Comparative Assessment of Software Programs for the Development of Computer-Assisted Personal Interview (CAPI) Applications

July 2011

Table of Contents

ACKNOWLEDGEMENTS

This publication was prepared by Arthur Shaw, Lena Nguyen, Ulrike Nischan, and Herschel Sy of the IRIS Center at the University of Maryland, with assistance from Margaret Richards.

However, a report of this length and breadth is never really undertaken by its authors alone. Therefore, we would like to thank several people for their invaluable contributions.

For funding, inspiring, and guiding the software comparison project, we would like to thank the staff of the Living Standards Measurement Study team at the World Bank. In particular, we would like to thank Talip Kilic for supporting, coordinating, and managing this effort; Gero Carletto and Kinnon Scott for directing and guiding this project through their valuable feedback; Kathleen Beegle for her insightful CAPI questions before and during this project; and others on the larger team for reviewing and commenting on an early draft of this report.

For their informative explanations and helpful clarifications, we would like to thank representatives from the software reviewed in this report: Jim O'Reilly from Westat, the providers of Blaise technical support for North America; Jeff Royal from the University of California Berkeley's Social Science Computing Lab, the developers of CASES; Julio Ortuzar and Ruben Hume from Serpro, the developers of CSProX; Nicolette Chin-Shue and Rodel Flores from Techneos, the developers of Entryware; Bart Orriens and Bas Weerman from RAND Corporation, the developers of MMIC; Yaw Anokwa and Mitchell Sundt from the ODK development team at the University of Washington in Seattle; Thomas Roth and Ivan Phillips of Pendragon Software; Joachim De Weerdt and Louise Broadbent from Economic Development Initiatives, the developers of Surveybe; and, although the software was not included in this review, Carmita Signes of Nova Research Company, the makers of QDS.

For their invaluable insights from the field, we would also like to thank the users of CAPI software with whom we spoke: Neil Hendrick from the Human Rights Center at UC Berkeley; Clayton Sims and Anton DeWinter at Dimagi; Marcos Vera-Hernandez and Bansi Malde at the Institute for Financial Studies; the staff at the CHARLS project at Peking University; Fred Wensing from Softscape Solutions, Australia; Dave Hockman-Wert, a user from the Pendragon user group; and Michelle McConnaughay and Elizabeth Koshy from Innovations for Poverty Action.

We have benefited from the collective wisdom and individual insights and perceptive corrections from all the above-named individuals. Our report is better for their contributions.

Any remaining errors are those of the authors.

ABOUT THE IRIS CENTER

The IRIS Center at the University of Maryland (College Park) is a non-profit international development organization housed in the College of Behavioral and Social Sciences on the College Park campus. It is a dedicated research, advisory assistance, and idea dissemination organization located on campus. In addition to the home office at College Park, IRIS establishes and manages field offices in those countries where we implement long-term research or assistance projects. These special purpose support facilities are governed by IRIS managers and according to IRIS standards.

The IRIS Center is administered by University Research Corporation International (URCI), a non-stock, non-profit corporation created in 1991 to serve the educational and scientific purposes of the University of Maryland at College Park. URCI is authorized by the Board of Regents of the University of Maryland System and governed by a Board of Directors, which includes (as Chairman) the President of the University of Maryland, College Park and (as President) the Dean of the College of Behavioral and Social Sciences, as well as other members of the UMCP faculty and staff.

The IRIS Center

University of Maryland at College Park
Office of the Dean, College of Behavioral and Social Sciences
3106 Morrill Hall
College Park, MD 20742, USA

Phone: +1 (301) 405-3110
Fax: +1 (301) 405-3020
Web: www.IRIS.umd.edu

1. INTRODUCTION

The World Bank, national statistics offices, and other institutions are keenly interested in moving from household surveys based on paper-and-pen interviewing (PAPI) to those based on computer-assisted personal interviewing (CAPI). However, most organizations migrating from page to screen--from PAPI to CAPI--know neither their software options nor what software package best meets their needs.

This dilemma underscores the paucity of consolidated information on CAPI software. There are five dimensions to this informational problem. First, there is no standard in CAPI software. Outside of the World Bank, different organizations use different software for their surveys. The United Nations Children's Fund (UNICEF) relies on RapidSMS for many of its surveys; the United States Agency for International Development (USAID) uses CSPro for DHS; and marketing firms use still different software packages. Even within the World Bank, different divisions rely on different software solutions. The Living Standards Measurement Study (LSMS) team within the Poverty and Inequality Group of the Development Research Group (DECRG) has used Capture with Enhanced Survey Technology (CWEST) for consumption experiments, while the DECRG Finance and Private Sector Unit has used Pocket Survey for measuring profit and sales of microenterprises.

Second, there are no CAPI user communities. There are no large user communities for CAPI software whose various experiences could inform the LSMS team's software choice. CWEST, which has now transitioned into Surveybe, has only been used by a handful of organizations outside of the LSMS team. RapidSMS has been used for only a few disparate UNICEF deployments, mostly one-off surveys. In addition, Open Data Kit (ODK) has been used by a scattered group of academic researchers and service providers.

Third, there are no prior systematic, publicly available comparisons of CAPI software packages. Many CAPI software developers—such as Serpro, EDI, and UNICEF—are known to have made internal comparisons and then developed their own systems to fill an implicit gap in the marketplace. Others who adopted CAPI software programs have chosen a specific alternative for particular reasons such as cost, functionality, or vendor support, but these reasons are known only within their organization.

Fourth, there is a glut of publicly available information. However, none of it offers an immediate answer about whether and to what extent a given package meets the needs of complex household surveys. Furthermore, the publicly available information on competing CAPI software packages is of differential quality and quantity. Some software packages give a comprehensive overview of functionality while others leave it to the end-user to see what the package can do. This complicates any attempt at a simple comparison of features across packages.

Fifth, a limited share of the publicly-available information is informed by hands-on experience. There is a considerable gap between second-hand and first-hand understanding of how CAPI software works. The product descriptions offer at best an incomplete and passive understanding. The real understanding of CAPI software comes through using it.

While this report aims to fill the informational gap on the whole, it also serves as a foundation for the medium-term decisions on the part of the LSMS team concerning the transition from PAPI to CAPI as part of the surveys supported under the LSMS-Integrated Surveys on Agriculture (LSMS-ISA) initiative.[1]

The challenge, then, is not simply to identify CAPI software packages, but to find ones that are robust enough in performance and broad enough in functionality to support a complex household survey. Consequently, the standard is the administration of a questionnaire with intricate skip logic and complex consistency checks, coupled with a reliable system of data and survey management.

This report's approach towards the fulfillment of the aforementioned informational gap is to search for software, screen for suitability, develop a rating system, and evaluate the packages. The first section of the report explains the methodology. It outlines the procedures followed for finding CAPI software packages. Subsequently, it explains how software were screened for suitability, details how objective evaluation criteria were devised, and outlines the procedures for evaluating software.

The second section offers an individual overview of each software package considered in this report. Interested readers can find more details on each package in Appendices A and B, which offer a compact features checklist and a detailed features checklist for each package, respectively.

The third section provides a comparative evaluation of the software packages relative to one another within each area of evaluation. Those who would like a numerical scoring of this discussion should refer to Appendix C – Meta Evaluation.

The final section provides an extensive written evaluation of each software package with regards to each dimension of the main evaluation criteria. As additional resources, the report provides short descriptions of previous survey efforts that relied on the featured software packages in Appendix D. Where possible, efforts have been made to find deployments that are as close as possible in complexity and context to surveys done for international development research. In particular, these illustrative deployments were chosen to highlight CAPI surveys as similar as possible in breadth of topics, scale of implementation, and level of complexity to LSMS-ISA surveys and other integrated household surveys.

Appendix E presents screenshots from various stages of questionnaire development in each featured software package. Finally, Appendix F includes a glossary of technical terms that are used throughout the report. These principally cover terms for data structures and functionality of CAPI software.

[1] The LSMS-ISA project is an innovative household survey program established with a grant from the Bill and Melinda Gates Foundation and implemented by the LSMS team. Under the LSMS-ISA initiative, the World Bank is supporting countries in Sub-Saharan Africa to establish systems of multi-topic, panel household surveys with a strong focus on agriculture. In each partner country, the project supports at least two rounds of nationally representative household panel data collection. In some countries, additional waves are being funded from other sources. The surveys under the LSMS-ISA are modeled on the multi-topic household survey design of the LSMS, and are designed and implemented in full collaboration with partner national statistics offices. In addition to the goal of producing policy-relevant agricultural data, the project emphasizes the design and validation of innovative survey methods, the use of technology for improving survey data quality, and the development of analytical tools to facilitate the use and analysis of the collected data. The micro-data produced under the project is fully documented and publicly available within twelve months of the completion of each survey round. Visit www.worldbank.org\lsms-isa for more information.

2. METHODOLOGY

2.1 Search for software

The IRIS Center used four complementary sources of information for finding suitable CAPI software packages:

1. Institutional knowledge
2. Internet research
3. Software choices for other major surveys
4. Software choices of other researchers, survey professionals, or survey organizations

Institutional knowledge: The IRIS Center had conducted a CAPI software search for internal decisions about survey software, during which process IRIS considered a number of survey software options before eventually choosing CSProX.

In addition, IRIS previously explored CAPI software for mobile phones. Through this research, it had identified one package featured in this report (ODK), as well as several others that were less suitable for surveys of the size, complexity, and mode of implementation considered here.

Internet research: IRIS extended its existing knowledge by two types of internet research.

The first type was based on keyword searches performed through general-purpose search engines like Google. This cast a very broad net that caught many CAPI packages and survey resources.

The second type was through careful reviews of specialized websites apt to mention CAPI programs. These included general survey resource websites like the International Household Survey Network[2]; network websites for survey professionals like the Association for Survey Computing[3]; international organization websites like Paris 21[4] and UNESCAP Stats[5]; and survey technology websites like Meaning Limited[6], Open Mobile Consortium[7], and MobileActive.org[8].

IRIS also combined these two types of research to good end. The keyword searches uncovered several unexpected specialized websites. The review of specialized websites, often each with their own jargon, in turn refined keyword searches.

Software choice for other major surveys: While CAPI surveys are relatively new in the field of international development, CAPI is not a novel methodology in the context of marketing or survey research. With that in mind, IRIS explored the various software packages that other major surveys or monitoring activities have used for their CAPI data collection. These included UNICEF, which uses RapidSMS for monitoring; Macro International, which uses CSProX for certain DHS surveys; Academy for

[2] www.ihsn.org
[3] http://www.asc.org.uk/
[4] http://www.paris21.org/
[5] http://www.unescap.org/stat/index.asp
[6] http://www.meaning.uk.com/your-resources/software-database/
[7] http://www.open-mobile.org/
[8] http://www.mobileactive.org/

Educational Development, which uses Epihandy for its NetMark surveys; the National Comorbidity Survey, which used Blaise; and World Food Program, which developed its own software for early warning and monitoring activities.

Software choices of other researchers, survey professionals, and survey organizations: IRIS contacted other survey professionals and their organizations to learn more about their CAPI software packages of choice. Among others, these included Macro International, which uses CSProX for some DHS surveys; the Institute for Social Research at the University of Michigan, which uses Blaise and CASES for its surveys; and Innovations for Poverty Action, which uses Blaise, Pendragon, and PocketSurvey for various survey projects around the globe.

2.2 Inclusion criteria

Having cast a wide net and determined upon a number of potentially suitable CAPI software packages, IRIS developed and applied objective criteria for screening what it had found. Working in consultation with the World Bank, IRIS concluded that CAPI programs, to be useful for complex household surveys, must meet minimal criteria in the following areas:

- Data capture
- Questionnaire navigation
- Skipping/branching
- Data quality control
- Data management
- Case management

These criteria— a sub-set of those recorded in the short evaluation in Appendix B—are as follows.

Data capture: Complex household surveys contain a relatively wide variety of question implementation options, such as numeric, multiple choice, and ranking. CAPI programs must be able to emulate these question implementation options, and if possible build on them.

Complex household surveys ask a large number of questions for a single questionnaire. Therefore, CAPI programs must be able to record a large number of data points in a single application.

Complex household surveys capture data at many different levels of hierarchy, in rosters for (i) household members, (ii) durable assets, (iii) food and nonfood expenditures, (iv) plots, (v) crops on plots, etc. CAPI programs must be able to present questions and capture data in these diverse and multifaceted data structures.

Complex household surveys are typically implemented at a national scale, and collect thousands of household observations. CAPI programs must be able to capture and store this large number of survey records.

Consequently, IRIS required that CAPI programs, to be considered further in the study, must allow a reasonable variety of question types, capture a large number of data points, capture data points at different levels of hierarchy, and collect and store large numbers of survey observations.

Questionnaire navigation: Complex household surveys, though highly structured, are simply human conversations that may stray from the point as often as they stick to it. CAPI programs must therefore allow field users to move relatively freely around a survey instrument in order to capture proffered answers and to accommodate the field realities of momentarily absent respondents. However, they should also offer facilities for the designer to impose restrictions on navigation, if necessary.

IRIS required that CAPI survey software considered for this study include basic questionnaire navigation abilities, such as the possibility to move backwards in an interview, to pause an interview and resume it at the last answered question, and to complete a questionnaire in a non-linear way by moving from one non-sequential module to another.

Skipping/branching: Complex skips over irrelevant questions or branching to follow-up ones are a mainstay of integrated household surveys. CAPI programs, to be suitable, must allow survey programmers to replicate on a computer the same complex instructions communicated on a page in a traditional paper-and-pencil survey.

IRIS thus required that CAPI programs under consideration have, at the least, basic and complex skip instruction capabilities. Basic skips are those that use only the current question's answer to determine whether the next question is relevant--for example, whether medical expenses are relevant if the respondent has not recently visited the doctor. Complex skips are those that either use answers from several questions in different modules and/or answers in different hierarchical levels of the questionnaire to determine whether the next question is relevant.

Data quality control: Complex household surveys require complex data quality controls when paper surveys are processed, edited, and potentially revised on data entry application platforms in the post-interview period. CAPI programs must at least offer the same scope of quality controls during as well as after the interview.

For this reason, IRIS maintained that CAPI packages must provide at minimum two types of data consistency checks: those within modules and those across modules. The logic and graduated level of increasingly greater requirements between intra-module and inter-module consistency checks is the same as with basic and complex skips.

Data management: Complex household surveys generate large volumes of data that are systematically stored in the data file format of one of the major statistical analysis software packages. CAPI programs, to equal data processing in PAPI surveys, must make transfer to these file types no less onerous.

IRIS thus required that CAPI programs must generate data sets, ideally with value and variable labels, in SAS, Stata, and/or SPSS, or at least in CSV format that can then be integrated into one of the aforementioned packages.

Case management. Complex household surveys owe part of their complexity to the hierarchical structure of survey operation and the associated delineation of responsibilities, which revolve around the overarching goal of complete and accurate data. From enumerators and team leaders in the field to survey managers at the headquarters, survey staff have a unique set of responsibilities in accordance with the point at which they are integrated into a process that is characterized by three phases: pre-interview, interview, and post-interview.

Accordingly, IRIS required that CAPI programs must have comparable sets of tools for enumerators, supervisors and survey managers to employ in the management of their work.

2.2.1 Exclusion criteria

In practice, the assessment excluded several types of CAPI packages. Many programs for marketing research failed to fulfill the essential requirements, due to (i) their concentration on different modes of data collection (e.g. web surveys that require constant connectivity), (ii) their lack of controls for anything more than the most simple surveys (e.g. rosters were either unavailable or unwieldy to program), and (iii) their limitations on the number of data capture fields or survey cases stored (e.g. a handful of fields and only a few hundred observations).

Several programs for mobile phones also failed to meet the initial specifications for consideration[9]. Their shortcomings were essentially the same as those for marketing research products. Often explicitly by design, many programs were originally created solely for the capture of light monitoring surveys. However, two of these types of CAPI packages – Entryware and ODK from the domains of marketing research and mobile phones, respectively – demonstrated enough potential to be considered in further detail within this report.

2.3 Development of evaluation criteria and a rating system

While searching for suitable software, IRIS attempted to articulate the CAPI needs of complex household surveys in order to devise a system for evaluating the extent to which particular packages meet those needs.

2.3.1 Ideal CAPI package

To do so, IRIS considered an ideal CAPI package for complex household surveys, with the sets of characteristics outlined below.

First, the CAPI software platform would emulate, enhance, and extend the PAPI data collection system. To equal paper, the platform would first support complex skip patterns and simple questionnaire navigation capabilities. Beyond this, the platform would automate mundane but error-prone aspects of data collection, such as the filling in of ID codes, and would augment other aspects through multi-media cues to the enumerator and/or respondent, such as photos of food quantities or prompts about the logical consistency of answers. Finally, the platform would facilitate the seamless capture and effortless integration of non-traditional data, such as audio-visual clips and geographical coordinates.

Second, the CAPI software solution would offer a simple but powerful interface for enumerators. The software platform would recreate the simplicity of the paper experience via a easy-to-use interface for the recording and correcting of answers as well as a straightforward method of navigating the instrument. Additionally, the platform would provide a powerful in-built suite of interview aids, including access to survey support materials, pictures that clearly define terms like "large pile", multiple language versions of the questionnaire, and interactive prompts to check the consistency and accuracy of answers.

