemed an effort that would only set new boundaries and would again force us to limit the ways in which we could structure our Blaise programs.

It was decided to try the reverse approach; instead of trying to get the structured translatable elements out of a Blaise questionnaire, we would identify the translatable elements in the Blaise source code. We coded question texts with 'tags'. To the text strings in the original source code some xml like tags (</>) surrounded the original text. This approach would also allow using the LMU for other purposes as well, for other structured texts we could design tags that could be understood and displayed by the LMU

eca132

```
"<t id='IS:eca132' type='questionnaire:text'>Do you want to receive  
a feedback report that compares your response to the survey with the  
average response of enterprises in your sector and country?  
The report will be send to the e-mail address on which you received  
the invitation to complete this survey.</t>": T_YNN
```

These tagged strings made it possible to easily import the source code into the LMU, extract the tagged elements and store in the proper place based on their type and id. Also returning a translated text back into the original questionnaire was easy. Simply search the original tags and replace the full string with its translation.

eca132

```
"@#Haluatteko palauteraportin, jossa vertaillaan antamianne  
vastauksia maanne ja liiketoiminta-alanne keskiarvoihin?  
Raportti l&#228;hetet&#228;&#228;n samaan  
s&#228;hk&#246;postiosoitteeseen, johon saitte kutsun  
t&#228;ytt&#228;&#228; t&#228;m&#228; kysely.@#": TYesNo
```

In a web questionnaire html codes for problem characters or Unicode can be used as well.

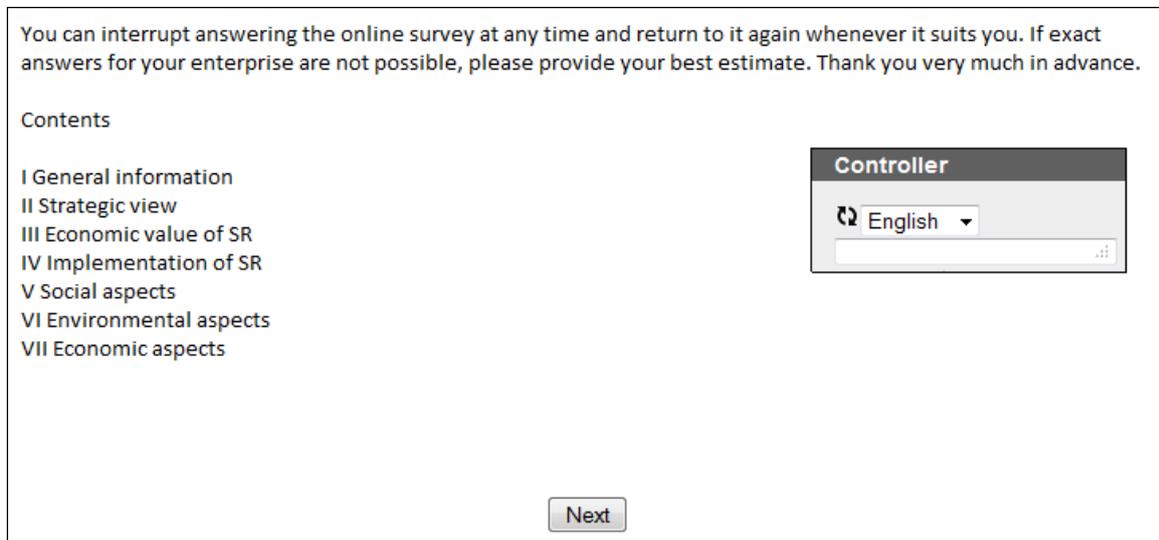
This worked reasonably well. It was annoying to manually tag the Blaise source, but you had to do it only once. Our questionnaire scripters got used to it fairly quick.

With this structure in place and the twelve countries responding to the questionnaire it was time to see if we could use this in a more generic manner. A special version of the questionnaire was exported. It didn't replace translation texts in the source code but replaced the tags with set a html-div block with an id and class.

eca132

```
"@#<div class=""RemoteTextLoad"" id=""IS:eca132""></div>@#": TYesNo
```

Using some JavaScript, adapting the style sheet and introducing floating div control menu this questionnaire was tricked into loading questionnaires directly from the LMU database, extracting translated texts at runtime. This seemed to give us the possibility to set up a version of the questionnaire where translation changes could be reviewed immediately without running a process to paste translations into source code and compile it. This meant testing of the questionnaire translation could be done immediately. This could shorten development time.

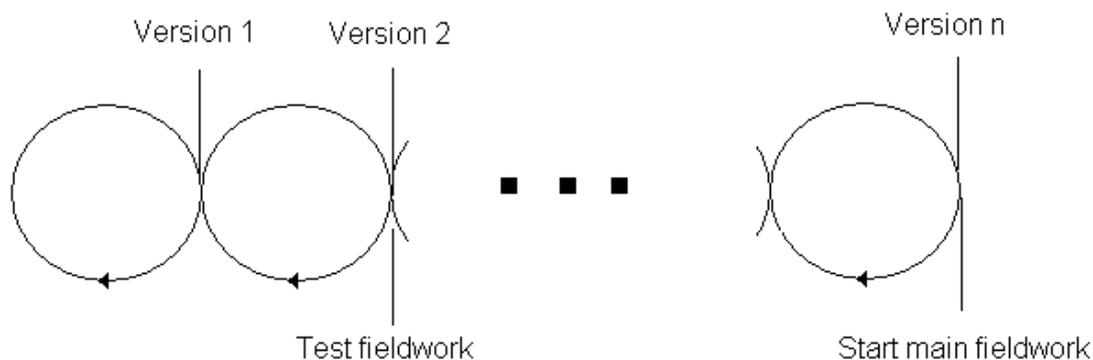


Unfortunately inserting the tagged-elements in the generic source code was not very user-friendly. To a certain extend we could have automated tagging elements but we would be trapped again in parsing source code.

This let us to investigate whether it is possible not to use the source code at all for this purpose. The backend of our translation software and Unitip were already built on the Blaise API. So it wasn't a big step to use the Blaise API to acquire the structure directly from the compiled questionnaire. This would however mean we had to redesign the LMU's database architecture completely to be more compatible with the internal structure of the Blaise Datamodel.

3 Ideas

The LMU for Share contains 4 waves of questions. Each wave consists between 7 and 13 cycles in which the questionnaire was adapted or translated. Some cycles end with a version of the questionnaire that is fielded; a pilot, pretest and main field work version. Each cycle itself consists of sub cycles where the questionnaire is fine-tuned, smaller bugs are resolved or translations are improved.



In each cycle a version of the questionnaire is saved for each translation. This led to a large collection of questions, questionnaire elements and their translations.

We got the idea that analyzing the translation process might say something about the quality of the questionnaire. If a question changes a lot through the translation process, this is an indication that it is hard to translate and therefore it is likely problems will occur that will influence the data collected with these questions.

To determine what questions changed and to what extent they changed the Levenshtein distance divided by the maximum number of strings provided was used. The Levenshtein distance is the number of changes you have to make to change a string into another string. Also an interface based on a 'track changes view' was created. This measure can also be used to determine to what extent a translation is changed. Comparing this with other translations or with the change in the original question could also be used to track potential problems.

281	BR001_EverSmokedDaily	w2, w4	100	The following questions are about smoking and drinking alcoholic beverages. Have you ever smoked cigarettes, cigars, cigarillos or a pipe daily for a period of at least one year?	The following questions are about smoking and drinking alcoholic beverages. Have you ever smoked cigarettes, cigars, cigarillos or a pipe daily for a period of at least one year?
282	BR005_WhatSmoke	w2	0	What ^FL_BR005_1 ^FL_BR005_2 ^FL_BR005_3? READ OUT; CODE ALL THAT APPLY	What ^FL_BR005_1 ^FL_BR005_2 ^FL_BR005_3? READ OUT; CODE ALL THAT APPLY
283	BR022_StoppedSmoking	w2, w4	96	Have you stopped smoking since we last interviewed you in ^FL_BR022_1?	Have you stopped smoking since we last interviewed you in ^FL_BR022_1?
284	BR010_AlcBevLastThreeMonth	w2, w4	94	I am now going to ask you a few questions about what you drink - that is if you drink. Please look at card 14 During ^ShowCardID During the last 3 months, how often have [n]did you drunk drink[n] any alcoholic beverages, like beer, cider, wine, spirits or cocktails?	I am now going to ask you a few questions about what you drink - that is if you drink. Please look at card 14 During the last 3 months, how often have you drunk any alcoholic beverages, like beer, cider, wine, spirits or cocktails?

To properly analyze the translation process we however need to know all the steps a translator went through. The version based setup of the LMU was not suitable for this.

While experimenting with an automatic import of previous translations by calling the Blaise API via PHP we created a few scripts. The old VB6 Paper version generator we used to create readable overview of the questionnaire routing was rewritten in PHP, an upload form was added and this made it possible to create the paper versions online.



```

IF MN101_Longitudinal = 0
PROCEDURE Txt_FL_DN029 (Procedure) ( {Parameterlist:}piIndex, MN002_Person{1}, Gender, FL_DN029_1, FL_DN029_2, FL_DN029_3)
Sec_DN2.Parents.Parent{2}.DN029_JobOfParent{0}
What was the job*FL_DN029_1*FL_DN029_2 had when you were about 10 years old?
Please give the exact description.

IWER:
E.g. not 'clerk' but 'forwarding merchant', not 'worker' but 'engine fitter'. In case of a civil servant, please get first official title, e.g. 'police constable' or 'student teacher'. Only if person did never do any work
for pay, enter 'housewife/-husband'.
ENDIF
PROCEDURE Txt_FL_DN051 (Procedure) ( {Parameterlist:}piIndex, MN002_Person{1}, Gender, FL_DN051_1, FL_DN051_2, FL_DN051_3)
Sec_DN2.Parents.Parent{2}.DN051_HighestEduParent
Please look at card *SHOWCARD_ID. What is the highest school certificate or degree that *FL_DN051_1*FL_DN051_2 has obtained?

IF DN051_HighestEduParent = a97
PROCEDURE Txt_FL_DN052 (Procedure) ( {Parameterlist:}piIndex, MN002_Person{1}, Gender, FL_DN052_1, FL_DN052_2, FL_DN052_3)
Sec_DN2.Parents.Parent{2}.DN052_OtherHighestEduParent
Which other school certificate or degree has *FL_DN052_1*FL_DN052_2 obtained?
ENDIF
PROCEDURE Txt_FL_DN053 (Procedure) ( {Parameterlist:}piIndex, MN002_Person{1}, Gender, FL_DN053_1, FL_DN053_2, FL_DN053_3)
Sec_DN2.Parents.Parent{2}.DN053_FurtherEduParent{1}
Please look at card *SHOWCARD_ID. Which degrees of higher education or vocational training does *FL_DN053_1*FL_DN053_2 have?

IWER:
Code all that apply

```

Other exports include overviews of fields and properties and of conditions that should hold for a field in excel. It should also be quite easy to export in xml. These are potential building blocks for data dissemination systems, question banks and the Language Management Utility.

4 Complete redesign

The experiments and ideas presented in previous sections made it clear that a redesign of the LMU could open up a whole new range of possibilities. We had to improve the versioncontrol, make the number of languages dynamic, support multiple questionnaires and make the backend system part of the webenvironment.

In the original design there were no dates attached to translations and there was only one save per version. This made it impossible to do proper analysis on the translation process. If every change was stored we could revert to a previous state when problems occurred. This would also open up the option to view history for each translated element to the translator. Therefore we removed the notion of versions altogether. A questionnaire gets updated, changes are now automatically tracked and flagged and a translator can come in any moment and translate elements that changed.

The translation languages were stored statically in the database before. If a language was added we had to add columns to the tables. This was adapted to a dynamic option. The elements table was introduced, it stores all translated strings with language_id as a foreign key. This was linked to a languages table that was extended to include settings for character sets for export.

The concept of procedures was unknown to the LMU before. If translations were stored in a procedure, we had to tag them with a code to find them. A procedure-table was included to find fill values that were loaded through procedures.

The LMU supported only one questionnaire per install. Now it is possible to insert multiple questionnaires simultaneous. As the number of questionnaires included grows, the questions and translations will be a rich source for new translation suggestions and make it possible to do cross questionnaires comparisons.

Translated versions can be generated directly from the web-interface. This means the Blaise Generator and Multi Blaise generator are no longer needed.

The screenshot shows the LMU interface. On the left, there is a sidebar with a 'Word count: Not available' and a 'Quick jump' section with links for Showcards, Capi Labels, Sms Labels, Standard Types, and Modules. Below this is the 'Current Module:' section, which lists 'Section_BR' and a long list of other sections (SC, XT, CM, DN1, SN, DN2, CH, PH, BR, CF, MH, HC, FP, GS, CS, SP, FT, HO, HI, CO). The main area is titled 'Section_BR - Danish (Denmark)' and contains three tabs. The first tab, 'BR001_EverSmokedDaily', contains the text: 'The following questions are about smoking and drinking alcoholic beverages. Have you ever smoked cigarettes, cigars, cigarillos or a pipe daily for a period of at least one year?' with options '1. Yes' and '5. No'. The second tab, 'BR002_StillSmoking', contains: '[The following questions are about smoking and drinking alcoholic beverages. /{empty}] Do you smoke at the present time?' with options '1. Yes' and '5. No'. The third tab, 'BR003_HowManyYearsSmoked', contains: 'For how many years have you smoked all together?' with a note 'IWER: Don't include periods without smoking Code 1 if respondent smoked for less than one year' and options '1. Ja' and '5. Nej'.

On the left side of the interface, the routing options are displayed, making it easier to access other modules. In the center a tab-controlled work-area is implemented. We can add new functionality easily by adding new tabs.

The screenshot shows the LMU interface for editing a question. The left sidebar has a 'Set status:' section with radio buttons for: Under development, 1. Awaiting translation, 2. Translated, ready for check, Check ok, 3. Check failed, 4. No translation needed, Imported from other translation, and 5. Imported. Below this is 'Current Question:' showing 'Section_BR > BR001_EverSmokedDaily' and a list of other sections. The main area is titled 'BR001_EverSmokedDaily - Danish (Denmark)' and has a 'translate' tab. At the top right of the main area are buttons for 'history', 'QbyQ', and 'discussion'. Below these is a 'Submit' button. The main content is split into two columns: 'Generic question:' and 'Translation for Danish (Denmark):'. The generic question text is: 'The following questions are about smoking and drinking alcoholic beverages. Have you ever smoked cigarettes, cigars, cigarillos or a pipe daily for a period of at least one year?'. The translation text is: 'De følgende spørgsmål drejer sig om rygning og alkohol. Har De nogensinde dagligt røget cigaretter, cigarer, cerutter eller pipe i en periode på mindst ét år?'.

The web-environment is a huge improvement. The translators in the fifth wave of share enjoyed working with this new approach to translating. The automatic import and direct options to export source code shortened the development process tremendously. The new Blaise DEP 4.8.3 supports different character sets much better than before, including full Unicode support. It seems we can now use the Blaise DEP to display questionnaires again and don't need our Unitip tool anymore for this purpose. This also means we don't have to work around the problems with accessing the activelanguage-property anymore. The described changes open up a wide range of future improvements.

Leveraging the Capabilities of the Blaise Editor Add-In Tool to Improve the Readability of Datamodel Source Code

*Chris Oz,
Survey Research Center, University of Michigan*

1 Introduction

One of the most pervasive issues that have plagued mankind's communication since the creation of the first written language has been one of readability. Regardless of the language used, books and newspapers need to be written in such a way that the information being conveyed is clear, concise and scannable—clear, so the reader understands what is being expressed; concise, so the thoughts and opinions expressed are not so overly wordy that they are incomprehensible; scannable, so the reader can efficiently read the text, with the minimum amount of re-reading possible. While a writer is not required to use proper grammar, punctuation and sentence structure to express thoughts and opinions, these basic rules of communication prevent the reader from becoming lost, confused, or even disinterested in the subject matter. This is the true litmus test for what makes a book well written, and software follows much of the same tenets. The world of computer code and software development is no different; the code that authors write is a list of instructions in a specific language that the computer is expected to carry out.

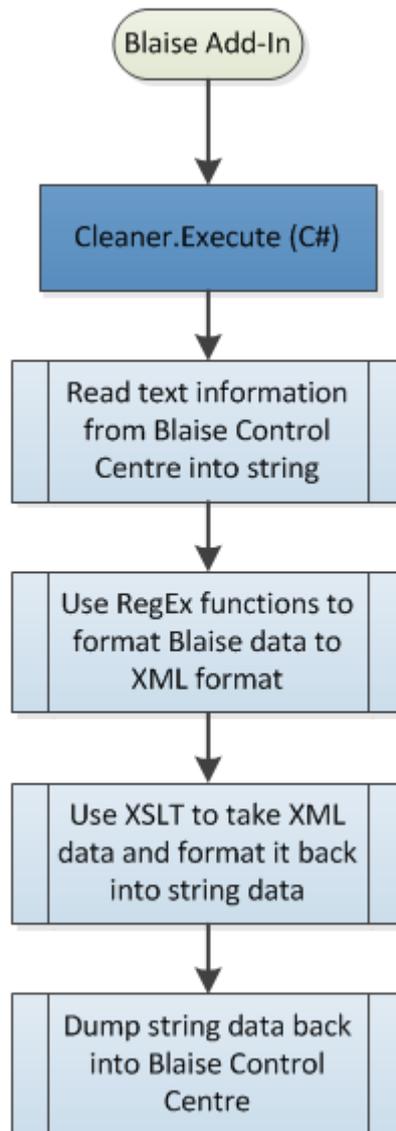
A software developer can write computer code that is for all intents and purposes correct, yet write it poorly enough that another author, unfamiliar with the code, would have a hard time following what is going on. This is a problem, because not only does it take longer to read and comprehend poorly written code, it also takes longer for someone to modify it. The Blaise Control Centre is a powerful and diverse tool for survey authors to create complex and extensible surveys; however, this flexibility allows for authors to write syntactically correct code with little regard for formatting and readability. To promote best practices in survey programming, this paper will discuss how a custom add-in was developed inside the Blaise Control to make code readable, scalable and scannable.

2 Overview & Approach

As just mentioned, this paper describes a solution for reformatting code using a custom add-in inside the Blaise Control Centre. There were multiple goals intended for this endeavor. Firstly, this programming solution needed to be fast, efficient and unobtrusive. Blaise authors creating production survey instruments needed a tool that would not hinder their work. Secondly, the solution needed to be flexible, so that research institutes can modify the formatting rules to their liking. Each research institute has their own standards and best practices for survey programming, so the solution developed in this presentation needed to easily accommodate them all.

A C#.NET class library was developed as the main add-in component for the Control Centre to interact with. This class library contained several sub-functions that serve as a step-by-step list of processes used to reformat Blaise source code.

First, information is read from the Control Centre application into a string variable. Second, regular expressions are applied to the string data to turn it into a strongly-typed XML document. Third, the XML data is passed into an XSLT to determine how the Blaise source code should be formatted. Finally, the formatted source code is “dumped” back into the Control Centre application. Below is a high-level overview of the custom class library that was developed.



Each sub-section of the diagram above will be discussed in detail further in this paper.

3 Retrieve Text from Blaise

The first sub-function that this custom add-in performs is reading the data from the Control Centre into a string variable. Ideally, the preferred method to achieve this would be to hook into the Blaise Control Centre, locate the active document window, and retrieve its contents into a string variable. However, because of the current functionality of the Blaise Control Centre add-ins, a different and slightly less elegant approach was implemented. Microsoft's .NET libraries were used to simulate a set of keyboard commands; select all (Ctrl – A) to select the entire text of the active window, and copy (Ctrl – C) to copy it to Windows' clipboard memory area. Once in the clipboard memory, another .NET library was used to access the clipboard memory. Its contents are transferred from the clipboard into a string variable that will be used throughout the programming of this custom add-in.

4 Format via regular expressions into XML

Once the text has been extracted from the Control Centre and stored in memory, the next step is to convert the text into some kind of standardized data format. This is an important step, because we

cannot rely on the current formatting inside the Control Centre as a basis for conducting the reformatting that is desired. For example, if two lines of code were in a Blaise instrument that used the USES statement, it would be difficult to account for the layout they currently have, and what they need to be. The picture below describes such a problem.

```

USES
    State      'State'

USES
Country      'Country'

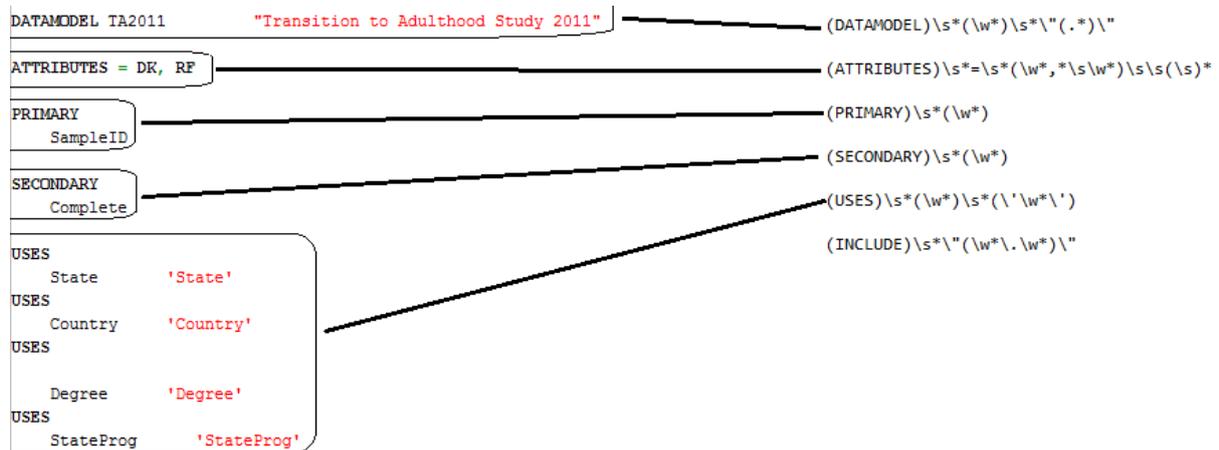
USES
Degree      'Degree'

USES
StateProg   'StateProg' |
  
```

Example of non-standard formatting.

From this example, it becomes clear that while this may be functionally correct, it is difficult to read, and also difficult to programmatically reformat. This is why a standardized data format like XML is so important.

To achieve this transformation to XML, the original text string is filtered through several regular expressions. Regular expressions are search terms and filters that provide a concise yet flexible means for authors to match strings, and search for character patterns inside a string of text. These regular expressions are what turn the Blaise source code into XML data. Below is an example of the process from Blaise instrument source code to XML transformation



Regular expressions to the right capture certain areas of the Blaise programming, on the left.

As shown in the screenshot, some clever regular expressions were needed to determine where each “chunk” of code starts and ends. It is important to capture these chunks of code accurately, so that the XML data is not malformed. Once the source code is passed through all of the regular expression filters, a complete XML data structure will be created. The XML elements are instrumental for the XSL transformation.

5 Transform XML through XSLT

Thus far, two sub-functions of this custom add-in have been covered. First, the Blaise source code has been retrieved from the Blaise Control Centre and stored in a string variable in memory. Second, the string variable has been filtered through several regular expressions, resulting in an XML formatted data document. The next step in this process is to take the XML data, and process it through an XSLT file to get the final Blaise source that will go back inside the Control Centre application.

An XSLT file is a declarative, data transforming schema used to turn XML type data into data of other types and formats. The XSLT approach was taken for this presentation for two reasons. First, XSLT files were specifically designed for strongly typed markup languages such as XML, and HTML to a lesser extent; other approaches such as the .NET library would be more difficult and time-consuming to implement. Second, XSLT files are easily customizable. This goes back to one of the principal requirements for this project, that the rules for formatting the code should be adjustable for each study center that would use this application. One study center may prefer two tab characters before an INCLUDE statement, where another study center may only wish to have one. Below is an example XSLT.

```
<xsl:template match="root">
  <xsl:apply-templates select="@* | node()" />
</xsl:template>

<xsl:template match="DataModel">
  <xsl:value-of select="'\r\n'" />
  <xsl:value-of select="'DATAMODEL '" />
  <xsl:value-of select="@ID" />
  <xsl:value-of select="' '" />
  <xsl:value-of select="concat('&quot;', @Desc, '&quot;')"/>
  <xsl:value-of select="'\r\n'" />
</xsl:template>

<xsl:template match="Attributes">
  <xsl:value-of select="'\r\n'" />
  <xsl:value-of select="'ATTRIBUTES = '" />
  <xsl:value-of select="@Attribute"/>
  <xsl:value-of select="'\r\n'" />
</xsl:template>

<xsl:template match="Primary">
  <xsl:value-of select="'\r\n'" />
  <xsl:value-of select="'PRIMARY '" />
  <xsl:value-of select="@Key"/>
  <xsl:value-of select="'\r\n'" />
</xsl:template>

<xsl:template match="Secondary">
  <xsl:value-of select="'\r\n'" />
  <xsl:value-of select="'SECONDARY '" />
  <xsl:value-of select="@Key"/>
  <xsl:value-of select="'\r\n'" />
</xsl:template>

<xsl:template match="Uses">
  <xsl:value-of select="'\r\n'" />
  <xsl:value-of select="'USES '" />
  <xsl:value-of select="'\r\n'" />
  <xsl:value-of select="@Name"/>
  <xsl:value-of select="' '" />
  <xsl:value-of select="@Value"/>
</xsl:template>
```

Sample XSLT style sheet that reformats Blaise statements.

As shown in the screenshot, this XSLT file looks for the “elements,” such as “Primary” and “Uses,” that we created in the first step when we made the XML file. Once the appropriate elements are found, they are replaced with the formatted text. This process is called transforming, and the end result of the transforming process is the formatted text that will ultimately be dumped back into the Blaise Control Centre.

6 Dump clean text into editor

At this point, the formatted text is ready to be transferred back into the Control Centre. As mentioned earlier, the very first step taken in this process was to transfer the text being worked on into the Windows clipboard by way of simulating keystroke commands. For the final step in this process, the text inside the Control Centre will be again selected in its entirety, in the event that the user interacted with the Control Centre during the reformatting process. The formatted text is pasted back into the Control Centre by again making use of simulated keystrokes. At this point, the user can review the formatted code, and decide whether or not it should be committed to use.

7 Lessons Learned

There were several lessons learned in the development of this paper, as well as the development of a proof of concept prototype. First, and perhaps most important, it was discovered that there is no pre-built library provided to authors for accessing multiple open files within the Blaise Control Centre. If one were to try to extend this application to reformat several open files at once it would be a difficult undertaking and require a great deal of advanced programming. At this time, the most efficient approach would appear to be to click on each file and run the custom add-in to clean the source code for each individual file. Furthermore, using the approach of simulating keystrokes to “Select All” and “Copy” allowed authors to input keystrokes manually, potentially breaking the Select All/Copy methods. If there continue to be no Blaise libraries available for accessing open files, it is highly suggested that additional programming be implemented in this custom add-in to temporarily disable the keyboard until the data are stored in the clipboard memory. There are multiple approaches available to achieve this, ranging from writing code to trap keyboard input and prevent it from being processed to using .NET libraries to lock out keyboard input altogether.

Additionally, in the course of developing a prototype, it was discovered that pasting reformatted code back inside the Blaise Control Centre cannot guarantee 100 percent consistent text formatting. For example, if an author was working on a Blaise instrument that had several USES statements, they could manually use spaces and tabs to format the USES statements properly, regardless of the length of the variables being used. Using the custom add-in developed as part of this paper, there is a slight variance in the spacing of these USES statements, because of the length of the variable names. Additional string formatting in the XSLT could address this. These issues, while noticeable, do not appear to provide any significant impediment to further development and usage of this custom add-in.

```
USES
    State      'State'
USES
    Country    'Country'
USES
    Degree     'Degree'
USES
    StateProg  'StateProg'
```

An example showing spacing irregularities between variables and their values.

8 Conclusion

As mentioned in the beginning of this paper, computer code is similar to books, newspapers, or any kind of printed media; it needs to follow standards of readability, scannability, and conciseness in order to maximize productivity and minimize time reading the code. Authors, like writers, need to use their chosen language to provide documents that are not just correct, but cleanly written. This paper has shown in detail how to achieve these best practices for Blaise instrument programming, by using the add-in manager to develop effective third party add-ins. While the Blaise Control Centre does not provide this functionality natively, the custom-add-ins provides a powerful and flexible method for creating custom programming that interacts with the Control Centre. It is hoped that while Blaise continues to evolve and improve as an industry leader for survey programming, more flexibility will be built into the Control Centre as well as the custom add-in manager so that survey authors can continue to strive to use best practices in software development to the benefit of everyone involved in the development and implementation of these instruments.

Data Transfer in Blaise Surveys with CAPI Collection Method

Lilya Luria, Central Bureau of Statistics, Israel

1 Introduction

Many surveys conducted by the Central Bureau of Statistics (CBS) in Israel are developed in Blaise. The most common data collection method in these surveys is the CAPI method, and therefore, there is a constant need to transfer data from the CBS network to the field and back.

The data from the field is transmitted through the 'Generic Broadcast System' - a user-transparent remote component which provides service for data transfer between remote environments and the internal network of the CBS. The data transfer is performed in both directions.

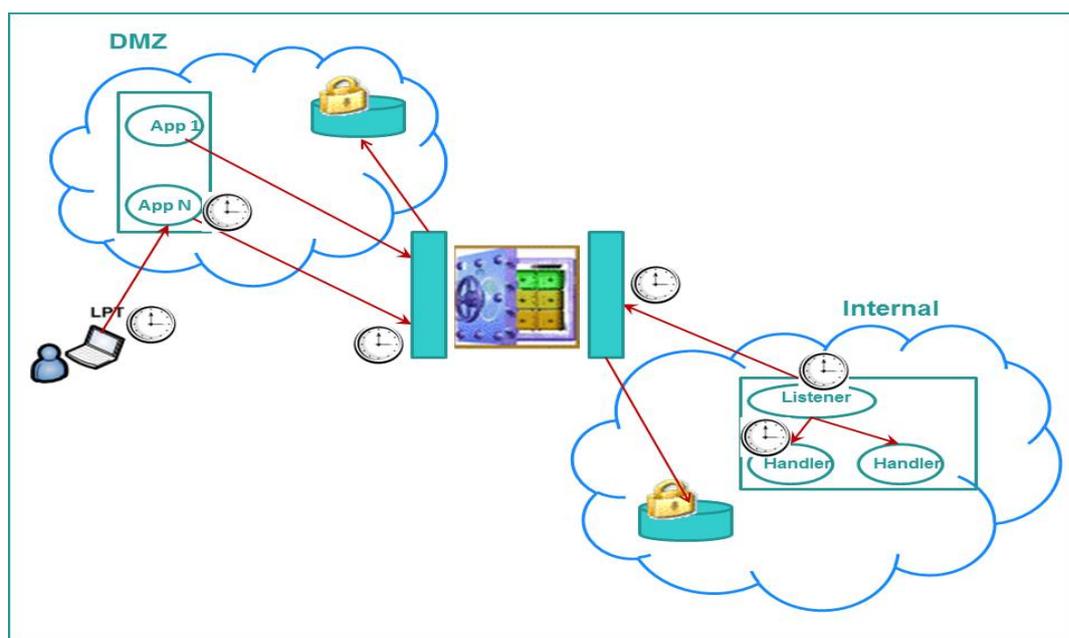
In order to connect the 'Generic Broadcast System' component, we have developed a set of generic applications which are used in all the surveys that are using Blaise for saving data on the one hand and using CAPI collection method on the other. These applications have been developed in Manipula and .Net using the BCP utility.

In this paper we will explain in details the process of sending Blaise files from the interviewer's laptop to the internal network and back and ways to troubleshoot.

2 The 'Generic Broadcast System' component

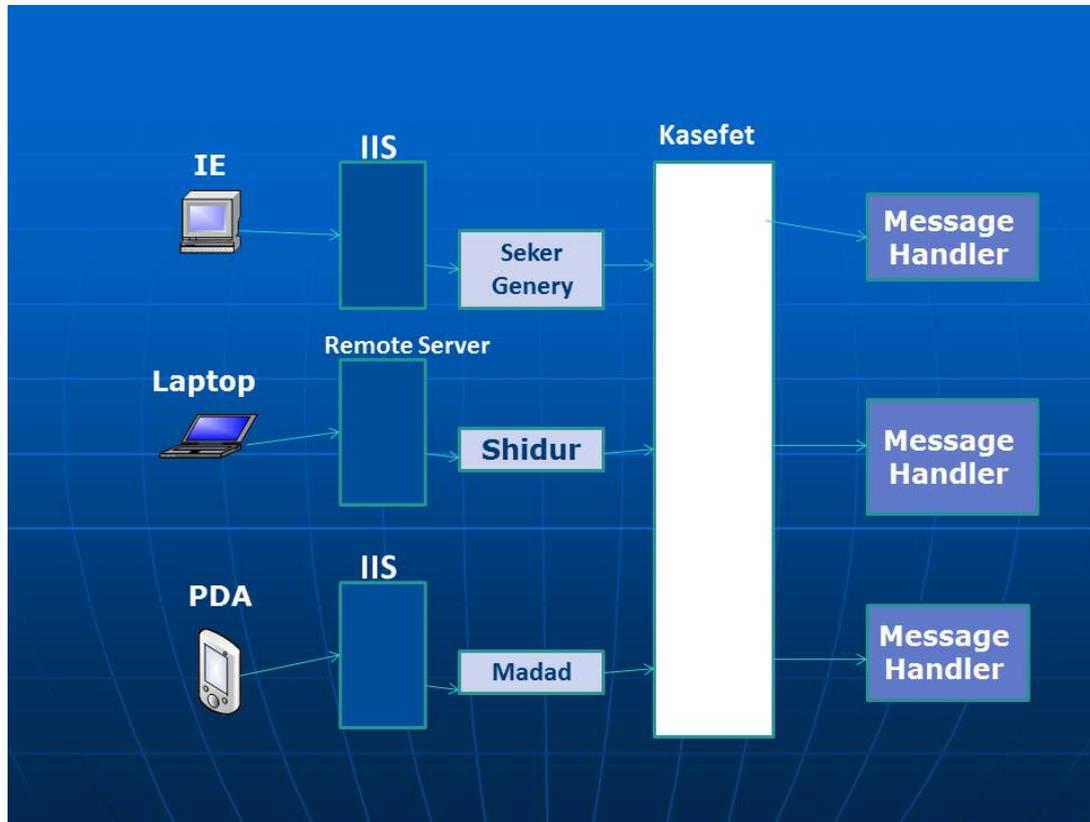
The 'Generic Broadcast System' is a component (.Net DLL) which has been developed by the team of generic applications at the CBS, and aims to meet the need to transfer data between different networks. This component is used today in all the surveys (Blaise surveys and others) which are using CAPI collection method.

The data is transmitted through the 'Kasefet' ("Safe") – the hardware & software component which main purpose is to store data and transfer it between networks.



In addition, the 'Generic Broadcast System' allows managing processes priorities, users' management and monitoring.

Data transferred via 'Generic broadcast' comes from different work environments: Web systems, laptop computers (CAPI), PDAs, etc., as shown in the following scheme:



In order to use the 'Generic Broadcast System', in each survey we need to implement a certain architecture which includes the following requirements:

Server Side:

- A dedicated collection server with folders for all interviewers: each interviewer has his own folder with a unique identification number (employee number).
- Within each folder, there are three sub-folders: Download, Upload and Backup.
- The Download folder contains files that are ready to be sent to the interviewers.
- The Upload folder contains files that have arrived from the interviewers.
- The Backup folder is used for a backup: at the end of each action all files are copied into the Backup folder. The Backup files are saved for a year - year and a half.

Client Side:

- In the laptop computers of all the interviewers we'll find the same three folders: Download, Upload and Backup.

- These folders are not per survey, they are used in all the surveys the interviewer works with.
- Each survey has an 'identification prefix' of 3 letters, for example, **Sch** for **SakaChodshi** (Monthly Labour Force Survey), **Pns** for **Pension** (Social Survey with an annual theme "Pension"), **Smr** for **SMArT** (Long-term Households Survey), etc. As a customization for each survey, the specific prefix is indicated in the configuration file allowing the 'Generic Broadcast System' to differentiate between the surveys applications/services.

3 Data transfer via 'Generic Broadcast System' in Blaise surveys

The Blaise development team has developed a set of generic applications for transferring data via the 'Generic broadcast'. These applications are used in all Blaise surveys at the CBS.

Two of these applications are used on the server side (CBS statistic network), and perform the following functions:

- Preparing and 'packing' Blaise files that are ready to be sent to the interviewer's laptop and placing those files within a dedicated collection server.
- Updating the Central Blaise DB at the CBS network with the data received from the field via the 'Generic Broadcast System'. This action is executed through different management applications and could be performed by the initiative of survey coordinators or automatically.

Two other applications are used on the client side (interviewer's laptop), and they perform the following functions:

- Preparing and 'packing' the data collected on the laptop to be transmitted via the 'Generic Broadcast System' to CBS statistic network.
- Updating the Blaise DB on the laptop with the data received from the CBS network, as well as software version updates.

4 Server Side Applications (Statistic Network)

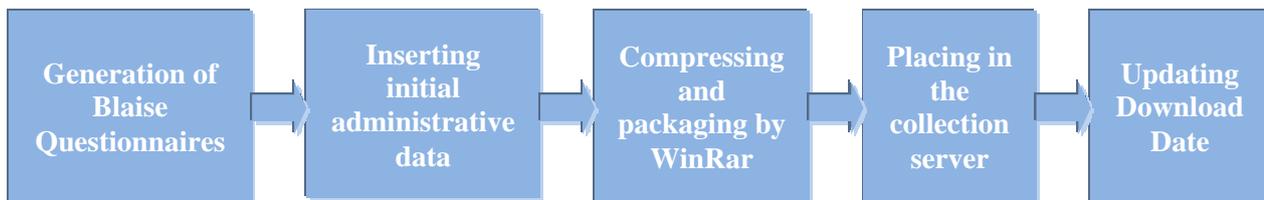
4.1 Delivery Preparation Application

This application is activated by region coordinators. The region coordinator sets collection of questionnaires to each interviewer, and then performs 'Delivery Preparation'. This delivery includes all the questionnaires that are set to the interviewers and not been sent yet.

The main functions carried out by this application are:

- Filtering all sampled that were set to a specific interviewer and have not been sent yet.
- Generating Blaise questionnaires for those sampled.
- Inserting initial administrative data into the Blaise questionnaire, such as data taken from the Population Register, property tax files and other sources.
- If this is not the "first cycle" of the questionnaire (the questionnaire has been sent to certain interviewer and then returned to the coordinator, and now the coordinator set this questionnaire to another or the same interviewer), in this case the information produced by the previous interviewer inserted into the questionnaire, such as Actions Record Table – a table documenting all actions taken in this questionnaire by all the interviewers who had handled it.

- Generating separate Blaise DB for each interviewer who supposed to receive the package, with only her/his questionnaires included.
- Compressing and packaging of all Blaise files which supposed to be sent to the interviewers (.bdb, .bdm, .bmi, .bjk, .bfi, .bpk) by WinRar software.
- The generated RAR file name is composed of the following parts:
 - First 3 letters are representing the name of the survey
 - The fourth letter is **M** which means the file is sent from the CBS center (**Merkaz**)
 - Partial date composed of the month, day and time.
 For example, the file named **SchM01231715** belongs to the Monthly Labour Force Survey (**Saka Chodshi**) sent from the CBS Center (**Merkaz**) to the interviewer on 23/01 at 17:15.
- Placing the created RAR file in the 'Download' sub-folder of the interviewer's folder (The interviewer's "cell") in a dedicated collection server.
- Updating Download Date at the Central DB



Although we do not know when the interviewer will draw the files sent to him from the collection server, once the questionnaires were sent they are considered as being under the interviewer's handling and the survey coordinator cannot perform certain operations on these questionnaires.

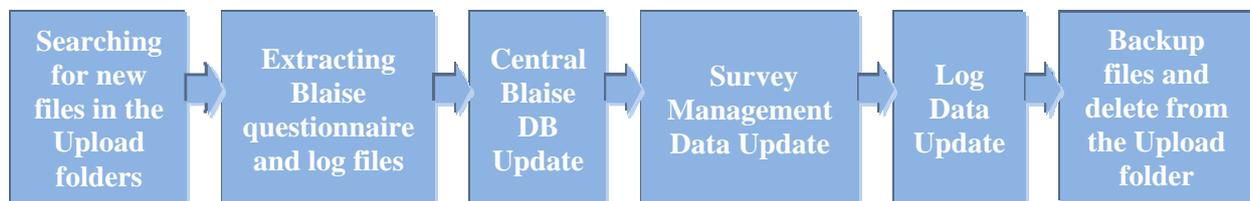
4.2 Receiving Questionnaires from the Interviewers Application

The purpose of this application is to withdraw from the collection server all the material sent by the interviewers and update the data in the Central Data Bases. In some surveys this is done automatically, on Windows Tasks Scheduler, usually at night, and in other surveys the material withdrawal is triggered manually by region coordinators through the survey management application.

The main functions carried out by this application are:

- Retrieving from the Survey Coordinator's management system a list of all interviewers working under the coordinator who activated the application.
- Check for each interviewer's number in the list: whether there is a material (files) in the 'Upload' sub-folder of the interviewer's "cell" in the collection server - those are the last files sent from the interviewer's laptop and have not been collected by the survey coordinator yet. For each survey, the application searches only for the files belonging to the same survey, and this identification is, as mentioned above, by the three first letters of the file's name, representing the name of the survey.
- The files arrived from the interviewers are compressed files with a '.RAR' extension and can be one of two types:
 - A file containing the questionnaires handled by the interviewer;
 - A file containing the log data (daily work reports each interviewer fills in the laptop).

- The received file name consists of the following parts:
 - First 3 letters representing the name of the survey;
 - The fourth letter is **S** (Soker - Interviewer) in a file containing the questionnaires, or **Y** (Yoman - Log) in a file containing the data from the daily reports;
 - Partial date composed of the month, day and time.
 For example, two files found in the 'Upload' sub-folder of the interviewer X:
 SchS01261825.RAR
 SchY01261825.RAR
- The Blaise files (.bdb, .bdm, .bmi, .bjk, .bxi) are extracted from the RAR file at the collection server.
- Analysis of management information in each questionnaire that arrived, especially from the "Actions Record Table": the questionnaire status (completed / non-response), the reason of non-response, the follow up status, etc.
- The questionnaire management information is checked to determine if the questionnaire meets the criteria for updating, if it is - the arrived questionnaire is being saved at Central Blaise DB.
- Updating management data of the survey in accordance with management data in the questionnaires, such as Upload Date, Questionnaire Status, the reason of non-response, etc.
- Updating the log data in the survey management system.
- Backup files in the 'Backup' sub-folder of the interviewer's folder.
- Deleting files from the 'Upload' sub-folder of the interviewer's folder.



5 Client Side Applications (Interviewers' Laptops)

Each interviewer who works in one of the Blaise surveys, has another icon on her/his laptop's desktop in addition to the survey's application icon – the adjusted generic application by which 'Generic Broadcast' is activated.

Clicking on the icon activates the application consisting of 4 main parts:

- Preparation of Blaise questionnaires 'package' ready to be sent to the Central DB.
- Call the 'Generis Broadcast System' DLL which performs data transfer in both directions.
- Updating Blaise DB on the interviewer's laptop with the questionnaires arrived from the Central DB.
- Software Version Update (if relevant).

5.1 Preparation of Blaise questionnaires delivery

The application checks for questionnaires that must be delivered to the Central DB. In fact, the selection logic for sending questionnaires from the laptop is fairly simple - every questionnaire that has at least one action recorded in Action Record Table – is taken to the delivery. This application does not perform a thorough analysis of the questionnaire: whether it must be updated in the Central DB, what status it should be given, etc. This operation is performed on the server side.

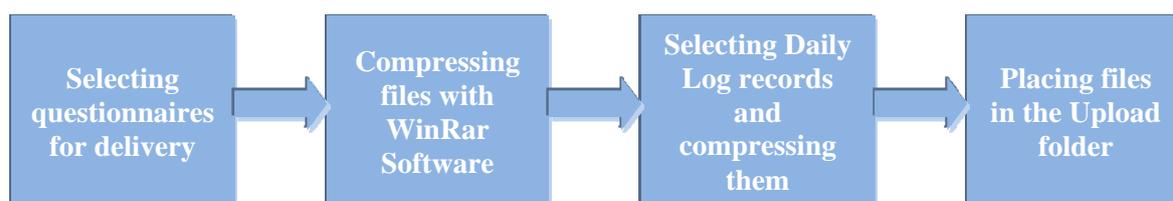
All questionnaires that meet the delivery criteria are copied to a separate Blaise DB intended for the delivery.

Then these Blaise files are compressed and 'packaged' by the WinRar software with a '.RAR' extension (similar to Delivery Preparation on the server side).

In addition, there is another Blaise application installed in all interviewers' laptops called "Yoman" ("Daily Log") - an application through the interviewer fills her/his daily work reports, and it is the same log for all the surveys.

While preparing the delivery, if the interviewer has filled new records in the log (daily report), they are packed for delivery as well, though in separate RAR file.

These RAR files are copied to the 'Upload' folder on the laptop, as a preparation for data transmission.



5.2 Data Transmission

After preparing the delivery, our application triggers the 'Generic Broadcast' DLL.

For this purpose the interviewer must be equipped with landline or mobile phone (for transferring data between networks) + special identification token (for information security reasons).

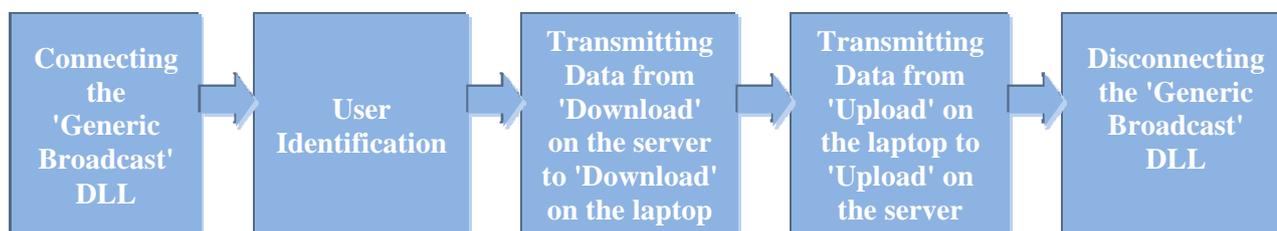
The interviewer receives the 'Generic broadcast' log-in window. The interviewer is required to enter predefined username and password, where the second part of the password is generated by clicking on the token, and expires after 30 seconds (for information security reasons).

Once connection is established, user authentication and password are being verified in front of the Central Broadcast DB.

Then the DLL turns to the collection server and searches in the 'Download' folder of the given interviewer whether there are files waiting to be sent from the collection server to the interviewer's laptop. If there are, the 'Generic broadcast' transmits these files from the 'Download' folder on the collection server to the 'Download' folder on the laptop.

Next step, the Broadcast component performs the inverse operation: searches in the 'Upload' folder on the laptop whether there are files waiting to be sent from the laptop to the collection server. If any, 'Generic Broadcast' transmits these files from the 'Upload' folder on the laptop to the 'Upload' folder on the collection server.

At the end of these operations our application disconnects from 'Generic Broadcast' DLL and performs data update on the laptop.



5.3 Blaise DB Update on the laptop

If there are files in the 'Download' folder on the laptop, our application will extract these files (an UnRar action). The arrived files are Blaise files containing questionnaires sent by the coordinator. These can be new questionnaires or questionnaires that have been sent in the past, returned to the coordinator and now sent back to the interviewer (same interviewer or another one).

The application updates Blaise DB in the laptop with the arrived questionnaires and with the management data that is also found inside the Blaise questionnaires.

Rarely happens that the file arrived from the collection server, has the same name as already existing file in the 'Backup' folder, so the application checks the RAR file name before extracting files from it. If for some reason file with the same name already exists (including the date and time), the new file is saved in the 'Backup' folder with another extension (in order to not overwrite the existing file), for further clarification.

5.4 Software Version Update

Once our application has finished the questionnaires update process in the laptop, it checks whether there are also software version update files in the Download folder.

Occasionally it is necessary to update the software version (Blaise questionnaire or interviewer's management system) on the laptops. For this purpose, in addition to the usual file containing collection of questionnaires for the interviewer, software version update files are also sent from the central management to the laptop.

If the application detects there are version update files, the interviewer gets a notification about that.

For version update, 2 files usually sent to the interviewer's laptop:

- RAR file containing all the Blaise and Manipula files that must be updated on the laptop. If the new version of the questionnaire has different structure from previous version, the RAR archive also contains an '.msu' file that copies data from the Blaise DB on the laptop to the new structure.
- .BAT file containing all the commands to be performed on the laptop.

The application runs the BAT file, which usually performs the following commands:

- Extracting files from the RAR Archive.
- Running MSU file that transfers data from the old structure DB to the new structure DB (if required).

- Copying files from the 'Download' folder to the questionnaire DB folder in the laptop.

At the end of the process a backup of all files is done in the 'Backup' folder and all files from the 'Upload' and 'Download' folders are being deleted.

6 Issues / Problems & Troubleshooting

There are several problems / issues which arise from time to time during the process of data transfer.

The main issues are:

- **System Folders Update**

The hard disk in the laptops is divided into drives C and D.

For security reasons there is a very clear distinction between types of files installed on both drives.

All the System files are installed on drive C, including Blaise System files.

Drive D is designated to applications' installations of various surveys.

When we send a version update via the 'Generic Broadcast', in terms of the interviewer's permissions on the laptop we can update only the files located on drive D. In most cases this mode is completely satisfying, since the survey files that need to be updated are all located in drive D.

However, once in a while we need to update files in the Blaise System folder - a folder located on drive C.

For example, in laptops using the BCP component it is necessary to update Blaise license every year - quite problematic when the interviewers with their computers are distributed across the country. Theoretically, we could send a file with the license key through a version update for the interviewers, but since this file must be placed in the Blaise folder on drive C, for security reasons the interviewer cannot receive it through the 'Generic Broadcast'.

Therefore, as for today - just before the old license expired, all the interviewers arrive physically to the CBS regional offices and got their Blaise license updated manually by the technical team under Administrator permissions.

There is no doubt that this situation is not very convenient, especially in large surveys with a large number of interviewers.

Apparent solution in the near future:

One of the CBS development teams is working these days on developing a generic application for software versions updates on all laptops. The application will be activated while transmitting data via the 'Generic Broadcast'.

One of this application advantages is that the update will be performed under Administrator permissions, so it will be possible to update the files that are installed on drive C.

- **The need for a collection server as an intermediate stage**

One of the questions that arise from time to time, especially in new surveys, is why when the interviewer initiates the data transmission operation in the laptop, it is not fetching the material directly from the Blaise DB and survey management system on the internal network, or directly updating this systems with files sent from her/his laptop.

The fact that the data transmitted in both directions "stops" in an intermediate stage which is collecting server, and "waits" to be picked up, forces the survey coordinators to activate two actions: sending material to the collection server and picking up material from the server to update the system.

The first reason for this is data security. To reach the maximum security level, the material passed between the networks stops at the collection server and does not go directly to the survey servers while the communication channel is open.

Only after the interviewer had disconnected from the broadcast and the communication channel is closed, the survey coordinator can fetch the arrived material from the collection server to the survey servers.

Another reason for the existence of the collection server is a technical reason - updating the survey systems (Blaise DB and management system) directly from the interviewer's laptop will extend the duration of the transmission process, which could cause technical problems, and also for reasons of data security it is better to shorten the transmission time as much as possible.

- **Technical problems on servers**

Rarely, various technical problems occur on collecting servers, such as a slow server, sometimes resulting in disruptions in data transmission process.

As for today, a specific technical solution is given to any problem that arises locally, and the development team together with the technical staff restores files that have not been transmitted, in case it happens.

There are plans in the near future that the development team and technical staff will find more global solutions to such situations.

7 Summary

Data transfer from the field to the CBS internal network and back in Blaise surveys with CAPI collection method is performed in a generic way, same for all surveys.

This generality is expressed in using the 'Generic broadcast' DLL on the one hand, and the set of generic applications developed by the Blaise development team and using the 'Generic broadcast', on the other hand.

Each new CAPI survey conducted on by CBS joins this infrastructure, which enables adding each new survey to the data transfer mechanism in a convenient and relatively simple way, with only minor changes between the surveys.

CCMS: Capi Case Management System

Gerrit de Bolster and Peter Sinkiewicz, Statistics Netherlands

1 Intro

Data collection with Blaise is not limited to the design of questionnaires. The process of data collection also needs a logistic system to get sample data and instruments to the interviewers' laptop, and to return the respondent data back to office. Most organizations already made their own custom toolset, adapted to their specific infrastructure and standards. For organizations new to Blaise, reinventing such toolset is rather inefficient. Making use of existing knowledge present in the Blaise community is an appropriate way to go. Started as an initiative of the BCLUB, the Case Management working group focused on the requirements for a CAPI case management system. CCMS is developed to meet those requirements and will be made available to the Blaise users.

Figure 1. Logo of CCMS



1.1 Some characteristics of CCMS

- Based mainly on Blaise technology
- Cheap: additional functionality provided by open software/freeware
- To be used at the office side, by the interviewers (and their supervisors)
- Fully automated synchronization of data between office and interviewer
- Build with the KISS principle in mind: Keep It Simple and Secure
- Robust and foolproof; self-recovering
- Secure

1.2 The scope of CCMS

CCMS is not a complete survey management system. Its functionality is limited to the logistics between the office, supervisors and interviewers and to the case management system on the laptops of the interviewers and the supervisors. So on the office side the functionality of CCMS is restricted: it only takes care of storing interviewer and supervisor data and interchanging those data with the other parties in a robust and secure way. Which respondent should be assigned to which interviewer is outside the scope of CCMS.

2 Scenarios

CCMS recognizes 3 parties: the office, the interviewers and when applicable a third in-between party e.g. a regional office or a supervisor. The surveys and the survey data of respondents are considered as “payload”. There are different scenarios possible in the communication and tasks between the 3 parties. Unlike the office and the interviewers a supervisor/regional office is not always present in all the scenarios.

Due to technical or organizational limitations no single solution can suit the needs of each organization. We have considered the situation where an internet connection is available to all or part of the interviewers and the situation where interviewers have to visit their ‘regional’ office. Besides transmitting data through the internet also transmitting data by means of a memory stick or memory card has been taken into account.

2.1 Scenario “A”

In this scenario only two parties are involved: the office and the interviewers. The office provides the payload; the interviewer on its return will treat the payload and then return the results. In CCMS it is foreseen that the transport of the data between both parties can be organized in different ways: ideally the interviewer connects with its laptop on a regular basis through the internet to a (s)FTP-server located at the office, but the transport of the payload can also be accomplished in another way, like by a memory card through regular mail.

2.2 Scenario “B”

This scenario involves a third party between the office and the interviewer, like a regional office or agency or a local supervisor. There can be various reasons for having such third party: a traditional regional infrastructure, outsourcing of the interviewing, limited ways of transport or limited communication as in mountain regions. The role of the third party can be as little as just being a serving hatch, passing on the payload. But also other tasks could belong to this party, like quality control, payment of remunerations and so on. For some of those tasks it is possible that this third party needs access themselves to the contents of the payload or part of it. This is the case when the third party has the task to divide the workload between a group of interviewers.

The main concern of CCMS is to offer a solution for the logistics. As it comes for instance to the reassignment of the payload to another interviewer it is considered best to be done on one place, at the office. Any suggestions of scenarios to be taken care of by CCMS are welcomed.

3 Data communication

The data communication system must meet three basic requirements:

It has to be protected.

It must be robust.

It must be very easy to use (usability).

3.1 Protection

There are several communication protocols available. They can be divided into two groups: protected protocols (sFTP, FTP-s, HTTPs) and unprotected (FTP, e-mail, memory sticks/cards). Using a protected protocol it is not necessary to protect the payload. However, when applying an unprotected protocol it is a must that the payload is protected by encryption. With Blaise build 4.8.2.1606 strong encryption (AES) was introduced for the zip and unzip commands in Blaise. With this encryption it is very easy to protect the payload. It can cause a problem if e-mail is used. Providers can have filters that block encrypted attachments. In that case the only solution is to use a provider that, by agreement, does not block the encrypted zip files.

In the current CCMS we applied sFTP and memory sticks/cards and AES 256 bits protected zip files with a key of 64 characters. Each interviewer/supervisor has his own key provided by the office.

3.2 Robust

During a contact between the interviewer and the office (or supervisor) files are exchanged in two directions: from the office to the interviewer and vice versa. This synchronization must be robust. In other words: it must not be possible that a communication fails without being noticed. The best way to secure such synchronization is to work with confirmation messages. Especially when indirect communication is involved (e-mail, memory sticks/cards by mail) this is a must.

The synchronization is always initiated by the interviewer (client side) by starting this function in the CCMS. There are different reasons and preconditions for an interviewer to start this synchronization. In case of an online connection the connection must be established. In case the exchange takes place with offline media it can be the reception of a memory sticks/card from the office.

At the office side CCMS will be used to check on uploaded files from the interviewer and preparing files for download. In the majority of cases the office will have a scheduled plan for this action. This

is for certain the case if the synchronization takes place by sending memory sticks/cards as the process will not be started for each stick/card separately.

Given this situation it is better to perform on the client side (interviewer, supervisor) first the download and subsequently the upload. Applying this order, the confirmation of the downloaded packages can be included immediately in the upload. The confirmation of the upload by the office will be included in a later download. To secure the synchronization there must also be procedures installed that warn the parties involved when confirmations seems to be missing. They could be triggered by time and/or by e.g. a broken sequence number.

All interchange actions are recorded in a logging. The logging on the interviewer side can be send to the office on initiative from the office to check what happened in case of a problem. If necessary the office can update the logging on the interviewers' side. The logging is also used to register confirmations and to avoid double actions.

3.3 Usability

The data communication must not be too complicated to use especially on the side of the interviewer. Just a click on the button will invoke all the necessary actions. The system not only supports the interchange of surveys and respondent data but also the communication between the interviewer and the office/supervisor. Hence an additional separate e-mail system is not necessary.

4 Laptop access

The protection of the laptop itself is not an issue for the CCMS. It is assumed that the laptops are protected according to modern standards. Therefore the stored surveys, respondent data and statements are not additional protected. However, the use of CCMS on the laptop has its own (limited) protection to avoid unauthorized use (deliberately or accidental).

In the laptop part of CCMS there are 2 files essential for its operation: the **configuration** file and the **interviewer** settings file. More than one interviewer can use the same laptop. Each interviewer has its own sub-folder with surveys and respondent information.

The configuration file contains the supported data communication modes, data communication access info, used symbols and some general settings. It is a Blaise file with a code/value structure of which the value (a string) is encrypted with a special developed encryption method. To decrypt this value a main key and a local key are needed. The main key of 255 characters is integrated in the Blaise setup, the local key is stored in the settings file of the interviewer.

The interviewer settings file contains the interviewer number (for verification), the local key to access the configuration file and the 64 characters ZIP key to access the AES 256 bits protected zip files used for the interchange of surveys and data with the other parties. It is also a Blaise file with a code/value structure of which the value (a string) is encrypted with the same special developed encryption method as used for the configuration file. To decrypt this value a main key and a local key are needed. The main key (the same as used for the configuration file) is integrated in the Blaise setup; the local key must be typed in by the interviewer in combination with the interviewer number at startup of CCMS. If the CCMS is not able to decrypt the interviewer settings file the access to the CCMS is denied.

As well the local key as the zip key of an interviewer can be changed at the office side and sent to the interviewer within CCMS. The system takes into account that some time may be needed before a new key is operational on the interviewers (or supervisors) side.

5 Open architecture

5.1 The software

As CCMS should be applicable for different organizations the kind of software involved plays an important role. It must be clear that Blaise has the preference. E.g. instead of using batch files it is possible to create a text file that is used by a Maniplus setup to invoke all kind of commands. In case there is no direct or indirect Blaise solution for hand (like in the case of sFTP or the installation program) CCMS is using freeware and/or open source solutions to perform the task. This makes CCMS accessible to more organizations. Of course CCMS should also be able to support licensed communication software. To achieve that goal, the interface with the data communication software is placed in a separate Maniplus setup containing procedures with a standardized call. It can easily be replaced by any other implementation.

5.2 Languages

CCMS supports different languages to be able to be used by different organizations. It concerns the language of the user interface of the software, as well at the interviewer side as at the office side. The surveys have to be designed supporting their own language(s). To be able to support different languages CCMS works with external language files.

On the interviewer side CCMS is multilingual. Not only can an existing language file be replaced by another language file, it is also possible to add more language files. This is useful in countries with more than one official language or in the case interviewers with a different language are hired to address a special minority. In the current version 2 languages are included: Dutch and English. The language can be changed within the user interface by the interviewer.

On the office side only one language file is supported at a time. It can easily be replaced by another language file.

5.3 The surveys

To maintain an open architecture while supporting all kind of different surveys (using different Blaise data models) is a challenge on its own. The (primary) key may consist of one or more fields varying from survey to survey. Therefore the key information of a form of an interview is stored in a general identification field in the respondent information of CCMS. This identification field has been defined as one single string which holds the values of the different key fields, separated by a semi-colon (;). When starting up an interview this key is automatically applied by the CCMS.

Another challenge is formed by the administration forms in which an interviewer can report about the interview itself. Some organizations include this form as a parallel in the questionnaire itself and some define it as a separate data model. In CCMS it is possible to indicate that the administration form is not included and one can specify which separate form should be used.

Support of an audit trail can be added to the surveys by choice. It is possible to activate the audit trail for all surveys, a selection of surveys; for all interviewers or only a subset of the interviewers. It is even possible to activate the audit trail for only a part of the forms (respondents) of a survey.

There are still some wishes to be fulfilled. In some organizations a main survey form is used for all respondents in the sample. Based upon the answers automatically a follow-up task should be generated for another survey. And for sure there are more wishes.

6 Technology applied

6.1 Data package & objects

The data is interchanged between the parties using data packages. These data packages (extension .cmp) are encrypted zip files. They can only be unpacked by the specific interviewer using the 64 character long zip key unique for each interviewer. The office has all the keys and the supervisor those of the interviewers within the supervisor's group.

Each data package contains one or more so-called data objects (extension .cmo). These objects only contain the data (properties and “payload”) and not the methods. When starting the development of CCMS the choice has been made to use separate Manipula programs which contain procedures for these methods, as procedures enable the use of parameters for import, transit and export. The procedures are used as alien procedures from other Manipula programs of CCMS, and will hereinafter referred to as Manipula libraries or Alien libraries. Older Blaise versions expected these libraries to be stored in the same folder of the application were the other Manipula programs were stored. Due to that restriction the methods were not added to the objects. The latest version of Blaise now supports the storage of alien procedures in other folders. However, during development of CCMS this separate storage of methods proved to be very developer friendly so the construction was not changed.

A data object (in reality a zip file) contains object properties and survey data. The properties are used by the methods (procedures) defined in the Manipula libraries to interpret what should be done with the data (payload). There are different types of data objects. Each type has a different role in the system. The type is included in the file name of the object:

<type>{<code>}<date>T<time>.cmo

<code> = see “Object types”

<date> = yyyyymmdd

<time> = hhmmss

Sample: QNR{Commute1}20110602T150600.cmo

The following types of the objects have been defined:

LNG - Languages

NTF – Notification

<code> = function (logging, messages, received, zipkey, etc.)

QNR - Questionnaire

<code> = survey code

RPD - Respondents

<code> = survey code

SWU - Software update

At this moment only the NTF, QNR and RPD are implemented in CCMS. The properties of the data object are stored as an xml file (_object.xml). The other files are stored in a zip file in the data object (data.zip).

6.2 Alien libraries

In CCMS procedure libraries (prepared Manipula setups) are used for the processes around the data objects. These libraries have the advantage that it is possible to use parameters for input, transit and output when calling the procedures stored inside the library. In some cases these alien procedures are invoking procedures of other alien libraries. The procedures (and functions) in such library can read from and/or write to files including temporary files. The alien libraries are linked to a certain object type and the same library will be used in the different environments (office, supervisor and interviewer). Furthermore a common library was added to support all kind of actions regarding the logbook, so this Manipula library is not connected to a specific data object type.

A disadvantage of using these libraries is that if a runtime error occurs when executing an alien procedure, it is not reported to the calling program: it just stops processing the commands. This makes debugging difficult.