[9] For a useful compilation of mobile phone data collection programs and deployments, please visit www.mobileactive.org

Third, the CAPI package would offer a user-friendly development environment for creating, modifying, and updating digital survey instruments. The CAPI development environment would be simple enough that the system would not be overly burdensome to create, so as to ensure that LSMS teams and their local implementation partners would be able to appropriately use the technology. In addition, the development environment would be straightforward enough that CAPI applications could be modified after questionnaire pre-testing and fixed if flaws are identified during survey implementation.

In other words, such a CAPI package would be:

- Equally effective for users at different stages of survey operation;
- Flexible enough for the broadest possible circumstances, or best for a particular one; and
- Open to redesign and improvement.

Equally effective for users at different stages of survey operation: The ideal CAPI program should be robust and easy to use during questionnaire development, field deployment, and data management. For developers, the program should provide powerful tools and a facile environment for creating a questionnaire. The program should have a set of instructions—a programming language—that provides complete control over the implementation of the questionnaire. The software should also have a studio – a development environment – for programming questionnaire instructions and accomplishing all other associated tasks of questionnaire development. The studio should ideally allow for simultaneous questionnaire development assignments across multiple programmers whose work could be consolidated at a later point into one application.

For field users, the CAPI package should be as intuitive and easy to use as a paper questionnaire. Interviewers should be able to move through a digital instrument as effortlessly as they would through a paper counterpart, and should be able to understand how to use the software environment without special instruction or additional technical skills.

For survey managers, the CAPI software system should provide tools for survey management and for the prompt export of data into a desired format. Much as with paper, survey managers at various levels should have the means to track the progress of a survey, in regards to both the completeness of questionnaires as well as action against any errors that may arise. A CAPI package should also provide an easy facility for exporting data – labeled and clean – into the preferred format. Furthermore, a CAPI package should support changes to CAPI applications that are already in the field.

Flexible enough for the broadest possible circumstances, or best for a particular one: Software is only as useful as it is adaptable to the circumstances and skills of those who use it. Hence, the ideal CAPI program should be flexible enough for most users. Through support and documentation, it should be approachable for a broad class of users, or at least best suited for users with specific skills. An appropriate pricing schedule should render it cost-effective enough for deployments of many scales. Furthermore, its hardware and software needs should be open and adaptable to a variety of existing and possible future survey devices.

Open to redesign and improvement: In the same way that paper survey instruments can be improved and redesigned, so too should CAPI software. Whether through regular updates or an in-built means for extending current commands, the ideal CAPI software should be able to change and grow in tandem with its users and their needs.

2.4 Evaluation areas

From these desirable traits follow the evaluation areas of this report. As the ideal CAPI software must be effective at every stage of survey implementation, each package is evaluated through the three complementary lenses of survey designers, survey interviewers, and survey managers. As the ideal CAPI software must be flexible for the broadest possible circumstances, each package is evaluated with respect to the human, computing, and financial resources that it requires. As the ideal CAPI software must be open to redesign and improvement, each package is evaluated by its potential for expanding and extending current functionalities.

2.4.1 Tools for Design

- Programming language
- Development environment
- Questionnaire implementation

Programming language: The programming language is the set of computer instructions, or language for creating instructions, that determines the functioning of a questionnaire. In the simplest case, these are the computer equivalent of skip instructions (e.g. “if Q23=2 then skip to Q2”) that paper questionnaires provide interviewers. In more sophisticated cases, these instructions cover the answer options to be displayed under different sets of conditions.

A programming language should provide extensive control over how a questionnaire works and appears. It should also simultaneously be able to provide complex instructions to the computer in commands that are simple to understand by potential programmers.

Development environment: The development environment constitutes the design studio for CAPI questionnaire creation. It is the assembled toolkit for programming instructions, modifying the graphical layout, and defining the content of CAPI questionnaires.

The tools of a development environment should provide control over all aspects of a CAPI questionnaire. These tools should be integrated together in a unified studio which is intuitive and easy to use.

Questionnaire implementation: The questionnaire implementation is a combination of the types of questions available to the designer and the ways in which a questionnaire's questions are supported. Question implementation options may include radio buttons, checkboxes, and text fields, among others. Question support may include definitions or pictures to support a question's terms as well as text in multiple languages that does the same.

The questionnaire implementation should provide a large array of options, for question types and question supports alike. It is preferred when the tools are powerful and the software is versatile enough to present and support questions in several different ways.

2.4.2 Performance in the field

For evaluating each package's performance in the field, we look at:

- Interface for field users
- Questionnaire navigation

Interface for field users: The discussion of interface for field users covers how interviewers and team leaders interact with the application to assign/receive assignments; navigate, complete and review questionnaires; track progress; and share outputs for management review.

The interface should be user-friendly for the average field staff, and the tools for accomplishing all survey tasks must be straightforward and intuitive.

Questionnaire navigation: The means and the ability for moving around in a questionnaire are its questionnaire navigation capabilities. This, simply put, is where and how an enumerator can move within a questionnaire during an interview.

The tools for questionnaire navigation should provide powerful and easy-to-use navigation abilities.

2.4.3 Tools for managing survey cases and survey data

For its tool for managing survey cases and survey data, we investigate:

- Case management
- Data transfer
- Data exporting

Case management: This tool concerns the methods by which (i) interviewers receive interview assignments and manage the completeness and quality of their questionnaires, (ii) field supervisors assign interviews, oversee the completeness and quality of interviewer outputs, and upload outputs for management review, and (ii) survey managers allocate workload across field teams and track the progress of survey operations.

Power and simplicity are the dual virtues of this evaluation area. The power of the case management tools must be commensurate with the challenge of field operations, and should be simple enough for users at each level of hierarchy in field operations.

Data transfer: This tool covers the way that data travel from the computers in the field to the computers in headquarters. For computer surveys, these tools are often specialized FTP or server applications, including transfer protocols that encrypt and protect sensitive survey data.

Data transfer features should be powerful in performing and tracking the success of data transfers, and function best for users when their complexity is hidden but their power not diminished.

Data exporting: This tool concerns the transformation of data from the way in which it is captured by a CAPI package to a form in which it can be analyzed by a statistical software package.

The end user should have control over what data are exported and their method of export, including variable and value labels in the languages of data collection. Capabilities are enhanced when this procedure, often painful and error-prone, can be performed with point-and-click simplicity.

2.4.4 Needs in human, computing, and financial resources

For its needs in the human, computing, and financial resources for survey activities, we explore:

- Support and documentation
- Hardware and software needs
- Pricing and upgrades

Support and documentation: This area of evaluation includes training manuals, training courses, and any other instructional materials, as well as software support services.

This area is assessed on its breadth and quality of the items and tools.

Hardware and software needs: This area concerns the minimum hardware requirements for installing and operating the survey software, in addition to any restrictions on the operating system on which it works.

Hardware and software needs should be appropriate for the type of devices--such as ultra-mobile PCs (UMPCs), mobile tablet PCs (MTPCs), PDAs, and smart phones--that are used in CAPI surveys.

Pricing and upgrades: This pertains to the pricing structure for software, software training, and technical support--as well as to the fees, if any, associated with eventual upgrades.

This dimension is rated on its value in regards to the appropriateness of the costs for the software and services provided.

2.4.5 Tools for expanding and extending current capabilities

To measure this final dimension, we investigate:

- Extensibility

Extensibility: This area concerns whether and to what extent the software provides users with the means of building on old or creating new software functionality—whether through open source code, integration to external programming languages, or facilities for creating user-defined functions.

This is rated by the extent to which the software—through in-built tools or the way it is built—is flexible enough to be changed by its end users.

Taken together, the areas of this evaluation are:

- Programming
- Development environment
- Questionnaire implementation
- Interface for field users
- Questionnaire navigation
- Case management

- Data transfer
- Data exporting
- Support and documentation
- Hardware and software needs
- Pricing and upgrades
- Extensibility

2.5 Evaluation of software packages

The CAPI software evaluation is comprised of four distinct tasks:

1. Complete a feature checklist;
2. Test features through the development of sample questionnaire modules;
3. Evaluate the performance of individual software packages; and
4. Compare the relative performance of each software package

Complete feature checklists: The first phase simply involves investigating whether a software package has each item on a pre-determined list of functionality within each area of evaluation. The checklist moves area by area carefully through all of the dimensions.

There is a short version and a long version of this feature checklist. The short version captures, in one simple listing, whether a software package has the features deemed most important for CAPI software. The more detailed, longer version delves more deeply into additional features in each area of evaluation, thereby providing a richer picture of software-specific performance.

The IRIS team conducted the first stage of investigation based on the best available information, which was then checked and confirmed independently by each of the relevant software developers. The LSMS team led the second stage of investigation in order to check the factual accuracy of the IRIS team's initial findings on software functionality.

Test features through developing a questionnaire: The subsequent phase entails putting found features to the test. Having established whether features exist, the evaluation tests how well they perform.

To preserve uniformity, the evaluation consists of attempting to build a CAPI questionnaire that emulates the most complex features of a template LSMS survey (see Appendix C). These features include: rosters of known and unknown dimensions (e.g. rosters of durable assets or household members); complex skip patterns and consistency checks (e.g. whether a household member is age-eligible for a module or whether a child falls within acceptable BMI norms for his age and region); and calling the contents of complex internal and external databases (e.g. a list of household members that meet certain criteria or a list of districts that fall within a given state).

Evaluate the performance of individual software packages: The third phase, built from notes taken during the first two phases, consists of evaluating the performance, area by area, of each software package in isolation as well as in reference to the ideal CAPI package outlined earlier in this section.

Compare the performance across software packages: The fourth and final phase compares each software package and highlights the strengths and weaknesses of each package relative to one another.

3. BRIEF OVERVIEW OF EACH SOFTWARE PACKAGE

The following section provides a brief synopsis for each software package reviewed in the report. The interest is to provide the reader with some basic facts about each program before moving on to comparative as well as software-specific evaluation.

This overview aims to provide readers with basic facts about each package, background on the software use, and a description of its major strengths and limitations. Hence, the information presented in this section will be less comprehensive with respect to the full reviews and functionality checks provided for each software package in Appendix A and C, respectively.

3.1 BLAISE

Developer: Statistics Netherlands
Version Reviewed: Blaise 4.8

Developed by Statistics Netherlands, Blaise is a powerful program for implementing computer-assisted personal interviews in many modes of data collection, including web, telephone, and face-to-face interviews. Blaise provides rich and wide-ranging controls for virtually every aspect of a questionnaire, as one would expect from such a comprehensive survey system.

Blaise has several strong features. For developers, Blaise boasts strong documentation and a vibrant user community that can help new developers assimilate the program's otherwise daunting command line programming development environment. Though the learning curve is steep, Blaise's programming language is well designed for most survey design tasks, ranging from issuing skip instructions to controlling the graphical interface.

Blaise also offers nice controls for interviewers. The most notable of these controls is a navigation panel for moving around the various modules of a survey instrument. This provides a facility of movement that is much needed for complex household surveys.

The major limitations of Blaise do not lie with functionality, but rather with interface. The Blaise development environment only offers command line programming. For technically proficient audiences, this can be a distinct advantage, not least for copying code from one survey to the next. However, for the average user with limited prior programming experience, the command line programming interface may prove rather difficult to master. The potential problems of this steep learning curve are somewhat offset by the quality and completeness of Blaise's support and documentation.

The Blaise interviewer interface suffers slightly from its relative reliance on the keyboard and on software menu options. Blaise does have some interface elements – notably tabs for moving between modules – that are touch-friendly, but the core interface functions still come from menus and keyboard shortcuts. To move from one screen to another (the CAPI display equivalent of turning a page in a paper questionnaire), the interviewer must select "Next Page" or "Previous Page" from the navigation menu. To change the language of the survey instrument, the interviewer must select the language option from the "Options" menu. This potential problem is somewhat allayed by the possibility of providing small buttons for most of these functions along the top of the interface screen.

3.2 CASES

Developer: Computer-assisted Survey Methods Program at the University of California, Berkeley
Version Reviewed: CASES 5.5 for CAPI, CASES 6.0 for case management

Developed by the Computer-assisted Survey Methods Program at the University of California, Berkeley, CASES is a versatile system for administering structured questionnaires through web, telephone, self-administered, or face-to-face interviews. Consequently, CASES has a large set of tools for creating and controlling questionnaires for every mode of data collection.

CASES provides survey developers with an astounding array of options. The software provides two separate modes of instrument creation. The more technically proficient can create an entire questionnaire through command line programming. The less technically oriented can garner the same control through a menu-based suite of controls. For both approaches, CASES offers considerable command over every aspect of a questionnaire, from its design on the screen to its internal navigation options for interviewers.

CASES provides field users with an intuitive interface after loading the survey instrument. It offers interviewers buttons for navigation and other designer-defined functions, such as changing the language of the survey instrument. It also has a minimalist software interface, since the survey instrument is flexible enough to provide interviewers the type of functions (e.g. navigation and scheduling a callback) that appear in the software interface in other CAPI packages.

However, CASES suffers from two shortcomings as a CAPI program, the first of which is customizability. Customizability is normally desirable in theory, but for CASES it is ultimately often undesirable in practice. CASES presents users with such an array of options for data fields that the design process can become a tedious experience. Developers often must write several lines of code to accomplish what either is available by default or can be done more easily and compactly in other CAPI programs.

The second issue is that CASES' development environment does not enable programmers to see the structure of the questionnaires and associated data sets that they are creating. This sometimes offers a significant obstacle to constructing complex rosters or testing that a CAPI questionnaire's data tables fit together well. This also may create confusion in grasping the hierarchical structure of the questionnaire (i.e. the level at which questions are asked and the level of analysis with which they are associated).

3.3 CSPROX

Developer: Serpro, S. A.
Version Reviewed: CSProX 3.3[10]

CSProX, developed by Serpro, a Chilean information system engineering company, is part of the old guard of survey software. Its predecessor program CSPro was conceived as a data entry package, aimed at bringing quality control to the process of keying paper records from complex household surveys. CSProX, an independent software product, adds computer-assisted interview features to that data entry core through functions such as pop-up menus, assisted-coding algorithms, and interactive error correction.

Blending power and simplicity, CSProX allows survey designers to implement difficult questionnaire designs with a relatively simple programming language and development environment. Its development environment offers an approachable blend of menu-based development and command line programming. Its programming language provides survey developers with an intuitive syntax and indicative command names (e.g. "skip to" provides a skip instruction).

For field interviewers, CSProX offers a relatively straightforward user experience. Although many features require the keyboard or software menus, CSProX provides interviewers with an intuitive interface. Answer options pop up in dedicated windows. Error messages can be programmed to do the same.

Versatile for deployments on different devices, CSProX works well with Windows and Windows Mobile operating systems, the operating systems typically available for UMPCs/MTPCs and PDAs. Indeed, CSProX can adapt the way it displays a questionnaire – whether multiple questions per screen or one per screen – to the device on which it is operating, all using the same data collection application.

For all its compelling CAPI features, CSProX still seems anchored in its past as a data entry program in two primary areas. First, the user interface seems to presume a keyboard and a relatively technically savvy user. Interviewers often need to use keyboard commands or make selections from the software menus in order to accomplish common survey tasks, such as changing the language of the survey instrument.

Second, the questionnaire navigation features are limited for a CAPI package. The interviewer can easily move forward by answering questions, as a data entry clerk would by keying records. However the interviewer faces some difficulties in moving around more freely in the ways that a live interview (as opposed to a data entry session) would require. Careful programming can lessen this problem somewhat, but fluid non-linear navigation is much less easy in CSProX than in other packages.

CSProX thus offers a moderately strong CAPI package – strong in its capabilities, but weak in what it fails to offer. Most notably, CSProX works best for linear questionnaire navigation, but encounters difficulties when the questionnaire completion process is not strictly linear, as is often the case with complex household surveys.

[10] Version 4.1 of CSProX was released after the close of this evaluation but before publication of this report. Hence, this evaluation fails to assess any new features made available in version 4.1. However, interested parties may find more details on this version on the software developer Serpro's website: http://www.serpro.com/.

3.4 ENTRYWARE

Developer: Techneos
Version Reviewed: Entryware 6.4

Entryware, first released in 2000 by Techneos of Vancouver, Canada, is a notably user-friendly mobile survey software package that is relatively customizable, though not necessarily compatible with the full needs of an integrated household survey. Originally designed for use in market research, it is highly flexible and powerful, with an attractive and professional appearance to its questionnaire and design interface.

Entryware Designer, the software used to create a survey, is a powerful and easy-to-use package. It boasts an integrated interface where everything is in one place, i.e. the data dictionary, question, and logic/scripting options. The program utilizes many point-and-click functions, as well as scripting for further customization. It allows more complex functions, such as non-linear completion, complex skips, and soft and hard checks. Looping and nonlinear completion are also possible but are not handled in an ideal way.