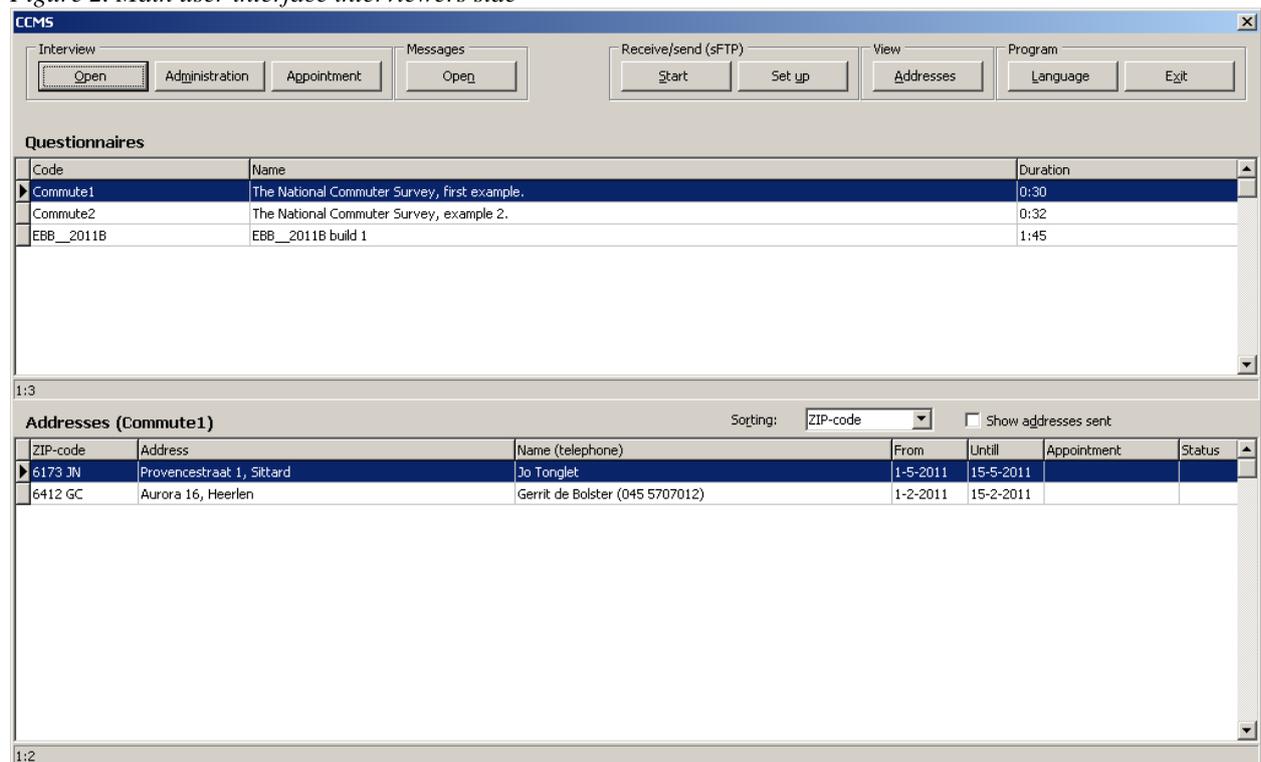
During development it appeared that some behavior of Manipula did not meet our needs. The actual Blaise version at that moment did not formally support that the alien procedures were stored in a different folder than the calling program. It was also impossible to use such alien if its path contained a space. For that reason it was decided to store the Manipula libraries in the folder with the other Manipula programs of CCMS and not to include them in the objects. As it appeared during further

development this decision made it easier to cope with (unavoidable) changes as test objects and data packages did not need to be recreated.

6.3 User interface

The user interface at the office side comes with a classical Windows menu. For the interviewers another approach has been chosen. The main user interface consists of a stand-alone dialog window, which functions as a dashboard. The size of this dialog is based upon the screen size of a netbook. A small VBScript is applied to place this dialog in the middle of the screen whatever it size may be.

Figure 2. Main user interface interviewers side



The layout of the dialog at the interviewer side is different too (see figure 2). Similar to modern interfaces the action buttons are placed on the top of the window, more or less like a ribbon. Below these buttons the lookups and other controls are placed. The main interface shows a split screen with the active surveys and the related addresses of respondents with status information. Changing the focus on a survey will change the set of addresses automatically to the one related to the focused survey. These addresses can be sorted in different order according to a choice in a dropdown list. By a click on an action button the split screen is replaced by a single list of all addresses. The related survey is now shown in a column of the list.

For the message function a separate sub dialog can be activated by a button in the ribbon, again with the buttons on the top of the screen. Below the buttons a list of the messages is shown. The interviewer can choose between received and sent messages.

6.4 Synchronization processes

This is a global description of scenario “A”. Actions e.g. concerning the logbook are left out.

Naming conventions: An interviewer is receiving (downloading) from the office and sending (uploading) to the office. Therefore the folders are called “Off2Int” and “Int2Off”, as well on the interviewers side, on the (s)FTP server and key/card and on the office side. The office is receiving the data packages uploaded by the interviewers from the “Int2Off” folder and sending data packages to be downloaded by the interviewers to the “Off2Int” folder.

Figure 3a. Data communication using a (s)FTP server

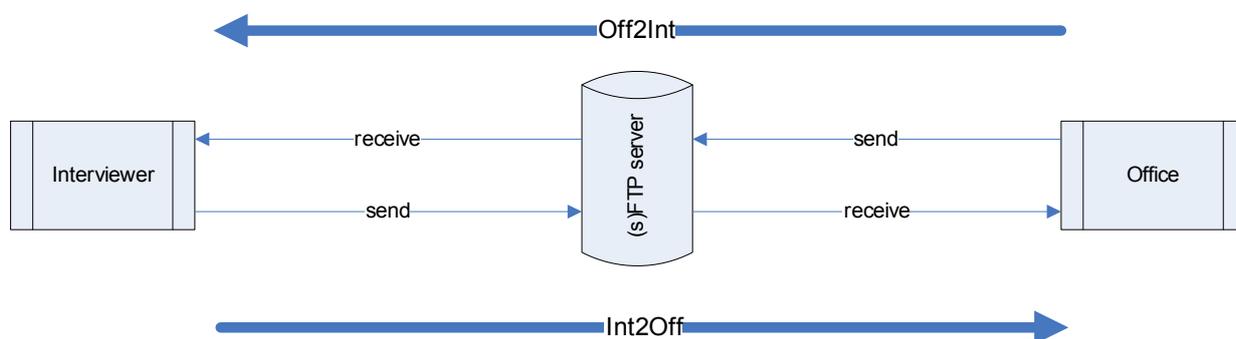
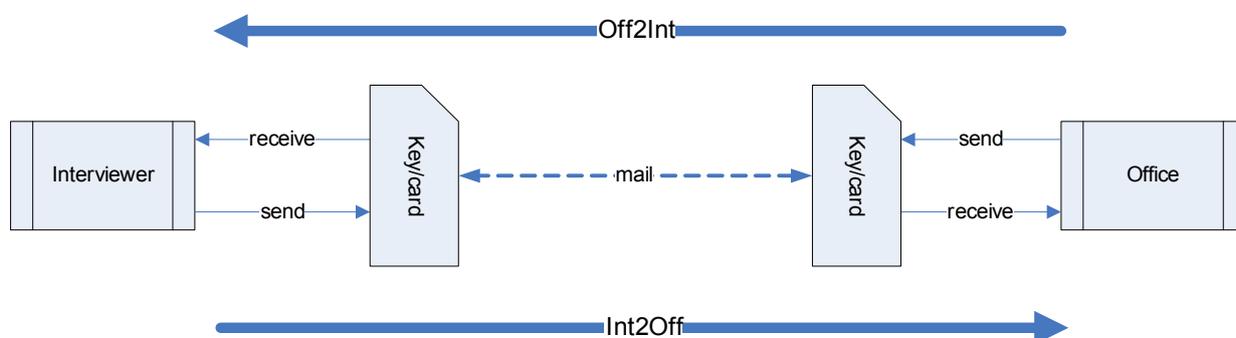


Figure 3b. Data communication using a Key/card



6.4.1 Synchronization process at the interviewer side

See appendix 1.

When the interviewer starts the synchronization process the data packages intended for this interviewer are copied or downloaded (depending on the setting: key/card or sFTP in the current CCMS) to the “Off2Int” folder (process 1[1/2]). These packages can be recognized as the file name contains the identification of the interviewer. The data packages are unpacked (process 2a) using the zip key from the interviewer and the resulting data objects are stored in the receive folder. If successful the data package is moved to the archive folder (process 2b). The data objects are then processed using the init method associated to this type of data object (process 3). A confirmation data object for the received data objects is created in the send folder as well as data objects with completed statements and messages. The data objects are packed in a data package and stored in the “Int2Off” folder (process 4). The data packages stored in this “Int2Off” folder are copied or uploaded (depending on the setting: key/card or sFTP in the current CCMS) to the key/card or remote upload folder (process 5[1/2]). If successful the data packages are moved to the archive folder (process 6).

6.4.2 Synchronization process at the office side

See appendix 2.

The data packages uploaded by the interviewers to the sFTP server are downloaded (or copied) to the “Int2Off” folder (process 1a). Data packages received on key/card are imported to the office system and copied to the same “Int2Off” folder (process 1b). The key/card contains an “Int2Off” and a “Off2Int” folder and only the data packages from the “Int2Off” folder are imported. It is therefore not necessary for an interviewer to clean the key/card received from the office before using it to send data packages to the office as any data packages sent by the office are stored in the “Off2Int” folder. De

data packages in the “Int2Off” folder are unpacked using the zip keys of the interviewers and the data objects are stored in the interviewers’ receive folder (process 2a). If successful the data packages are removed from the FTP server (process 2b). The received data objects are processed using the init method associated to this type of data object (process 3). Resulting data is placed in a shared folder to serve as input for the internal statistical processes. A confirmation data object containing the received data objects is created in the send folder as well as data objects with new surveys, respondent information and messages (process 4). The data objects are packed in a data package and stored in the “Off2Int” folder (process 5). The data packages are uploaded to the download folder on the FTP server (process 6a) and copied to the “Off2Int” folder on the formatted (process 8) key/cards (process 6b). The download folder on the FTP server is synchronized with the “Off2Int” folder removing data packages that already have been successfully download by the interviewers (process 7).

When keycards are used instead the contents of that card resembles the contents of the folders of the sFTP-server: all the data packages for all the interviewers are copied to the “Off2Int” folder of all formatted key/cards; hence this replication process is simple. So after the replication all the key/cards are identical with causes the packing in envelopes and sending the key/cards to the interviewers to be foolproof. An interviewer can only open its own data packages as they are encrypted with the unique zip key of the interviewer. The same applies to the download folder on the sFTP server.

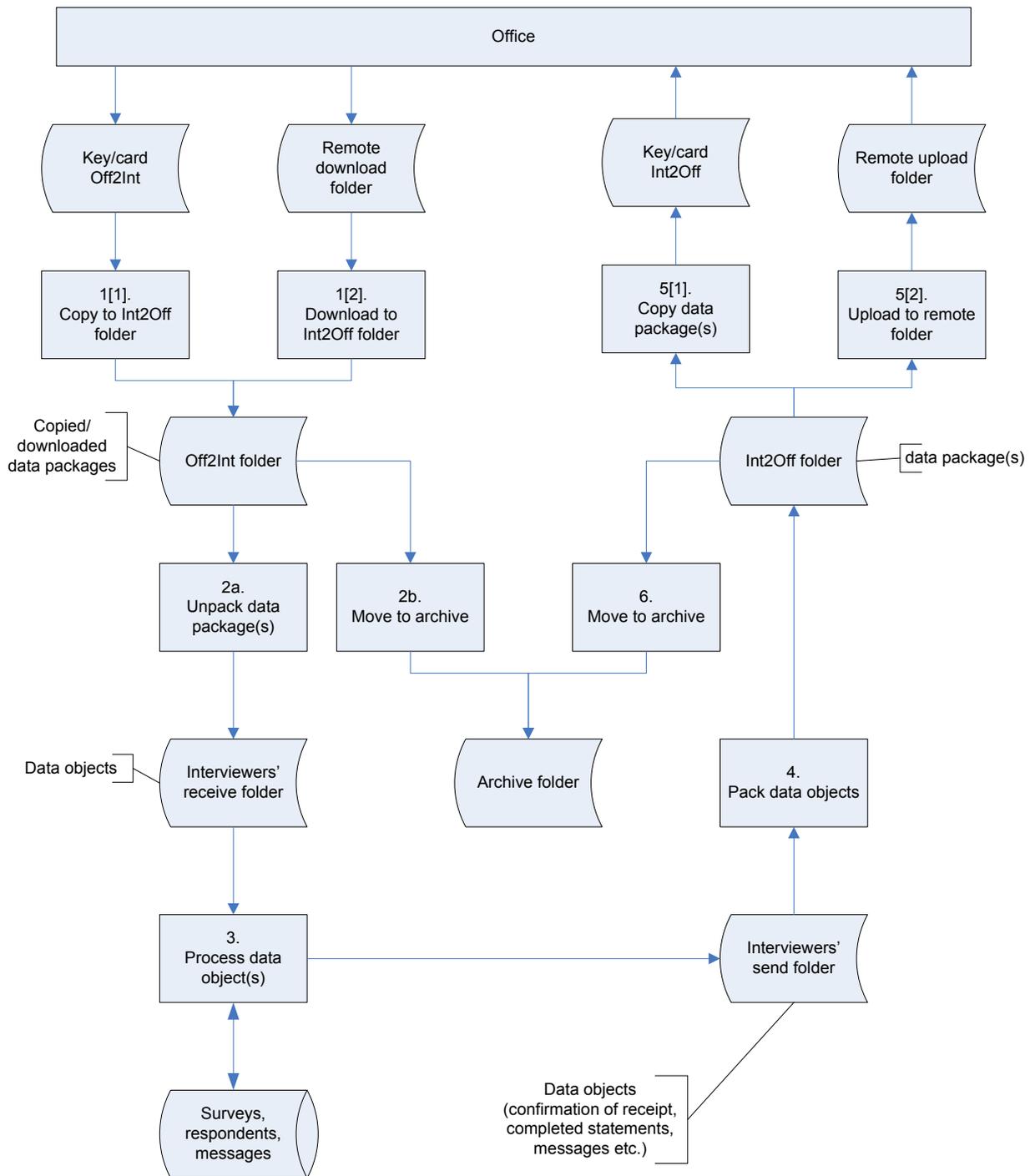
7 Summary

Within the defined scope the management system described in this paper is a complete solution for CAPI Case Management at very low costs. A working version is available for free. It includes both the applications as described in scenario A: for the laptop and for the office. It is based on Blaise in combination with freeware. The interchange of information between the office and the interviewer (and vice versa) is robust. The first version supports 2 different transportation media: sFTP and memory keys or cards by classic mail (or a combination of both). Even for the sFTP server a freeware solution is available. It is very flexible concerning the languages applied although the first version only supports the “left to right” languages.

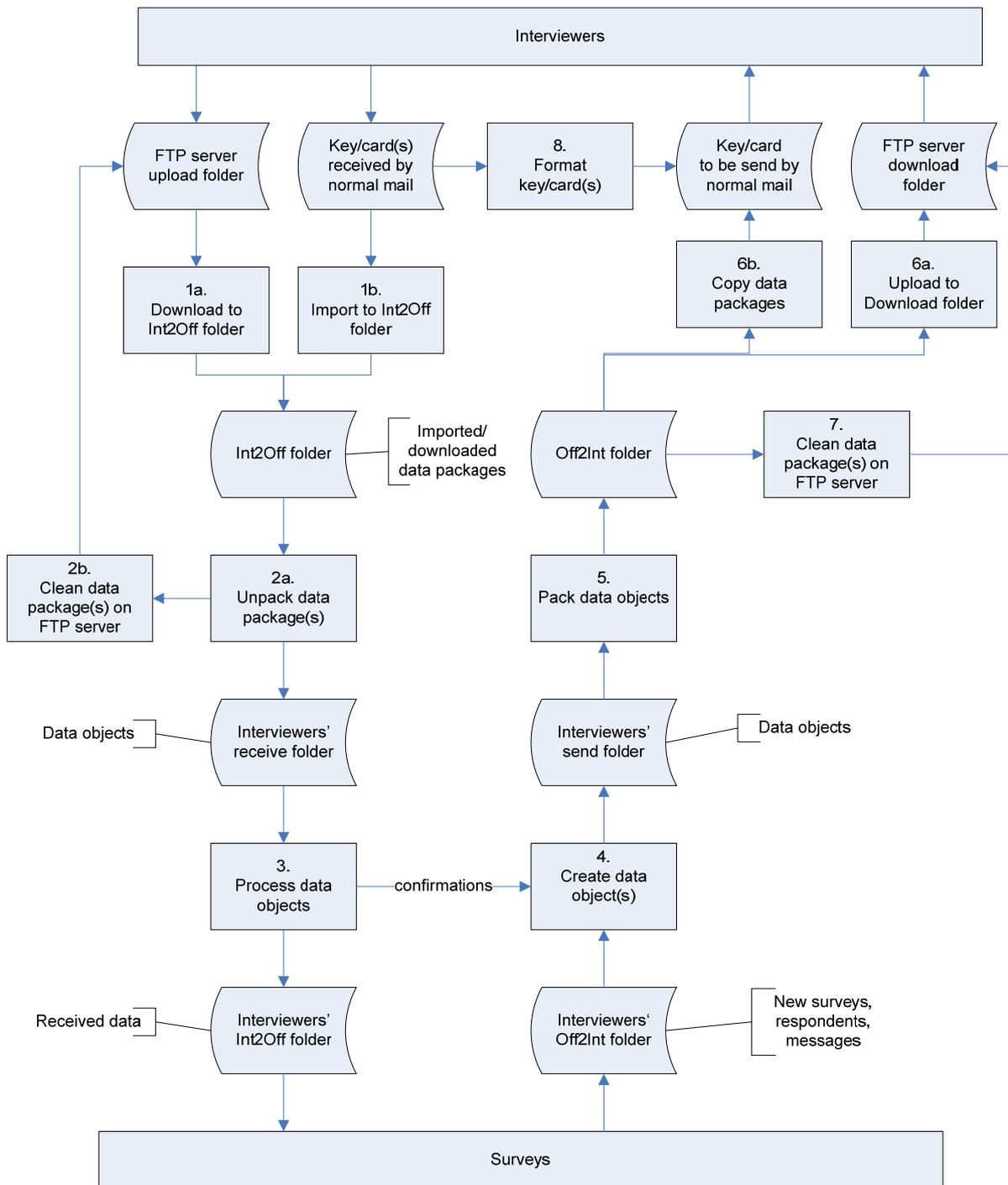
The development is set up as open software. Every interested person is invited to join the “CCMS community”. Not only to use it for free but also (if possible) to contribute. The software (including the sources and documentation as far as it is available) is distributed using DropBox (www.dropbox.com). Contact the authors of the paper for more information.

Appendixes

1. Synchronization process at the interviewer side



2. Synchronization process at the office side



3. System requirements

The same system requirements that apply to the Blaise software are applicable for using CCMS. CCMS does support the management of questionnaires created with older versions of Blaise like 4.6. However, for executing CCMS on the laptop/netbook of the interviewer the Manipula program (Manipula.exe) of Blaise 4.8.3.1735 (or higher) is required. CCMS in the office will also work with a Blaise 4.8.2 version, e.g. build 1700. At the office side in addition to the Manipula program CCMS also uses the Blaise dataviewer (dataview.exe). From the licensing point of view Blaise licenses are required as described on the website of Statistics Netherlands. Hence Capi end-user licenses will be required for the interviewer laptops/netbooks. Creating questionnaires or making adaptations to CCMS can only be done when the organization possesses a developer license as well.

For the data communication (client and server) and even the installer there is enough suitable freeware available on the Internet. In the first version some samples are included.

CCMS (interviewer and office side) will run on Windows XP as well as on Vista and Windows 7 (32 and 64 bits).

Implementing Audio Computer Assisted Self Interviewing in Basil

Thomas Spaulding, U.S. Census Bureau

1 Introduction

The National Health Interview Survey (NHIS), which is sponsored by the National Center for Health Statistics (NCHS), is the main source of information on the health of the non-institutionalized population in the United States. The survey provides information about the prevalence and distribution of illness, its effects in terms of disabilities and chronic impairments, and the kind of health services people receive.

As early as 2004, the NCHS had talked to the U. S. Census Bureau about the possibility of using Audio Computer Assisted Self Interviewing (ACASI) technology to ask questions about sexual orientation in the NHIS instrument. The NCHS was aware of the sensitive nature of the questions and wanted to use ACASI to create a private environment for the respondent to answer the questions.

In ACASI, the interviewer does not have to read questions aloud to the respondent nor does ACASI require the respondent to provide a verbal answer to the interviewer. An ACASI environment provides the respondent with a device (i.e., some sort of computer) with which to complete a questionnaire. The device displays questions and answer lists on a screen while the respondent listens to audio recordings of the same questions and answer lists through headphones. The respondent answers each question by pressing the appropriate computer key (or touching the screen appropriately if the device has touch screen capability). This approach would create the type of private environment that the NCHS wanted for its respondents.

In September 2010, the Question Design Research Laboratory (QDRL) at the NCHS began discussions with the Census Bureau about conducting an ACASI pilot project. The QDRL's initial plan was to have the Census Bureau's Field Representative (FR) who conducts the NHIS interview provide the respondent with a small touch screen device (e.g., iPad, iPod, or smart phone) on which the ACASI portion of the interview would be administered. The device would present just the question and response options on the screen, and respondents would only need to touch the screen to select the desired response. The QDRL had initially envisioned an ACASI questionnaire that was a multimedia application using sound, images, and video.

Unfortunately, the time frame for developing and deploying the pilot project precluded the use of such a device. There was not enough time to go through the procurement and security processes required to place a new device in the field. As a result, the pilot project would have to run on the existing FR laptops. Considering this constraint, the authors at the U. S. Census Bureau recommended to the NCHS that the pilot project use the ACASI functionality available in Blaise. After some investigation, the NCHS survey methodologists decided that the Blaise user interface was more complicated than what they wanted to use, and it could not be customized sufficiently to meet their needs. The NCHS methodologists wanted a user interface that would be simple in appearance and easy for respondents with limited technical knowledge to use when answering the questions in the ACASI module. Following discussions with Westat and Statistics Netherlands, the NCHS researchers decided to develop the ACASI module in Basil which is better suited for self interviewing situations.

This paper discusses some of the technical, methodological, and organizational challenges encountered while the NCHS and the U. S. Census Bureau worked on the ACASI module. It also discusses some of the advantages and disadvantages of using Basil for a project of this nature and some experiences from the first field test.

2 ACASI Project Requirements

The NHIS ACASI project was a new experience for the Census Bureau in a couple of ways. This was the Census Bureau's first attempt to field an ACASI application, and our first significant experience with Basil. In the past the Authoring Staff at the Census Bureau had researched ACASI technologies, but no inquiry from any of our customers had ever reached the point where we actually had to develop and field an ACASI application. This project was also the first time that the sponsoring agency wanted to develop a prototype of the application on its own and then turn the application over to the Census Bureau to complete. The NCHS survey methodologists and the NCHS programmers in the QDRL wanted to work together closely during the initial stages of the project so they could experiment with the ACASI technology and rapidly develop and test various methods of asking sensitive questions. Ultimately, the QDRL developed a "proof of concept" prototype of the ACASI module. The QDRL turned that prototype over to the Census Bureau's Authoring Staff, and it was the Census Bureau's task to complete the prototype and integrate it into the NHIS instrument so it would be ready for a field test.

2.1 The ACASI Prototype

While working on the prototype of the ACASI module, the QDRL's major concern focused on usability. The QDRL wanted to keep the user interface as simple as possible for the respondent. As a result, the QDRL decided to keep the number of keys the respondent had to use to a minimum. For most questions this meant the respondent only needed to know how to use three keys. The exceptions were questions that asked the respondent to key in a number or a string of text. There were only a couple of questions like this in the ACASI module, but they required the use of the letter and number keys on the laptop's keyboard.

The three main keys used in NCHS's ACASI prototype were the Enter key, the Space Bar, and the Caps Lock key. (The Caps Lock key was later replaced by the Tab key in the version of the module used in the field test.) All three keys were covered by color coded stickers to make them readily recognizable. The Enter key became the Green button, the Space Bar became the Red Circle key, and the Caps Lock key became the Orange button. (When the Caps Lock key was replaced by the Tab key, its color also changed from orange to blue.)

The Green button (i.e., the Enter key) saved an answer and moved to the next question, the Orange button (i.e., the Caps Lock key) backed up one screen in the instrument, and the Red Circle key (i.e., the Space Bar) allowed the respondent to scroll through the answer list and return to the question (if needed). Figure 1 is a screen shot from an early version of the NCHS's prototype. It is a depiction of one of the instructional screens used in that particular version of the ACASI module. This screen did not make it into the final version of the prototype, but it does show what the laptop keyboard looked like with the color coded keys.

Figure 2 is from the same prototype. (This particular question also failed to make it into NCHS's final prototype.) Figure 2 shows how the respondent can use the Red Circle key (i.e., the Space Bar) to scroll through the answer list to select a response. When the respondent first lands on the screen, the focus falls on the question text, and the ellipse is not displayed. Pressing the Red Circle key moves the focus to the first answer in the answer list which is then encircled by the red ellipse. Each time the Red Circle key is pressed, the focus moves to the next answer in the list until the last answer option is reached. Pressing the Red Circle key while on the last answer option moves the focus back to the question text.

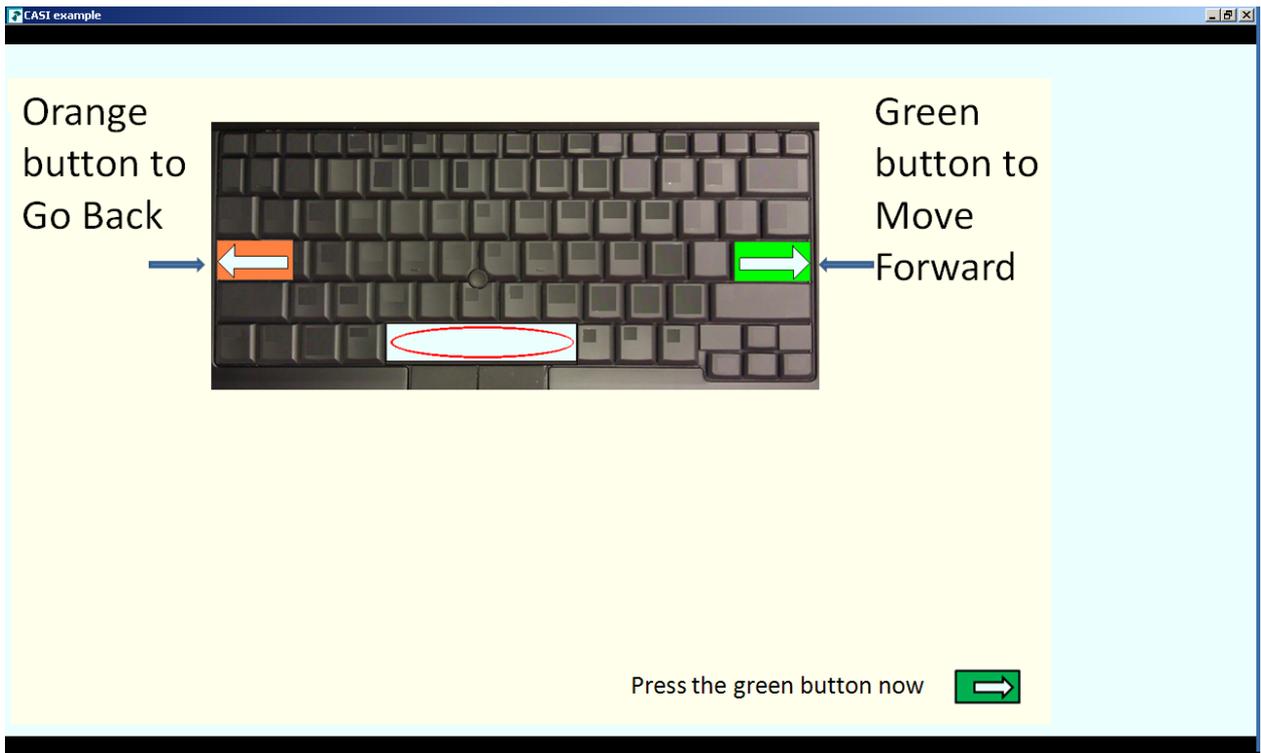


Figure 1- Instructional Screen from the Prototype Showing Color Coded Keys on Keyboard

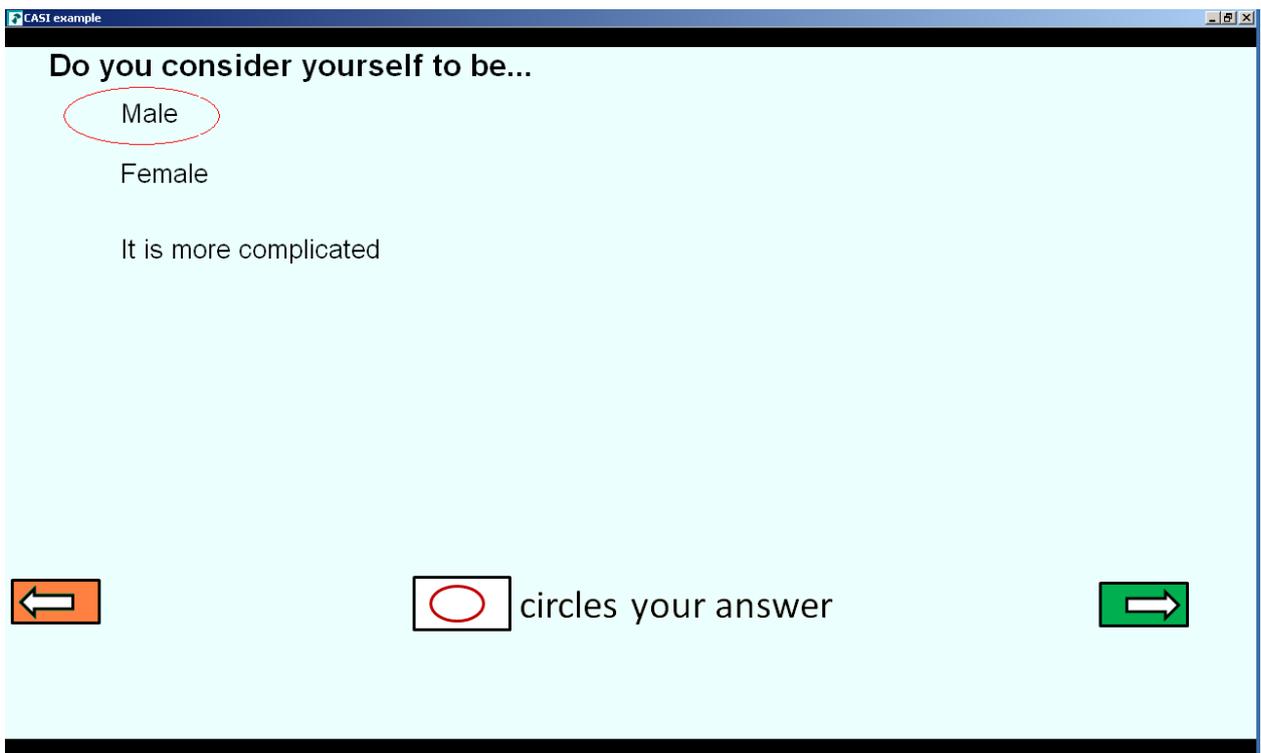


Figure 2 - Selecting a Response in the Prototype by Pressing the Red Circle Key

2.2 High Level ACASI Requirements

The NCHS provided the Census Bureau with its final ACASI prototype in September 2011, and this application served as the basis for the requirements that the Census Bureau would use to complete the ACASI module for the field test.

The subject matter specialists in the Census Bureau's Demographic Surveys Division (DSD) documented the following high level requirements.

The specialized, color coded keys to use in the field test instrument would be:

- Green Arrow key (i.e., Enter key) = Record an answer and go to the next question
- Red Circle key (i.e., Space Bar) = Scroll through answer categories and question text
- Blue Arrow key (i.e., Tab key) = Back up one screen

Other global instrument functionality included:

- To hear the audio for a question repeated, the respondent will scroll through the answer categories until the focus returns to the question.
- The instrument should allow a blank answer at each question. The respondent does not have to select an answer category in order to go forward to the next screen (i.e., allow Empty for each screen).
- For the first field test, once the respondent completes the ACASI section, neither the respondent nor the FR will be able to back into the ACASI section. There will be a screen at the end of the ACASI module that tells the respondents that they will not be able to go back and change their answers.

The NHIS Blaise instrument must launch the ACASI module.

Data collected in the ACASI module must be stored in the NHIS Blaise instrument's database.

- Audio recordings will be in .mp3 format. The audio files will be recorded using a human voice. (The NCHS prototype had used text-to-speech technology.)

2.3 Requirements for Field Testing the ACASI Module

The NCHS requested that the Census Bureau conduct three field tests of the ACASI module integrated with the NHIS instrument.

The Census Bureau conducted the first field test in November 2011. This was a small scale test that used only 6 FRs to administer the questionnaire. The goal of the test was to collect 50 completed cases. This test focused on the programming of the ACASI portion of the questionnaire and on the procedures that the field staff would use to administer the ACASI portion of the interview. Some production procedures were not employed to expedite the test. For example, the FRs only had to collect data for one adult, and it was not necessary for them to collect data for the entire household. The test also provided an opportunity to gauge initial reactions to the instrument from both the FRs and the respondents. Some experiences from this test are discussed later in this paper.

The second test that the NCHS requested requires a sample of 1,100 cases and will employ about 100 FRs. Its goal is to collect 500 completed cases. The Census Bureau will conduct this test in April 2012, and the test will utilize all the field procedures used in production for the NHIS. The NCHS hopes that this test will provided a realistic reflection of the field effort required to obtain completed ACASI interviews and of respondent acceptance of ACASI.

The third test is scheduled for July 2012. At this time, it is not known how many FRs will be participating in this test, but the goal is to collect 5,000 completed cases. There will be a split sample of 11,000 cases. About 60% of the sample will get the ACASI treatment. The other 40% will use flash cases to respond to the sensitive questions. The NCHS will use the results from this test to determine whether to implement the ACASI questionnaire for production data collection.

3 The ACASI Module for the 50 Case Test

The ACASI module for the 50 case test included instructional screens, sexual orientation questions, financial concern questions, questions on sleep, mental health questions, and alcohol use questions. This section of the paper describes the basic organization of the Basal datamodel, the type of questions

administered in the module, and the basic approach the authors used to implement a question in Basil.

3.1 Structure of the ACASI Module

The ACASI module starts with the required Basil application section. This section differentiates a Basil application from a Blaise instrument and is used to specify global information about the layout and behavior of the instrument. For example, the NHIS ACASI module's application section specifies that the application uses the entire screen (and that it is not divided into panels). The application section also specifies what actions should take place when the Basil questionnaire starts and when it closes, it identifies the location of the Maniplus procedures that are used by the Basil application, and can associate an event with specific key.

An AUXFIELDS section follows the application section. It defines all the Basil question sections that the ACASI module uses. A Basil question section contains the information required for displaying one question. In the NHIS ACASI module, the question sections are used to represent both the questions and the answer choices. Each question and each response option in a question's answer list are separate AUXFIELDS, and each AUXFIELD's Field description contains its own unique Basil question section.

The AUXFIELDS section is then followed by the FIELDS section that defines the actual variables into which a respondent's answers will be stored. The data in these Fields will later be copied to the Blaise instrument's database. A RULES section provides the logic for how the Fields are to be processed in the ACASI module. (As will be noted later, the ACASI module makes considerable use of Maniplus procedures to perform event handling that assists in navigation, range checking, and data storage during the ACASI interview.) Finally, the LAYOUT section specifies which of the AUXFIELDS containing Basil question sections appear together on the same page.

3.2 Types of Questions in the ACASI Module

The ACASI module uses only three types of "questions." The first type of question is actually an instruction. These instructional items provide the user with information about using the ACASI questionnaire. Most of these appear at the beginning of the questionnaire. There are also questions with an answer list (many of which include 'Don't know' and 'Refusal' options), and there are a few questions that ask the respondent to enter either a numeric or text response into an input box. Examples of all three types of questions appear in the following three screen shots.

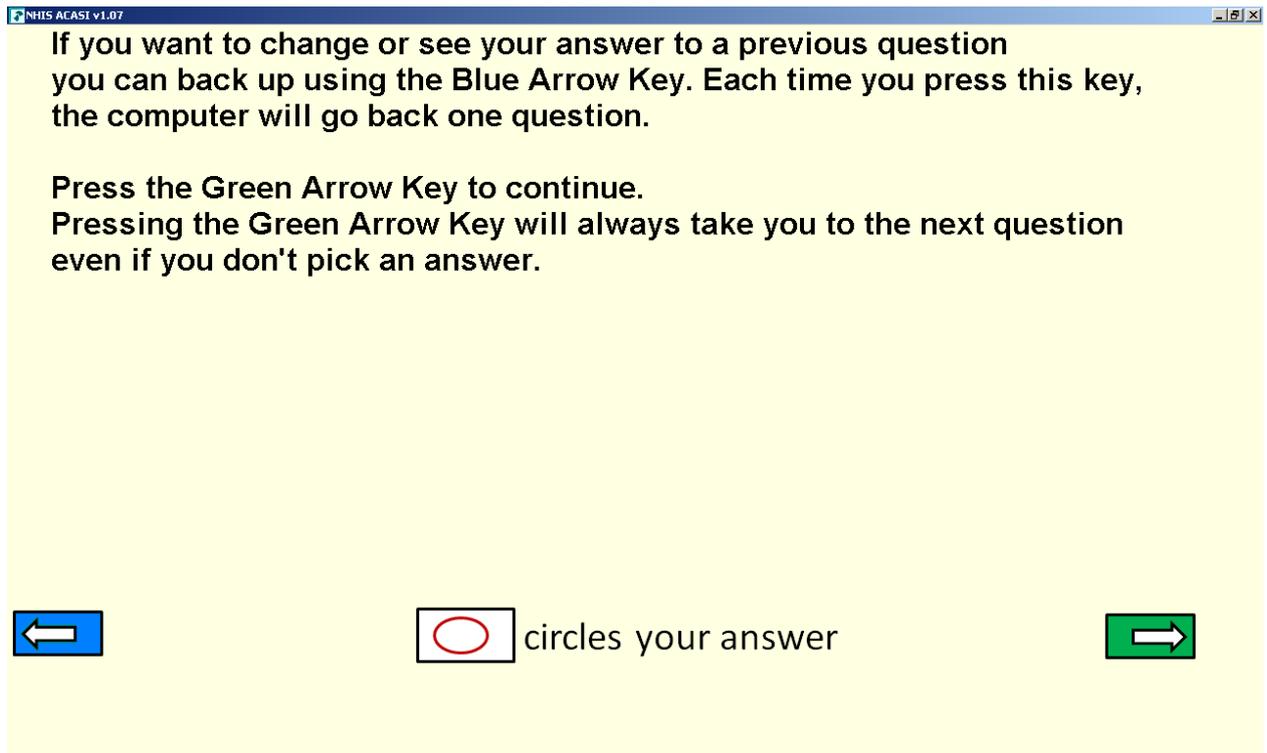


Figure 3 - An Instructional Screen

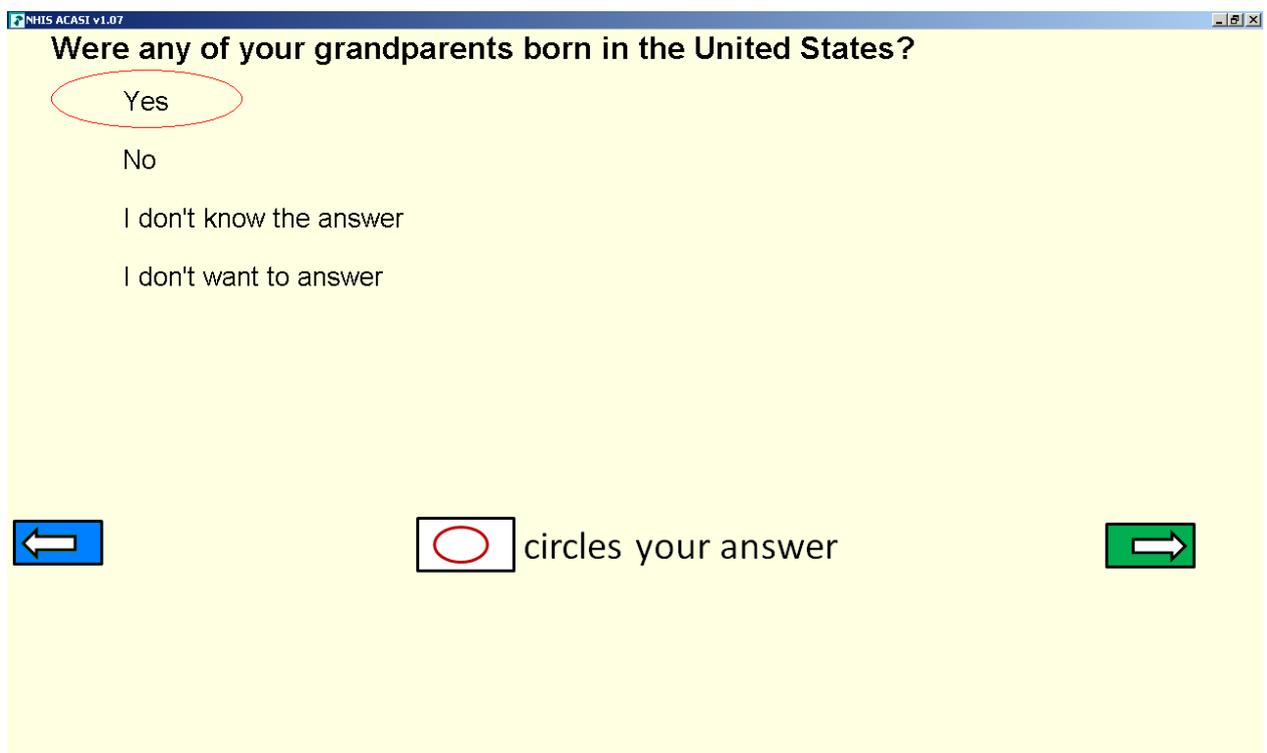


Figure 4 - A Question with an Answer List

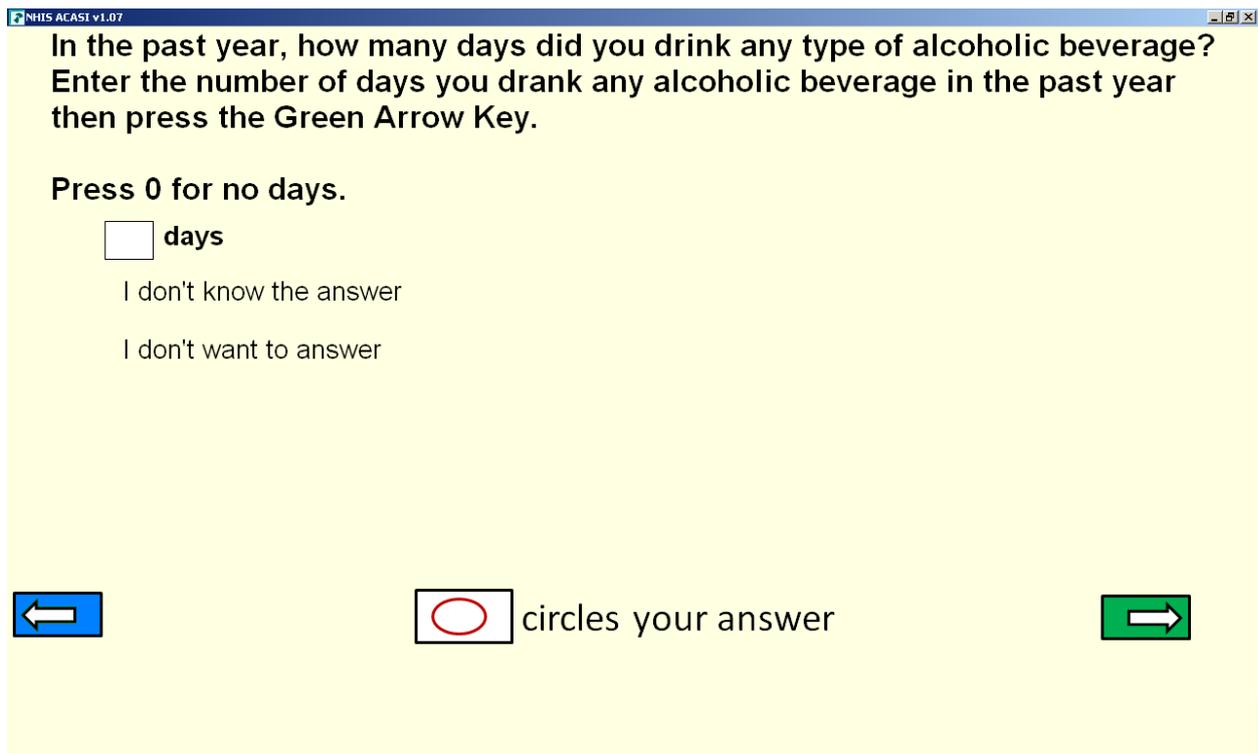


Figure 5 - A Question with an Input Box

3.3 Implementing Questions in the ACASI Module

As stated above, each question and each response option in a question's answer list are defined as separate Fields in the AUXFIELD section of the ACASI module, and their Field descriptions contain Basil question sections that provide the information needed to display the question or the response option.

The authors implemented every question in the NHIS ACASI module using basically the same technique, and they used slight variations on this approach to achieve the desired results for each question. This section of the paper reviews the general technique the authors used to implement a question. It uses the ACIALCDS alcohol use question (shown in Figure 5) as an example.

3.3.1 Basil Question Section for the ACIALCDS Alcohol Use Question

The Basil question section for the ACIALCDS alcohol use question appears below in Figure 6.

```

ACIALCDS_Question
  "<question>
    <audio name=acialcds_q src='ACIALCDS_Q.1.mp3'>
    <label left=45 width=1200 text='<font size=24><B>In the past year, how many days did you
    drink any type of alcoholic beverage?<br>Enter the number of days you drank any alcoholic
    beverage in the past year then press the Green Arrow Key.<br><br>Press 0 for no days.'>
    <input left=100 top=200 width=50 height=40 columns=1
    onenter='blaise:playmedia(acialcds_q)'
    onexit='Blaise:stopmedia(acialcds_q)'
    onreturn='runtime_ACASI_procedures.ACIALCDS_RangeCheck;blaise:save();runtime_ACASI_procedures.ACIALCDS_Check;blaise:nextpage()'
    onkey_1='blaise:gotofield(ACIALCDS_DontKnow)'
    onkey_2='blaise:save();runtime_ACASI_procedures.Special_backup_8'>
    <label left=160 top=200 width=100 text='<font size=20><B>days'>
  </question>" : INTEGER[3], EMPTY
  
```

Figure 6 - Question Section for Alcohol Use Question

ACIALCDS_Question is defined as an Integer Field with a length of three, and like all questions in the ACASI module it allows Empty.

The audio element in the ACIALCDS_Question question section specifies the identifier (i.e., acialcds_q) by which this question section references the audio file that is to be played. It also

associates a specific recording (i.e., ACIALCDS_Q.1.mp3) with the identifier. (Note that multiple audio files can be specified in the audio element if needed.)

The first label element that appears in this question section specifies the position of the label, the text characteristics, and question text for the alcohol use question.

The input element is a bit more complicated. It first specifies the position and size of the input field. It then lists several control attributes and specifies what actions Basil takes when these events occur.

- When entering the field (i.e., onenter control attribute), play the specified audio file.
- When leaving the field (i.e., onexit control attribute), stop playing the audio recording.
- When the Enter key is pressed (i.e., onreturn), several actions take place.
 - The ACIALCDS_RangeCheck procedure performs a range check,
 - the blaise:save() action saves the current record,
 - the ACIALCDS_Check procedure saves special values in the ACIALCDS storage Field if the focus was on either the Dontknow or Refusal fields when the respondent pressed the Enter key.
 - the blaise:nextpage() action moves the respondent to the next screen on route.
- When the Space Bar is pressed (i.e., onkey_1), the blaise:gotofield action moves the focus from the current field down to the ACIALCDS_DontKnow field which is the "I don't know the answer" option that appears just below the question text on the screen.
- When the Tab key is pressed (i.e., onkey_2), the blaise:save() action saves the current record, and the Special_backup_8 procedure determines what the previous screen was and loads it. That is, the procedure takes the respondent back to the previous screen on route.

The second label element that appears in this question section specifies the position of the label, the text characteristics, and the label 'days' for the input box.

3.3.2 Basil Question Sections for the ACIALCDS Alcohol Use Answer List

The Basil question section for the ACIALCDS Dontknow answer option appears in Figure 7.

```
ACIALCDS_DontKnow
"<question>
  <audio name=dontknow src='DONTKNOW.mp3'>
  <input left=100 width=310 height=30 top=15 columns=1
  AUTOSELECT=false SHOWSELECTCONTROL=false SHOWFOCUSRECT=false
  onenter='blaise:playmedia(dontknow) '
  onexit='Blaise:stopmedia(dontknow) '
  onreturn='blaise:save();runtime_ACASI_procedures.ACIALCDS_Check;blaise:nextpage()'
  onkey_1='blaise:gotofield(ACIALCDS_Refuse) '
  onkey_2='blaise:save();runtime_ACASI_procedures.Special_backup_8'>
  <ellipse left=45 width=460 height=60 visible=true transparent=yes FocusedBorderColor=red>
</question> " : (Dont_Know {1} "I don't know the answer")
```

Figure 7 - Question Sections for the Don't Know Answer Option

ACIALCDS_DontKnow is the first of two options in an answer list for ACIALCDS_Question.

The audio element in the ACIALCDS_DontKnow question section specifies the identifier (i.e., dontknow) by which this question section references the audio file to be played, and it associates a specific recording (i.e., DONTKNOW.mp3) with the identifier.

The input element specifies the position and size of the field. It then lists several events and specifies what actions Basil takes when the events occur.

- When entering the field (i.e., onenter control attribute), play the specified audio file.
- When leaving the field (i.e., onexit control attribute), stop playing the audio recording.
- When the Enter key is pressed (i.e., onreturn), several actions take place.

- the blaise:save() action saves the current record,
- the ACIALCDS_Check procedure saves special values in the ACIALCDS storage Field if the focus was on either the Dontknow or Refusal fields when the respondent pressed the Enter key.
- the blaise:nextpage() action moves the respondent to the next screen on route.
- When the Space Bar is pressed (i.e., onkey_1), the blaise:gotofield action moves the focus from the current field down to the ACIALCDS_Refuse field which is the "I don't want to answer" option that appears just below the "I don't know the answer" answer option on the screen.
- When the Tab key is pressed (i.e., onkey_2), the blaise:save() action saves the current record, and the Special_backup_8 procedure determines what the previous screen was and loads it taking respondent back to the previous screen on route.

The ellipse element that appears in this question section specifies the position of the ellipse and its attributes when the focus is on this Field.

Following the question section is the Field's type declaration. In this case, the ACIALCDS_DontKnow Field is an enumerated type with just one category identifier defined in its list of items. The category text "I don't know the answer" gets displayed on the page.

The Basil question section for the ACIALCDS Refusal answer option appears in Figure 8.

```

ACIALCDS_Refuse
"<question>
  <audio name=dontwant src='DONTWANT.mp3'>
  <input left=100 width=300 height=30 top=15 columns=1
  AUTOSELECT=false SHOWSELECTCONTROL=false SHOWFOCUSRECT=false
  onenter='blaise:playmedia(dontwant) '
  onexit='Blaise:stopmedia(dontwant) '
  onreturn='blaise:save();runtime_ACASI_procedures.ACIALCDS_Check;blaise:nextpage() '
  onkey_1='blaise:gotofield(ACIALCDS_Question) '
  onkey_2='blaise:save();runtime_ACASI_procedures.Special_backup_8'>
  <ellipse left=45 width=460 height=60 visible=true transparent=yes FocusedBorderColor=red>
</question> " : (Refused (2) "I don't want to answer")

```

Figure 8 - Question Section for Refusal Answer Option

ACIALCDS_Refuse is the second of the two options in the answer list for ACIALCDS_Question.

The audio element in the ACIALCDS_Refuse question section specifies the identifier (i.e., dontwant) by which this question section references the audio file to be played, and it associates a specific recording (i.e., DONTWANT.mp3) with the identifier.

The input element specifies the position and size of the field. It then lists several events and specifies what actions Basil takes when the events occur.

- When entering the field (i.e., onenter control attribute), play the specified audio file.
- When leaving the field (i.e., onexit control attribute), stop playing the audio recording.
- When the Enter key is pressed (i.e., onreturn), several actions take place.
 - the blaise:save() action saves the current record,
 - the ACIALCDS_Check procedure saves special values in the ACIALCDS storage Field if the focus was on either the Dontknow or Refusal fields when the respondent pressed the Enter key.
 - the blaise:nextpage() action moves the respondent to the next screen on route.
- When the Space Bar is pressed (i.e., onkey_1), the blaise:gotofield action moves the focus from the current field back to ACIALCDS_Question so the respondent can hear the audio for

the ACIALCDS_Question question again. This action and the playing of a different audio file are the major differences between this Field and the ACIALCDS_DontKnow Field.

- When the Tab key is pressed (i.e., onkey_2), the blaise:save() action saves the current record, the Special_backup_8 procedure determines what the previous screen was and loads it taking respondent back to the previous screen on route.

The ellipse element that appears in this question section specifies the position of the ellipse and its attributes when the focus is on this Field.

Following the question section is the Field's type declaration. The ACIALCDS_Refuse Field is an enumerated type with one category identifier in its list of items. The category text "I don't want to answer" is displayed on the page.

3.3.3 Basil Question Section for the Image on the ACIALCDS Screen

The ACIALCDS_Instructions question section in Figure 9 simply specifies that the image in the Instructions4.bmp file should be displayed at the bottom of the page.

```
ACIALCDS_Instructions
"<question>
  <label top=180 text='<img src='Instructions4.bmp' transparent=true width=100 height=100 stretch=true'>
  </question>" : STRING[1], EMPTY
```

Figure 9 - Question Section for Image

4 Integrating the ACASI Module with the NHIS Instrument

The NHIS instrument consists of four main components. These are the Demographic, Family, Sample Child, and Sample Adult questionnaires. The ACASI module was added to the end of the Sample Adult questionnaire. This section of the paper briefly describes how the authors integrated the ACASI module into the instrument.

4.1 Modifications to the Blaise Instrument

To add the ACASI module to the NHIS Blaise instrument, the authors created a new block (named ACI) and included it in the Sample Adult questionnaire.

The ACI block defines Fields that correspond to the data Fields collected in the ACASI module. The data collected in the ACASI module is passed back to the Blaise instrument and stored in these Fields. As a result, the data captured in the ACASI module is saved in the NHIS instrument's Blaise database, and all the data collected during an NHIS interview is available in one place for the sponsor.

The ACI block also defines two introductory screens that the NHIS instrument displays when it is time for the FR to transfer the laptop to the respondent for the administration of the ACASI module. The first screen prompts the FR to describe ACASI to the respondent, and the second screen provides FR instructions for transitioning the laptop to the respondent. These screens appear below in Figures 10 and 11 respectively.

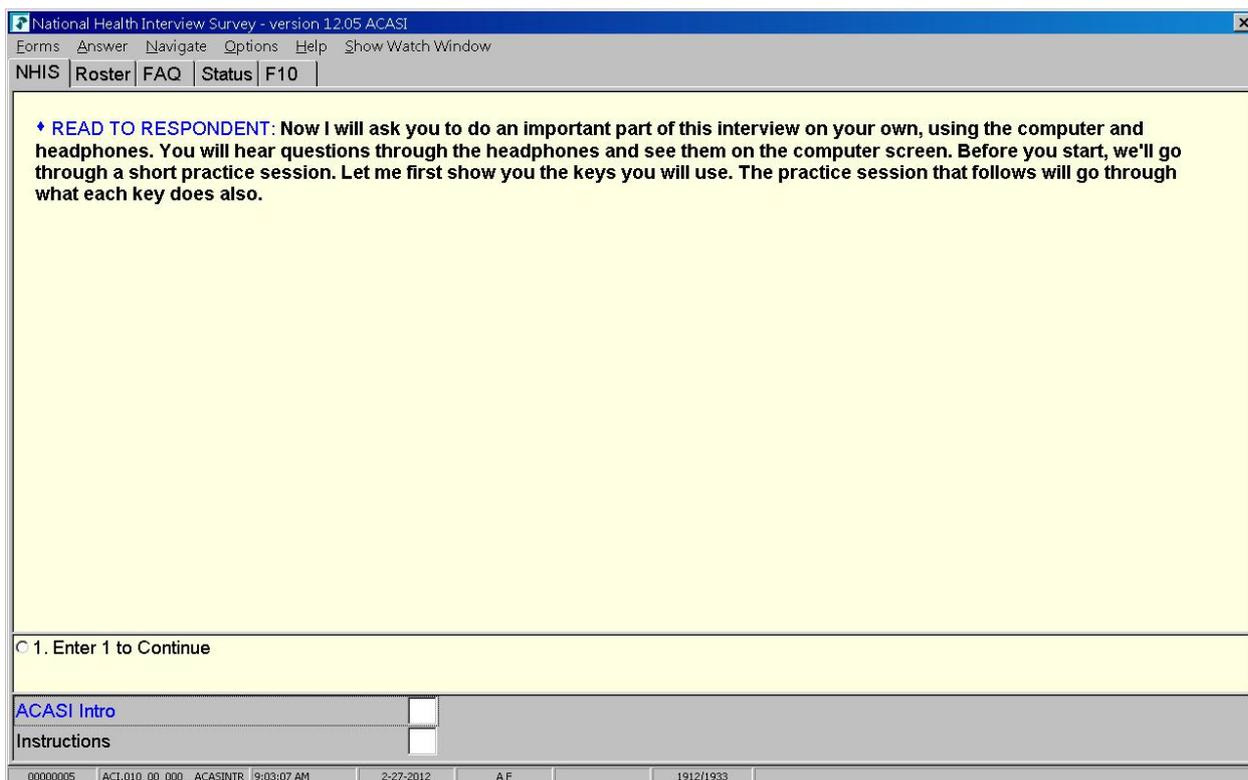


Figure 10 - Explaining ACASI to the Respondent

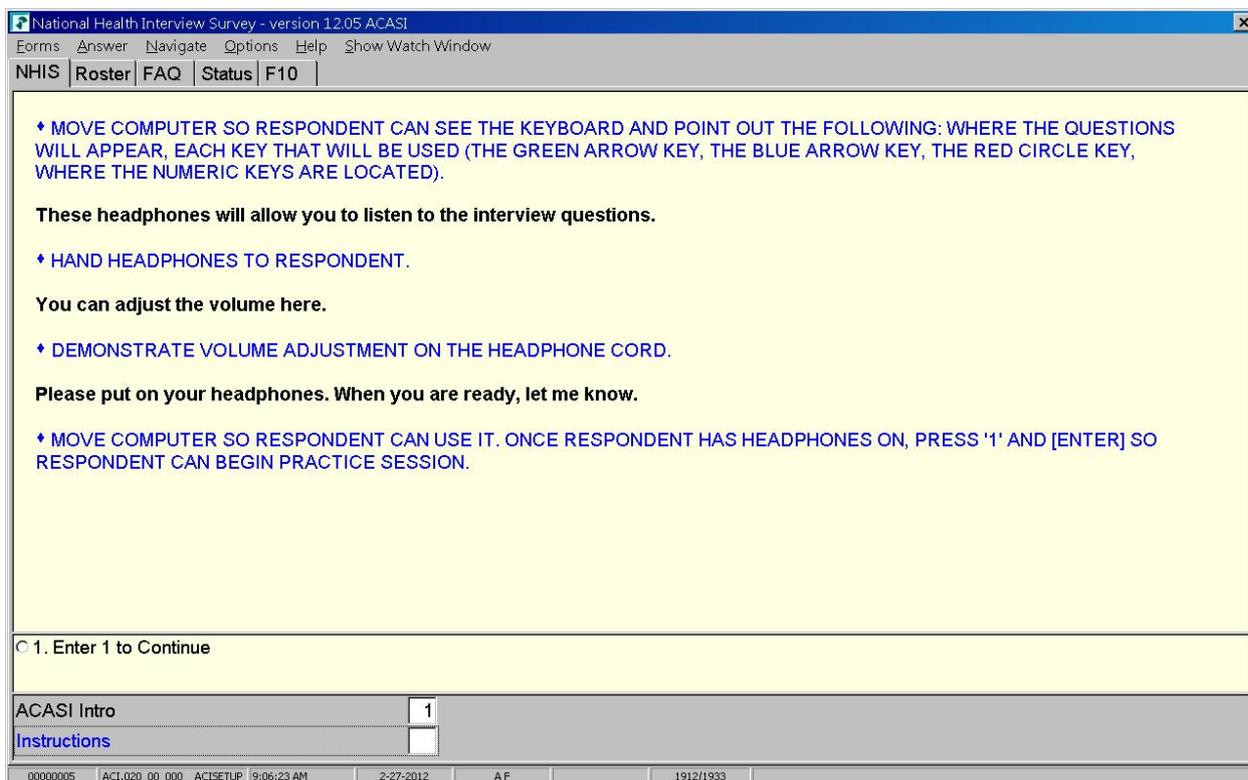


Figure 11 - Transitioning the Laptop to the Respondent

After administering the introductory screens, the ACI block calls the procedure `prc_launchACASI`. This procedure in turn calls the alien procedure `script_launchACASI` to launch `Basil.exe` and run the ACASI module.

4.2 The script_launchACASI Alien Procedure

The first thing the script_launchACASI alien procedure does is create and open a new database for the Basil ACASI module (i.e., NHIS_ACASI.bdb). The procedure then transfers the values of all the Fields (defined in the Adult.ACI block) from their storage locations in the Blaise database into the corresponding Fields in the Basil database. The Field must exist (with the same name) in both the Blaise instrument and the Basil instrument, and AUXFIELD values are not transferred.

The complete transfer of the Fields from the Adult.ACI block in the Blaise instrument's database to the Basil database might seem a bit excessive in light of the following facts.

- Only a few of these Fields (e.g., SEX) are populated at this point in time.
- Neither the FR nor the respondent will be allowed to re-enter the ACASI module once the respondent completes it.

However, it was a simple feature to implement, and if the requirement to lock the users out of this module is ever removed in the future, then the functionality for repopulating the Fields in the Basil instrument's database is already in place.

After the Field values are transferred from the Blaise database to the Basil database, the procedure calls the Basil.exe and passes the name of the Basil database and its primary key (CASEID) as parameters. The WAIT argument is used to suspend execution of the procedure until the new process (i.e., Basil.exe) terminates.

When the Basil.exe closes and control returns to the alien procedure, the procedure transfers the values of all Fields from the Basil database to the corresponding storage locations in the Blaise database (in this case the block Adult.ACI). Again, the Field must exist (with the same name) in both the Basil instrument and the Blaise instrument, and Auxfield values are not transferred.

The procedure turns on a flag (i.e., ACASIFLG) to indicate that the ACASI section is complete. This is done so that the rules in the Blaise instrument route to a "lock screen" to prevent the FR from re-entering the Basil module.

Finally, the procedure deletes the Basil database since it is complete and will not be accessed again by either the respondent or the FR.

5 Issues Developing and Deploying the ACASI Module

There were a number of issues that arose during the design and development of the ACASI module for the 50 case test. This section discusses some of the more noteworthy issues.

5.1 Navigation in the ACASI Module

Getting the navigation in the ACASI module to work correctly was problematic at times. Typically, the module worked as expected when moving straight through it without backing up. However, irregular navigation (such as repeatedly backing up and moving forward or backing up, changing an answer, and moving forward) would sometimes cause a problem. For example, the focus might not fall on the correct field on the screen or the instrument might completely freeze.

In general, we found that these problems were the result of a conflict between the rules and a task performed by an event handler we had programmed. Ultimately, the authors reduced the rules section of the ACASI module to the bare minimum and controlled most of the navigation in the module with event handlers.

5.2 Off Path Data

Unlike Blaise, Basil does not handle data that has been backed over and then placed off route. Backed over data are retained in a Basil application when it terminates.

While it is possible to program a script to clean up the off path data when the ACASI module terminates, this is essentially performing a data processing task in the field. This is an activity that the Census Bureau's authors prefer to avoid. After discussing the issue with the subject matter specialists in the DSD and the NCHS, all parties agreed to leave the off path data alone. The sponsor's data processing staff would cull the off path data after it came in from the field.

The ACASI module is small, and there are only a couple of places in the module where the respondent can back up, change an answer, and move forward down a new path resulting in off path data. As a result, the problem seemed to be minor and easily handled by the data processors.

5.3 Time between Screens

The authors encountered an issue with the time it took to move from one screen in the ACASI module to the next. For the most part this was limited to one screen that took 5 to 10 seconds before the audio played and the text displayed. This particular lag in the ACASI module was consistent on all the desktop workstations that were used to test the ACASI module. One developer actually had a significant lag between all the screens in the module when he tested it on his workstation. His machine was an older model and probably an exception.

The authors have not yet determined the cause of the problem. Researching the problem became less of a priority when they discovered that the problem did not occur when running the ACASI module on the FR laptops.

There is a significant difference between the workstations used by the authors for development and the FR laptops. For example, the operating systems are different. The developers' desktops run Windows XP while the laptops run Windows Vista. There are probably differences in the Federal Desktop Core Configuration (FDCC) settings as well. Some recent research suggests that the problem has something to do with the .mp3 audio file used for the screen experiencing the delay.

5.4 Transmitting the Instrument to the FR Laptop for the 50 Case Test

The NHIS ACASI module required files that we do not usually deploy with our instruments. Obviously audio files were required for the ACASI module. These were created in .mp3 format. The module also used image files to provide visual clues to the respondent on how to use the instrument. These were bitmap files (.bmp).

Table 1 provides the type, number, and total uncompressed size of the additional files required for the ACASI instrument.

Table 1. ACASI Specific Files Required by the NHIS Instrument

File Type	Number of Files	Total Size in Bytes
.mp3	162	9,040,750
.bmp	8	1,456,024
Totals	170	10,496,774

These files quadrupled the size of the typical NHIS instrument "package" that we transmit to all NHIS FRs. Whenever we have large amounts of data to transmit to FRs, we always try to find an approach that has the least impact on the users. Since the 50 case test involved only 6 users, we decided to customize our transmission scripts so the additional files were sent separately from the instrument package and were sent only to the 6 FRs participating in the test. While all the NHIS FRs received the 50 case test instrument, only the 6 FRs working the test received the audio and image files. As we move into the larger tests or into production, all NHIS FRs will have to receive these files (which could possibly change with each instrument release) therefore increasing the length of time our quarterly NHIS instrument transmissions will take.