Entryware Mobile, the software used by the enumerator to conduct an interview, is also intuitive and easy to use. The software takes advantage of the advances in mobile technology by being touch enabled and automatically scaling its interface to accommodate different sized screens. In addition, it is very accommodating to the needs of field research, as it can be used online or offline. Entryware Mobile can run on devices operating on Windows 98 and higher, Palm™ Desktop software 3.1 or higher, and on Windows Mobile 5 & 6 PDAs and smartphones.

The tradeoff for having Entryware's sleek look and intuitive design platform lies in its functionality. The program only allows one question per screen, which could cause follow-up questions to lose context. While Entryware does allow rostering, it is not as dynamic a feature as an LSMS survey would warrant, requiring a clumsy, error-prone workaround. Entryware also lacks the ability to capture GPS or other peripheral input data. Lastly, the price could be prohibitive for a large survey, as the user software is quite expensive.

Entryware thus shines for its ease of use, but pales alongside the power of other CAPI packages. The package excellently fills a niche need, as it is easy to use, particuarly for organizations with limited technical capacity. In terms of basic surveys done on small devices, such as light monitoring on PDAs or smart phones, it is the clear leader.

Although Entryware can also fulfill many of the needs of more complex surveys, the program does not currently have the tools for handling complex surveys as well as it does for simpler ones. Handling complex rosters requires work-arounds and complex programming that defeat the purpose of Entrware's simplicity.

3.5 MMIC

Developer: RAND Labor and Population
Version Reviewed: MMIC v0.7.2b

Multimode Interviewing Capability (MMIC) is an open-source computer-assisted survey program designed by RAND for administering interviews in several modes of data collection, including web, telephone, self-completion, and interviewer-assisted.

MMIC is an exceptionally powerful and versatile software package capable of developing CAPI applications that can comfortably accommodate the essential needs of an integrated household survey operation in the interview and post-interview phases.

Its core programming language allows survey developers to do anything in the package that can be accomplished in other programs, while its other/external programming languages include web programming languages (e.g. PHP, HTML, and Javascript) that allow developers to do virtually anything that is possible on a web page. MMIC also pairs power for developers with simplicity for users. The field user interface in MMIC is minimalist but intuitive. The versatility of the package allows this to be the case, even for complicated questionnaires.

However, MMIC's power comes at a high price, which is borne primarily by programmers in the pre-interview phase. While simple questionnaires are relatively simple to construct, more complex questionnaires require considerably more programming. Although MMIC's internal programming language is simple, more complex surveys quickly come to require more complex coding in external programming languages. The price of mastering MMIC is made all the higher because the software's thin documentation. Mastering the tools to program medium to hard questionnaires therefore requires significant and painstaking experimentation or lengthy research into references for the web programming languages that MMIC uses.

3.6 OPEN DATA KIT

Developer: The University of Washington's Department of Computer Science and Engineering
Version Reviewed: Aggregate v0.9.4, Build v0.8.5, Collect v1.1.5[11], Manage v1.0, Validate v1.2

Open Data Kit (ODK) is an open-source CAPI software for Android phones that was developed by the Computer Science and Engineering Department of the University of Washington. As its development only began in 2008, ODK is a new software product that is still relatively rough around the edges. Despite its novelty, the software shows great promise as a result of its powerful and unique data collection toolkit.

ODK is extremely user-friendly for field users. The interviewer interface is touch-friendly by default, where other packages strive to be through programming. As ODK is built for touch-screen phones, all of the program's field interface is oriented to touch – touch selects answers, touch changes the questionnaire language, and touch navigates the questionnaire.

Another virtue of ODK is the ease with which it takes to media-intensive applications. What might pose a stumbling block for developers in the programming phase in other applications is accomplished with ease in ODK. Similarly, ODK captures media as an answer or displays media to assist in selecting an answer with equal ease.

However, the program's current lack of polish is visible on a number of fronts, the first of which is questionnaire development. ODK's current questionnaire development tools are fragmented into separate stand-alone applications for each major design task. Programming involves one tool (ODK Build), while testing involves another (ODK Collect).

The design toolkit is further complicated by the seeming functional overlap of tools as well as the availability of several options for every function. Testing a data collection application involves using two separate tools: ODK Validate to ensure that program code follows proper syntax and ODK Collect to confirm that the program code produces the desired effects. Programming a data collection application can be done in ODK Build, PurcForms, or command line. This fragmentation seems to obey the design philosophy of having single tools that accomplish single goals. The result, however, makes questionnaire design often unduly difficult for new or less technically proficient users.

Data management is another area in which ODK struggles. The package stores data in a convenient portable format and transfers it through a dedicated server application. However, the data storage format fails to allow for the variable and value labels that are crucial for the management of complex household survey data. Likewise, the server application warehouses the data well, but stores the data in a format that makes data export burdensome; rosters for each survey case are stored in separate files that must be concatenated for use.

ODK has deep flaws in its current incarnation, but the program is strong in many areas where other packages are weak, and is fast evolving where other programs stagnate. ODK's current strengths lie in its handling of media and of GPS. Unlike more traditional CAPI packages, ODK treats media capture as a logical adjunct of, rather than an add-on to, pre-scripted survey questions. The consequence is that

[11] A beta version of ODK Collect 1.1.7 was released after the close of this evaluation but before publication of this report. The assessment, hence, limits its evaluation to the features available in version 1.1.5.

media capture is effortless for developers to program and as easy as the click of a button for interviewers to perform. The same is true for GPS coordinates. This could make ODK a logical choice for quantitative surveys that have qualitative elements, or for facilities surveys that rely as much on images and coordinates as on answers to pre-coded questions.

ODK 's other strength is in the rapidity with which the program is currently evolving, and the degree to which its adopters can play a role in determining its ultimate form. ODK is constantly adding new functionality on a nearly monthly basis. It is also an open-source project that can benefit from a vibrant developer community, and its core product can and has been modified for the particular purposes of certain survey deployments. This makes ODK an interesting product to follow for implementers with an interest in novel features, or in an interesting product to shape for organizations with high levels of programming capacity with languages such as XML and JavaRosa.

ODK, at the very least, is a program to watch, if not use today – even if its current features do not precisely match the needs of complex household surveys.

3.7 PENDRAGON FORMS

Developer: Pendragon Software Corporation
Version Reviewed: Pendragon Form VI

Pendragon Forms has been a staple of data collection since 1996. Originally built for simple data collection on Palm Pilots and other small form factor devices, it has now released a new version that runs on HTML5-enabled browsers, opening the software to a larger set of devices and larger form factors.

Yet despite this recent migration to a new platform, the core product remains essentially unchanged. It is still optimized for use on small screens and continues to offer developers simple tools for creating questionnaires.

The hallmark of Pendragon is simplicity. For the developer, the software provides an intuitive design suite that guides users sequentially through stages of designing survey instruments. For the field user, the data collection interface is easy to master. The interviewer can easily advance in the questionnaire through forward and backward buttons as well as through a navigation menu. For data transfer, Pendragon provides effortless sync of data through its server application. Field users only need to press the "sync" button to accomplish this task.

However, Pendragon's simplicity is also the source of some of its limitations, which are visible in regards to presentation, design, as well as technical performance. Pendragon is optimized for simple surveys on small devices. This is a laudable feature except that it cannot be modified for devices with larger form factors, as Pendragon limits each screen to a maximum of 10 lines. It also provides very limited screen real estate for questions, leading to the forced truncation of text for longer questions.

Pendragon is designed for short surveys, and it therefore fails to anticipate the more demanding needs of complex survey instruments. In particular, Pendragon does not provide the type of looping functions needed for rosters (i.e. repeat a given set of questions for a given set of items). Instead, it often requires developers to create multiple fields for each element of a roster (i.e. a set of variables for each item in a roster).

Pendragon also imposes technical constraints on the number of data fields that a survey can feature. Surveys in Pendragon are composed of fields and forms: fields collect data, and each form contains several fields. Pendragon limits the number of fields to 250 per form, and the number of forms per survey to 25. These limitations may seem permissive until one considers that each row in a roster would have several fields, such that this limit may actually be binding for complex surveys.

Pendragon is thus similar to Entryware in that its emphasis on simplicity forecloses an ability to handle more complex operations. Pendragon still performs many of the needs of more complex surveys, but it does so at the cost of undue complexity and cost in instrument development relative to CAPI packages geared towards more complex use cases.

3.8 SURVEYBE

Developer: Economic Development Initiatives (EDI)
Version Reviewed: Surveybe, pre-release beta version

A CAPI program developed by the survey research firm Economic Development Initiatives, Surveybe was built by survey implementers with the field realities of complex household surveys clearly in mind. This focus determines to a large degree both the capabilities and limitations of the software package.

Surveybe provides survey developers exactly what they need and nothing extraneous: a limited design palette that is excellently implemented. In number, there are limited question types, limited graphical design tools, and limited templates for presenting groups of questions. However, these tools are suited for easily designing questionnaires both simple and complex. With drag-and-drop controls, developers can construct everything from simple questions to complex series of related rosters. Using a default setting that creates navigation tabs for each survey screen, developers can provide users with a powerful navigation feature without any additional programming. With a little programming, developers can deftly draw in past rounds of data or external databases to facilitate validating current data collection.

As accessible for interviews as for designers, Surveybe provides an user-friendly interface for field interviewing. By default, all survey input and action fields, from entering answers to moving between modules, are touch-friendly. All sections of the survey can be reached by a few clicks so that interviewers can complete survey modules based on the availability of respondents, rather than in a strictly sequential or linear way. Surveybe provides an interviewer interface that is both simple and powerful.

For all its strong features, Surveybe suffers from three principal shortcomings. First, while the non-linear navigation feature allows interviewers to navigate freely through a survey instrument, it does not show them where they have been or where should go next. It fails to constrain or guide interviewers in their non-linear completion.

Second, Surveybe does not currently provide a user interface for case management. The current case dashboard provides field staff nothing more than the name of the data file associated with a survey case. This limitation is problematic, as it implies that (i) the field staff may struggle to find the proper case for the continuation of an interview and/or to recover the completion status of a case without manually opening the interview file and going through the questionnaire, and (ii) the survey management staff lack the tools to assign workload, track progress, and broadly review the quality of the work.

Third, Surveybe offers a development environment that is intuitive for being based almost exclusively around menu selections. While this is a boon for new users, menu-based development could become a burden for experienced or heavy users. The package enables the reuse of previously developed questionnaires in their entirety, but does not allow the reuse of structural components of prior questionnaires (e.g. questions, rosters, screens/modules). Another related constraint is that the development environment does not allow for simultaneous development assignments across multiple programmers as part of the same survey operation, such that their separate work could eventually be consolidated into a single application.

4. COMPARATIVE EVALUATION OF SOFTWARE PACKAGES

This section compares the relative performance of each software package on the basis of the evaluation criteria retained for this report.

Each sub-section evaluates the performance of the software packages with respect to one particular dimension of evaluation.

The reader may find a graphical representation of this detailed ranking in Appendix C.

4.1 Programming

The programming language is very important for CAPI software packages. Its scope typically defines the parameters of possibility in designing and controlling survey instruments; its facility of use defines the ease with which an average user can harness a program's power.

For programming, the CAPI software considered here have one of three types of languages: proprietary, which are developed for the specific needs of a given CAPI software package; external, which are developed for software users outside of CAPI but also work within specific CAPI programs; or both, which feature both a proprietary language and at the least some components of an external programming language.

Power: Three types of power are on display with the CAPI package under consideration in this report: real power, potential power, and unique power.

Real power comes from clear control over the way that CAPI instruments work, the method of navigation for interviewers, and the appearance of the questionnaires. This clear control stems from the straightforward use of existing commands that are specifically designed for the objective they accomplish.

Blaise, CASES, and MMIC have real power, provided directly from the command line. Blaise's programming language features a broad and powerful toolkit for designing the appearance of questionnaires, structuring the storage of survey data, and determining whether and of whom questions are asked. CASES's programming language provides comparable, if not more extensive, command over every aspect of questionnaire design and administration. MMIC contains a comprehensive command language that builds on the strong legacy and structure of CAPI programs like Blaise and CASES. Blaise, CASES, and MMIC provide programmers a holistic design toolkit through the programming language. In these programs, virtually everything can be accomplished from the command line.

CSProX, Entryware, Pendragon, and Surveybe also have real power, but in lesser measure. This is because the programming languages of these packages are limited in scope and sometimes also limited in ability. CSProX provides a programming language whose commands perfectly fit both common and more demanding survey design needs, but much of CSProX's power comes from other internal development tools rather than from its programming language. The graphical layout, data dictionary, and question content of questionnaires are all defined with other internal development tools. Entryware and Pendragon are similar to CSProX in the limited scope of their programming languages, but their languages are also weaker. They handle rosters less compactly and with less ease, which

presents problems in calling external databases to populate answer options. Surveybe also follows CSProX's lead in limiting the usage and scope for programming, but breaks from trend in that its programming language is quite strong. Surveybe's programming language, the database language SQL, is ideally suited for that with which it is tasked: complex calls to databases, complex forward feeding of data, and complex data validation. Limited languages, then, work well for CSProX and Surveybe, but not for Entryware and Pendragon.

Potential power comes from commands that could provide considerable controls over CAPI instruments if slightly re-purposed. For the CAPI programs in this report, this potential lies in the use of external programming languages that can do many more things than provide interview instructions.

MMIC and Surveybe offer exactly this. MMIC builds on its Blaise-inspired core through the deft use of web programming languages like Javascript, HTML, and PHP. Surveybe succeeds in harnessing the power of full SQL, a powerful database query language, and partial HTML, a web markup language. These external languages enable developers not only to perform classic CAPI functions but also to execute features beyond the ability of their proprietary CAPI languages. Furthermore, these external languages equip survey programmers with a much broader and more flexible development toolkit than even the broadest proprietary CAPI languages are able to provide.

Unique power comes from unexpected language elements, of which ODK provides the clearest example. Through its utilization of XML, ODK can capture extremely unconventional data into a viable database, including photos, videos, and even barcodes. With its XML instruments, ODK can also administer CAPI questionnaires in unconventional ways. Through a project still in development, ODK can implement a survey via interactive voice response (IVR), where a computer conducts an interview by rendering text into speech and records answers by transforming a respondent's voice responses into numbers.

Taking into consideration real power, potential power, and unique power, a hierarchy of CAPI packages clearly forms, at the pinnacle of which is MMIC. This software package provides a strong core command language that benefits from the best of Blaise, and benefits further still from integrating the power of PHP, HTML, and Javascript.

Surveybe places second in the hierarchy. This new entrant in the CAPI market uses full SQL to provide core functionality as well as to lay a foundation for creating increased functionality. Surveybe's choice of programming language is particularly well-suited to many complex programming tasks encountered in CAPI instruments, including but not limited to i) calling external databases for codes and constraints ii) executing rosters-within-rosters for follow-up questions on a sub-set of rostered observations and iii) managing the relational links that tie different hierarchical levels of a survey together.

A strong contender, Blaise provides excellent functionality despite not utilizing any other command language beyond its own proprietary one. Through classic programming concepts such as loops and arrays, Blaise handles complex tasks with ease. By using its own concepts such as blocks and tables, Blaise provides strong controls with few lines of code and offers an elegant means for non-linear completion of a questionnaire.

Ease of use: Power cannot be the sole consideration for a programming language as power does not always correspond to facility of use, and where the two do not go hand in hand, the language cannot be used to its full potential.

For the purposes of this evaluation, ease of use comprises of being easy to learn and easy to program.

In regards to program languages, being easy to learn stems from being close to human languages, or being articulated in a clear way that requires little explanation. Three languages shine in that area. Closest of the three to human language, CSProX's commands have names that describe what they do (e.g. "skip to next member_name" gives instructions to skip to the next row in a roster). Only somewhat weaker, Entryware translates execution time markers into terms that even non-programmers can understand (e.g. “prequestion” is executed before a question, “postquestion” after one). Finally, Surveybe's commands, although in a proper programming language, read virtually as an English-language sentence and also handle subscripting of rostered data elements in an intuitive way (e.g. the current value of Q25 in SQL is “current.Q25”).

Pendragon and ODK are also relatively easy to use, but to a lesser degree than the aforementioned packages. Pendragon provides an intuitive command language for simple tasks, but becomes more opaque for higher-end functions, with minor annoyances even for simple instructions. Skip instructions in Pendragon are as simple and straight-forward as in CSProX, but calling external databases is burdensome. Pendragon’s simple syntax for simple tasks is also marred by the strange default convention of referencing variables by their position in the data set (e.g. the second variable is 2) unless variable aliases are explicitly programmed (e.g. an instruction to call variable 2 gender). ODK’s programming is easy to grasp overall and much easier to grasp in regards to multi-media functions than other CAPI packages. At the same time, it may be initially intimidating for programmers not familiar with XML or command line environments. The main functions in ODK require very little programming—and the media functions, such as including an image in a question, require radically less than other packages. However, the pure command line programming required for complex household surveys could be intimidating for new programmers unfamiliar with web programming languages. In particular, coding in ODK requires following the conventions of XML, which means that programmers must use proper tags for commands, comments, and other pieces of the questionnaire program code (e.g. “<label>How old is the household head?</label>”). Though these conventions are rapidly learned, they may present a problem for newer programmers.