5.5 Testing the ACASI Module

Testing an ACASI application can be time consuming. It is necessary to listen to every recording completely, and there can be quite a few recordings in an application. For example, in the NHIS ACASI module, a simple question like the one that appears below requires six audio files.

Were any of your grandparents born in the United States?

Yes

No

I don't know the answer

I don't want to answer

When the respondent lands on the question, two audio files play. The first one states the question and the second one reads the complete answer list. Then there are individual recordings for each of the response options in the answer list. These play as the focus moves from one answer to the next when the respondent uses the Space Bar to cycle through the answer list. Consequently, there are a total of six audio files that must be played in their entirety to test this question. This is a relatively simple question. Many of the questions in the ACASI module had much longer answer lists and therefore more recordings to test.

Testing an ACASI instrument that administers the questionnaire in a language that you do not speak is another challenge. In this situation, it is useful to have a copy of the specification with the question text in the foreign language and a flow chart of the instrument.

As with any automated questionnaire, it is necessary to test irregular navigation. Testing the ability to back up and move forward and back up and change paths in the instrument seems to even more critical in an ACASI application developed in Basil.

6 Future Initiatives

At the time this paper was written, the Census Bureau was completing work on the ACASI module for the 1,100 case test. The Census Bureau is scheduled to field test this instrument in April 2012. Several new features were added to the ACASI module for this test. Plans for the 11,000 case test were also being prepared. Some of the initiatives for those two tests are presented here.

6.1 Respondent Debriefing Questions

Eight respondent debriefing questions were added to the end of the ACASI module for the 1,100 case test. These questions ask respondents for feedback on their ACASI experience. The debriefing questions are listed below.

- Did the interviewer explain how to use this computer?
- Did you have any difficulty using the keys to choose your answers?
- Did you listen to the recorded voice for all of the questions, some of the questions, or none of the questions?
- Was the recorded voice too slow, too fast, or about the right speed?
- Was the recorded voice too low, too loud, or about the right volume?
- Did you use headphones or earbuds?
- How comfortable were the headphones/earbuds?
- Do you think it was important for this part of the interview to be private?

Many of these questions were included in the respondent debriefing based on observations made during the 50 case test. The NCHS wanted to get better feedback from the respondents about their ACASI experience.

6.1.1 Audio Recordings

After the 50 case test, there were some comments that the narrator in the (English) audio files was speaking too slowly. Respondents who listened to all of the recordings that came on route (in their entirety) took a considerable amount of time to get through the ACASI module. Several respondents took over 10 minutes to complete the ACASI questionnaire, and one respondent took 20 minutes to complete it. The survey methodologists plan to review the respondent debriefing data from the 1,100 case test to determine whether the audio files need to be rerecorded.

6.1.2 Audio Equipment

The 50 case test used headphones with disposable covers. The FRs sometimes had difficulty putting the covers on the headphones. This caused a delay in transitioning the laptop from the FR to the respondent, and in some cases seemed to cause some concern on the part of the respondent. One proposed solution to this problem was to use larger covers if they were available. Another suggestion was to offer the respondents a choice of the headphones or earbuds that they could keep or throw away. For the 1,100 case test, respondents will be given a choice between the headphones and the earbuds. Again, the survey methodologists will review the respondent debriefing data determine what equipment works best.

One participant in the 50 case test did not want to use the headphones. Others wanted to use their own headsets. The FRs had not been trained to handle this situation so they had issues getting the headsets to work in some cases. Sometimes the FR could not get the respondent's headset to work with the FR laptop. Headsets with an analog audio jack worked. Headsets with a USB connection might or might not work depending on whether the laptop had the appropriate device driver installed.

6.2 Audit Trails

The ACASI module used during the 50 case test lacked audit trails because there had not been enough time to implement this feature before the test. There will be audit trails for the 1,100 case test

The author implemented the audit trail functionality using Maniplus procedures. (Much of this code was borrowed from a Blaise example.) There were procedures for initializing a temporary audit trail file, formatting the audit trail records and writing them to the temporary file, appending the records in the temporary file to the Blaise instrument's audit trail file, and closing out the audit trail process when the respondent exits the ACASI module.

The initialization and close out procedures are called from the oncreate and onclosequery event handlers (respectively) that are present in the Basil module's application section.

The procedure that formats and writes an audit trail record (WriteAuditTrail) is called from within each of the module's question sections under several conditions. The procedure is called:

- when entering a field (onenter),
- when the respondent presses the Enter key to save an answer and move to the next screen (onreturn),
- when the respondent presses the Space Bar to scroll to the next option in an answer list (onkey_1), and
- when the respondent presses the Tab key to back up to the previous screen (onkey_2). (The onkey_1 and onkey_2 events are defined in the module's application section.)

When an event handler in a question section calls the WriteAuditTrail procedure, it passes a string to the procedure that identifies the question from which it was called and the type of event that occurred. This allows the WriteAuditTrail procedure to write out an audit trail record with the appropriate information.

In order to write to the Blaise instrument's audit trail file from the Basil application, it was necessary to modify the audit trail information file (.aif) so the CloseFile option is set to 1. This instructs the Blaise Audit Trail dll to open and close the audit trail file every time it writes to the file. Since the

audit trail file is closed when the ACASI module is running, the audit trail file is available for the ACASI module to write to it.

An example of the ACASI audit trail appears below. In this example, the first seven lines are from the Blaise NHIS instrument and the rest from the ACASI modules.

```

"2/28/2012 3:37:40 PM", "Enter Field:ACI.ACILANG", "Status:Normal", "Value:"
"2/28/2012 3:37:42 PM", "<KEY:>21<ENTRI"
"2/28/2012 3:37:45 PM", "Action:Store Field Data", "Field:ACI.ACILANG"
"2/28/2012 3:37:45 PM", "Leave Field:ACI.ACILANG", "Cause:Next Field", "Status:Normal", "Value:1"
"2/28/2012 3:37:45 PM", "Enter Field:ACI.ACASINTR", "Status:Normal", "Value:"
"2/28/2012 3:37:46 PM", "<KEY:>1<ENTRI"
"2/28/2012 3:37:47 PM", "Action:Store Field Data", "Field:ACI.ACASINTR"
$2/28/2012 3:37:51 PM
$2/28/2012 3:37:52 PM ENTER FIELD ACIHEAD
$2/28/2012 3:43:09 PM LEAVE FIELD ACIHEAD GREEN ARROW
$2/28/2012 3:43:09 PM ENTER FIELD ACIINTR1
$2/28/2012 3:43:30 PM LEAVE FIELD ACIINTR1 GREEN ARROW
$2/28/2012 3:43:30 PM ENTER FIELD ACIINTR2
$2/28/2012 3:43:49 PM LEAVE FIELD ACIINTR2 GREEN ARROW
$2/28/2012 3:43:49 PM ENTER FIELD ACIDLIC_Question ACIDLIC
$2/28/2012 3:44:09 PM LEAVE FIELD ACIDLIC_Question RED CIRCLE ACIDLIC
$2/28/2012 3:44:09 PM ENTER FIELD ACIDLIC_Yes ACIDLIC
$2/28/2012 3:44:11 PM LEAVE FIELD ACIDLIC_Yes RED CIRCLE ACIDLIC
$2/28/2012 3:44:11 PM ENTER FIELD ACIDLIC_No ACIDLIC
$2/28/2012 3:44:13 PM LEAVE FIELD ACIDLIC_No RED CIRCLE ACIDLIC
$2/28/2012 3:44:13 PM ENTER FIELD ACIDLIC_Question ACIDLIC
$2/28/2012 3:44:16 PM LEAVE FIELD ACIDLIC_Question RED CIRCLE ACIDLIC
$2/28/2012 3:44:16 PM ENTER FIELD ACIDLIC_Yes ACIDLIC
$2/28/2012 3:44:19 PM LEAVE FIELD ACIDLIC_Yes GREEN ARROW ACIDLIC 1 Yes
$2/28/2012 3:44:19 PM ENTER FIELD ACINUM_QUESTION ACINUM
$2/28/2012 4:03:33 PM LEAVE FIELD ACINUM_QUESTION GREEN ARROW ACINUM

```

Figure 12 - ACASI Audit Trail

Note that the ACASI audit trail records differ significantly from the Blaise audit trail records. The format of the ACASI audit trail records was the Blaise author’s attempt to make the ACASI section of the audit trail as readable as possible. While this approach is creative, it has a downside. Both the Census Bureau and the NCHS use applications that parse the audit trail files to obtain timing information and other data. These applications will not be able to process the ACASI audit trail records without some modifications. Since the NCHS uses an application developed by another organization, modifying their application is not an option. For the 1,100 case test the ACASI audit trail format will remain as it is. The DSD data processors will strip the ACASI records (all of which begin with a “\$” to make identifying them easy) from the audit trail before passing them along to NCHS. After the 1,100 case test, the authors will need to modify the ACASI audit trail procedure so it writes out records that have the same format as the Blaise audit trail. The resulting ACASI audit trail might be less readable, but it will conform to all the systems that use the audit trail files.

6.3 Spanish

The ACASI module for the 1,100 case test will have a Spanish component. Unfortunately, Basil does not support multiple languages the way Blaise does. The NCHS programmer’s solution for the multiple language issue was to create two separate datamodels – one in English and the other in Spanish. The Census Bureau’s authors had hoped to find a better solution, but ended up settling on a very similar approach. The Census Bureau’s authors created a single datamodel in which there were two paths – one for English and one for Spanish. This approach is essentially the same as the NCHS method. Neither technique is satisfactory. Both complicate the programming and testing of the ACASI module. The authors have to keep the logic of the two paths in synch, and the testing effort doubles because there are two unique paths to test. The Census Bureau’s Authoring Staff continues to look for a better way to implement multiple languages in Basil.

The issue with the lag between screens in our development environment got worse when the authors added the Spanish to the ACASI module. The lag at the one screen in the English path remained the same, but several screens in the Spanish path had similar delays before the audio played and the question text appeared on the screen. Like the lag on the English path, this was a problem on the developers’ workstations only. The Spanish component of the ACASI questionnaire ran without the delays on the FR laptops.

Adding Spanish to the ACASI module essentially doubled the total size of the audio files needed to administer the questionnaire. This further aggravates the issue of transmitting the instrument to the FR laptops.

6.4 Edit Checks

Edit checks for numeric fields were not available in the 50 case test instrument. The authors had programmed the edit checks, but they were in a version of the NHIS instrument that had not been thoroughly tested. As a result, an earlier version of the instrument without the edit checks was used for the test.

The 1,100 case test instrument will have edit checks for the numeric fields. These are implemented using Maniplus procedures and dialogboxes. The figure below displays an example of one of the error messages.

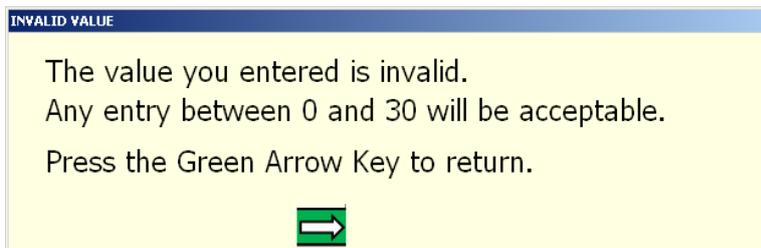


Figure 13 - Range Check Dialog Box

While these edit checks are adequate, the dialogbox does not play audio that corresponds to the text. The edit checks that use the Maniplus dialogboxes will be replaced for the 11,000 case test. When respondents enter a value that is out of range in the 11,000 case test, the ACASI module will direct them to a new screen in the instrument that will inform them visually and acoustically that there was an error. The ACASI module will then return them to the previous screen where the respondents can make the correction.

7 Conclusion

There are some advantages and disadvantages in using Basil to develop an ACASI questionnaire. Basil's two big advantages are that it gives the developer considerable control over the appearance of an application's screens, and it also gives the developer control over actions that need to take place upon the occurrence of certain events. When compared to Blaise, Basil's main disadvantage is that it is not as easy to program. The Basil documentation that comes with the Blaise Control Centre states, "By default your BASIL application does not do too much." This is true. The programmer must manage a number of issues that Blaise would handle automatically or could be implemented in Blaise with less effort than in Basil. This paper addressed some of these issues such as the support of multiple languages, irregular navigation, handling data that is off route, writing an audit trail, and so on.

An ACASI module would have been easier to implement and maintain in Blaise, and it would have been more robust and less problematic in the field. However, a Blaise ACASI module would not have provided the simple user interface that the customer requested.

While the ACASI pilot project is progressing satisfactorily, the project has not been problem free. An initial lack of experience with Basil at both the NCHS and the Census Bureau was an issue. The compressed project schedule probably contributed to some of the issues we encountered as well. Rather than taking the prototype developed by the NCHS and simply expanding the application as we did, it might have been useful to take the time to analyze the prototype and determine whether there was a better way to implement it.

Since the NCHS provided the ACASI prototype, it has grown substantially. We would like to find a way to reduce the amount of code and simplify the task of maintaining the ACASI module. The authors at the Census Bureau have started to reevaluate how the Basil module was designed to determine whether there is a better approach. Most questions in the NHIS ACASI module have several question-specific event handlers programmed in Manipula to control some functionality. The programmers are investigating whether they can use a more generic approach that might make use of generalized templates to avoid all the question specific procedures. There are probably other ways to simplify the Basil ACASI module. For example, instead of customizing the ellipse to fit the various sizes of the answer categories, it would be easier to use a generic rounded rectangle that spans the page. Such a rectangle would fit all answer categories, and there would be no need to customize the ellipse for each answer in each question's answer list. (Because this sort of change affects the ACASI module's GUI interface, we would have to negotiate the implementation of such a change with the sponsoring agency.)

8 Acknowledgements

The author would like to acknowledge the following people for input into this paper.

Andrea Piani, U.S. Census Bureau
Andrew Stevenson, U.S. Census Bureau
Ann Daniele, U.S. Census Bureau
Anna B. Sandoval Girón, National Center for Health Statistics
Jason Arata, U.S. Census Bureau
Karen Bagwell, U.S. Census Bureau
Malcolm Robert Wallace, U.S. Census Bureau

9 Disclaimer

The views expressed on (statistical, methodological, technical, or operational) issues are those of the author(s) and not necessarily those of the U. S. Census Bureau.

Challenges and Lessons Learned Using Blaise IS with SQL Server

Max Malhotra and Jas Sokhal
Survey Research Center, University of Michigan

1 Introduction

The University of Michigan's Survey Research Center Survey Research Operations unit (SRC/SRO) is engaged in a collaborative Army Study to Assess Risk and Resilience in Service members (STARRS). This is the largest study of mental health risk and resilience ever conducted among military personnel with a goal to collect data from approximately 100,000 Soldiers over a 2-year period. The National Institute of Mental Health (NIMH) assembled a group of experts to carry out this research, including teams from the Uniformed Services University of Health Sciences (USUHS), University of Michigan, Harvard University, the University of California-San Diego, and NIMH. The Technical Services Group (TSG) at SRC was assigned the task of Systems Development for the survey data collection, the management of data, and the control of data.

Computer Assisted Interview (CAI) data collection had to be performed at the remote Army installations located around the United States with no access to the Army network infrastructure. This created unique challenges in terms of maintaining data, confidentiality, integrity and availability.

The requirements included:

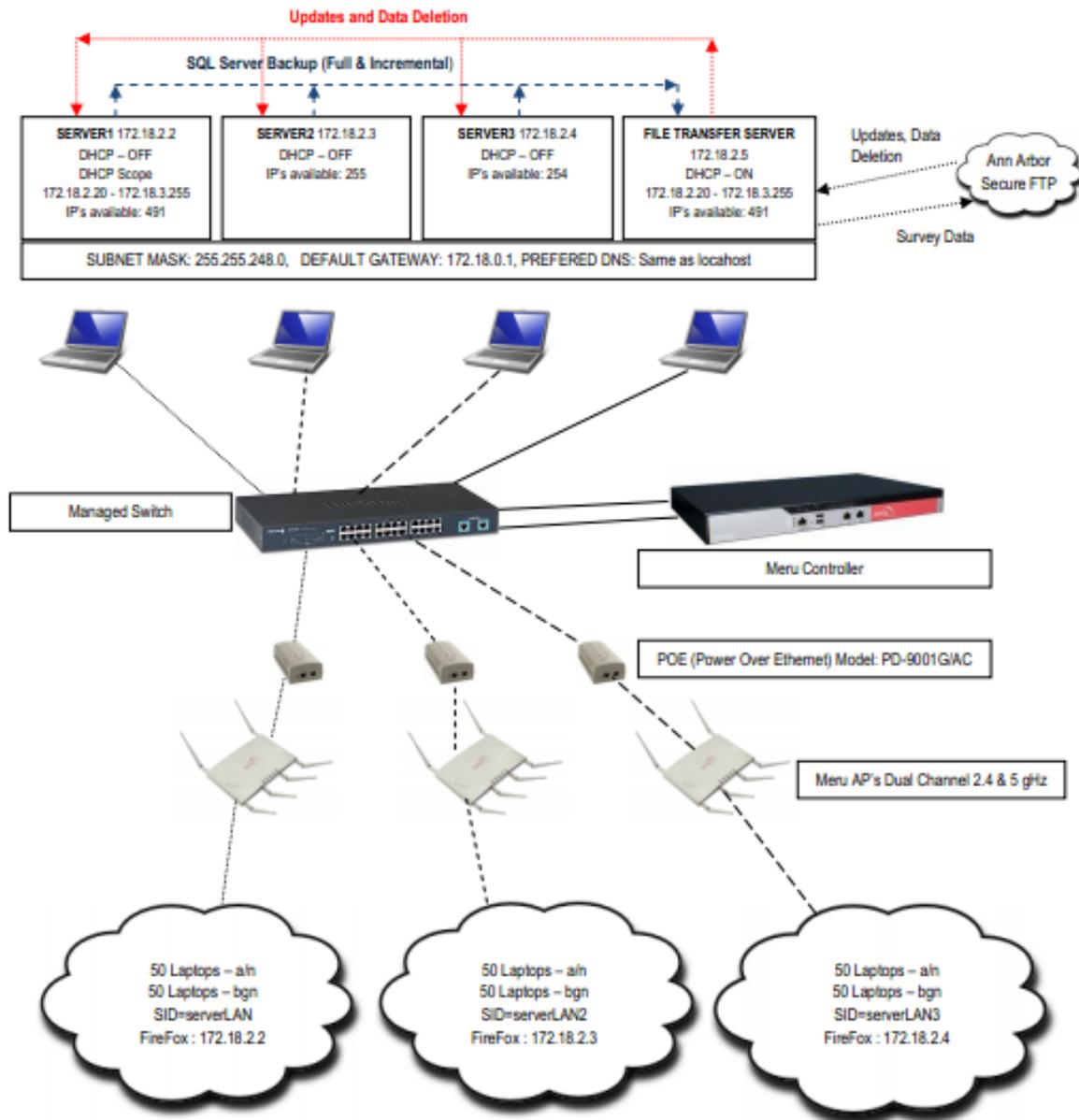
- Federal Information Processing Standards (FIPS) Compliance
- Bitlocker Encryption
- A secure relational database support Transparent Data Encryption (TDE)
- A Secure protocol for transferring data to and from the Data Management Center
- Data deletion methodology on remote servers
- The capability to integrate Neurocognitive testing

After an extensive evaluation of available data collection software and tools, TSG chose Blaise IS as the main data collection instrument front end with a MS-SQL Server back end. The data collection started in winter 2010, and as with any first Production launch, it had its fair share of complications, particularly with Blaise lockups using MS-SQL Server.

With the lack of traditional methods of communication and no access to a self-contained network infrastructure, addressing and remediating issues encountered in the field posed particularly unique challenges. This dictated that the Quality Assurance (QA) processes for the data collection equipment had to be flawless. In addition, a lab environment replicating the field had to be established for troubleshooting. We will explain in this paper the issues encountered during data collection, troubleshooting in the lab, collaboration with Statistics Netherlands resulting in software releases, and remediation plans.

2 Systems Architecture

The data collection system architecture for 300 concurrent participants, which was the maximum number per session, is illustrated in the diagram below. After extensive benchmark load testing it was ascertained that the optimal number of users connecting to a server in this architecture was 100 users. This was due to the fact that all participants started the survey at precisely the same time causing a peak load condition.



There are **12** different major pieces of equipment in use, not counting the printer, cables, power strips, or power cords! There are 24 points of connection among those 12 pieces of equipment, so that means that there are *at least* 36 different points of potential failure in the hardware setup. Each of these points of failure would need to be tested to eliminate the potential that this was where something had gone wrong. The technical problems are further compounded by the fact that each component in the network architecture plays a pivotal role in a large scale site administration, so that a failure of any one of these points could bring data collection to a halt for 100 survey participants. If this is not enough, all the equipment has to be assembled and then disassembled for each data collection session.

Each of the components had been designed with redundancy in mind so that each had a hot swap duplicate ready if a malfunction occurred.

The points of failure could include network architecture, server configuration, hardware failure, software malfunction, and other unknown variables. For the infrastructure to work correctly the components must follow an established path of communication, which implies that the controller and servers must connect to the switch, and the switch must be connected to the Power Over Ethernet

(POE) devices, and the POE must connect to the Access Point (AP). The Access points will then serve as the broadcast media to the clients and perform bi-directional communication to the servers.

Troubleshooting becomes compounded when there are 300 participants answering questions in a tightly limited time-constrained group survey session, particularly since there is no opportunity to re-take the survey. As such, we had to build in redundancy and countermeasures for every component in the infrastructure. System monitoring tools were utilized to observe the health of the controller and the computers connected to the network. The cabling was color-coded making it easier to swap out backup equipment at a moment's notice in case of any systems failure.

3 Security Requirements

This study required stringent security measures to preserve the integrity of the data. The following security measures were implemented on the data collection servers.

- **Federal Information Processing Standard (FIPS):**
System Cryptography: Use FIPS compliant algorithms Enable
<machineKey validationKey="AutoGenerate,IsolateApps"
decryptionKey="AutoGenerate,IsolateApps" validation="3DES"
decryption="3DES"/>
- **Bitlocker:** With USB Key boot up.
- **SQL Server - Transparent Data Encryption (TDE)**
- **Secure File Transfer Protocol (SFTP)** to send/receive data

In addition to system security the equipment had to be supervised during data collection and all the equipment locked in a secure location while not in use at the Army base.

The file transfer server was the only piece of equipment that was authorized to be taken off site to perform a send / receive of the data.

4 Issues and Troubleshooting

Many issues were encountered initially and the problems were assigned to one of the following categories:

- System setup - connecting together all the components, location of Access Point (AP)
- Hardware failure - Server, Client Laptop, Controller, POE, Switch, AP
- Configuration - Server, Client Laptop, Controller, AP
- Software Failure - SQL Server, Blaise IS, IIS, Operating System, Firefox, etc.
- Other - Military signal scrambling, solar storms!!!

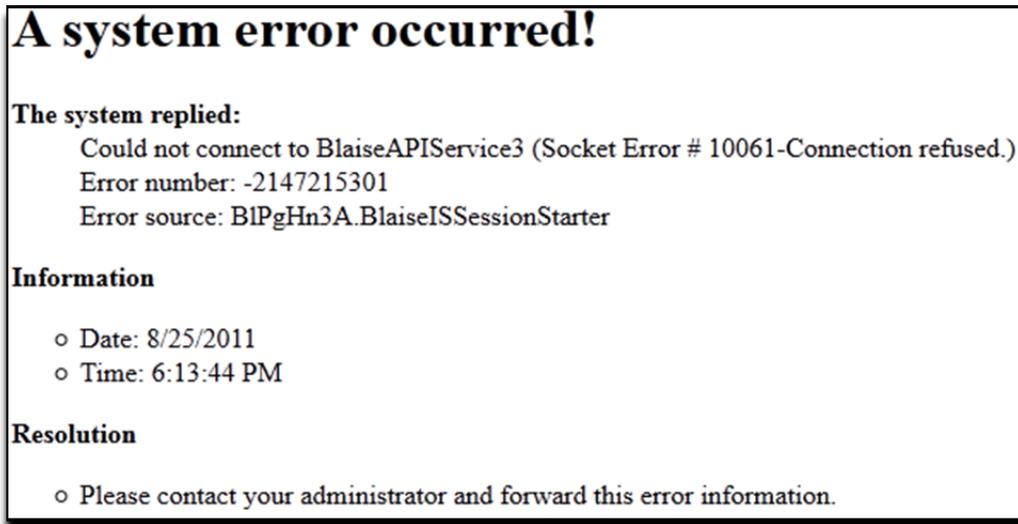
The focus in the following sections is primarily on the troubleshooting the Blaise issues and neurocognitive test.

4.1 BDB vs SQL Server

Concurrent benchmark testing was performed using both BDB's and MS-SQL server. This was conducted to isolate the issues related only to Blaise, and to verify that none of the other numerous components were inadvertently causing the Blaise lockups. It was observed that while using BDB's, no Blaise lockups were encountered due to the Data Link and performance was better than using SQL Server with Blaise.

4.2 Blaise API (Data Link) Issues

How did we discover it was Blaise API? On the client machine an error was received in the browser similar to the one shown below.

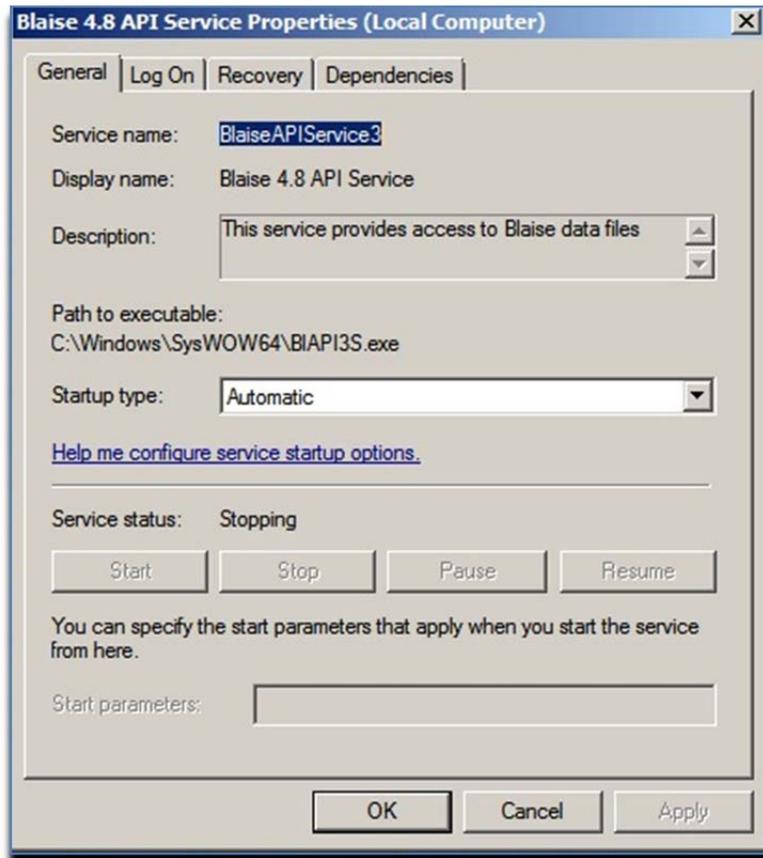


4.2.1 Blaise API Service

The error above would appear simultaneously for all users, usually 100, connected to a particular server. By reviewing the client machine, we could determine very quickly which server had the issue. The first step was to look to see if the Blaise API service was still running on the server. If it was running, then the assumption was that the Blaise API service had an internal problem and was hanging. After restarting the service we noticed that this resolved the issue in most cases. However, in some instances the Blaise API service failed to restart and the message below was displayed:



Following this error, the ability to Start or Stop the service is no longer available. The solution was to open the task manager and end the process for BI-API3S.exe and then restart the Blaise API service.

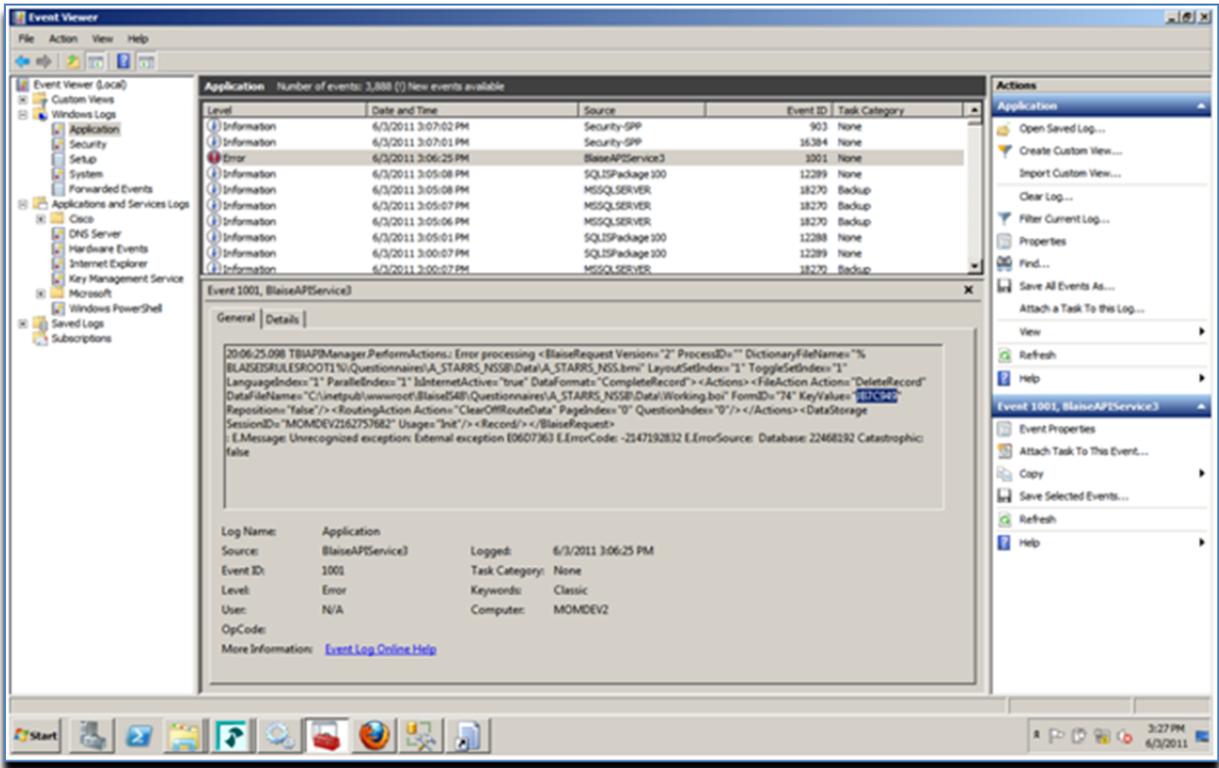


NOTE: By restarting the Blaise API service the session data disappears, but the browser still has a valid asp session. By pressing F5 the new interview page is determined by the API service with an empty session so it returns to the first page. This new empty session has an empty key value. Therefore the save actions to the working database fail.

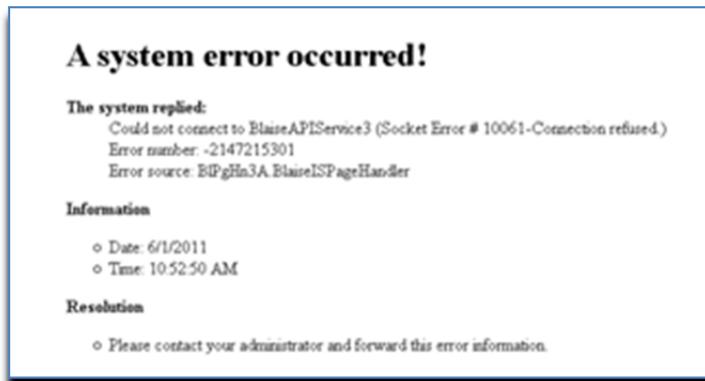
Most errors related to the Blaise API service were similar to the one above and multiple case studies and corresponding Windows Logs and Blaise Garbage logs were shared with Statistics Netherland.

4.3 More Troubleshooting with the Blaise API Service

We observed that after stopping the Blaise API service during the course of a test, the Neurocognitive (NC) tests continued. In the Event viewer we noticed that there was a BlaiseAPIServie3 error that occurred.