The programming languages of the remaining packages – Blaise, CASES, and MMIC – have relatively steep learning curves, for two primary reasons. First, the programming languages for these packages, as noted in the Power sub-section above, offer control over almost every aspect of the CAPI questionnaire, from the storage of data to the appearance of the questionnaire to the formulation of questions. The consequence of such expansive power is that the programming languages for these packages offer a dauntingly large number of commands, sub-commands, and options. For example, CASES requires the programmer, through command options, to be very specific about the appearance of a drop-down menu (i.e. with codes and answer options, with answer options only, with protected answers, with editable answers, and so on).

Second, the programming languages for Blaise, CASES, and MMIC follow coding conventions that take time to learn. Though in slightly different ways, each of these packages requires the programmer to break code into different ordered functional blocks. In CASES, the programmer first programs where the data will reside (e.g. data element *age* will start at character 6 and occupy 2 characters), then defines the question content (e.g. the data element *age* will have the associated question “How old is X?”), then defines how the data will be captured (e.g. the data element *age* will have answer options 0-10, 11-30, 30-100), and finally establishes the skip patterns that apply (e.g. if the answer to *age* is 0-10, then skip the employment question). In Blaise and MMIC, there is a similar breakdown of code into functional

blocks. Questions and their answer options are first defined in one block of code. Question order and skip rules are then separately defined in subsequent blocks of code. These conventions are not always straight-forward or intuitive, particularly for new programmers.

Although the clarity of its manuals does play a role in rendering a programming language more intelligible, this was not a decisive factor for the CAPI packages under review. Ironically, the programs whose languages are already closest to human language also provide the best pedagogical materials. CSProX, Entryware, and Surveybe all have excellent manuals, while MMIC, one of the most difficult programs, has the least user-oriented documentation. Blaise breaks with this trend: its programming language is relatively difficult, but the package provides an excellent manual as an explanatory source.

Being easy to program can be just as important for ease of use, and Entryware is by far the easiest of the packages to program, thanks to a menu-based tool for creating program code. However, its language is also the least powerful of other easy-to-use programs. While Surveybe also provides ease of programming, its ease of use does not take away from its power. Like Entryware, Surveybe has a script builder, but provides increased functionality with the little programming performed by this coding tool.

Therefore, although Entryware offers the easiest programming language to use in absolute terms, Surveybe is the easiest to use of the powerful languages.

4.2 Development environment

For CAPI software, programming and other design tools exist within a questionnaire development environment, which is essentially a digital design studio for creating and modifying survey instruments. Whatever the inherent merits of individual design tools, they are often only as good as the development environment they inhabit. In other words, individual design tools are only as useful as they are easy to find in the development environment, and only as effective as they are easy to use in conjunction there. Development environments function best when they are intuitively arranged and their tools are well integrated.

For CAPI packages, there are essentially three classes of development environment: i) menu-based environments that perform all development tasks through one or several linked windows, ii) command line environments that develop questionnaires solely through programming code, and iii) mixed environments that utilize menu-based tools for some development tasks and command line programming for others. Entryware and Pendragon are principally menu-driven, while Blaise, CASES, and one development mode of ODK are command line. CSProX, MMIC, Surveybe, and an alternate mode of ODK development are mixed.

Ease of use: Simple programs are typically those that are easiest to use. Entryware is very simple in that it provides instrument designers their full set of development tools in one view, where simple tools are distributed intuitively around the screen and more complex tools provided through clear contextual menus. Blaise and CASES, in its all text development mode, offer another blend of simplicity. Both rely completely on command line programming. All development work occurs within the same window, and every development tool typically has its proper place in program code.

However, simplicity is not always synonymous with ease of use. For non-programmers, Blaise's development environment is not at all intuitive. The interface is simple, but requires some substantial sophistication on the part of the user to execute every development task in it. For non-programmers,

menu-based environments, like in Entryware or Pendragon—or mixed ones like CSProX's, MMIC's, or Surveybe's—are much more intuitive. Through clearly labeled and contextual menus, they show users where to click for major design tasks and orient users in accomplishing tasks within a given development activity.

Integration also matters enormously for ease of use. For those who can master them, command line programs draw much of their power from integration. There is no need to scour for one particular function or another through a byzantine maze of menus. There is also no need to click on the same menu each time a question is created when these same controls can be achieved through commands that apply to several questions and/or can be reused (e.g. copied and pasted) later in the survey or in another survey entirely.

For a less technically proficient audience, menu-based development environments have clear intuitive appeal, but are not always well integrated. Entryware, Pendragon, and Surveybe each have almost every tool available within a unified interface. However, there are substantial differences between the three. Entryware is perfectly integrated in that its development environment lays out all major design tools within a single window and brings further contextual menus to the user as they become relevant. Surveybe is conceptually integrated but functionally segmented. Almost all development tools are available within the software's main interface, but developers must often go hunting through windows and menus in order to complete the next relevant task. For example, Entryware's menus walk developers through drafting a question, defining its answer options, and determining its completion rules--all within a single menu window. Surveybe provides prompts for the first and last tasks, but places answer option definitions (partially) in another development tab. Pendragon's integration is likewise flawed, in a major way. Pendragon integration for each discrete design task is comparable to Entryware's, but fails to provide developers a clear enough overview of the design process. Complex household surveys require developers to link several forms (e.g. housing conditions and demographic details), but Pendragon refuses to show users more than one form at a time, leaving developers fumbling to find the name of the form to which they need to link.

Furthermore, ease of use is often only defined relative to a user's experience, and to an organization's time horizon for using a CAPI product. For the average user with somewhat limited prior exposure to programming, menu-based programs have clear appeal over command line ones. They are typically the easiest to learn and are certainly the easiest to use over the short term. The seasoned programmers with substantial computer programming experience may be the only users to prefer command line CAPI programs to menu-based ones, at least in the short term. Also, prior knowledge of a particular CAPI packages may push potential users in one direction or the other. For example, Blaise users will find MMIC an easy transition since several concepts and programming constructs are quite similar across packages. Likewise, Entryware and Pendragon users will find Surveybe easy to adopt since some development environment interface elements are similar.

However, in terms of ease of use, as an organization's time horizon lengthens, prior experience matters less, while efficiency matters more. What is initially difficult to use may, over time, become much easier than the initially easy alternative. That is, the marginal returns to menu-based programs decrease over time while those to command line programs increase.

More specifically, within an organization that regularly fields surveys, it is not hard to imagine that the same household roster setup may be used several times. In a menu-based program, that structure may need to be developed anew from one survey to the next, since structural elements are nearly impossible

to share.[12] Surveybe is an exception among the menu-based programs in its ability to reuse a select number of pieces (e.g. answer options) from previously developed questionnaires.[13] However, in this category of software packages, there is still no product that provides tools to integrate, as part of a new CAPI application, structural components (e.g. questions, rosters, modules) of different questionnaires developed for earlier survey efforts.

Another related constraint is that menu-based survey development is sequential rather than simultaneous. Rather than allow several programmers to work concurrently on different aspects the same application and combine their work later, the existing menu-based programs impose a sequential workflow--where either a single programmer develops an application in its entirety or a team of programmers make their contributions each in turn. [14]

In command line programs, by contrast, the code that creates any questionnaire structure could be cut and reused in other survey efforts, and the code developed independently to create various components of the same questionnaire instrument can be merged with relative ease.

This highlights, then, that ease of use may have a temporal element. What is easy to use now may present problems in future. What is hard now may be worth struggling to master for future gains. Yet this discussion acts as if menu-based and command line CAPI programs are separated by an unbridgeable chasm. On the contrary, this gap could well be filled by a hybrid model – one where developers have the choice of menu-based or command line development, and where menus provide a pedagogical tool for command line coding.

Unfortunately, no such program yet exists. CASES and MMIC have elements of this approach. CASES' graphical design suite allows developers to create commands through its menu interface. MMIC provides developers with a drop-down menu of command lines that affect questionnaire layout. Both partially bridge the divide between menu-based and command line systems, but still remain far from the ideal hybrid.

This being the case, this report is left to identify the package whose ease of use best approximates the ideal. In this, two software packages stand out, for somewhat different reasons. Despite its heavy reliance on menus, Surveybe builds a small bridge between menu-based and command line programs. Command line programming is included in several portions of its development environment – commands in HTML are used for formatting text, while commands in SQL used are for simple skip instructions and more complex data quality checks. However, a few menu-based interfaces accomplish things for which other packages would use command line. For example, sharing answer options across surveys is done through menus in Surveybe, but through copying and pasting code in command line programs.

---

[12] Sharing an entire roster (i.e. questions, answer options, data dictionary, skip logic, etc.), for example, would likely involve sharing an entire questionnaire program file, stripping away the unwanted pieces through multiple development interfaces, building up a new questionnaire from the desired core, and testing that no legacy code from the shared questionnaire creates issues with what the programmer is now trying to develop.

[13] Both Surveybe and Entryware allow users to create a replica of a previously developed questionnaire in its entirety, should that provide efficiency gains in planning for a future survey effort.

[14] EDI reports that (i) ability to reuse structural components of previously developed questionnaires as part of new CAPI applications, and (ii) simultaneous development capability are among the development priorities for Surveybe.

Thanks to the structure of its development environment, MMIC makes command line programming a little less daunting. Command line programming typically relies on the developer to impose structure on program code: to separate code into logical units, to split code into functional parts, and to ensure that all of the parts fit into a coherent whole. Command line programming in MMIC accomplishes this by forcing developers to code a questionnaire in the sort of pieces one would normally encounter in a menu-based program. Questions are created, drafted, and defined in one section, rules for the order of questions are developed in another, and the relation of questions to modules is drafted in still another. Within these logical divisions of code, MMIC provides a few small bits of menu interface in what is otherwise command line programming. Developers, for example, may click to create a new question or select questionnaire layout commands from a drop-down menu.

For these reasons, Surveybe fares fairly well overall, despite its sometimes unwelcome overreliance on menus that have short-term as well as long-term implications for how the work could be shared across several programmers and reused in future efforts. MMIC is the clear winner, despite its sometimes difficult-to-navigate development environment.

4.3 Interface for field users

For field users, the most important issues concern the appearance of a CAPI program and its ease of handling during a live interview. Ultimately, power in back-end functions – the concerns of programmers – means little to nothing if front-end interface is lacking for the specific needs of field interviewing.

The quality of the field user interface is considered through the twin virtues of appropriateness and user-friendliness. Appropriateness is the degree to which the interface fits CAPI surveys specifically, such as its fit for smaller survey devices and use by less technically savvy interviewers. User-friendliness concerns the ease with which the interface can be exploited for common interview tasks, such as changing the interview language or moving quickly to another section of the survey.

Appropriateness: For software as with all else, design is destiny. Those software packages whose field user interface matches least well with CAPI implementation tend to be those that were not originally designed for this purpose.

CSProX, the cousin of the leading tool for data entry of completed questionnaires, CSPro, comes up short in the experience it offers for live interviews. The software interface contains unnecessary complexities for field users, such as reliance on software menus or keyboard shortcuts, a downside for devices without a keyboard or mouse. As an early package for market research and inventory management, Pendragon has many strong features, but user interface is not one of these. It works very well for completing short forms, but its interface shows too little on the screen at one time to be an effective software for supporting questions. Developed explicitly for smart phones, ODK currently suffers the same problem as Pendragon. However, ODK will soon be releasing a version that displays several questions on the same screen. Likewise, Entryware's otherwise exceptional interface is marred by this same restriction. These are not, in most cases, catastrophic issues with appropriateness, but rather noteworthy deviations from what would have been desirable.

Meanwhile, the remaining software packages boast appropriate field interfaces, for diverse reasons. Most are multi-mode packages--designed to work equally effectively for CATI, CASI, and CAPI interviews--that provide particularly fitting interface features for CAPI interviewing. One is desgined specifically for CAPI. Blaise, although also built for CATI and CASI, provides an on-screen structure that makes CAPI

interviews easy, notably through tabs that correspond to separate survey modules. MMIC, also a multi-mode program, fares extremely well in CAPI, thanks to its near complete lack of software interface aside from the customizable questionnaire. CASES, the last of the multi-mode trio, offers a similarly interface for CAPI, matching MMIC in appropriateness for the end user. Surveybe, however, was built specifically for CAPI surveys, and it shows. There is little software interface to distract from the questionnaire itself, and the survey instrument shows question context in compelling fashion.

User-friendliness: Design also plays a large role in user-friendliness. Those things that are carefully constructed are most often those that are easiest to use.

Such is the case with the software originally created for CAPI purposes. Designed for touch-screen devices, Entryware and ODK are quite touch-friendly. Entryware has buttons for navigation and large user-friendly input fields. As a smart phone software, ODK defaults to touch as its mode of interaction, as does Pendragon.

Such is also the case for programs originally designed for other needs. Born from a predecessor program developed for processing paper surveys, CSProX's core design works somewhat against its user-friendliness for CAPI surveys. The user interface is what one would expect from a data processing program: heavily reliant on keyboard commands and selections from the software menus.

However, design is not a necessary condition for good functioning. Blaise is surprisingly touch-friendly, not least with its navigation panes. CASES and MMIC offer the ability to provide buttons for many user functions. Surveybe makes everything touch-friendly, from input fields to navigation panes.

What sets certain software apart from others is how common survey functions, such as questionnaire navigation, questionnaire review, and changing interview language, are made easy for interviewers. CASES provides programmable buttons to be used for these functions. Entryware gives interviewers navigation buttons throughout. ODK offers a touch-friendly main menu that has been translated into several languages. Blaise manages inter-module navigation through touch-friendly tabs for each major survey division.

The best performances in this area, however, are MMIC and Surveybe. MMIC provides a highly customizable questionnaire environment, including compelling features such as command buttons and the ability to create color-coded dashboard of accessible modules, where a color indicates to interviewers whether or not they can currently complete a given module. Surveybe, for its part, provides an exceptional questionnaire environment by default. Default navigation is through tabs that correspond to survey sections. Large clickable buttons guide interviewers through follow-up questions. Almost all entry fields are touch-friendly.

Taking into consideration both appropriateness and user-friendliness, MMIC and Surveybe surpass other CAPI software currently being considered. MMIC fares so well because of how well it can be made to fit the needs of a CAPI survey. Surveybe leads MMIC slightly because it possesses the most desirable CAPI interface characteristics natively, without the need for any additional programming.

4.4 Questionnaire implementation

What a software can do – its power and versatility – should be considered alongside how well it does it. Good software accomplishes a single goal effectively, but better software can accomplish multiple goals excellently.

For CAPI software, power is judged by how well it can handle complex or media-intensive implementation of questions. Versatility is gauged by the breadth of available options for presenting questions and capturing their answers.

Power: The hallmarks of question implementation power are a solid command of the basics, a possibility of variety, and the integration of multi-media. All currently considered CAPI software provide core question types. CSProX, Entryware, and Pendragon offer the necessary core of answer options and question support, and each extend well beyond this core: Pendragon allows image capture, Entryware supports answers along a sliding scale, and CSProX provides an excellent assisted-coding algorithm for mapping open-ended answers into a large set of pre-coded answer options (e.g. assisting coding "farmer" into a labor category "103 – Subsistence farmer").

However, only a few shine for the way in which they go beyond the basics. Through artful exploitation of SQL, Surveybe can determine the contents of questions in unique ways; a recent choice experiment saw respondents vote on a set of service options, part of which were randomly selected at runtime. Equally at home with CATI, CASI, or CAPI surveys, Blaise and CASES convincingly deliver basic question types while integrating media. CASES, for example, can associate several files with both a question and its individual answer options, which is an important feature for sensitive questions. With virtually everything on web pages also possible in the package, MMIC provides a powerful variety of ways to implement questions. One user provided an on-screen game for adding digital pebbles to vote on desirable attributes. Very much in a class of its own, ODK stands out through its compelling capture of GPS, audio, and video alongside classic survey data.

Versatility: Variety is another source of strength for the way CAPI software implement questions. Most packages provide only a few options but do them well. CSProX provides users relatively few but convincingly implemented ways of asking and answering questions. Surveybe also provides relatively few options, but executes them exceedingly well.

Some provide a great degree of design freedom. Blaise and CASES allow developers to indicate in very specific detail how questions and their options should be presented. Building implicitly on this legacy, MMIC provides this same degree of user-defined variety while adding to it the flexibility that only external programming languages can provide.

Thanks to its power and versatile controls, MMIC provides the best questionnaire implementation controls. Its only limitation is the programming costs involved in realizing the program's potential.

4.5 Questionnaire navigation

For complex household surveys, how interviewers get to questions is just as important as how the questions are asked, if not more so. The defining characteristic of complex household surveys is that they are complex. They contain many unrelated streams of questions. They require several respondents, not all of whom may be available to complete questions in strict sequential order.

The quality of navigation controls is judged from the complementary perspectives of questionnaire developers and questionnaire users. From the developer's perspective, navigation is about control. The quality of control is gauged by the degree to which designers can both restrict and strategically relax interviewers' control over navigation, thereby providing the necessary freedom of navigation while imposing some constraints on the questionnaire order. From the interviewer's perspective, navigation is about simplicity. Its simplicity is measured by the ease of use for standard and complex navigation tasks: for answering questions and for correcting any conflicts, particularly across modules, in answers.

Power: Control is the key to power in navigation. Part of control is what the software lets the interviewer do. The ability to move around the questionnaire more freely is a feature that all CAPI packages offer, but that varies vastly in power from one package to another. Some, like CSProX, provide meager tools for non-linear navigation. With a relatively weak toolkit that requires programming, CSProX only offers one of two unsatisfactory options: one the one hand, the default ability to move forward and backwards one questionnaire screen at a time; on the other hand, the marginally programming -intensive feature of jumping forward to pre-appointed places in the questionnaire, or the considerably more programming-intensive option of allowing specific questions to be skipped at the cost of more program code . Others – like Entryware, Pendragon, and ODK – provide simple tools for navigation that require little to no programming. These packages offer an in-built ability to jump around a CAPI instrument, but move only to particular questions specified by the interviewer. Still others – like Blaise, CASES, MMIC, and Surveybe – offer more extensive controls over navigation, but require some limited programming. These packages allow the programmer to designate certain sections as "parallel blocks" that can be completed independently of other sections. Blaise, CASES, and MMIC require a small amount of programming to provide this ability. Surveybe provides this by default for all survey sections, yet provides no tools for adjusting this default setting.

An adjunct of what software lets interviewers do is how it lets them do it. CSProX is keyboard-dependent for navigation by default, but can offer some scope for more touch-friendly navigation. Questionnaire movement requires either keyboard shortcuts, selections from the program menu, or clicking a question from the questionnaire case tree. Blaise, Entryware, MMIC, ODK, Pendragon offer touch-friendly buttons to move from one survey screen to the next. Blaise and Surveybe offer a panel of navigation tabs to move from one module to another. Blaise and CASES offer programmable jump menus that enable interviewers to access only those sections that are currently relevant and allowed.

The other parts of power are what software prevents interviewers from doing, and how it constrains them. By default, CSProX constrains interviewers to the programmed path of the interview, not as an option but as a rigid default, which does not provide any navigational power whatsoever. Entryware, Pendragon, and ODK err in the opposite direction by allowing relatively free movement that is uncontrolled by any constraints, which gives rise to its own set of problems. By careful programming, CSProX provides comparable freedom, but unlike the above does so at the cost of developing more program code. However, Blaise, CASES, MMIC, and Surveybe provide navigation with some constraints. CASES and MMIC allow developers to provide interviewers with non-linear navigation abilities, but also endow programmers with tools for constraining the scope of non-linearity. CASES and MMIC can prevent interviewers from entering certain blocks of a questionnaire without having first completed other blocks. MMIC, for example, can conditionally enable (and disable) entire survey modules based on whether or not a prior module has been completed. Blaise and Surveybe provide much freer movement than either CASES or MMIC, but also have lesser ability to constrain movement. Their only constraints are the normal ones: preventing interviewers from moving down an irrelevant branch of

questioning. Yet the example of completing a roster in a questionnaire demonstrates that this is still a feat. In Surveybe the enumerator sees an entire roster at once, can enter data on any person in that roster, and move forward into any sub-subsection of the roster for any given person. Thanks to programming tools, Surveybe can prevent completely free movement by disabling the sub-sections of a roster until such time as an interviewer has provided requisite information on whether that sub-section is relevant for a given person.

CASES and MMIC offer the most power in what navigation they allow and constrain interviews to have, although this power comes at programming costs. Both packages offer non-linear completion through designating blocks of a questionnaire as "parallel blocks". Both packages can prevent interviewers from completely non-linear movement by requiring that certain modules be completed before others can be begun.

Blaise and Surveybe offer non-linear completion with much greater ease than CASES and MMIC. Blaise requires a small amount of programming to designate sections as "parallel blocks" and then creates tabs for interviewers to access them. Surveybe offers this feature with no programming requirements at all. Questionnaire sections in Surveybe are by default "parallel blocks". However, these packages fail to constrain non-linear navigation significantly.

Simplicity: Powerful controls, to be useful, must also be simple. Most software packages offer buttons for forward and backward movement, but a few also offer many more user-friendly navigation features. Blaise and Surveybe present a clickable navigation panel for moving from one independent module to another. CASES provides programmable buttons that offer users a more sophisticated set of navigation options. MMIC can create color-coded buttons that indicate which modules an interviewer has completed and where they are allowed to go next. These all result in the simplicity of one-click navigation.

However, Surveybe leads the pack for simplicity of navigation for interviewers. Through its navigation panel, it allows interviewers to move freely from one module to another by a click of a button at any point during the interview. The only other CAPI packages in which this readily appears are Blaise and Entryware. Through its unique questionnaire layout where buttons provide easy access to more detailed questions, Surveybe renders the otherwise confusing or tiresome navigation through forward and backwards buttons unnecessary. Through all of its navigation features, Surveybe provides interviewers a common mode of moving around: click on the labeled item where you want to go. This holds true for sections, each of which has its own labeled tab, and for sub-sections of follow-up questions, which also have a button with a brief description of their content.

Another aspect of simplicity in navigation concerns the ease with which an interviewer can move around to fix any errors or inconsistencies that the CAPI package may have identified. Although all software under consideration offer the traditional means of paging forwards and backwards to fix an errant field, only four of them offer an interactive means of doing so that exploits non-linear navigation, in two fundamentally different ways. Blaise and MMIC both prompt interviewers to revise potentially errant fields and provide them with the on-screen means for skipping to those fields. CSProX offers comparable functionality through a pop-up menu. This is all triggered when errors occur, and can be designed as "soft" or "hard" errors – that is, as possible errors that may be bypassed or as definite errors that must be addressed immediately. Surveybe takes a slightly different tack. It offers the interviewer the means for jumping directly to problematic or potentially problematic fields with a single click on the appropriate row displayed in the list of errors. It differs in its approach, however, in that these errors

may be raised and addressed at any point, after the interviewer has elected to "validate" part or all of a questionnaire. This allows enumerators to continue their work and make revisions later, while the other packages force errors to be addressed immediately. These are differences in design philosophy, but the overall approach –interactive error correction and provision of links to problem fields – is the same.

Surveybe is both powerful for developers and simple for interviewers. Much of Surveybe's navigation capabilities come from design features that require no programming. Other packages, like MMIC, come close to challenging Surveybe's power, but replicating Surveybe's features with such functionality comes at the cost of program code.

4.6 Case management

Collecting cases is not the same as managing them. Most CAPI packages have separate solutions or interfaces for managing survey cases.

One cannot take for granted that good performance at collecting data guarantees good performance at managing it. Nevertheless, the metrics for the performance of the case management system are comparable to those of data collection tools.

Considering in turn the perspective of the developer and the field user, case management will be rated in two dimensions. The first is power. It reflects the quality and the scope of information that the developer can capture and present to users about each case, as well as the software's ability to provide a structured system for making and adjusting survey work assignments at several different levels of hierarchy. The second is simplicity. It concerns the ease with which field users can use and understand the case management system, as well as the ease with which they can receive, sort, and act on their survey case assignments.

Power: Case management involves, on the one hand, gathering and updating information on the status of cases and, on the other hand, making and adjusting work assignments on the basis of this information, typically through an assignment file distributed in part or in full to supervisors and/or team members.

One criterion of power, then, is the quality and scope of information that survey software gathers on survey cases. Most pure CAPI packages have little or nothing to offer in this regard. ODK and Pendragon limit themselves to simple data collection, as do Entryware and Surveybe, while CSProX simply records the status ("not started", "in progress", and "complete") of interviews through its Control system.[15] However, the CAPI packages that also offer multi-mode interview capabilities – Blaise, CASES, and MMIC – collect rich data on interviews. These packages allow programmers to determine the breadth of data (e.g. number of contacts, status of the interview, case notes, etc.) that is being collected as well as how and when the data are updated.

What is true for the scope of information gathered is also true for the other criterion of power: the ability to provide a structured system for making and adjusting survey work assignments at different levels of hierarchy. In general, pure CAPI packages entirely lack this system. In CSProX, there is a partial counterexample. Through its Control system, CSProX can specify which teams are assigned to which

[15] EDI plans to make available a case management suite in Surveybe sometime in the second half of 2011. No information is currently available on the exact set of planned features.

clusters, and which interviewers are assigned to which household. This provides effective controls but is also rigid in that it requires regeneration and redistribution of assignment data files when supervisors' field assignments change.

More aligned with this objective are the multi-mode interviewing programs: Blaise, CASES, and MMIC. Blaise provides programmer tools, through Maniplus, for creating a custom-tailored case management dashboard for each level of survey hierarchy (i.e. interviewers, team leaders, and survey managers). CASES offers the same functionality, through a larger design toolkit, for defining the parameters of case management dashboards, for designing the look of the dashboard, and for providing the means of synchronizing data and assignments across various levels of hierarchy. MMIC offers a roughly comparable suite but adds the capability of entering and reviewing qualitative notes about interviews to the equation.

The case management systems offered by these multi-mode interviewing programs offer functions for all levels of survey staff. For interviewers, these programs offer a dashboard with assigned cases and their status, where interviewers can resume an interview or start a new case.

For supervisors, these programs provide a summary of survey operations under their supervision, showing cases that the supervisor has assigned to interviewers as well as cases that the supervisor has not yet assigned to interviewers but that have been assigned to their team. This is where the supervisor can make assignments for their team, review completed cases, and generate reports on progress.
For survey managers, these programs provide both a mechanism for making survey assignments to teams and for reviewing their progress through reports. This is where the survey manager allocates and adjusts field assignments, as well as generates reports, in some cases for broad units like clusters or more granular ones such as individual interviewers.

Each multi-mode program offers an excellent case management system. However, CASES and MMIC stand out most, each in their own unique way. CASES provides a solid suite for creating a user-defined case management system and managing assignments between several layers of survey hierarchy, but distinguishes itself through its vendor neutral tools. CASES' case management suite integrates perfectly with its data collection tools, but can also work well with the data collection tools from other vendors. Indeed, CASES has used its case management tools to work with Blaise and Egonet. MMIC also provides a case management system on par with its multi-mode peers, but distinguishes itself in its ability to generate reports on data collection. MMIC allows supervisors and managers an easy means of tracking survey progress for individual field workers, clusters or surveys.

Simplicity: Although the amount of setup required by the case management systems of Blaise, CASES, and MMIC may not be easy for developers, the systems themselves are simple to use. For the field user, the case management dashboards of these multi-mode programs is mainly driven by accessible menus and dialog prompts.

Other case management systems are excessively simple. CSProX is at the same time too simple and too difficult. The portion that links with Control is simple, in that the data collection program automatically updates the status information for the survey case when any action is taken on it. However, the portion linked with CSProX proper is less simple since its cases are designated by an unintelligible string of identifying information that is not very accessible to field users. The data presented in Surveybe's case management dashboard is no more than the interview file name.

For their respective combination of power and simplicity, CASES and MMIC currently present the best case management systems.

4.7 Data transfer

What is true for case management is also true for data transfer: most CAPI packages deploy an entirely separate system to handle these needs. The quality of these systems, as before, is judged by their power and simplicity. Power concerns what the system can do. Simplicity concerns, from the interviewer's perspective, how easy the data transfer operations are to perform.

For CAPI packages there is a two-fold typology of data transfer systems: end-to-end solutions and partial solutions. Those that are end-to-end provide services from data encryption to data transfer to data transfer management. Those that are partial only handle one segment of the data transfer system, typically getting files from the field to headquarters.

Power: CSProX, Entryware, and MMIC demonstrate the range of power provided by the CAPI packages in this report. Representative of several CAPI packages under consideration, CSProX provides only a data synchronization solution, operating either through USB or an FTP. Blaise and Pendragon fall into this category.

Building on this, Entryware provides data encryption along with data transfer. Entryware encrypts data at collection and syncs data with a remote secure FTP. This provides end-to-end data security. CASES emulates the example of Entryware. Surveybe also follows this model, although its sync functionality is currently provided through third-party programs.

Bringing still more to the data transfer system, MMIC exploits data sync not only to synchronize survey data but also survey paradata (i.e. when data interviews were started and what their status is) and survey case assignments (i.e. which household an enumerator should interview). CASES resembles MMIC in also providing this feature.

A three-fold typology emerges: programs that solely provide data synchronization, programs that provide synchronization and end-to-end security, and those that provide synchronization along with other services. Somewhat unique and also an outlier in the typology, ODK offers the ability to track the success and failure of data transfer attempts, in addition to server synchronization solutions.

For data encryption, there is also a typology based on the packages that offer encryption as an in-built feature as compared to those that are open to outside solutions. Entryware and Surveybe provide encryption as an in-built feature, while all others allow end users to determine which encryption solution to use.

Simplicity: Beyond the power of these systems, there is concern for how well they work for field users, who are their ultimate end users. Though many packages are not easy to use, the best of them make major operations quite effortless. CASES, Entryware, Pendragon, and ODK make syncing data to the server simple - in most cases, the interviewer simply clicks a button to carry out this process. CSProX and MMIC do the same, but without the interviewer needing to select this option.

For another aspect of the data transfer process, Entryware and Surveybe make encryption easy. These packages encrypt data at the point of collection and only decrypt it at the home office with a special key. This all occurs automatically, in the field and in headquarters.

The data transfer is provided in most packages as synchronization from field interviewers directly to survey headquarters (e.g. CSProX, Entryware, ODK, Pendragon). In other packages, the synchronization must first be approved by supervisors, or by headquarters but requiring supervisory sign-off, before a case is considered ready for integration with the database. The way data is transferred and handled at headquarters depends entirely on the case management system. Blaise, CASES, and MMIC are the programs that handle data transfer as more than simple synchronization, thereby helpfully maintaining a comparable quality control system--of approvals at several levels of management-- that exists in most PAPI data collection efforts.

The best performers in this category are MMIC and Entryware. Entryware is one of the few packages to offer an end-to-end solution, which it does compellingly with its simple data synchronization.

4.8 Data exporting

Exporting data to the desired format can often be taken for granted by CAPI users. However, this area of performance can make or break the power of computer assisted interviews. Done well, data exporting can help users fully realize the CAPI potential for having data ready for analysis very quickly after a survey. Done poorly, it can make operations almost as expensive, and certainly more frustrating, than dealing with paper surveys.

Power: In data exporting, control is power. The more control a package exercises on which data are exported and the method of export, the more powerful it is.

Blaise and CASES have exceptional power. Their respective data exporting facilities allow users to extract, manipulate, and export user-defined segments of data into a user-defined format. CSProX provides a great deal of power as well. It allows users to select from a questionnaire case tree the segments of data to export, and from a menu, the desired pre-defined data format. It can export several formats from the same raw data at the same time.

Simplicity: However, these powerful solutions are mostly lacking in simplicity, of one sort or other. For Blaise and CASES, data exporting is an unnecessarily complex multi-stage process that involves first exporting data into ASCII format and then transforming that ASCII data into formatted final analysis files. Once this is done, the data files will be in the proper structure with the proper labels—that is, in long format with variable and value labels.

Indeed, the less powerful programs are generally much simpler to use. CSProX, Entryware, MMIC, ODK, Pendragon, and Surveybe allow data managers to export data in a few simple clicks, although the simple clicks may not be the end of the process for many packages. CSProX, Pendragon, and Surveybe all produce data that is in “long” format (or in “wide” format if preferred) and that has all appropriate variable and value labels. However, this is not the case for Entryware, MMIC, and ODK. Entryware exports data to “wide” format. Users that prefer “long” format must reshape the data set accordingly. ODK also exports to wide format, but unlike Entryware does not contain variable or value labels. This

formatting must be applied separately from ODK, as the current build of the program does not support this functionality.

Several packages provide power without simplicity, while others provide simplicity without power. Surveybe provides the best combination of these virtues. Though its range of export options is quite restricted (to only CSV and Stata), Surveybe provides an export toolkit that is far easier to use and better aligned with the desires of survey managers than the other CAPI options in this report. More powerful programs require many more steps to export data—for example, to select it, to export it, and to apply variable and value labels. Surveybe does this in a few simple clicks, and automatically creates as many data sets as there are data tables in the survey. This saves data managers the dull routine of exporting data one data table at a time. Simpler programs may export to undesirable formats or fail to provide a record of how data were formatted. Surveybe implements the proper labels and also generates a Stata file to show how data were exported from CSV to Stata format. This makes checking the program's work easy, and provides survey mangers a clear record of how the data set was generated.

4.9 Support and documentation

Software is only as good as a user's ability to exploit it. Support and documentation, which should show new users how to use and exploit the software's functionality, are a large determinant of that ability.

The utility of support and documentation is rated in two complementary ways. One is breadth, which concerns the variety of support and documentation, as well as the number and type of places to which a user can turn for instructions. Another is quality, which concerns the clarity, usefulness, and comprehensive nature of the support and documentation.

The CAPI programs evaluated here present three different profiles for support and documentation: i) internal, where all references and resources are a part of the proprietary package ii) external, where all references and resources are outside of the software and iii) mixed, where some references and resources are internal and others are external.