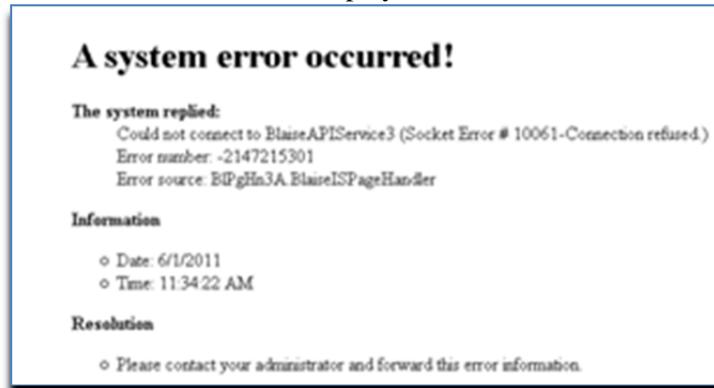
Also the following screen was displayed in the browser:



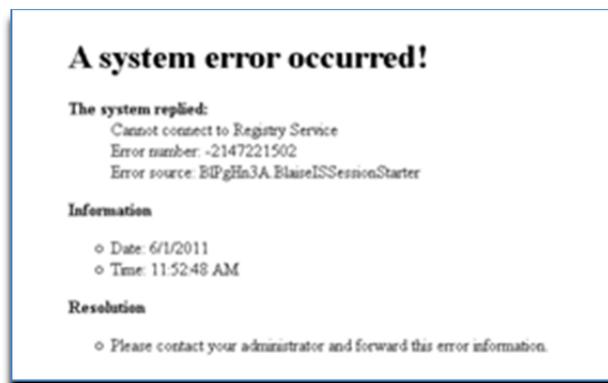
The Blaise API Service was restarted on the server followed by pressing the F5 refresh key on the client machine. This resulted in another error shown below. We had to shutdown Firefox and restart it and log back in, which allowed the survey to continue.



Next we stopped the Blaise 4.8 Internet Survey Manager Service and we saw the same error as when we stopped the Blaise 4.8 API Service. In addition, stop the Internet Survey Manager Service, causing the Blaise API service to stop as well. No errors were generated by stopping this service in the event log. However, the screen below was displayed on the client machine.



Next we stopped the Blaise 4.8 Registry Service this again caused the Blaise API service to stop; additionally this caused the Blaise Internet Survey Manager Service to stop as well. The following errors were displayed on the client machines.



We saw the following error as well on the client machine.

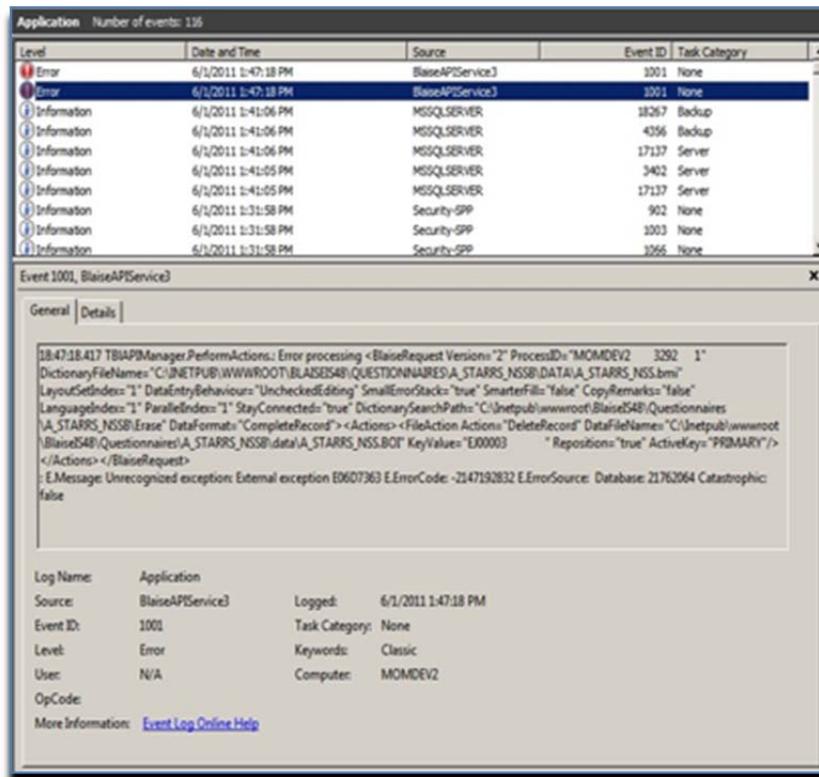


After services were restarted and survey participants were back in the survey, we shut down the Portal Service, Server Pack Service, and the Service Monitor Service which had no adverse effect on the survey.

Many similar battery tests were conducted and the results were shared with Statistics Netherland.

4.4 Data Deletion Roadblock

While troubleshooting Blaise issues we stumbled upon survey data in the windows event log written by the Blaise API service similar to the one shown in the diagram below.



The problem appeared to be happening while data were moving from the working to the main database.

This led us to believe that there was an issue with the Blaise API service deleting records in the working database. To test this theory we added 10,000 rows of data to a test table and wrote a utility using the Blaise API service in manipula to loop through the 10,000 rows and perform consecutive 'ReadNext', 'Delete' statements. We consistently observed that the API service crashed between 400 - 470 rows.

We successfully ran the same utility using BDB without any issues.

1. The import was much faster
2. The delete was faster
3. 10200 cases were deleted without any errors in the event log.
4. Even if the API service is stopped, the BOI deleting script is run just as fast.

From this, we concluded that the Blaise API service had a problem with the BOI database.

This observation was shared with Statistics Netherland and the utility forwarded to them. They were able to replicate the error in their environment and discovered a bug in the DBLink component resulting in a Blaise release.

4.5 BOI File Corruption

On several occasions the Working.BOI file gets corrupted and Blaise is no longer able to access tables for the Working.BOI database. The temporary solution to this issue is to open the main BOI file for that database and rename the tables to WORKING... Subsequently we do a "Save As" and save it as

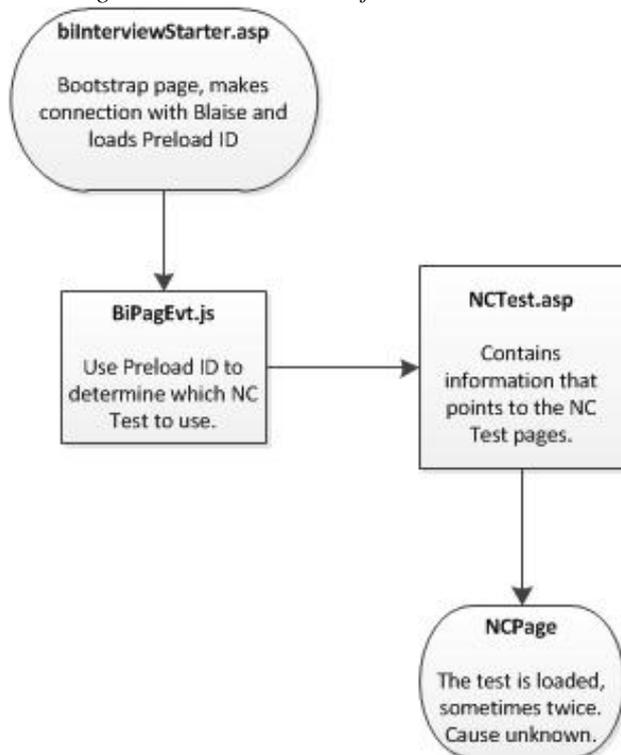
the Working.BOI file. The cause has yet to be determined. One theory is that an old version of the .BOI-file is restored by the Registry Service.

4.6 Duplicate Neurocognitive Tests

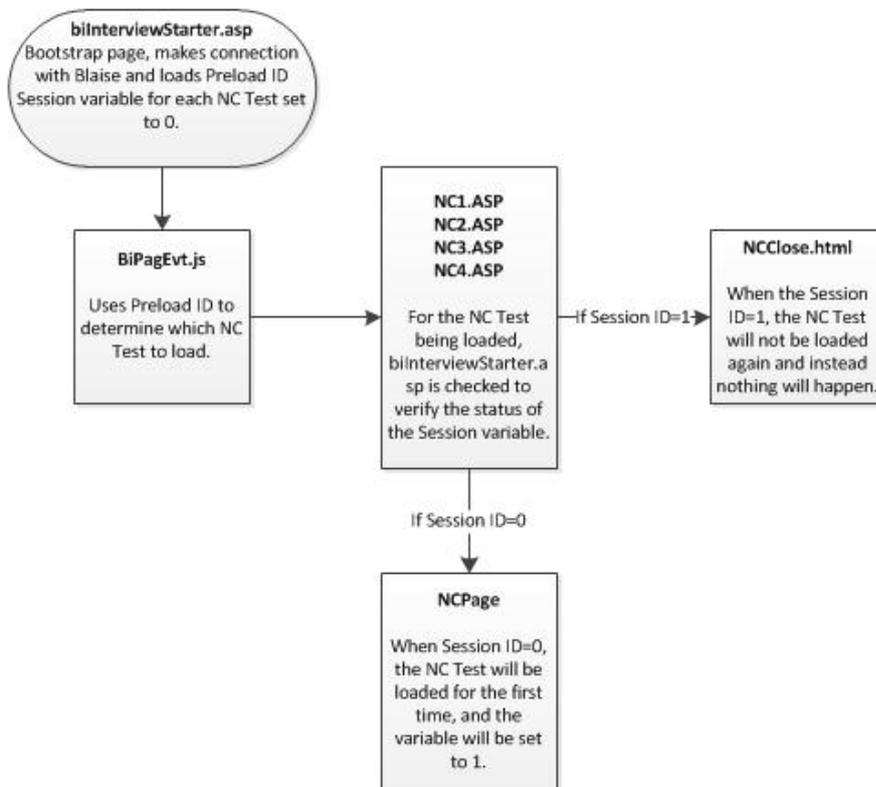
During the course of the survey neurocognitive (NC) tests are administered to the soldier, written in Flash and launched from Blaise IS survey. These test are designed to gather baseline thinking/reasoning/reaction measures. We noticed that, in some cases, two instances of the same NC test would launch when a soldier started the first test. When the first one was completed the second would come to the foreground, and the soldier, naturally, would do the test a second time. This error occurred with an alarming frequency of about 20%.

The code was redesigned to circumvent this error, as diagramed below.

Neurocognitive Test Process - Before



Neurocognitive Test Process –After



4.7 Missing Data

We have cases where the survey data are missing but the paradata are available. Paradata of a survey consists of data about the process by which the survey data is collected.

The survey data can be reconstructed from the paradata. Since the Blaise API service was being utilized to write the paradata from the client to the database, this posed a high risk of complete data loss for some cases due to Blaise API lockup frequency.

To mitigate this risk a paradata backup table, named *Paradata_Alternate*, was created. There was an addition to the front-end code to also write the paradata to the Alternate table. However, instead of using the Blaise API to write the data, the native Microsoft ActiveX Data Objects (ADO) component was used to write directly to the database. The clear advantage to this approach is that if a Blaise IS service crashes the alternate table will not be affected.

One of our major concerns was what impact this change may have on performance. Even though we test it in the lab, it's difficult to replicate the exact production environment. To mitigate the risk we tested this in a small scale field test. We also built in a toggle function to allow the feature to be turned on and off on specific servers. Currently, we still have cases where the survey data are not written by Blaise to the Working or Main database even though the Blaise API service appears to work fine for other users on the same server. We are researching this issue and trying to replicate it in the lab.

5 Quality Assurance

Quality Assurance (QA) is the systematic process of checking to see whether the system being developed is meeting the requirements. Since remote access for troubleshooting in the field was next to none, airtight QA processes were established. This was accomplished by several daily reviews of

the processes and continual refinement. The equipment was configured and rigorously tested in the QA lab before deployment.

In particular, the servers running Blaise were setup using a very specific set of instructions in order to maintain consistency throughout the project. These installations are then tested by the Business Team, Programmers, Data Managers, and the Lab Technical Support staff. We also used Visual Studio for load testing.

Once all survey and security tests are passed, the servers are imaged to a cloning server to quickly replicate the setup. The efficiency gained by this process to build a QA certified server, was to achieve this in several hours as opposed to 2-3 days. Images of servers deployed in the field were stored in the QA lab and an exact duplicate of the server could be re-created in the lab for troubleshooting.

Sanity check lists were created with sign off from all the teams mentioned above, which established accountability for every piece of equipment that left the lab.

A sanity check is a process that systematically verifies all equipment, configuration, and data components, such as server builds. It encompasses software and hardware design and specifications.

For example, the sanity checklist document for the data collection server consists of approximately 50 steps with each step checked and signed off on for every server ready for shipping to base (see sample below). In this way, a quality control paper audit trail is maintained.

Q3 SANITY CHECKLIST v3.1.0 doc SANITY CHECKLIST

SANITY CHECKLIST

Checked By: _____ Date: _____ Time: _____

Server ID (On bottom of server): _____

MOMDEV1: _____ MOMDEV2: _____ MOMDEV3: _____

MOMDEV4: _____

(Refer to Run Book) Appendix C):

Post Name: _____ Post ID: _____ Post Letter Indicator: _____ Project _____

Initial Setup _____ Perform all the steps below.

Specific Army Base Setup _____ Perform only the steps with an X in the AB column.

Primary Servers _____

Back Up Servers _____ CLONED: YES/NO (Print the first page and Cloned Section)

IF CLONED, TAG NUMBER OF PRIMARY SERVER: _____

MD1 _____ MD2 _____ MD3 _____ MD4 _____

#	Description	1	2	3	4	AB	Initials
1.	Verify the Sticker on the servers to indicate appropriate information. (Ex. Q2_FT_LEONARD WOOD 104-W MOMDEV1 PRIMARY/BACKUP). Make sure to put a sticker on the front and one on the top. If the sever is a cloned hard drive, make sure that serial number of the hard drive is stored in the Bitlocker key spreadsheet so you can later verify the location of the hard drive.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	X	
2.	CHECK TO VERIFY IF BIT LOCKER is enabled, if not then enable it following the instructions in the Installing Bitlocker with TPM Disabled run book.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	X	
3.	Verify with Lab Manager that he has the Bitlocker Keys	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	X	

6 Summary

Implementing Blaise IS with MS-SQL Server has not been without challenges, but the journey has been extremely fulfilling. The key to success was to establish QA best practices early, create a lab environment replicating production, engaging Statistics Netherland early in the game, and keeping the technical team continuously engaged at University of Michigan.

The following releases have been implemented, some of which were a direct result of the issues encountered.

1. Release 4.8.2.1589
2. Release 4.8.2.1606
3. Release 4.8.2.1618
4. Release 4.8.2.1639
5. Release 4.8.2.1649
6. Release 4.8.2.1653
7. Release 4.8.2.1656
8. Release 4.8.2.1700
9. Release 4.8.3.1717
10. Release 4.8.3.1735 ← The most stable so far
11. Release 4.8.4.1737

Acknowledgments/References

¹Supported by NIMH U01 MH87981, funds provided by the Department of the Army with supplementary funding by NIMH.

²O'Reilly, Jim. Paradata and Blaise: A Review of Recent Applications and Research
International Blaise User Conference 2009, Riga, Latvia

Servicing Blaise instrument needs in a multi-mode environment: Some technical examples

*Richard Frey, Kathleen O'Reagan, and Nona Brown
Westat*

International Blaise Users Conference, April 2012, London, UK

Summary: *As response rates have dropped and budgetary constraints limit telephone surveys, many projects today are operating in multi-mode project environments that offer more than one way for respondents to submit survey answers. This paper looks at how Blaise 4.8 capabilities and features work with multi-mode projects requiring fast setup and processing. It includes a technical discussion of issues encountered, what worked well and lessons learned in using Blaise as the solution. We will discuss the use of a single database and how to handle mode differences such as question types and text.*

Introduction

In a perfect world, specifications would be written that are universal across modes. That said, some projects are not planned as multimodal and subsequently need to be modified to accommodate the differences between modes. The goal is to attain a seamless integration of all modes, in this instance Computer Assisted Telephone Interviewing (CATI) mode, Computer Assisted Web Interviewing (CAWI) mode, and paper surveys, resulting in a unified system. We chose to make use of a single SQL database for all modes to minimize issues associated with moving files back and forth between databases when the mode of interviewing changes. The management system will still keep track of which mode currently owns an interview. A case in data collection in one mode can create work activity in another mode or close out work in another mode. The decision to use one database requires planning but this is much less effort than would be required to maintain and synchronize a different data model and database for each separate mode.

Multi-Mode Management System

While a Multi-Mode Management System (MMMS) is not the focus of this paper there should be a mention of the role it plays in a multimodal environment.

The role of an MMMS is to provide a single interface that project systems can use to load cases and manage the complexity of interfacing with each individual mode. Figure 1 depicts an example of an MMMS that can:

- Control the activation of cases across multiple modes.
- Create new cases based on the completion status of prior cases, e.g., if you get a completion for a web survey, schedule a follow-up call 1 month later.
- Provide a user interface to permit managers to query case status and manually activate/deactivate cases or change modes.
- Implement a service layer so that updates between MMM and modes can occur in real-time.
- Allow for new modes to be brought online more easily.

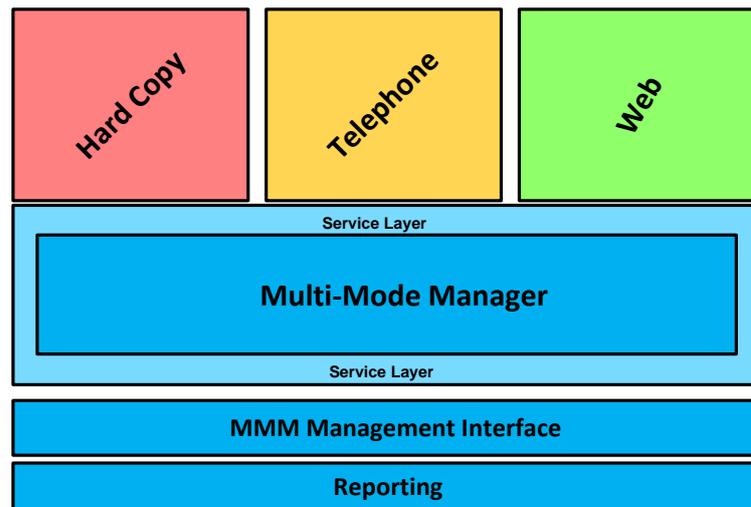


Figure 1. Example of Multi-Mode Manager

Using a Single SQL Database

In many multi-mode environments a mode is usually designed with a myopic view. Each mode has a data collection instrument, a database and a management system all different than other modes. As data collection proceeds, synchronization between modes becomes more and more difficult and time consuming. Programs have to be written such as to determine what mode a case is currently assigned, where it came from and completion status. Eventually the data will have to be merged for analysis, which again, necessitates writing programs to merge the mode data.

We chose to use Blaise for our multi-mode solution. Integrating the diverse modes using Blaise is relatively straight forward if you keep a few things in mind. To have one database you first need the instrument structure to be the same across all modes. In other words, instruments for each mode will have the same fields as any other mode. What will make each modes instrument unique will be the rules of the main datamodel and blocks. Conditional Defines are used to direct prepare directives to establish the rules of the datamodel for a mode.

Once the datamodel structure is finalized, a Blaise OLE DB Interface (boi) file can be created along with the tables in the SQL Database. The boi file is then used by each mode to read, write and update the database. However, each mode will only store data based on the rules of its instrument.

Customizing the Mode Library

Each collection mode will have its own customized Mode Library, also referred to as the modelib. Within a multi-mode environment there are times when you might want to combine the layout sets of a modes modelib. In doing so you increase the number of layout pages generated when the instrument is prepared. For each layout set in the modelib, the prepare process produces the individual pages with screen layout information needed for the DEP. The total number of pages is limited to 16,384 pages, easily reached with a large datamodel containing many arrays.

When using separate mode libraries the appropriate library must be used when preparing an instrument for a collection mode. Choosing the proper mode library is done at prepare time using prepare directives, specifically the {\$MODELIB} prepare directive. This directive is used at the very beginning of the instrument as shown below.

```

{$IFDEF WebLayout}
  {MESSAGE 'Preparing Web Version of our Instrument'}
  {MODELIB C:\SampleInstrument\Config\Web_Inst.bml}
{$ELSE}
  {MESSAGE 'Preparing DEP/CATI Version of our Instrument'}
  {MODELIB C:\SampleInstrument\Config\CATI_Inst.bml}
{$ENDIF}

```

For the CAWI mode, one of the design decisions required the display of all enumerations and sets in one column, and indentations five positions from the left side of the question text. In addition, we increased the size of the radio buttons and further separated the radio button from the category text. These global changes were made to the properties of the Answer List default FIELDPANE in the Internet layout set of the CAWI modelib. The property changes are shown in Figure 2, the Answer List control.

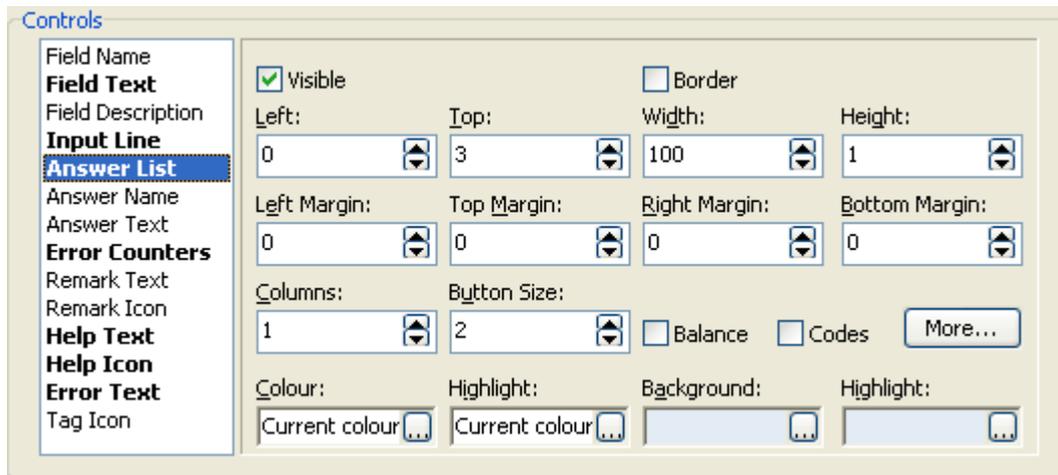


Figure .2 Example of Answer List Control

The radio button and text positioning were set using the Answer List Layout Specifier which is launched by selecting the More... button on the Answer List controls property panel as shown in Figure 3.

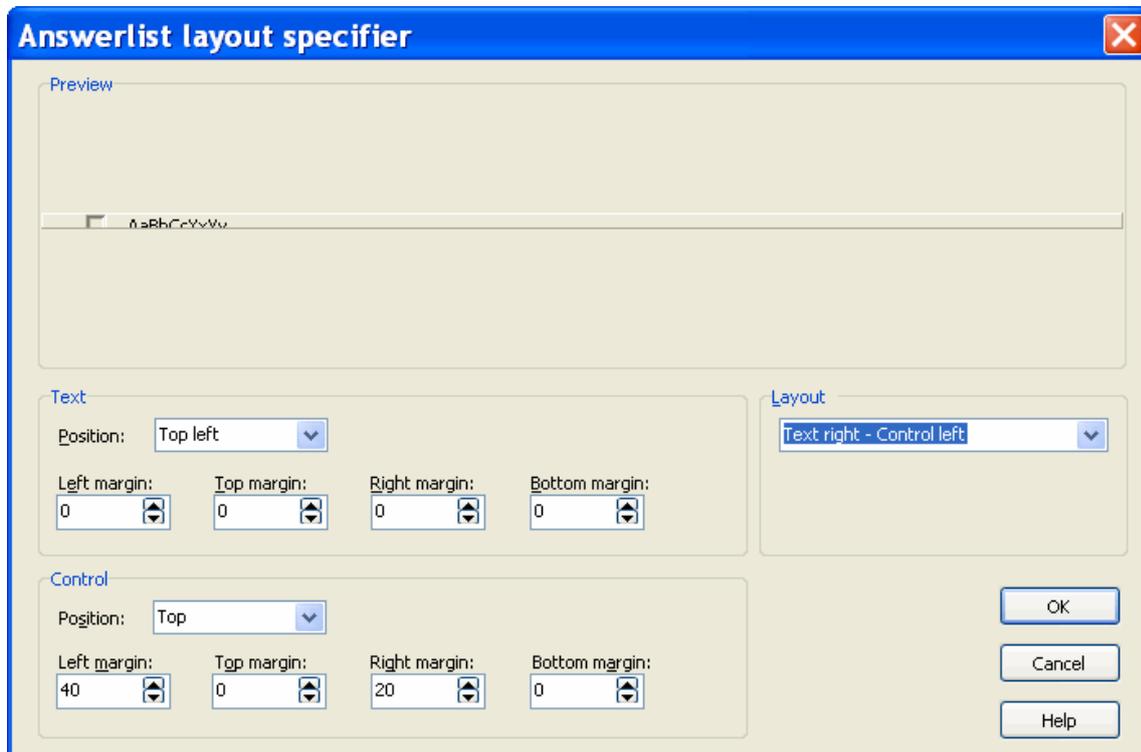


Figure 3. Answer List Layout Specifier

Lastly, the color scheme of the project's web site had to be incorporated into the layout of the survey to provide a consistent look and feel across pages of the projects web site. Using the color scheme tool the default project colors were easily set for each controls color related properties.

Question Layout Decisions

One of the important decisions when designing the screen layouts is determining how the survey questions will be displayed in CATI mode or grouped on a page in CAWI mode. When examining the survey, you'll see the majority of questions will display across modes without any or minimal changes. The remaining will need some modifications, especially for the overall look and feel of the web page including the header, content area and footer. Modifications will include the formatting of questions and question groups such as multi-column, group tables, or other specifics as part of an enumeration.

A multi-column group type is useful for positioning several questions next to each other. Examples of applications include quantity-unit questions where the respondent has to specify both a quantity and the unit, or date of birth questions which include month, day and year.

Figure 4, shows a date of birth question using the multi-column format. Typically there is lead in text for the month question and then instructions for the day and year responses. In a multi-mode environment, prepare directives determine the text to be used. The following shows the month, day and year field definitions using prepare directives:

```

ChildBirthMonth
{ $IFNDEF WebLayout }
"What is ^HHChildrenNames.ChildrenNames.Person[pChildPointer].ChildFName's
birthdate?
  @/@/@/@I[ENTER MONTH.]@I"
{ $ELSE }
  "What is
^HHChildrenNames.ChildrenNames.Person[pChildPointer].ChildFName's birthdate?
  @/@/@IENTER MONTH@I"
{ $ENDIF }
  : TMonth,DK,RF
ChildBirthDay
{ $IFNDEF WebLayout }
"@E[What is
^HHChildrenNames.ChildrenNames.Person[pChildPointer].ChildFName's
birthdate?]
  @/@/@/MONTH: ^ChildBirthMonth@E@|@|@I[ENTER DAY.]@I"
{ $ELSE }
  "@/@/@IENTER DAY@I"
{ $ENDIF }
  : TDay,DK,RF {TI1_31}
ChildBirthYear
{ $IFNDEF WebLayout }
"@E[What is
^HHChildrenNames.ChildrenNames.Person[pChildPointer].ChildFName's
birthdate?]
  @/@/@/MONTH:^ChildBirthMonth @|@|@DAY:
  ^ChildBirthDay @E@|@|@I[ENTER YEAR.]@I"
{ $ELSE }
  "@/@/@IENTER YEAR@I"
{ $ENDIF }
  : TYear,DK,RF {TI1990_2012}

```

An alternative would be to first duplicate the group type questions, then separate the two sets using prepare directives and modify one set for Web display only.

Figure 4. Alternative way to display the fields

Additional layout items for the example include setting the month, day and year types to display as dropdowns and using the internet FIELDPANE Dropdown. The survey specifications include the dropdown view for all modes.

Another group type, group table, has a lead in question followed by several enumeration questions all having the same answer options. This group type is useful for positioning questions below each other in table form such as a series of Yes/No questions.

The multi-mode changes for this group are the same as the multi-column. The Blaise fields used in the layout as shown in Figure 5, were duplicated and differentiated for the web by prepare directives. An AUXFIELD was created to hold the text for the lead in question and given the SHOW method. The SHOW method causes the web page to skip the AUXFIELD text and give the first row in the table the focus.

Has SUSAN lived in any of the following places for at least 2 weeks in a row in the last 12 months?

	YES	NO
a. Camp?	<input type="radio"/>	<input type="radio"/>
b. Boarding School?	<input type="radio"/>	<input type="radio"/>
c. Juvenile Detention Center?	<input type="radio"/>	<input type="radio"/>
d. Mental Health Facility?	<input type="radio"/>	<input type="radio"/>
e. Hospital or Medical Facility?	<input type="radio"/>	<input type="radio"/>
f. Foster Care?	<input type="radio"/>	<input type="radio"/>
g. Any other place?	<input type="radio"/>	<input type="radio"/>

Figure 5. Example of web question using Show method

The final group type we'll look at is the Other Specify. This group type uses a category of an enumeration to associate the input field for the other specify. It is relatively simple to implement and the only decision is whether the text used to 'Please describe:' is contained in the enumeration category text or as the question text of the other specify field. Figure 6 shows both ways of formatting the other specify.

What is SUSAN's relationship to you?

- Biological child
- Stepchild
- Adopted child
- Grandchild
- Niece/nephew
- Foster child
- Brother/sister
- Cousin
- Romantic partner's child
- Other

Please Describe SUSAN's relationship to you:

What is her race? You may choose one or more races.

- White,
- Black or African American,
- Asian,
- Native Hawaiian or other Pacific Islander, or
- American Indian or Alaska Native?
- Some other race, please describe:

Figure 6. Ways of formatting Other Specify

Interviewing Instructions, Differences between CATI and CAWI Modes

In addition to layouts, there are a number of other differences that should be taken into consideration when programming for a multi-mode survey. One difference that needs to be accounted for between CAWI and CATI mode is interviewing instructions. Standard CATI interviewing instructions may not be appropriate for a CAWI. Case formatting of text in CATI indicates whether the text is read to the respondent. If the text is in all upper case, the text is not read to the respondent and if in upper/lower case, the text is read to the respondent.

The simplest way to accommodate the interviewing instructions and upper/lower case differences between interview modes is through a combination of prepare directives and display fields.

An example would be a code all screen that typically has two interviewer instructions. One states ‘CODE ALL THAT APPLY’ and the other states ‘PROBE: Anything else?’ While the first instruction may be appropriate for a CAWI, the second is not. We accommodated the second interviewer instruction by creating a display field and using prepare directives set the field to the appropriate value as shown below.

```

{ $IFNDEF WebLayout }
  dispProbe := "PROBE: Anything else?"
{ $ELSE }
  dispProbe := EMPTY
{ $ENDIF }

```

Figure 7 depicts a CATI code all screen and Figure 8 shows the same question displayed on the web.

What is her race? You may choose one or more races.

CODE ALL THAT APPLY

PROBE: Anything else?

11. White,

12. Black or African American,

13. Asian,

14. Native Hawaiian or other Pacific Islander, or

15. American Indian or Alaska Native?

91. SOME OTHER RACE

Figure 7. Code all screen example in CATI

What is her race? You may choose one or more races.

CODE ALL THAT APPLY

White,

Black or African American,

Asian,

Native Hawaiian or other Pacific Islander, or

American Indian or Alaska Native?

Some other race

Figure 8. Example of code all screen on web

Although the first person is not used frequently in CATI interviews, there might be a question that contains wording such as “I just listed 2 children...” as depicted in Figure 9. The corresponding web text might have “Besides the 2 children you just listed...”, as shown in Figure 10. Just as interviewing instructions were handled in the code all screen above, so too would the question text displays be handled using prepare directives as shown below.

```

{$IFDEF WebLayout}
    AuxScrText := "I just listed 2 children,"
{$ELSE}
    AuxScrText := "Besides the 2 children you just listed,"
{$ENDIF}

```

I just listed 2 children, are there any other children 18 years old or younger who live or have lived in your household part-time under a shared custody arrangement, but did not live in your household for 2 weeks in a row?

1. YES
 2. NO

Figure 9. Example of CATI question phrasing

Besides the 2 children you just listed, are there any other children 18 years old or younger who live or have lived in your household part-time under a shared custody arrangement, but did not live in your household for 2 weeks in a row?

Yes
 No

Figure 10. Example of web question phrasing

As part of interviewer training, the interviewers are taught not to read to the respondent anything that is in all capital letters. This includes enumerated type questions such as YesNo whose answer categories are all capitalized. Some enumerated type questions end in an ellipsis and these types have their answer categories in mixed case. We wanted to make the web survey look as uniform as possible. To that end, when we were close to the end of CATI development we made a copy of the Types file, renamed it, changed all the lettering to mixed case, and used prepare directives to include the new version of the file for the CAWI interview.

```

{$IFDEF WebLayout}
    INCLUDE "AdultWEB_Types.inc"
{$ELSE}
    INCLUDE "Adult_Types.inc"
{$ENDIF}

```

Whereas interviewers are trained to put in a DK response and a comment when a Hard Check occurs and the respondent insists that the data inconsistency is correct, CAWI respondents have no such training. In order to prevent the Web respondent from becoming frustrated and ending the interview early, we have found it advisable to use Soft Edits, where possible, instead of Hard Edits.