The CAPI programs also fit another type of profile: established or start-up. Established software tends to have a larger set of documentation and support services available. Start-up software tends to have fewer or more fragmented support.

Breadth: Two CAPI packages, both established, stand apart from the rest in their superior breadth of support and documentation. Benefiting from the references and resources of both its free and licensed software, CSProX has substantial breadth. Through its predecessor CSPro, CSProX has a beginner's guide, an extensive user manual, a set of example programs, and technical support from the US Census Bureau (although the manual for this separate product covers very little CAPI functionality). Through its licensed version, CSProX offers a compendium of new commands and fee-based training and consultative services. The latter can be done in several languages, the former only in English. Another member of the old guard of CAPI products, Blaise has accrued an impressive array of support and documentation over the years. It offers a set of user manuals, numerous example programs, free message boards, fee-based support, user conferences, and fee-based training. Its user community is large and vibrant.

Another member of the old guard, CASES also provides extensive support and documentation that falls just short of the example set by CSProX and Blaise. Like these two packages, CASES provides large manuals for beginners as well as for those digging more deeply into the software. By way of further

assistance, CASES also provides a large number of programs that demonstrate programming concepts, in addition to excellent and timely technical support by phone and e-mail. Its only relative weaknesses are its relatively smaller and less vibrant user community, and its relative lack of less technical documentation.

Of the new guard, Entryware and Surveybe distinguish themselves. Somewhat established as a software, Entryware offers a clear and concise manual for the basics, a set of examples for more sophisticated features, and free technical support service via phone or e-mail as part of the software contract. Although a start-up software, Surveybe provides clear interviewer and designer manuals, useful programming examples inside the designer manual, moderated message boards for users to discuss issues and share ideas, and fee-based support services that range from basic technical assistance to personalized consultative assistance[16].

Other members of the new guard fare less well. MMIC only offers survey developers limited online documentation and fee-based technical support services with which to supplement it. ODK offers a vibrant online user community and some external programming references, but very little in the way of consolidated, ODK-specific documentation. This mainly owes to the start-up status of these two software providers; MMIC is still writing its online documentation and ODK is still actively building and rapidly extending its software product.

However, another explanation is based on the fact that both of these products are open source. In contrast to the other products, which are all fee-based, MMIC and ODK offer their software product for free. For fee-based products, the software providers are the ones that write and provide documentation and support. For the open source products, it is largely the user community or outside companies that provide documentation and support. Although MMIC does provide fee-based technical assistance and service-based consultation, the overarching norm is that implementers pick up these products and run with them, finding support in the larger community or in specialized software firms. ODK, for example, provides limited support through its message boards, but encourages large scale implementers to contract companies like DiMagi to provide more timely and tailored assistance.

Quality: Quantity, of course, does not always make for quality. CSProX, Blaise, and CASES all have some shortcomings in quality despite their sheer breadth of resources. CSProX's coverage of new commands contained in the proprietary package is seriously lacking; it functions as a rapid listing of commands more than a manual. CSProX's main documentation, in fact, comes from its freeware counterpart CSPro. Blaise's coverage of CAPI in particular is not quite consolidated for users to read. Its manual provides a general overview of its functionality and spends as much time on CATI as CAPI, a relatively small but noteworthy issue of focus. CASES' manual is large and difficult to absorb without outside assistance (in contrast to its excellent technical support through phone and e-mail).

Meanwhile, several newer software packages have exceptional, if not broad, sets of references and resources. Entryware's manual is short but exceptionally clear, and its support services readily help to explain what the manual may overlook. Surveybe, in similar fashion, offers a consolidated and clear manual that is specifically oriented to complex household surveys. It provides very helpful extended

[16] Surveybe provides individual technical assistance that can come in many forms, from trainings to feasibility trips for agencies looking to transition to CAPI. This could also take the form of field-based design, monitoring, and management of survey operations.

examples of design and programming issues that are particular to this type of survey, in addition to a clear overview of the software itself.

MMIC, ODK, and Pendragon deviate somewhat from this trend, with lackluster formal documentation but excellent informal support. MMIC's documentation is too thin to be immediately helpful but is complemented by its excellent direct assistance. ODK's documentation is difficult for less technical users but is somewhat compensated for by its extremely active message boards. Pendragon's manual and message boards offer first time users relatively poor guidance, but direct assistance quickly answers technical questions.

Taking into consideration breadth and quality, Blaise presents the best support and documentation package, both as the cause of and testament to its success as a CAPI program. As another part of its appeal, Blaise has a large user community, which makes its message boards an equally strong source of support as any other piece of the package.

However, Surveybe deserves an honorable mention because of its clarity, comprehensiveness, and helpful orientation to complex household surveys. Although as a start-up its documentation may not rival the breadth of more established players, Surveybe certainly matches more established packages on quality, judging from the resources currently available. The package's manual, for example, is the clearest one of all the CAPI packages considered for this report. Time will tell whether the other pieces of the support package maintain the high standard of quality.

4.10 Cost effectiveness

All power comes at a cost. However, the question is whether that cost is appropriate for the software services provided (i.e. whether the CAPI costs lead to value in data collection).

The CAPI products under consideration roughly follow three pricing models: software as services, where purchase of the product grants the user unlimited technical assistance; software as software, where the package costs a per-unit fee and assistance is extra; and software as freeware, where the package is free to all users.

The CAPI products include several software products, as well as some service ones. Cost effectiveness must therefore consider the price of each component cost and what it means for survey operations.

Each CAPI package considered here includes two to three programs that are typically priced separately. The first piece, responsible for creating and modifying a CAPI program, is the designer software. This is typically purchased in relatively few units for a small team of CAPI programmers. This can typically be considered as a fixed cost of CAPI survey operations.

The second piece, responsible for running the CAPI program in the field, is the implementer package. This is purchased for each field interviewer, and can thus be considered a marginal cost. Projects of larger scale will typically require more licenses.

The third piece, responsible for moving data from the field to the survey headquarters, is the server software. Like the designer license, this can be considered a fixed cost of CAPI survey implementation. For some software, this is a separately priced software product; for others, this is integrally integrated into the core CAPI program.

Alongside software is technical support. There are a few different modes of support on offer. One is one-off technical training (e.g. classes on how to program) that is incurred as a set-up cost of the CAPI implementation. Another is on-going technical assistance that can either be a fixed cost (e.g. as-needed technical assistance that acts as an insurance policy against catastrophic CAPI failure) or a flexible cost that increases with the scale and complexity of the CAPI survey (e.g. a technical adviser that assists with design and management).

What is rated here is how well the costs for these items translate into value for the CAPI implementer.

Value: Although the final calculus of cost effectiveness depends on the scale of deployment and the technical competence of the implementing agency, a few cost trends merit emphasis. First, software as service (i.e. software provision with unlimited remote technical assistance) may seem expensive but could pay dividends for organizations with lower technical capacity.

For example, Entryware has high unit costs for designer and implementer programs, but paying these fees entitles users to unlimited remote technical assistance via phone and e-mail. The rate seems high until one factors in the price of technical assistance. However, this type of pricing model may be too costly for organizations with higher technical capacity. These organizations could benefit from more modestly priced software whose technical assistance is priced separately and only paid for on an as-needed basis.

CASES and Pendragon also largely follow this model. Both provide software but also offer unlimited remote technical assistance as part of their product. This support must implicitly be factored into the product price.

Second, software as software (i.e. software provision not inherently inclusive of any technical assistance) can be an inexpensive pricing model as long as users do not require technical assistance. Technically competent organizations may benefit, all other things being equal, from software whose price is divorced from any substantial technical support. They can simply pay for the price of the software and rely on internal human resources for absorbing and implementing the technical content of CAPI programs. Less technically competent organizations, on the other hand, may find this pricing model expensive. The software packages that offer technical support as a separate item typically charge dearly for that support.

Blaise, CSProX, and Surveybe adopt this model. Software is priced and sold separately from support services. End users may use software without paying for support, but will have to pay for support if it is needed.

Third, free software, though appealing, often has hidden costs. MMIC and ODK illustrate this point well. MMIC is a free software package, but using it effectively would likely require training and costly technical assistance. These are services that RAND provides for a fee alongside its free software. ODK is also free, but programming with it could require considerable learning costs, as ODK does not offer much in the way of technical assistance. Therefore ODK adopters would likely need to hire an outside software firm, find competent programming resources, or task staff with the arduous task of learning all of the components of programming an ODK questionnaire (i.e. XML, JavaRosa, etc.). Free, then, is not always entirely free.

While cost effectiveness may be an organization-specific concept, there still seem to be a few CAPI packages considered here that offer an interesting value proposition. In the software as software group, two packages offer real value. Blaise offers competitive fixed costs for developer software and reasonable variable costs for interviewer software. Surveybe may provide an attractive pricing model for some – where fees match usage, as measured by data points collected, rather than the number of machines – but its appropriateness is highly dependent on scale. Perhaps a deal for small surveys, Surveybe can become increasingly expensive for either large sample sizes or large questionnaires.

In the software as freeware group, both MMIC and ODK could be strong value propositions, but on the condition that users have a high level of programming expertise, particularly with web programming languages. MMIC may require some up-front training costs in order to get users acclimated. ODK is much more approachable, but also more limited in functionality. The value proposition is also large if organizations are willing and able to incur relatively higher adoption costs, whether through dedicated staff time or hiring external organizations to train. Once the fixed cost is incurred, these free software packages have no marginal cost.

In determining the best priced package, organizations should consider the features that are strictly necessary for its survey deployments, the level of existing internal programming expertise, and the degree of technical assistance required to get a CAPI system operational, given the quality of a software's documentation and the organization's technical absorptive capacity.

Additionally, it is worth mentioning that most of the featured CAPI software providers also offer bulk pricing. The pricing and cost effectiveness analysis here relies on publicly available pricing information that is more relevant for smaller scale surveys. Those interested in larger scale implementations should contact CAPI software providers directly to inquire about larger purchase agreements.

4.11 Extensibility

Another consideration of a CAPI program's fit is the extent to which the program can grow with the user and be profitably modified by the user.

For the CAPI programs considered herein, there are essentially two types: those that are extensible, whose functionality can be modified or extended by the user; and those that are not extensible, whose functionality only matures through new software releases.

The vast majority of proprietary packages are not user-extensible, or are only extensible in extremely limited ways. These packages also constitute the lion's share of software considered in this assessment. On one end of the spectrum, programs like CASES offer no scope for user-defined commands or extensions of existing functionality. The user must make adept use of the programming toolkit to push functionality in ways that its software designers may not have anticipated. On the other end, packages like Blaise, CSProX, Entryware, and Pendragon offer limited scope for creating user-defined functions. However, those functions are mainly limited to combinations of existing elements of the programming language.

The few highly extensible programs under consideration are those that use an external programming language for part or all of their functionality. Surveybe brings some real and potential power through its reliance on SQL and HTML for specific development tasks. SQL allows users to exploit internal and external data tables (e.g. tables of constraints or filtered rosters) in a way that most other proprietary

packages would not allow. MMIC offers much of the power of the web through its integration of programming languages like PHP, HTML, and Javascript, which allows users virtually as broad a design palette as those available in website creation. ODK is perhaps the most highly extensible, but has some constraints. ODK is open source and can be modified by technically trained users. However, while ODK exploits some very powerful programming tools, it is ultimately constrained by the main programming tool—JavaRosa, an open standard for XML surveys—that constitutes its core. ODK often must wait for JavaRosa to resolve an issue before it can provide certain functionality.

However, extending the functionality of a software program can often entail substantial programming costs on the part of users. Surveybe's usage of SQL seems the least costly. SQL is a widely used and relatively straightforward language whose power can be harnessed to provide novel ways of implementing questions and defining their answers. MMIC's usage of PHP, HTML, and Javascript makes it easy for web programmers to add power that its proprietary language does not provide. ODK may be the least easy to extend since the languages it uses are less widely known.

All things considered, MMIC provides the most extensibility at the least cost. In the hands of web programmers, MMIC could be quite easy to extend. However, Surveybe is not far behind. In the hands of database programmers, Surveybe could provide strong features, although the scope for extension may not be substantial.

4.12 Operating requirements

Like all computer products CAPI programs require certain hardware, software, and operating systems. To be most useful, their requirements must match the type of machines preferred for complex household surveys: UMPCs, MTPCs, or other hardware whose form factor makes the device easily portable but whose screen is large enough to display and contextualize survey questions.

The CAPI products considered here follow a two-fold typology: those that are tied to a particular operating system (OS) and those that are usable on many different ones.

Appropriateness: Overall, the CAPI software packages in this report have requirements that are well aligned with the types of devices typically used for survey deployments. The quality that sets some software apart from others is their appropriateness of requirements for the largest set of hardware/OSes, or their appropriateness for hardware/OSes that make particular sense for CAPI data collection.

The main constraint to choice – hardware requirements being quite minimal across the board – is software. Most CAPI packages under consideration work on Windows operating systems, as well as on a handful of others, including mobile operating systems (e.g. PalmOS, Windows Mobile, etc.). Software like Entryware and Pendragon, which in the past were specifically tied to the OSes of specific mobile devices (e.g. PDAs), now have versions that can be deployed on a wider array of operating systems.

More so than the operating system, it is the additional software requirements that restrict use. MMIC only requires PHP and a web browser. Surveybe requires the full Java stack. Pendragon VI requires only a modern web browser. However, those restrictions are not substantial enough to foreclose any large set of hardware and OS choices.

ODK offers one notable exception to this trend of general openness to a large variety of hardware and OSes. Indeed, ODK works only on Android, and may be stronger for its OS choice. In particular, Android is free, touch-friendly, and usefully spare in its interface. These three features all seem to be virtues for CAPI surveys. The price of data collection devices is substantially driven by the OS on it. Android could make devices more inexpensive, and survey operations thus less costly to scale.

Touch is typically an advantage for computer interface. Android, unlike most OSes, is specifically built for touch friendliness. Furthermore, its upcoming version, code-named Honeycomb, will exploit the greater screen real estate available on UMPCs/MTPCs while still working well on smaller form factor smart phones. This will bring touch interface more convincingly to larger form factor devices.

Limited OS interfaces are also convenient, in that they are typically easier to learn and offer less scope for interviewers navigating outside the intended restrictions of a survey.

Yet these virtues stem from the operating system, not from ODK. Thus, these virtues could also extend to other CAPI packages that also work on Android. Most CAPI packages reviewed here do not specifically support Android as of this writing. Entryware and Pendragon support Android on their most recent software release, while Surveybe and CASES are currently exploring Android support. Other packages, despite a lack of current plans to this effect, may adopt Android support in the future if the OS and device market shifts in Android's favor.

The one program that seems to have the least restrictions on use and the fewest software and hardware demands is MMIC. This program can work anywhere PHP can be installed and a web browser is available. MMIC operates on the widest range of operating systems, running on OSX, Linux, and Windows platforms.

5. EVALUATIONS OF INDIVIDUAL CAPI PROGRAMS

The following appendices provide detailed insights on the performance of each CAPI package in each area of evaluation. These appendices are meant to provide readers with a more thorough understanding of each software. They highlight the major strengths and weakness of packages within these areas.

Interested readers concerned with whether a given package provides a particular function are also advised to consult a thorough checklist of functionality compiled for each package in Appendix C.

5.1 BLAISE

Programming

Power of programming language: Blaise is a powerful computer-assisted interviewing and processing system. It performs well for both simple and complex objectives. Despite not being open-source, Blaise does allow a fair amount of extensibility due to its compatibility with ActiveX and Dynamic Link Library (DLL) files.

Overall, Blaise's programming capabilities are well-rounded; complex enough to suit advanced programmers, and easy enough and with room to grow for the beginner.

Functionality of Power: Blaise allows a user to perform complex skip patterns, validate data, skip modules, complete surveys in a non-linear method, call on external databases to facilitate an interview, code in hard and soft checks, among other functions. Blaise also has unique features such as allowing concurrent interviewing of two or more people, as well as relatively easy and workable codes for the creation of external databases to perform various sorts and searches, such as hierarchical code frame, alphabetical search, trigram search, or a combination of the three.

Blaise offers several less common features that aid in the survey implementation process. The software provides the ability to call up a help file or training manual during the interview. Unlike some of the other software packages under review, Blaise can call up answers from a previous round of panel data collection for complex data validation, either as a reference or to populate the survey. In addition, questions can be supplemented with media. Adding media to a Blaise survey requires the use of ActiveX and DLL; it is even possible to display clickable images. Moreover, DLL can be used to incorporate a GPS tracking system into the survey. Blaise is flexible enough that it can even capture input from other devices. These features help make the survey process more consistent and systematic.

Limitations of Power: Blaise is able to perform all of the functionality necessary to complete a full-scale survey on par with the LSMS. No limitations were discovered during the software review process.

Familiarity: Blaise follows programming constructs such as ifs, endifs, looping, etc. It looks similar to Pascal, Delphi, and Visual Basic. There is a learning curve to the more advanced features of Blaise, but the program is pleasant to use once a user gains a thorough grasp of the software, especially with the excellent help that programmers receive in the user forum and the thorough documentation provided by Westat.

Ease of Use: Novice programmers might find coding in Blaise a little challenging, such as with rosters of unknown size and elements. However, Blaise provides extensive documentation and has a very active user community, allowing the beginning user to quickly gain considerable proficiency in Blaise's programming language. A programmer with intermediate to advanced coding skills should find Blaise easy to use, even when using the more advanced features.

Development Environment

Integration: In Blaise, survey design is accomplished within an integrated development interface. From the command file, the developer can create the variable name, question text, answer type, checks and logic. Blaise does not support drag and drop form creation, which means that one would have a harder time customizing the layout of Blaise.

Survey Creation: In order to create a survey form, a programmer needs to declare a *FIELD* section where one creates a variable name, question text and answer type (how answers will be accepted). Question notes are optional but highly desirable, as this gives instructions on how to ask a question and on the appropriate flow of the interview.

Logic Programming: To facilitate the flow and integrity of the survey, the programmer can write skip patterns and data checks (hard checks where the error needs to be corrected before one can go to the next field or soft checks where the error can be ignored) in the *RULES* section of the command file.

Dictionary Creation: The data dictionary is automatically created in the *FIELDS* section.

Application Testing: Blaise has a built-in compiler for testing applications. A programmer need only click on the compile button to test the code. If there are errors in the code, the compiler will report the error. A programmer can also hit the *run* button. The program compiles the code and requires that the survey be free of syntax errors before it will run the program. If the survey is error free, Blaise automatically compiles and generates a questionnaire form.

Unique Features: A unique feature of Blaise is the ability to code collaboratively. Blaise allows different people to work on the same project and code different modules at the same time. The programmers will need to save their sections as a separate files called 'include' that can be later collated into a cohesive program called a 'datamodel.' 'Include' files not only function as collaborative tools but can also be reused in subsequent surveys.

Another interesting feature of Blaise is the ability to code *procedures*. Procedures are used to perform repetitive tasks within or between applications or to perform functions that are not explicitly provided for in Blaise. This is akin to developing a small sub-routine that can be called as needed within a program.

Interface for Field Users

Interface Aesthetics: Blaise looks clunky and lacks the ability to make large-scale changes to the questionnaire form's layout. However, this aesthetic clunkiness does not present an obstacle to the program's functionality. One unique feature of Blaise is that it boasts a screen-based presentation (multiple questions on the page) which helps interviewers navigate the questionnaire and contextualize questions. This feature helps maintain the context of the interview by showing the question flow as it

unfolds and by displaying the answers to previous questions.

Blaise surveys can be implemented on laptops or UMPCs with a Windows environment (Windows 2000 and up). As it can only work with an operating system of Windows 2000 and beyond, Blaise is not suitable for a PDA, Android or an iPhone.

Questionnaire Navigation: Questionnaire navigation is intuitive and is typically implemented by using backwards and forwards buttons, which are small icons just below the software menus. In addition, Blaise accommodates non-linear completion by allowing the enumerator to skip modules, questions, and to return to complete unfinished questions at a later time.

*Questionnaire Review***:** Once a questionnaire is filled out, Blaise saves the answers in a database. From there, enumerators can review the answers posted.

Language Capabilities: Language can be changed in a few keystrokes or by clicking the menu for multilingual instruments.

Questionnaire implementation

Supported Question Types: Blaise provides a wealth of questionnaire implementation options. There are 9 supported answer types: *classification, open text (paragraph form), date, time, real number, integer, multiple choice (radio buttons), check box, and string*. Blaise also supports external look-ups where one can search for the appropriate answer based on the available dictionary.

Question Support Features: Blaise is very accommodating to the demands of a full-scale survey. To enhance question clarity, Blaise allows audio, picture, and video to be embedded within the question prompts. Blaise also supports multiple language versions for text, audio, and video, to accommodate enumeration areas with multiple primary languages.

To help enumerators, Blaise allows comments to be entered at any point in the interview and provides access to question-by-question links to help files.

Questionnaire navigation

Blaise's questionnaire navigation is fairly intuitive. Depending on how the program is created, one can move backwards and forwards, through, within, and across modules and screens, and through rosters with ease.

Navigation is achieved by clicking the fields with a mouse or by hitting *enter* or the *arrow buttons* in the keyboard. Accessing the different modules is accomplished by selecting the different tabs on top of the info pane. By clicking the appropriate tab, the user is brought to the desired module or interview.

Although Blaise offers tabs to facilitate movement through modules, there is no easy way to access a given field within a module, e.g. to immediately move to a given question about which an interviewer may learn more information during the answer to a later question.

Case Management

In the strictest sense, Blaise does not offer a dedicated case management system. Instead, it provides survey developers with the tools for building one. Using Blaise's Maniplus program, developers can construct case management dashboards for the survey. They can even devise distinct dashboards for staff at every level of the survey hierarchy: for interviewers, supervisors, and survey managers.

Enumerators: For interviewers, Blaise allows developers to use the interviewer dashboard as a place to launch new interviews and to see information on those that have been completed. This makes creating new cases and calling old ones easy for interviewers.

Supervisors: For supervisors, Blaise allows developers to program higher level controls. It allows supervisors to access the same cases as interviewers, but with more information about who conducted the interview and who else handled the data. At the supervisory stage, it is also possible to assign different tasks to different users by storing a list of available users for an interview.

Survey Manager: For survey managers, the case manager application providers an even higher level view of all the survey cases, with information on the interviewers, supervisors, and teams who did the work. At the survey manager level, it is possible to load and remove the questionnaire on the machine, package the completed questionnaires, and send them to the home office.

This entire system is facilitated by Blaise's ability to collect data on when and by whom a survey was administered, as well as the length of time that it took to complete. These paradata – data on how data were collected – populate the various information and status fields that field staff see about survey cases.

Headquarters staff can either assign households or clusters of households to teams manually or automatically, based on a database containing geographical information for each enumerator. Team leaders can download their assignments using an open telecommunications system. Once the enumerators receive the assigned households and are able to interview them, enumerators can share the information with their team leader, who can then check the statuses of the interviews. If a questionnaire is not complete or contains errors, the questionnaire may be sent back and the household will be interviewed again. Blaise also has the capability to randomly choose households for dual coding. The chosen households are re-interviewed by another member of the coding team, after which the data are checked against the initial interview entry to ensure the quality of the data. Once the data have been checked, they can then be exported to a statistical package using Cameleon.

Data transfer

*Server Overview***:** According to the Westat Support, when working with CAPI a centralized server is not relevant and it is therefore not provided by Blaise. For web surveys, a server is central, so Blaise does provide this.

Data Security: Data security is not provided by Blaise. However, most organizations use full disk encryption. Other approaches decrypt and encrypt Blaise data files using a data compression software, PKZIP, or other third-party tools before and after the interview. Typically, to transfer data, Secure File Transfer Protocol (sftp) is used.

Data Transmission Protocol: LAN/WAN and the internet are the only data transfer protocols supported by Blaise. It is also possible to transmit files using the standard techniques, i.e. flashdrive, etc.

Data Exporting

File Formats: Blaise has a sophisticated data export utility called Cameleon that reads information from the database and translates it into other file formats. Blaise provides pre-packaged export options for 18 packages including SAS, SPSS, SPSSdot, OracleC, OracleD, ASCII, ASCIIRel, blockstrc, questblk, dic, techdesc, papersim, and fieldlist. Cameleon also allows programmers to create custom export protocols.

Transfer Protocol: While this utility may seem complex, a user can easily export the data via Cameleon through the point-and-click interface or through a user-generated code.

Features: Through this utility, Cameleon maintains all the variable and value labels defined by the user in their Blaise datamodel. In addition to Cameleon, Blaise has a program called Maniplus/Manipula that can create batch programs that read data from files and manipulate that data. One can use Manipula to reformat, update, merge and sort the data. Manipula is also used to derive new fields based on existing fields, as well as to split a data set into two or more data sets. Manipula is good for batch processing and file handling, especially when handling large volumes of data.

Blaise, through its rich data export facility, can provide output data files in either wide or long format.

Support & documentation

There is extensive documentation available for Blaise, which can be found at the Blaise website under Online Help. Once a user pays to get the full license, the user can also access hundreds of sample versions of code. There is also a basic quick start guide that comes along with the documentation.

Blaise has an established support group. To date, there are two organizations/ clubs for Blaise: International Blaise User Group (IBUG) and Blaise Corporate License User Board (BCLUB). IBUG is an independent organization open to all Blaise users from around the world. The group publishes a periodic newsletter and hosts a users conference about every 18 months. BCLUB is open to organizations holding a corporate license for Blaise. The group meets annually to discuss current usage and future plans for Blaise as they relate to the development plans and priorities.

Blaise also has an active online user community/forum. Users are encouraged to ask technical questions and share ideas, solutions, tips and tricks for developing Blaise applications. Users are actually encouraged to use this forum first before contacting Blaise support. Reply time is dependent on the difficulty of the question. For easy questions, reply time can be as short as 2 minutes. For harder questions, times may be longer (or requests will be forwarded to Blaise support). In addition to these free, user-supported services, Blaise License Plans also include product support by email for questions concerning syntax, functionality and some technical questions. As a complement, Westat also offers fee-based support including Blaise Training and other consulting services.

Pricing & Upgrades

Since the LSMS is a multi-country collaborative study, pricing would be discussed considering scope, usage and support required. Pricing for international collaborative studies does not follow the pricing

model shown on the Blaise website. Details on regular and corporate licensing are specified below. Depending on the licensing plan that is most suitable for an organization, Blaise offers a Regular Licensing Plan, and for greater than 100 application users, a Corporate License Plan is offered. The Blaise Component Pack (BCP) is an enhancement available for the Basic System version 4.5 or above. The BCP consists of a number of COM and ActiveX components that allow users to develop software that integrates with the Blaise system in real-time and that accesses Basic Blaise system's metadata and data. The BCP is useful when Blaise needs to be extended by the addition of a thermometer, GPS, clickable images, and other features.

Under the Regular Licensing Plan, the fee per developer per year is as follows:	**Under the Corporate License Plan, the fee is calculated based on the total usage units. [Total usage unit = Number of developers + (0.25*Number of (enumerator) Application Users)] The fee for total usage per year is as follows:**
• 1st developer for a basic system: $2652/year • 1st developer for a basic system with BCP: $5057/year • For the next 2-10 developers for a basic system: $1992 each/year • For the next 2-10 developers for a basic system with BCP: $2846 each/year	• For 25-49 total usage without BCP: $28,623 • For 50-74 total usage without BCP: $39,778 • For 75-149 total usage without BCP: $72,824 • For 25-49 total usage with BCP: $34,346 • For 50-74 total usage with BCP: $47,733 • For 75-149 total usage with BCP: $87,393

An important licensing element for Blaise is that all licenses are for simultaneous users, not installed instances. Consequently, an organization can manage its license so that a certain level of simultaneous usage is observed over a larger number of users if the timing of use meets the license terms. It is also possible to increase one's usage level during the current license year if needed, paying only a pro-rated amount. The pro-ration for adding Blaise licenses during a license year is the months remaining divided by twelve, multiplied by the annual license charge.

Under the Regular Licensing Plan, the fee per (enumerator) Application user per year is as follows:

- CAPI interview method without BCP: $134/year
- CAPI interview method with BCP: $162/year

When a license plan is obtained, users are entitled to:

- Use of all licensed Blaise software in a single organization up to the usage limits
- Free updates to the software
- Beta versions or other pre-production releases of Blaise Software in order to preview, test, and provide feedback to Statistics Netherlands on emerging versions (only for a corporate license plan)
- Support through email concerning installation, syntax, functionality, appropriate use and other technical questions

Training classes are available at Westat for a fee of $1500 per person per module, except for the Maniplus training which is $1000 per person per module. There are four modules: Basic Blaise Training, Blaise Basic CATI Training, Blaise Internet Services (BlaiseIS) Training, and Maniplus Training. Client

location training is also available, with pricing depending upon the trainer's travel and expenses. (http://www.westat.com/westat/statistical_software/blaise/training.cfm)

Extensibility

The Blaise software does not have a multi-language version interface. According to the Online Assistant, if a user has a special need that Blaise cannot handle, a programmer can extend the system by using the Blaise components. Using Dynamic Link Libraries (DLLs) and ActiveX components, one can also extend the system in various ways beyond the basic Blaise system. DLLs can help display graphics, read data from serial ports, invoke specialized coding software, or perform complex calculations already programmed elsewhere.

It is also possible to add specialized capabilities to the Control Center main menu with the Add-Ins menu selection. Add-ins launch an ActiveX control, which is a framework for defining re-useable software components in a programming language.

Hardware and Software Needs

Blaise only supports the Windows environment (Windows 2000 and up), but Blaise Internet can support any browser (Safari, Internet Explorer, Firefox, etc.). According to the documentation, Blaise works equally well on normal machines as on low-power machines such as UMPCs/MTPCs.

Known software users: Statistics Netherlands; Statistics Finland; Korea Labor Institute; Office for National Statistics (UK); Israel Central Bureau of Statistics; Agricultural Research Service, USA; University of Michigan; CentERdata, The Netherlands; National Center for Social Research, England; Central Statistical Bureau of Latvia; ISTAT, Italy; RTI International, USA; Statistisches Bundesamt, Germany; Innovations for Poverty Action.

5.2 CASES

Programming

Power of programming language: CASES' programming language is powerful in the sense that it covers every aspect of the CAPI survey: from the content of questions, the array of answers, the flow of the questionnaire, and the finer details of implementation (such as which fields are open for answer, when, and under what conditions). It is also powerful insofar as it has commands that govern graphics, including the conditional display of questions and other graphical elements.

Functionality: The programming language allows survey designers to exercise a great degree of control over the flow, content, and manner of implementation of survey instruments.

Limitations: CASES's programming language tries to be readable by both humans and computers (or computer programmers), but fails somewhat in that goal. It endeavors to be intuitive through its use of commands with indicative names (e.g. "goto" governs skips, "choices are" defines multiple choice answer options), but falls short of that goal because of unnatural syntactical elements and complicated rules on where commands must be. It also simultaneously strives to look like other programming languages though the use of structures common in other programming languages (such as if/then constructs, loops, arrays, etc.), but obscures outward similarity by unconventional syntactical elements (e.g. data items require inverted arrows (><), commands require command brackets ([]), and variables have an "at" prefix (@)).

Familiarity: CASES' programming language attempts to make code as readable as prose, but bungles this with strange rules about where certain commands can be placed relative to questions. On the one hand, a CASES instrument coded with its programming language has some immediate intuitive appeal. Questions are created and their answer options defined in rapid succession within programming code. On the other, a CASES program unsettles the expectations of both a general survey designer, by locating skip logic after all of a screen's questions are defined, and of a programmer, by placing the answer option definition commands of some question types alongside skip logic (but not all answer option definitions).

The programming language also has one last bit of a challenge. Because of its power – its ability to exercise control over virtually everything about a survey instrument – it has a dizzying array of commands. Since commands are typically short descriptors of action (e.g. goto goes to, call calls, etc.), they have a large number of options and various syntaxes in order to offer the vast amount of control available within CASES.

Ease of Use: For a general audience, CASES' programming language can be positively puzzling. For survey programmers, it can be overwhelming, since CASES' language exercises every imaginable control over a survey, from developing the database to issuing the skip instructions to dictating the graphical instrument layout. For computer programmers, it can be confusing because of CASES' confusing blend of known structures (e.g. loops, arrays, if/then conditions), strange syntax (e.g. usage of >< for items instead of <>, placement of commands in brackets and question text in none), and irregular rules (e.g. some answer definitions come in the portion of a screen's program that follows the scripting of questions, while others strangely share space with skip logic).

CASES therefore has a steep learning curve, but the software is ultimately easy to use once a user has

mastered its rules, syntax, and structure. Having done this, the program's considerable power can be unleashed.

Development Environment

Integration: CASES offers users all of its development tools in one of two separate design suites: graphical or command line. The user can accomplish all development tasks – from dictionary development to form creation to logic programming – in either one suite or the other. The graphical suite accomplishes this through point-and-click accompanied by menus, while the command line does the same through several commands and options. The development environment is in this sense completely integrated into a tight toolkit provided by either graphical or command line development.

Yet CASES' complete integration also has a downside: the user rapidly becomes overwhelmed with all of the options in either mode of development. The graphical suite contains far too many menus, menu tabs, and options within a given tab. The command line environment requires too many commands, contains too many options, and is too cluttered a program for even relatively simple questionnaires. The result, for all but the master of CASES development, is a rather confusing experience.

Efforts are made to counteract this. The graphical suite tries to separate features into menus of items that belong together. The command line has syntax highlighting to visually underscore that differently colored code does different things. However, the result is still an instrument development environment that is overwhelming for new users, and perhaps also so for more seasoned ones. The visual confusion makes things hard to find; the quantity of commands makes achieving an effect sometimes a long slog through manuals on options and syntax.