In addition, we found there were edits used in CATI that were not appropriate for CAWI. It is important to remember when adding or removing an edit with prepare directives, a RESERVECHECK needs to be added or removed depending on the collection mode. The following prepare directive code shows how to do this.

```

{ $IFDEF WebLayout }
  RESERVECHECK
{ $ELSE }
  CHECK
  IF (MisReportedChild = RESPONSE) THEN
    (ChildFName <> EMPTY)
    "@EMPTY LINE CANNOT BE MARKED AS MISREPORTED.@R"
  ENDIF
{ $ENDIF }

```

Using Help in the Different Modes

The requirements for the project had no Help specified for the CATI and very little for the CAWI. The help requirements for CAWI were to add web links to a contact page, a list of study definitions and list of helpful numbers as shown in Figure 11. These were to be available for the respondent throughout the entire interview.



Figure 11. Example of web help options

Conclusion

Using a central repository for the storage of data from different collection modes worked well. The repository is not an assembly of different databases each bound to a specific collection mode. Rather it contains one database shared among the different modes. Blaise capabilities and features support rapid multi-mode survey development and implementation.

The first step to a successful implementation of a multi-mode survey using Blaise is to understand how Blaise instruments can be programmed to operate in different collection modes and share one SQL database. What worked well was the establishment of a standard data structure that included common and unique fields from each collection mode. The structure was then translated into SQL tables using the Blaise Datalink component.

In addition to a establishing a standard data structure, prepare directives allowed each mode's instrument to have a look and feel appropriate for the collection platform, while not compromising the integrity of the data.

In taking on an endeavor like a multi-mode survey planning is important. The primary items encountered in this case were formatting fields and question text, and identifying critical fields that would be used to refresh text e.g. certain fill strings.

A major lesson learned was make sure the survey specifications take into account the differing multi-mode characteristics. For example, text for a CATI mode should have corresponding text for a web mode. However, projects are not always planned to be multi-mode, and may become multi-mode after the initial programming has been completed. In these instances projects may spend additional time and resources modifying the specifications to accommodate the differences between modes.

Event History Calendar program development and anticipated effects - Main functions and characteristics of the KLI CAPI-EHC

Kimin Kim, Hyomi Choi, Korea Labor Institute

1 Introduction

An Event History Calendar (EHC) refers to a survey method or mode in which events are surveyed in the order or occurrence in a calendar-like format. The greatest difference between the EHC and the Event History Data is that the EHC survey looks like a calendar. Events are recorded on an EHC calendar which is known to facilitate interviews by utilizing interviewee past experience to induce recollection of related events (Robert F. Belli; 2000 etc.). EHCs are most useful for surveying major life events such as education, vocational training, employment, marriage and housing, and are used in order to reduce recall errors between past event time points and thereby increase data reliability. EHC can be used in both paper-based (PAPI) and computer-based (CAPI) formats. The KLI used a paper-based EHC in the 2007 Korean Longitudinal Study of Ageing.¹ In 2010, a computer-based EHC (CAPI-EHC) was developed for KLI use. This paper seeks to describe the main functions and characteristics of the CAPI-EHC developed by the KLI.

2 Main functions and characteristics of the KLI CAPI-EHC

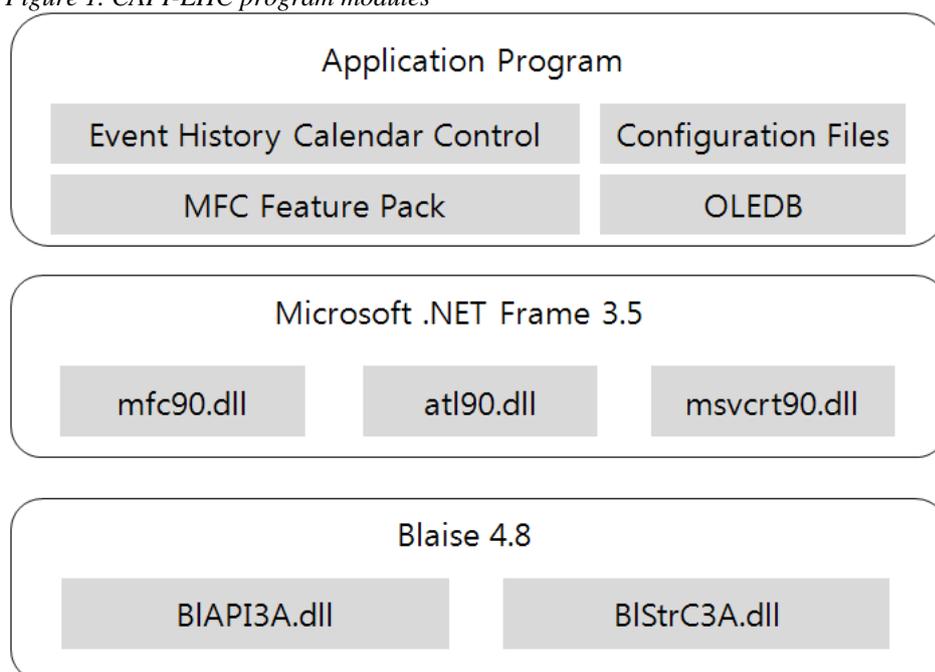
This section looks at three aspects of the CAPI-EHC program developed by the KLI. First, we look at the CAPI-EHC modules; second, we look at the CAPI-EHC screen characteristics; and third, we look at the KLI CAPI-EHC survey design to examine the main functions of the CAPI-EHC.

Our description is based on the fact that the KLI CAPI-EHC was designed on the basis of the life history survey.

2.1 CAPI-EHC modules

Figure 1 shows us the CAPI-EHC modules. The CAPI-EHC program uses three Microsoft.NET Frame 3.5 dll (Dynamic Link Library) files – mfc90.dll, atl90.dll and msvcrt90.dll – and BIAPI3A.dll and BISTC3a.dll provided by Blaise 4.8.

Figure 1. CAPI-EHC program modules



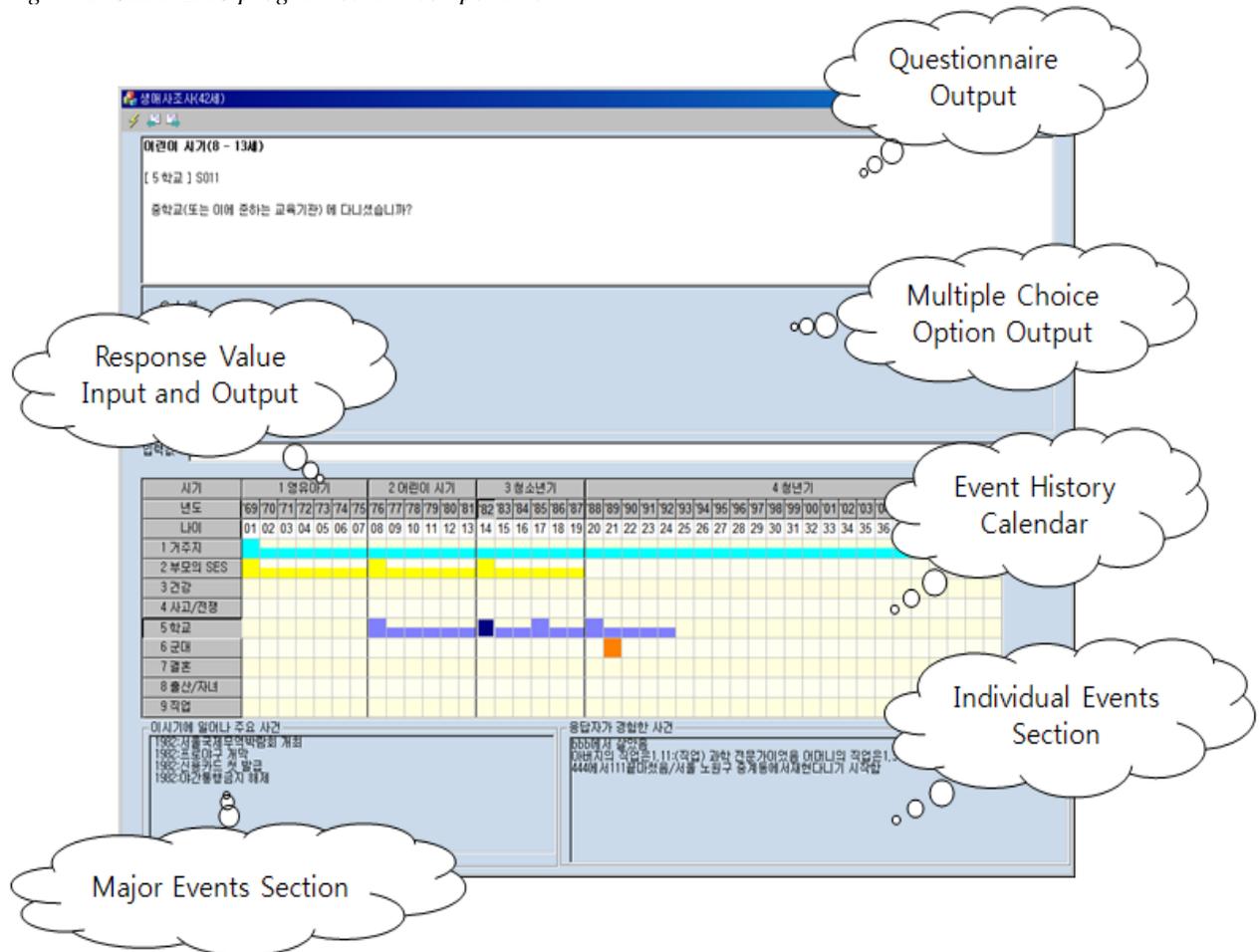
The CAPI-EHC program has 4 components: the Event History Calendar Control component that controls the overall calendar screen; the OLE DB (Object Linking and Embedding Database), an API developed by Microsoft that allows access to various types of databases; the MFC (Microsoft Foundation Class) Feature Pack that facilitates the development of Windows-based applications that have added functions such as the true color toolbar and ribbon menus; and the configuration files. The Event History Calendar Control component allows us to set the size and color of each calendar zone, while the OLE DB lets us link the EHC with various types of databases so that information from other surveys or from previously collected information can be quickly accessed. Meanwhile, the configuration files allow us to reconfigure the program according to the needs of the particular survey. Buttons, toolbars, input windows and frame structures on the calendar screen are designed to enhance interviewee convenience.

2.2 CAPI-EHC screen characteristics

The basic screen composition of the KLI CAPI-EHC is shown in Figure 2. The screen consists of a questionnaire section and an Event History Calendar section, each of which take up about one half of the screen. The questionnaire side has controls for questionnaire output, multiple choice option output and response value input and output, while the Event History Calendar side consists of an Event History Calendar where responses are shown on the calendar, a major events section for a particular year and an individual events section for the year.

As the questionnaire section is not very different from the existing CAPI questionnaire screen, we choose to look only at the Event History Calendar section. As mentioned above, the Event History Calendar section consists of an Event History Calendar, major events from a particular year and individual events for the year. Figure 3 shows us the Event History Calendar which can be adjusted to align the start point and the end point of the calendar display to the actual timeframe addressed in the survey. For life history surveys, interviewee convenience can be enhanced by separating the response periods according to the life periods of early childhood, childhood, adolescence and young adulthood.

Figure 2. CAPI-EHC program screen components



If multiple themes are being surveyed simultaneously, the various themes can be displayed together on a single screen with different colors used for each theme for easy recognition by interviewees. The whole box is colored in for the start and end date of an event, while only half of the box is colored in for dates in between. If the cursor is brought to an event start or end box, a tooltip immediately appears with information on the contents of the event; and if the year is clicked, a window with a submenu for the months of the year pops up.

In addition, the screen is programmed to maximize recollection by interviewees by allowing, for example, the user to simply click on the box and move directly to another input screen if the interviewee starts to provide information on marriage or jobs while responding to a question on housing.

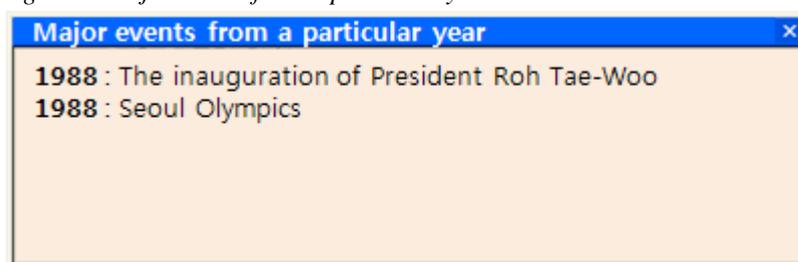
Figure 3. Event History Calendar

Moment	1 Babyhood							2 Childhood							3 Adolescent							4 The Delusion of Youth																					
Year	'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	'03	'04	'05	'06	'07	'08	'09	'10	'11
Age	01	02	03	04	05	06	07	08	09	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
1. Residence																				I moved to Daechi-Dong from Haengdang-Dong.																							
2. SES	[Red]							[Red]							[Red]																												
3. Health																																											
4. Accident																																											
5. School								[Purple]							[Purple]																												
6. Army																																											
7. Marriage																																											
8. Children																																											
9. Occupation																																											

Major events from a particular year are displayed in the lower left side of the screen, while individual events for the year are displayed in the lower right side. Figure 4 shows us an example of major events in a year. If – as in the example below – the interviewee is providing information for the year 1988, the screen displays major events such as the inauguration of President Roh Tae-Woo and the Seoul Olympics in order to help them recall memories of that year. Previous responses by the interviewee are drawn upon to display personal individual events that occurred in the year in the lower right corner.¹

This feature of the CAPI-EHC takes full advantage of the merits of being a computer-based tool, whereas it was not easy to automatically display major events or individual events from a particular year in the PAPI-EHC.

Figure 4. Major events from a particular year



Because there is a limit to the number of events that can be displayed on a single screen, the CAPI-EHC allows users to choose characteristics for the major events that occurred in the year that they wish to have displayed on screen and thus lessens the scroll burden. For instance, if the interviewee is not interested in sports, all sports events can be deselected and hidden from view, while still allowing the interviewee to look at cultural and scientific events from the particular year.

¹ The individual event display is not shown here as it is not very different from the major event display.

Figure 5. Major events selection



2.3 CAPI-EHC major functions

This part of our paper focuses on describing the KLI CAPI-EHC program configuration in order to explain more about the main functions of the KLI CAPI-EHC program. The KLI CAPI-EHC is configured, first, in terms of screen components, second, the display of inputted questionnaires within the calendar and, lastly, linkages to previously obtained information or other non-EHC questionnaires. The KLI CAPI-EHC program is designed so that users can easily customize the program to the purpose of the survey by making changes on a single configuration file.

2.3.1 Screen configuration

The KLI CAPI-EHC can easily link chosen Blaise Database (questionnaire) files to the EHC program and allows users to very easily set the format of the calendar. The length of the survey period (i.e. years, months, days displayed on screen) can be set in the top row, while themes such as jobs and health are configured in the first column. Configuration instructions are outlined in Table 1

Table 1. Basic screen configuration

	Example
i) Assign the CAPI-EHC database	Blaise database=lifehistory.bdb
ii) Assign the color of screen	Color 0=15391434
iii) Assign the number of years	YearsOnSheet=49
iv) Assign the number of rows in EHC	NumberofRows=9

The KLI CAPI-EHC allows configuration of calendar themes and also allows simple configuration of the colors for each theme. Interviewee convenience is also enhanced as the year setting at the top of the screen can be set to display specific decades such as the 1970s or the 1980s or specific life periods such as early infancy or early childhood.

The program used for these purposes is presented in Table 2. For instance, simply changing the '16776960' in "SelColor 1=16776960 (Raw4/SkyBlue)" will change the color for a certain theme.

Table 2. Calendar screen composition

	Example
Initial Header	InitialHeader=Life History
i) Assign the Label of Row(default)	Row Label 1 = Moment Row Label 2 = Year Row Label 3 = Age
ii) Assign the Label of Row	Row Label 4 = Residence Row Label 5 = Health Row Label 6 = Occupation
iii) Assign the Color of Row	SetColor 1=16776960 (Row4/bright blue) SetColor 2=65535 (Row5/yellow) SetColor 3=255 (Row6/blue)
iv) Assign the Label of Column	Col Label 1=1 babyhood Col Label 2=2 childhood Col Label 3=3 Adolescent
v) Assign the number of columns	Col Size 1=7 Col Size 2=6 Col Size 3=6

2.3.2 Calendar display of inputted questionnaires

The KLI CAPI-EHC program allows users to configure the calendar start and end dates and the questions and responses to be displayed on the calendar. Since EHC questionnaires generally repeatedly ask the same questions, users can also configure the maximum number of repetitions in the program design.

In this process, there are some fields that are required in the questionnaire to be able to link the Event History Calendar and the Blaise program, the array for which is set forth in Table 4. As the program will not work if these required variables are absent, particular care must be taken in the design of the Blaise questionnaire (refer to figure 6).

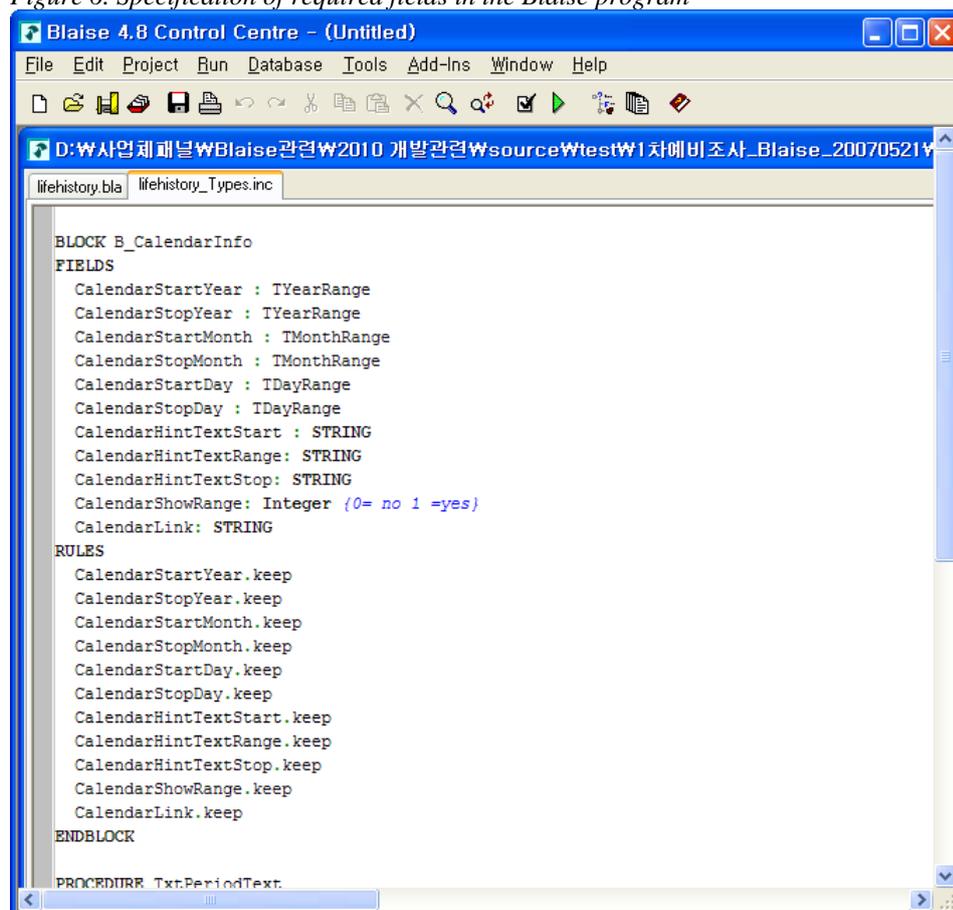
Table 3. On-screen questionnaire content components

	Example
i) Assign the Blaise database	User Name=preload_name
ii) Assign the Primary key	Primary Field=Bid
iii) The beginning question in EHC questionnaire	Start Question=IntroCalendar
iv) The end question in EHC questionnaire	Stop Question=Comp_stat
v) The beginning year set in BDB	BirthYear=preload_birthday
vi) Set up the number of loop in repeated question	Row 1 Loop=20 Row 2 Loop=40

Table 4. Required fields

	Subdivision	Description
Calendar Info	CalendarStartYear	The year of beginning
	CalendarStopYear	The year of end
Calendar Info	CalendarShowRange	
	CalendarHintTextStart	
	CalendarHintTextRange	
	CalendarHintTextStop	
	CalendarLink	Set up the linked field name

Figure 6. Specification of required fields in the Blaise program



2.3.3 Linking external data

EHC surveys help interviewees recall past events by displaying events in calendar format, and thus help to reduce recall errors and enhance the quality of data. Recall errors will decrease further if, along with the display of responses in calendar format, past information or responses to other questions are utilized to check for consistency.

The KLI CAPI-EHC has the additional functionality of being able to link with other data to block errors. This program was developed in consideration of linkage speed so as to ensure that the overall flow of the survey is not disrupted by linked data. Table 5 describes how to link external data to the CAPI-EHC program.

Table 5. How to link to other databases

	Example
i) Link the Database	DataBaseConnection=Provider=Microsoft.Jet.OLEDB.4.0;DataSource=Preload.mdb
ii) Assign Table Name	TableName=OldRespondentList
iii) Assign Table ID	Field 1=FID
iv) Assign EHC Blaise ID	Blaise 1=BID

3 Summary and conclusion

So far, we have discussed the CAPI-EHC program developed by the KLI.

Used mostly to survey major life events such as education, vocational training, employment, marriage and residence, the EHC increases readability to reduce errors in recalling time points at which past events occurred, and thus improves data reliability; but the design and data entry process of EHC is complicated, and requires a lot of time and money.

These weaknesses are expected, however, to be significantly alleviated with the introduction of CAPI,

as CAPI surveys characteristically require a lot of time and money in the early development process, but involve lower cost burdens in later years. Also, since CAPI data input and output occurs simultaneously with fieldwork, the input/output burden is that much lower. In particular, since error checks occur on a real-time basis during fieldwork in the CAPI-EHC, we are able to collect more consistent life event data.

The KLI CAPI-EHC program was developed with a focus on ensuring fast operation speeds, easy links with pre-collected information or other CAPI programs, and an easy-to-configure EHC program that can be customized to the questionnaire and survey design. The KLI CAPI-EHC program allows the 7 basic variables required for calendar composition to be specified in the Blaise questionnaire file, and is designed so that it can be quickly utilized in surveys just by designating the program title, calendar format and design and external linked files within the configuration file.

The KLI CAPI-EHC system is a very practical and useful program that has been developed on the basis of know-how accumulated by KLI panel researchers in the course of conducting numerous panel surveys, and which brings together the strengths of CAPI and EHC. This CAPI-EHC program has not, however, been used in the field so far, and is therefore limited in terms of verification of the program's stability and effectiveness. Through future cooperation with other panel surveys and through further improvements to the program, we hope to achieve higher quality and further progress in life event surveys and recalled data.

The Application of Blaise CARI in Chinese Family Panel Studies

Shuang Yu, Jiahui Yao, Peking University

Abstract

There are 4 kinds of questionnaires in Chinese Family Panel Studies 2011 (abbreviated as CFPS2011) includes community, family, adult and child questionnaire. To supervise the activity of interviewers and to guarantee the quality of survey data, we hope each question in the questionnaire can be recorded by sound.

In CFPS2010 we used an external record program. Interviewers terminated the record program according to the answer of respondent (when the respondent said she/he don't want to be recorded anymore). But there are several problems. Firstly, interviewers are easy to make a mistake of operation. Secondly, the record program can not terminate automatically as the questionnaire ends. Finally, all the questions are recorded to one sound file in a questionnaire, it's difficult to locate the question we want. To solve such problems, in CFPS2011 we used Blaise CARI.

This paper presents the solution of CFPS2011's CAPI questionnaires integration with Blaise CARI, and discusses the challenges that we meet when using CARI in CFPS2011, and also discuss the questions we still have in using CARI.

1. Background

Chinese Family Panel Studies (CFPS) is aimed at three levels data collection, including individuals, families and communities. The project studies the changes of society, economy, population, education and health, which is reflected by the collected data. It is a key social science project that serves academic research and policy decision supporting. The first phase of the survey is 12 years (2008-2020). CFPS is early funded by phase II of 985 plan of the Ministry of Education, beginning preparations in 2005, setting up an institution in 2006. In 2007, preliminary work completed, including two test surveys, 140 families totally, in Beijing, Hebei, Shanghai, and the questionnaire tools basically mature. Exploratory survey in 2008 in Beijing, Shanghai, Guangdong, expanded scale to 2400 families. Tool testing follow-up survey in 2009 in Beijing, Shanghai, Guangdong, tested the stability and reliability of CAPI technology, real-time management techniques for survey process, real-time technical support system for survey process, real-time data quality monitoring techniques. In 2010, the survey all over China (Tibet, Qinghai, Xinjiang, Ningxia, Inner Mongolia, Hainan, Hong Kong, Macao and Taiwan are not listed) was formally implemented, the scale was up to 16,000 families, and a follow-up survey was implemented annually. Every year from 2010, field surveys are being conducted from March to July, data is being cleaned up from August to October, and from November to February of next year, provide the cleaned data for the teachers and students in Peking University and partner universities, write the key indicators report "China Report", which is published by Peking University Press. The key indicators report and data is published to public in March. There are 4 kinds of questionnaire in CFPS, including community, family, adult and child questionnaire. The community questionnaire is used to obtain the household residence of the basic environmental information; family questionnaire is used to obtain family economic, demographic status information; adult and child questionnaire is used to obtain the individual social, economic, educational and health information.

In order to ensure data quality and supervise the behavior of the interviewers, the interview process needs recording. In 2010, we used an external recording program. Its advantages are respondents who encountered sensitive issues can deny the recording and the recording program can be interrupted at any time without affecting the interview. Recording program's switch can be controlled separately. When the program starts another time, a new audio file will be generated. But the drawback of this approach is as following:

There is no way to record one by one field, and finally the entire interview generates a single audio file with more than one field. So it is unable to quickly navigate to the recording position of some

field, which causes great inconvenience in the data quality checks and takes more time to listen to some field in need of verification.

If Interviews forget to turn off the recording program, it has been recording, which results in larger files that take a lot of space, and the file uploading process will be relatively slow.

2. Introduction to CARI

Computer Audio Recorded Interviewing (CARI) is a technique that records the conversation between the interviewer and the respondent on the computer during an interview. The Blaise Data Entry program (DEP) supports CARI. CARI does not require interviewer intervention for starting or stopping the recording in the DEP. Besides sound recording and screen capturing during the data entry process, the DEP can also play back the recorded files.

The system offers various settings to influence the recording process, for instance which questions to record, it allows respondent consent to be given or revoked at any time and it offers ways to specify the naming of the sound files that contain the recordings. The CARI settings can be specified in a CARI settings file. In this CARI settings file the recording process can be configured such that neither the respondent nor the interviewer knows exactly which portions of the interview are being recorded. Blaise CARI is an important product, which has following advantages: audio files can record questions one by one which means it can generate an audio file for each question, this is very convenient to search; it has several formats for record file including wav, wma and mp3; audio files' naming is flexible, it can be named as sample's key value and question field's name; audio files' storage directory can also be specified in the setting file; it can also choose questions to record, not all questions, for saving disk spaces.

3. Application of CARI

At CFPS in 2011, the Institute of Social Science Survey at Peking University (ISSS) decided to apply CARI. The Blaise DataModel file in CARI is same as in other Blaise products, e.g. CAPI. After the completion of DataModel file for questionnaire's logic, should configure CARI for the questionnaire. This paper presents the configuration and the CARI's application in CFPS family questionnaire.

Run Blaise Control Center, open the CARI configuration window through menu *Tools->CARI Specification*. Enter the node *Audio->Record->File Parameters* in the left tree, shown in Figure 1. There are two parameters can be set in the right panel: *File Name* and *Path*. *File Name* parameter specifies the audio file's naming style, its default value is $\$KEYVALUE+\$FIELDNAME$, which means the file will be named by sample's key value and question's field name. *Path* parameter specifies the questionnaire audio files' storage directory, its default value is $.\backslash\text{SoundRecordings}$, which means the audio files will store in a subdirectory named as *SoundRecordings* under the questionnaire's data directory.

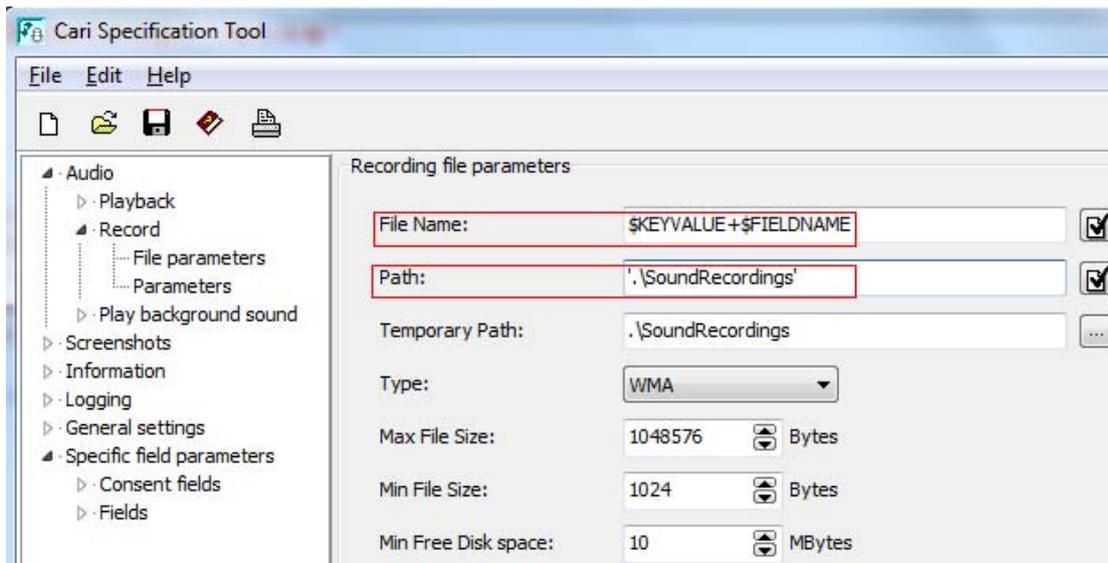


Figure 1. CARI Audio Record File Configuration

Next, enable CARI recorder. Enter the node *Audio->Record->Parameters* in the left tree, shown in Figure 2. The default value of parameter *Enabled* is false, set the value to true to enable the recorder. The parameter *Max. nr of fields* specifies the maximum number of fields to record, its default value is 40. If the recorded fields' number exceeds the parameter's value, the CARI recorder will stop. Set the value to -1 means the number of fields is unlimited.

Other parameters can all be with default values, and more information can be got through F1 help. After all the settings are specified, save them as a .bci file.

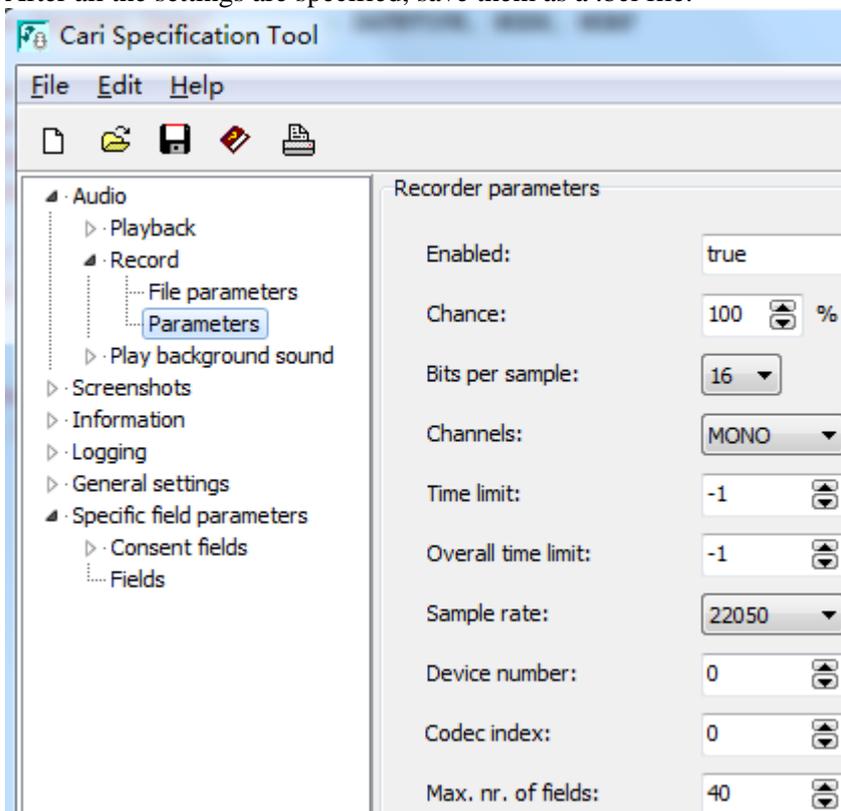


Figure 2. CARI Recorder Parameters Specification

Two approaches to run CARI, one is to run with Blaise Control Center. This approach should set the DEP's run parameter: CARI settings file, shown in Figure 3. Run Blaise Control Center, open the *Run*

Parameter window through menu *Run->Parameters*. Enter the panel *DEP->Files*. Specify the parameter *CARI setting file*. Set its value as the Blaise CARI specification file (*.bci) that is saved previously.

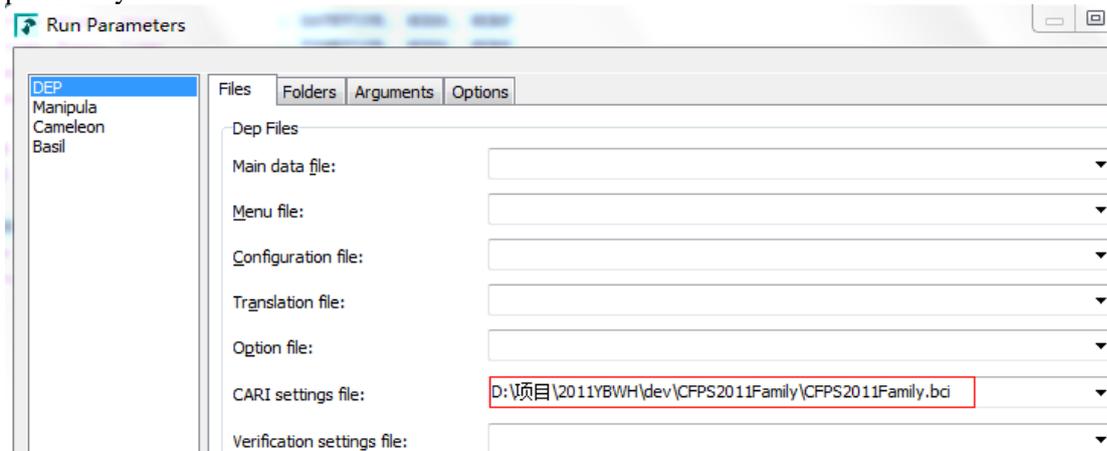


Figure 3. Specify CARI Settings File

Additionally, need to enable CARI. Same as specifying CARI setting file, enter the panel *DEP->Options*, check the option *Enable CARI* and *Record*, as show in Figure 4. After above setting is completed, run DEP in Blaise Control Center, then CARI recording is started at the same time.

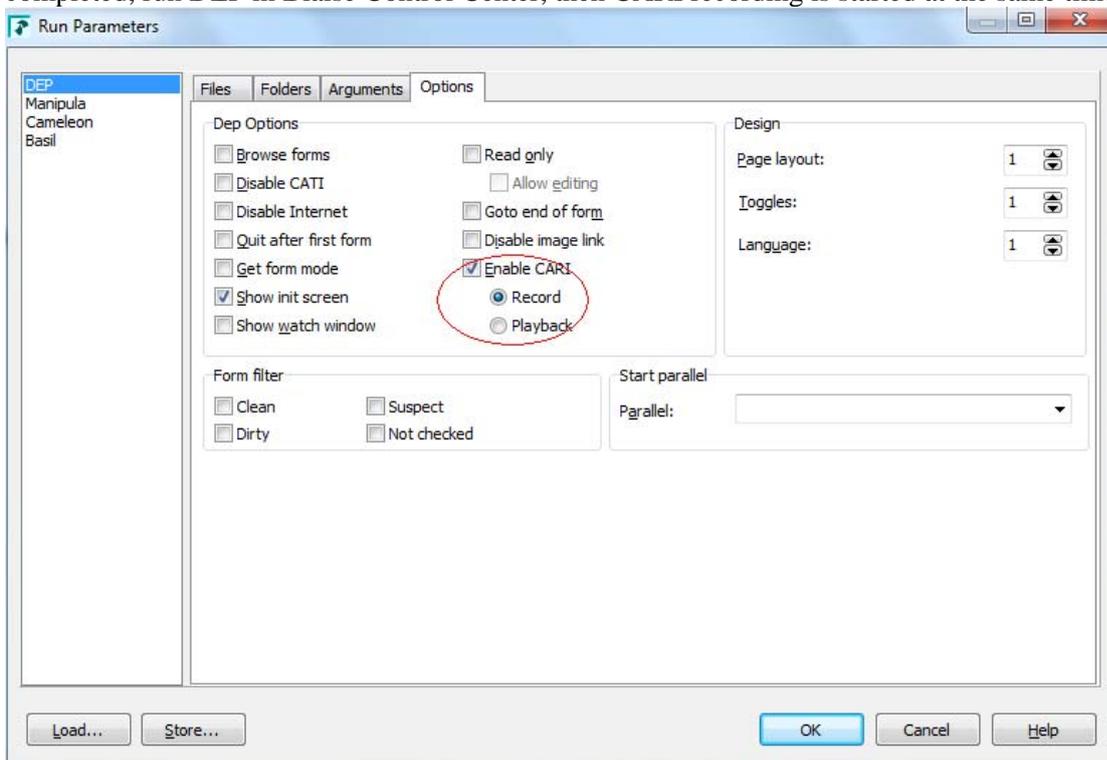


Figure 4 Enable DEP with CARI

Another approach to run CAI is to set DEP's command parameter with CARI relative parameters. An example is as following:

```
C:\blaise\blaise48\DEP.EXE C:\BLproj\CFPS2011\CFPS2011Family\work\CFPS2011Family.bmi
/MC:\BLproj\CFPS2011\CFPS2011Family\work\CFPS2011Family.bmf
/CC:\BLproj\CFPS2011\CFPS2011Family\Work\CFPS2011Family.diw
/EC:\BLproj\CFPS2011\CFPS2011Family\work /WC:\BLproj\CFPS2011\CFPS2011Family\work
/BCI=C:\BLproj\CFPS2011\CFPS2011Family\work\CFPS2011Family.bci /CARI=RECORD
#C:\BLproj\CFPS2011\mm\blaise.tra /X /K
```

The first approach as development mode applies to questionnaire programmers. It can be used in the debugging process. And the second approach as production model applies to the actual course of survey.

4. Problems

After the application of CARI, we hope that CARI should make some improvements. When respondents who encounter sensitive questions refuse to record, CARI should be paused. Because the audio files record questions one by one, if some question's time is very short and the interviewer goes on next question, then question's recording will be incomplete. So the audio file should also records previous and next questions' recording.

5. Summary

During the survey of CFPS in 2011, ISSS applied Blaise CARI. The application achieved that audio files record fields one by one and facilitates inspectors to locate each question. Practice has proved that the application with CARI compared to approach with external recording programs, which greatly facilitates the inspectors for verification, and improve the work efficiency of the inspectors. CARI is running with interview at the same time, the recording automatically closed at the end of the interview. So the interviewers do not need to concern the recording is turned on whether to stop, and thus also to some extent it improves the work quality of the interviewers.

Using Audit files to get the time of each question (TIEQ) in China Family Panel Study (CFPS)

Yongjian Zhang, Shuai Sun, Jiahui Yao , Peking university

Abstract

To verify the work of interviewers and ensure the high-quality interview of the survey, we want to know all the operations of interviewers during the interview. With the Blaise system all the interviewer's operations are recorded in audit trails files. We can get a lot of para data through analyze audit files, it contains a so many information that we can't use these files directly. So we should develop special softwares to help us analysis audit trails files.

In China Family Panel Study(CFPS),we use audit files to compute the answertime of each question(TIEQ). With TIEQ data we can do lots of analysis, such as verify the completion of the work of interviewers, determining the Interviewer's remuneration and labor costs, providing the basis for judging whether this sample need to re-visit and so on.

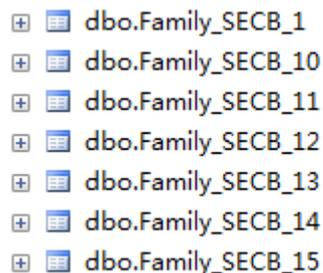
This paper describe the solution of getting TIEQ from audit files and challenges we encountered. And also discuss the future use of adt files in helping the analysis of para data.

1 Create Sqlserver Table

In China Family Panel Study(CFPS),our development environment is .net and Internet Information Services, So we had to use the sqlserver database.Now Our data analysts want to read data directly from sqlserver,but not bdb files or sas, so we need to copy the bdb data to sqlserver database. At the beginning, we tried to use the Blaise ole db workshop, but The effect is not very satisfactory, so we developed an application to implement this, fields in the sqlserver table and fields in the bdb files are one-to-one correspondence.As we know, in sqlserver the maximum number of a table is one thousand, Unfortunately, our questionnaires have several thousands, even more than ten thousands fields, in this case, the Ole db workshop will not be able to meet our needs. So we had to find something else to solve this problem

Now,we have several questionnaires in one project, we create the sql server tables named questionnaire name and '_' and block name,for example, the questionnaire named family,there are many fields named SecA.QAName, SecA.QAAge, then we can create a table named Famliy_SecA, QAName and QAAge are the properties of the table. In this way, it will be well organized.

picture p.1 show the sqlserver struct



```
dbo.Family_SECB_1
dbo.Family_SECB_10
dbo.Family_SECB_11
dbo.Family_SECB_12
dbo.Family_SECB_13
dbo.Family_SECB_14
dbo.Family_SECB_15
```

p.1.1

2 The Application named HelloBlaise

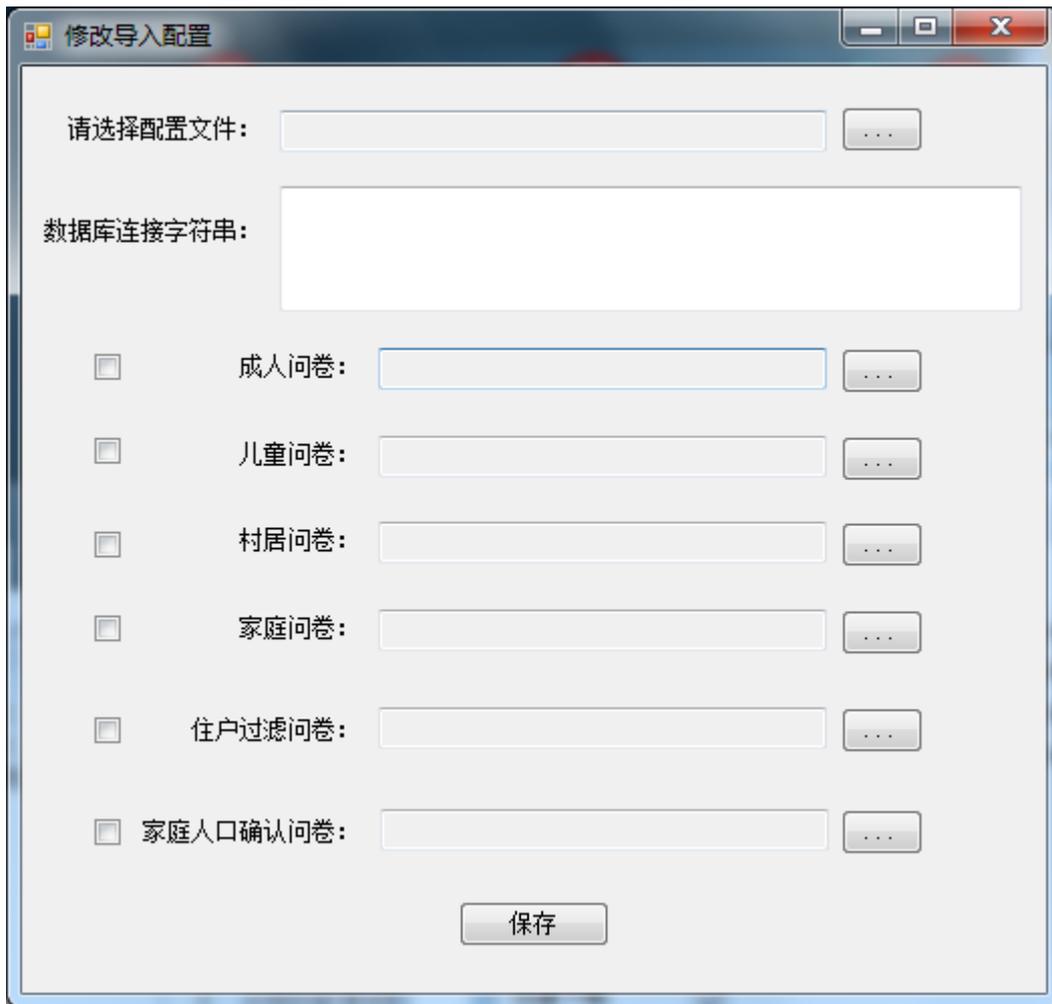
In developing, we use an application called hello blaise, this application can get all the fields contained in bmi files, and you can also list the fields that you did not want, the application will remove them. Despite this, we still have a problem, there are so many fields, should we write them



p.1.4

5 Audit trail files' path config tool

As shown in p.1.4, we have six questionnaires, the truth is the audit trail files are stored in the Hard disk in different paths, if we write these paths in the source codes, should we copy the files to the path that we write in the source codes when we use this import application? How do we control this? Don't worry, we have another tool, use this tool, we can choose the path that the audit trail files be stored, also we can set the sql server database.



P.1.5

6 Summary

The Blaise System open many interfaces, That we can use its dll files ,then we can do lots of things by ourselves,we can development lots of applications to assist our survey ,that's amazing.

The main purpose of our survey is to provide survey data with a national sample, regarding various aspects of social phenomena, different scholars or organizations may use this data to help them to do some research ,when the government can launch policies they can refer this data. As the Data gatherers, we have an obligation to ensure the survey data is maintained correctly.so we collect TIEQ, to assist the checks to check the interviewers' work.

Social Data Collection Transformation Project: Transforming CATI data collection and data processing for the Labour Force Survey

Michael Hart, Office for National Statistics

1 Overview

The Labour Force Survey (LFS) is one of ONS's most important household surveys. It is a survey focused on collecting data on the employment circumstances of the UK population. It is the largest household survey in the UK and provides the official measures of employment and unemployment. The survey provides information needed to help the Government decide its economic, especially labour market, policy. Key external users of the data include the Department for Work and Pensions (DWP), the Department for Business, Innovation and Skills (BIS), the Department for Children, Schools and Families (DCSF), the Welsh Government, the Scottish Government, the Bank of England and HM Treasury.

The Social Data Collection Transformation (SDCT) project aims to deliver new data collection systems for the LFS which future projects can use as templates for other major household surveys. The project has 2 tranches, which will be developed concurrently:

- Tranche 1 aims to redevelop the “core” data collection systems for the LFS - associated with a) case creation (and delivery to the existing communications infrastructure), b) operational case management within the HQ systems, and c) passing data through to the statistical operation.
- Tranche 2 aims to redevelop the LFS Telephone Operations Call Scheduler (TOCS)

For both tranches, the project will aim to deliver the operational management information (MI) required to manage the data collection process. The project will deliver the data outputs fed into the current strategic management information systems to allow these to continue to function.

Operational and strategic MI are defined as follows:

Operational	<p>The information required during data collection to:</p> <ul style="list-style-type: none">• monitor the collection process• decide on reallocations during collection• decide on reissues into a following collection period• change the data collection strategy for the current or following field period <p>Current operational MI for the LFS is heavily focused on monitoring progress in terms of the cases that have been started and completed (and the response rate) within the field period, and managing reallocation and reissues. Operational MI is required for running the telephone operation (TO).</p>
Strategic	<p>This covers the information required to monitor the overall performance of the social survey data collection operation, with the aim of identifying opportunities to improve the efficiency and quality of the operation and/or identify change initiatives (e.g. recruiting additional interviewers, altering the relative priorities of surveys, changing the call pattern strategy). It also covers Reports used by managers to monitor the performance of individual interviewers or the wider field force/TO e.g. for performance management.</p>

The data collected is currently processed from the proprietary Blaise database into the Dbase management system and the underlying .dbf file formats for operational (data collection) purposes, and (currently via SPSS) into the new Oracle repository for statistical processing. The operational applications used in the current LFS data collection systems (e.g. for case creation, case management, telephone call scheduling, the rotation of case and survey data, etc) are mainly written in the CA-Clipper™ language, which runs on the Microsoft® DOS operating system on personal computers .

2 Scope

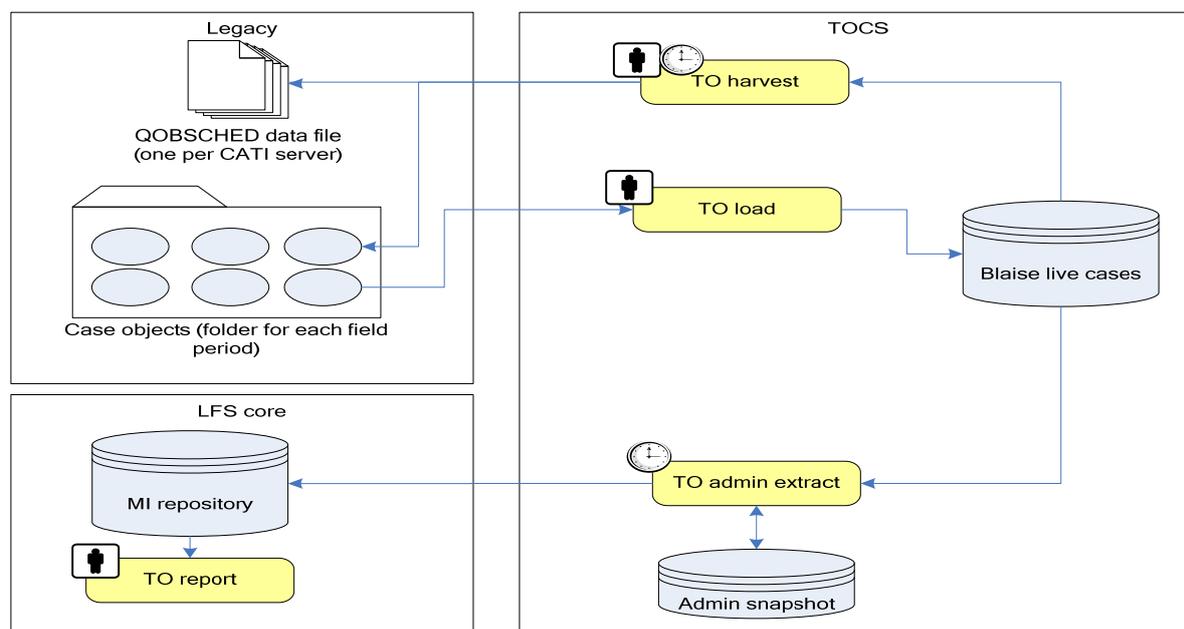
The project defined the following as within scope:

- Telephone Operations LFS Call Scheduler (TOCS).
- Core LFS Systems. This will deliver the following
 - A respondent data store which stores identifying and survey data securely,
 - Integration with new case management applications (Case creation and delivery to the existing communications infrastructure for field and CATI servers for TO; processing completed cases from both modes of collection; rotation of case and survey data into the next wave of interviewing)
 - Data cleaning in the form of validation and editing and produces defined outputs (e.g. passing data to the statistical processing repository, output of operational management information reports, data outputs for existing strategic management information systems, storage of management information data, etc).
- A platform to support the implementation of other major household surveys.

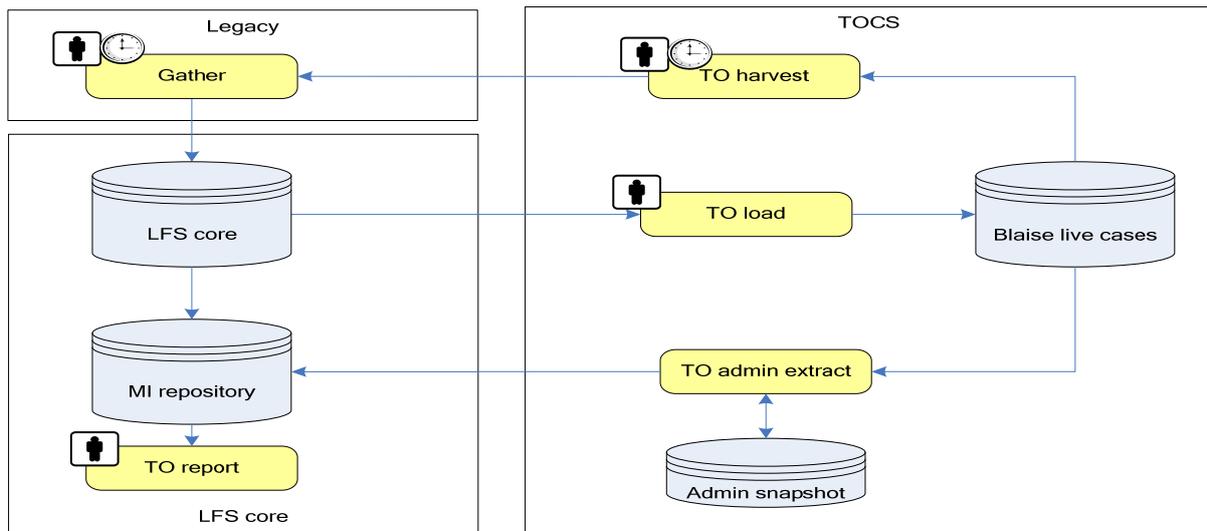
The following diagrams give a high level indication of:

- The proposed parallel development tranches
- The scope of each tranche
- Some of the key interfaces to legacy systems (sampling, allocation, existing communications and case management systems for the field force, etc)
- Some of the outputs, e.g. management information

Tranche 1 Integration



Tranche 2 Integration



3 Business Objectives

ONS Social Survey aims to deliver the following high level objectives:

Enhance the effectiveness of our surveys through better organisation and stronger programme and project management, through smarter design and a better understanding of users needs;

Identify, test and implement further options for increasing the efficiency of survey operations.

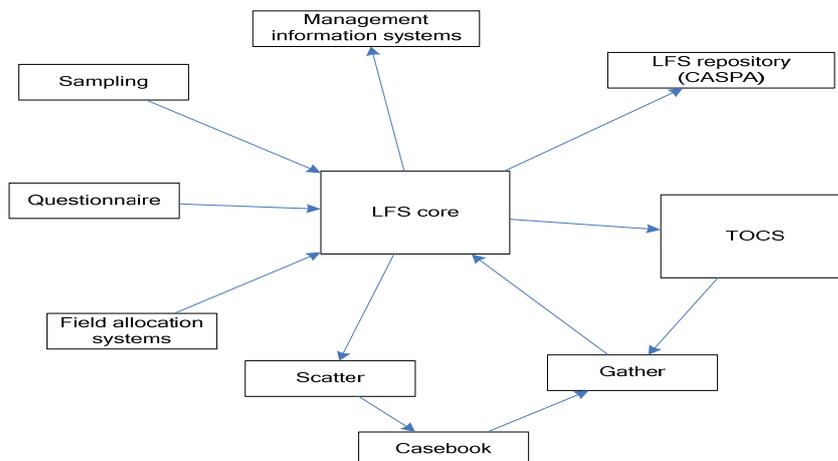
The project has two main aims:

Aim 1: To eradicate Clipper from the LFS core systems and LFS Call Scheduler.

Aim 2: To modernise and standardise systems, deliver a template for standard Social Survey Data Collection.

4 Interfaces

The following diagram is a representation of the interfaces from a business perspective for the project



5 Business Change Implications

5.1 Overview

The project will provide solutions implying a number of changes to the business. The key elements are:

1. An off-the-shelf call scheduling product that will require changes to the business process (e.g. less intervention in the default scheduling, a different way of managing appointments with LFS respondents, etc)
2. A reduction in overall staffing levels supporting the LFS operation on both the business and Information Management (IM) application support sides.
3. A reduced need for IM application support to be involved in running the business process.

As the project involves two parallel development tranches or streams, this section aims to identify where changes are implied by a particular tranche/stream.

5.2 Changes to job specifications

The LFS core systems development implies some changes to details of job specifications for Field Office and Survey research team staff, due to the objective of reducing the need for IM application support involvement in the business process (particularly in the preparation for the quarter).

5.3 Training and Communication

TOCS: While there is generally a desire for change, the nature of the TO organisation can make some parts of the change more difficult and so it will need careful planning to work with the interviewers. To mitigate this risk:

- A communication strategy has been developed, which includes information on a range of methods that will be used to communicate with staff;
- There will be a training work package within the project - to ensure business users receive adequate training before customer acceptance testing and implementation

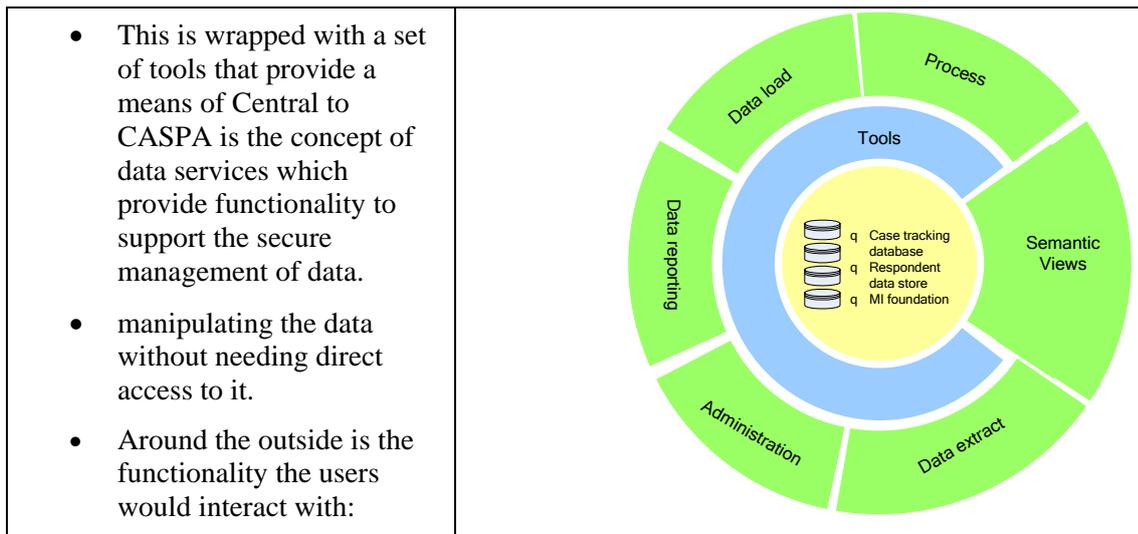
Business Process Change

Changes to business process will not be fully determined until the business has had some exposure to the solution, but there is an expectation and acceptance that there will be a need to change business processes to meet the constraints of the solutions being delivered. The business also intends to use the opportunity to address the need for some of the current business processes that have 'evolved' over time.

The TOCS development in particular implies a number of areas where the business process will change, as the intention is to implement a commercial off-the-shelf (Blaise CATI) solution to replace a bespoke solution. Processes expected to change in the telephone operation include evaluation of the flow of work through the telephone operation and the interviewing process itself. The LFS questionnaire will also need to change to accommodate the proposed solution which will have a small impact on processes associated with questionnaire design, testing and implementation into live production.

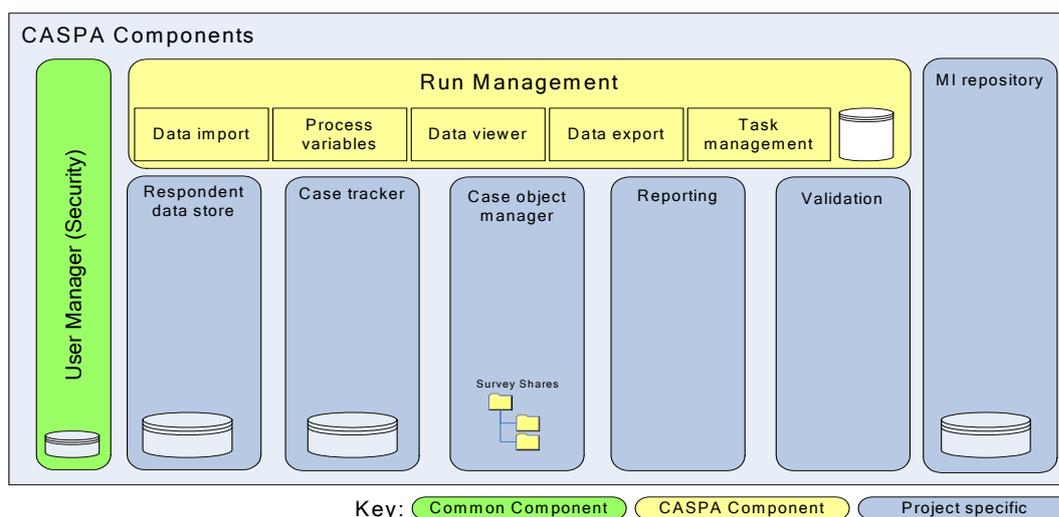
6 LFS Core Solution

The LFS core part of the solution is responsible for coordinating the work done during the social data collection process. The project is intending to deliver this functionality as a CASPA application. CASPA is a standard application architecture used within ONS to deliver various statistical processing applications. The following diagram provides a logical view of CASPA:



6.1 Component Model

The following diagram shows the proposed component model for the LFS core application.



6.1.1 User Manager

Common component used by multiple applications to provide user management functionality.

6.1.2 Run Manager

Existing bespoke CASPA component which is responsible for the orchestration of any processes needed to perform the business process. The run management component currently supports a number of features:

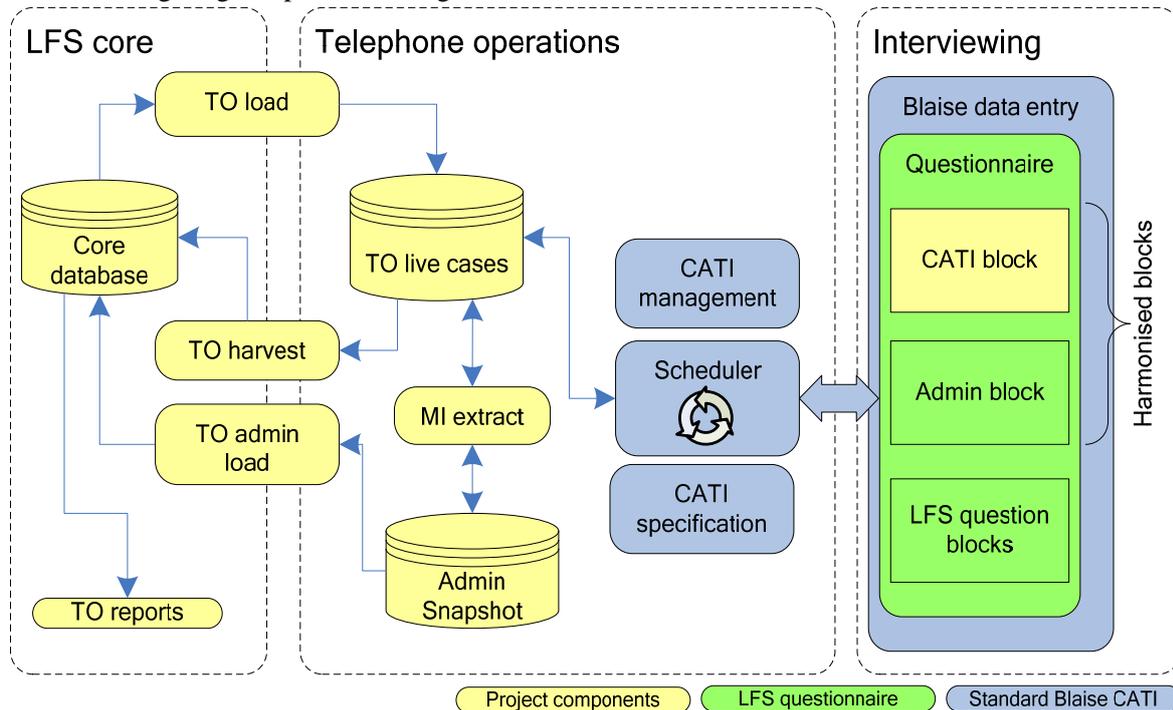
- The ability to sequence individual steps
- The ability of a step to invoke system functions provided by other components. For this project this includes:
 - Constructing the list of cases to be interviewed (including the rotation process).
 - Building case objects for FTF interview
 - Managing the data cleaning process.

- The ability to loop through a series of steps (not expected to be needed for this project)
- The ability to allow a user to enter properties which would be passed as needed to the individual functions.
- The ability to allow the user to start and stop a process.
- The ability to report status of a process.

The essential functionality to be provided by this component is the coordination of data flows between the various components of the solution.

7 TOCS Solution

The following diagram provides a logical overview of the TOCS solution:



8 Integration to legacy components

The following diagram shows the key systems which need to work together to deliver an end-to-end collection process and also shows the key interfaces between the various systems.