Furthermore, CASES' integration is functional but disorganized. Both the graphical and command line development environments put a variety of design tools at the developer's disposal within one interface. However, those tools are hosted, in both design tracks, through a series of related but separate windows, or single windows where important information is often hidden out of sight. In the graphical design suite, the developer must juggle a form manager (which is the central listing of screens displayed to the interviewer), a window for each form (which contains both the graphical layout of the screen and the skip logic), and a command line file (which ties the screens together as a CAPI instrument and provides further commands on how to administer questions). Together, this contains a large amount of information in several free-floating windows. However, the developer has no central display of variables created for the study, whose names are instead hidden individually inside the menus that developed the questions that collect their data. In the command line environment, things are much the same, but the visual confusion instead consists of attempting to find variable names and to parse database structure from countless lines of code, which contains everything from variable names and labels to question text and background color.

While each development mode has strong and well implemented features, the overall development environment is unfortunately less than the sum of its parts. The development environment would ultimately be much more user friendly and effective were the best mode of development used for certain questionnaire attributes and not others, e.g. by leaving graphic design to the graphical suite alone, and leaving the command line environment (which has analogous graphical commands) to skip logic.

Interface for field users

Although CASES' look may appear antiquated alongside that of other packages, its user interface has a great deal of modern appeal. It is touch-friendly, thanks to an abundance of buttons. It is highly customizable, thanks to the power of its considerable graphical and programming design suites. And it is ultimately easy to use for most survey needs, thanks to its lack of software menus and abundance of buttons for survey control.

CASES is touch-friendly in two ways. First, almost all of its question types have a clickable version. Yes/no questions have radio buttons. Multiple choice questions have combo boxes. Second, CASES reserves a total of 24 slots on the periphery of the screen for programmable "hot keys" that provide interviewers with an easy, clickable way to execute most survey operations, such as navigation, changing instrument language, or scheduling a callback.

CASES is highly customizable. Layout controls dictate the position of questions, other survey instructions, and the size of their text. Programming power imbues buttons with the requisite responsiveness to every user's survey administration needs. The result is a user interface that is both easy to use and powerful.

CASES, for the field user, is user friendly. For field use, CASES has no software menus, only survey controls offered within the instrument, mainly by customizable buttons. There is no need to move through difficult to understand menus, for example, to navigate the questionnaire or change the instrument language.

However, these virtues do not come by default. CASES only provides these desirable attributes through programming on the part of CAPI instrument developers. Buttons, for example, must be programmed and linked with a set of system commands or user-written instructions.
Despite the difficulties found in using the programming language, it enables survey designers, through its many commands and many more options, to craft highly user-friendly survey instruments.

Questionnaire Implementation

Supported Question Types: CASES provides designers with a wide range of questionnaire implementation options, and considerable control over how those options function. CASES offers a core of answer types – text box, radio buttons, checkboxes, list boxes, and combo boxes – that designers can tweak and tailor through the program's command language. For example, instrument designers can enable or disable the field user's ability to edit and/or add to the options in a combo box.

Although CASES provides survey designers with a large set of tools, there are still features that are lacking or missing. Lacking is the facility for simple implementation of cascading selections. Although CASES can accomplish this, the programming code for it is not straight-forward. Also missing is an in-built algorithm for assisted coding of text answers into a large set of pre-coded answers.

Question Support Features: CASES provides designers with excellent facilities for supporting questions through user-defined help screens, or user-created pictures or audio files. CASES offers an easy way of providing users a link to a separate help screen to explain questions. CASES also provides a rich facility for associating audio files not only with questions but with the answer options themselves.

The large set of question implementation options, while confusing, can produce a useful instrument. CASES offers many means of displaying and supporting questions and their associated answers.

Questionnaire Navigation

CASES enables users to move freely within areas of the questionnaire that the instrument designer specifically allows--within a screen, across screens, within rosters, and across interlinked rosters. CASES provides clear and effective interviewer navigation controls through buttons that map to navigation commands, menus that facilitate free movement within a survey, and classic forward movement that comes through simply answering questions.

CASES offers compelling non-linear navigation. Interviewers may move freely across the instrument through backward and forward controls. They may also move even more freely around explicitly allowed areas of the questionnaire using a special navigation menu, jumping from one screen to another more distant one. This provides the freedom of movement needed for administering a complex survey.

However, CASES' non-linear navigation has some limits, some of which are desirable through designer controls over the extent of allowed non-linear navigation, and others which are less desirable as a result of shortcomings in the package's capabilities. Thanks to a suite of clever control commands, CASES designers can limit where and under what conditions interviewers can navigate to certain questions. This includes disabling later, related questions until prior, linked ones are answered; restricting non-linear access to sections, for example, to those previously encountered through linear navigation; and selectively disallowing non-linear access to certain sections while access to other sections is freely granted. The CASES command language allows this and other more complex but desirable restrictions.

However, not all limitations to non-linear access are preferable. CASES allows constrained non-linear navigation through aptly named jump menus, which are special navigation menus that allow interviewers to jump freely from one part of the questionnaire to another. Yet this ostensibly promising feature is marred by the way it appears. Rather than an intuitive menu with user-defined labels, the jump menu is a confusing collection of collapsible jump targets that are identified by typically uninformative variable names. The navigation capability is there. However, its flawed implementation constrains its utility for field users.

Case Management

CASES provides a complete suite for case management for all levels of hierarchy in the survey organization.

Enumerators: For field interviewers, CASES provides a relatively intuitive interface for seeing the status of and following up on all assigned survey cases. From the case management view, an interviewer can sort through his/her survey assignments and start or resume interviews.

Supervisors: For supervisors, the case management dashboard is where cases are received from headquarters and assigned to interviewers. This portal allows supervisors to receive case assignments from headquarters, to update the status of cases, and to make assignments to interviewers.

Survey Managers: For survey managers and study designers, CASES offers tools for designing the case management system as well as for utilizing it to make work assignments. At the inception of the project,

study designers can use CASES' toolkit to define the variables that will be tracked for survey cases, and what the case management interface will look like for users at lower echelons of the survey hierarchy.

During field operations, the case management view can be used to assign work assignments to supervisors and to track progress.

CASES also has unique features. It offers vendor neutral tools that allow CASES' case management interface to work with third party CAPI software. The system simply requires gleaning a few small bits of information from the third party CAPI software and passing it to the CASES case management system. This unique features has already been tailored for Blaise and Egonet. However, the tools are generic enough that a technically astute user who understands how both the third party software and CASES system work could develop this tool for other packages.

Data transfer

Server Overview: CASES uses a relational database to store the collected data. The software uses the popular MySQL as the database for the server. By default, CASES sets up the server locally on a computer designated by the user. Users have the ability to designate up to nine computers as the central server. After setting up the server computer, users then connect the computers the interviewers will use (known as "clients") to the server by entering certain information about the server into the CASES version on the clients. When this task is performed successfully, the client computers will be able to communicate with the server.

Data Security: CASES provides the ability to encrypt not only data other files associated how the survey is completed.

Data Transmission Protocol: CASES has a Synchronizer application which is the main mechanism for transferring data between the server and clients via the Internet. When users click on the Synchronizer application icon, CASES automatically uploads newly collected data to the server. The Synchronizer menu displays the names of files that have been uploaded to the server in that session. One limitation is that Synchronizer will encounter errors when there are discrepancies in questionnaires between sessions such as significant changes to variable properties. This limitation could be a severe hindrance if a questionnaire must be significantly altered during the course of data collection.

Data Exporting

File Formats: The only data format that CASES can export into is ASCII. Along with the data, CASES will also automatically export an accompanying layout file that includes information about the type (i.e. string, number), column and width of each variable. The user also has the option of exporting a Data Definition Language (DDL) file that contains descriptions for its matching data.

Export Protocol: Users use the Output application to export data into an ASCII file. This can be performed through a simple point-and-click interface. There is an option in the Output application menu to also export the matching data description file for the data.

Features: Although CASES cannot directly export data into a Stata, SPSS or SAS file, it does offer a workaround for this limitation. The data description file, the DDL file, is utilized by popular statistical software packages to append labels to the variables and values to the ASCII data file. Through this

combined method, CASES is able to export data with variable and value labels that can be read by Stata, SPSS, and SAS. This method also allows users to export data with variable labels into an XML format.

CASES also provides users with the option to export either all the data from a survey questionnaire or only the data associated with select modules or variables. The option is available in the point-and-click interface of the Output application menu.

Support & documentation: CASES provides thorough support and documentation. It has an introduction for beginners and a more extensive reference manual. It has downloadable examples that offer excellent demonstrations of questionnaires constructed purely from command line programming, of arrays and loops properly used in practice, and of how to associate audio files with questions and their associated answer options. It has a thorough tutorial that instructs beginners through most of the aspects of programming and instrument development, with video tutorials on the way. It has a user group that operates through a listserv. Unlimited technical assistance from the CASES developers is also included as a part of the product price.

However, quantity does not equal quality. Some pieces are excellent. Direct technical support is exceptional. The beginner's guide, which offers a welcome software walkthrough, does an outstanding job of familiarizing new users with the CASES environment. The resource programs, although short on instructive comments, provide clear examples of how to implement several features, functions, and questionnaire frameworks. However, the reference manual fails to pick up where the beginner's guide leaves off. It fails to provide a useful detailed overview of how CASES works, how its pieces fit together, and the broad rules that govern the system. Instead, it confounds the reader with the excessive exceptions, strange quirks, and legacy features of CASES, rather than a straightforward synthesis of CASES' capabilities.

Pricing & Upgrades

Unlike other CAPI software packages, CASES cannot be purchased but rather must be obtained by paying for an annual membership in the Association for Computer-assisted Surveys (ACS). In addition to obtaining CASES, ACS members also receive software upgrades, access to online resources, and unlimited remote technical assistance through e-mail and phone, all at no additional cost.

The annual membership fees are dependent on a sliding scale of pricing based on scale of usage measured in staff hours. Users sign up for membership in a certain category with a predetermined fee and amount of usage, and users are responsible for alerting the CASES developers if they exceed their membership category's predefined level of usage.

CASES has the following use-based membership rates:

- $1,500 per year for each PC, with no web surveys or SDA use
- $4,100 per year for 2,000 to 4,000 staff hours of use and up to 100,000 web questionnaires administered
- Higher annual rates for greater staff hours and numbers of completed web surveys

For more information about the pricing scale for CASES, please refer to:
http://cases.berkeley.edu/membership_fees.html

Also of note is that association membership provides members access not only to CASES but also to Survey Documentation and Analysis (SDA), a feature-rich program providing online documentation and analysis of survey data.

Extensibility

CASES has no capability to allow users to extend the functionality of the software, a common downside of packages with proprietary programming languages.

One unique feature of CASES is that it contains a vendor neutral case management system that may be used with other third party data collection software programs. The case management system requires a different configuration to be compatible with different third party data collection programs. CASES 6 can currently be configured to be compatible with two data collection software programs: Egonet and Blaise. One important limitation of this feature is that configuring the system is complex and even when correctly configured, the system might still not be fully compatible with all the available data collection software programs. Regardless, this feature would have the potential to add case management features to other CAPI software programs that do not have them natively.

Hardware/software needs

CASES has very modest hardware and software requirements that fit the type of devices needed for most CAPI deployments.

For hardware, CASES requires a 466 MHz process, 64 MB of RAM, and 55 MB of storage space.

For software, CASES has two requirements: Java, and a modern Windows operating system, such as Windows XP, Windows Vista, or Windows 7.

Known software users: Survey Research Center at Penn State University; Missouri Family Study (MOFAM) at the Washington University School of Medicine; National Election Studies at the University of Michigan; Bixby Center for Reproductive Health & Research at UC San Francisco; State Health Access Data Assistance Center (SHADAC) at the University of Minnesota; Caucasus Research Resource Centers at the Eurasia Partnership Foundation (Tbilisi, Georgia); University of Thailand Chamber of Commerce (Bangkok, Thailand).

5.3 CSPROX

Programming

Power of programming language: A rare achievement in software, CSProX boasts a concise programming language that marries simple syntax with strong programming power. Interactive error correction is a prime example. This function relies on a compact command that checks a user-defined logical condition (if-then), issues a user-scripted error message if the condition is violated, and provides user-specified options of how to resolve the issue by providing associated buttons that point back to potentially errant answers. In CSProX, this functionality is accomplished with a few lines of code, easily half of what is required by other programs.

Functionality of Power: CSProX provides a broad array of advanced functionality. It can handle simple and complex rosters, call both internal and external databases, and both display and capture certain media.

To enhance functionality, code can be created for all data and survey design elements (e.g. data fields, groups of fields, and screens of fields). Code associated with individual data fields has an effect solely upon those fields. Code associated with groups of fields (e.g. rosters, screens of a survey) affect all of the data elements in the group. In this way, a few simple lines of code can have a large effect.

Limitations of Power: Despite its power and simplicity, the CSProX programming language has some serious limitations. There are two notable features that it cannot do well: non-linear completion and the calling of external devices.

Designing non-linear navigation of questionnaires is quite difficult to program in CSProX. The software is more adapted to PAPI data entry than to CAPI applications. For this reason, commands are executed in three series of sequential events (pre, present, post) that work faithfully during forward navigation of questions, but break down when navigation is non-linear. Even if the programmer wanted to design a non-linear survey, CSProX does not offer navigation features that allow enumerators to advance screens or to skip sections.

The second flaw is the programming language's limited ability to interface with and capture the output of peripheral devices such as GPS. While CSProX does provide a command for capturing GPS coordinates, this function only works on the Windows CE operating system. Additionally, the software provides theoretical versatility for using other peripherals, but does so through launching the peripheral device's program, thus lacking the ability to pull in and store the captured data within CSProX.

Familiarity: CSProX's language has structural and conceptual similarities to common computer programming languages (e.g. "if/then/else" structures that implement commands under certain conditions, "while" loops that execute commands iteratively while a condition is true, and arrays that capture lists for later use in the program). Despite this, CSProX's programming language remains eminently approachable for the less tech-savvy.

Ease of Use: CSProX's core programming language, inherited from CSPro, offers a limited set of commands that still provide a broad array of functionality, compact commands that accomplish complex tasks with few lines of code, and commands with evocative names that make coding easy for the less technically astute (e.g. "skip to" provides skip instructions from one question to another).

Nevertheless, with a handful of notable exceptions, the simplicity of the core commands does not carry over to CSProX's specialized functions and features. Specialized commands tend to have considerable programming complexity that requires a higher level of programming prowess. Unfortunately, these commands tend to be more sparsely documented.

Development Environment

Integration: The CSProX development environment is comprised of three pieces: the data dictionary, which defines questions and their answers, the form developer, which governs the digital questionnaire's graphical layout, and the logic, which dictates questionnaire flow and data validation. Each tool is accessible as a separate tab within the questionnaire development environment. CSProX thus provides a tightly integrated development environment comprised of strong tools for each piece of the questionnaire development process.

Nevertheless, the functional division of the development environment into a tripartite system can often make development harder rather than easier. For first-time users, the layout is less intuitive and harder to learn than fully integrated environments. For experienced users, this tripartite layout can often facilitate work on particular tasks within a functional area (e.g. copying and pasting "yes/no" as answer options for several questions), but can equally often make simple modifications unnecessarily complex (e.g. deleting a question requires deleting it not just from the graphical view but also from the data dictionary).

Furthermore, this imposed division of the development environment also imposes a consequent workflow on developers which may be far from optimal. For example, questionnaire designers would generally prefer to write questions and their answer options in the same view. Yet CSProX divides this single task into several tasks within several different parts of the development environment. First, one must create the question (as a variable) in the data dictionary. Having done this, one may add answer options as value labels. Next, one must draft the data element onto the questionnaire view. Once this is done, the developer can finally draft the question text in a sub-view of the layout developer.

Survey Creation: The form designer determines how data elements defined in the dictionary appear on the screen and are presented in the interview, i.e. a survey's graphical layout and a question's content. The process is relatively simple and is accomplished by using an intuitive drag-and-drop interface for arranging data elements on separate screens and attaching question text to the respective data elements.

Logic Programming: CSProX's logic programming provides the questionnaire application with instructions on when questions are to be skipped, which answer options are applicable, and whether a given answer makes sense in light of previous answers, data from previous sections, or even answers from previous surveys. Logic can be attached to any "data node" – that is, a group of variables, such as a roster, a given screen, or the whole questionnaire. Logic can also be viewed, usefully, as a single program that draws together commands from all levels of the survey.

Dictionary Creation: Using the data dictionary, the survey designer creates variables to capture the answer to each survey question. Within the data dictionary, the designer determines the allowable range of answers, and potentially the pre-scripted answer values (e.g. 1=yes, 2=no, etc.) for each section. For questions or groups of questions, the designer determines the maximum number of times a

question is asked of a survey unit (e.g. the maximum number of household members for which demography questions can be asked). This interface is also where the survey designer applies variable and value labels.

Application Testing: The application tester provides the developer a preview of how an instrument will work. It compiles all program logic, loads all forms, and runs through a questionnaire as with regular data collection. It allows the developer to check that all code works as desired and that no errors arise during data collection.

Interface for field users

Interface Aesthetics: Although its default look is a bit antiquated, CSProX's data collection interface is customizable. Users can easily change the size of question text and labels, and even the background color of the questionnaire. CSProX also allows users to customize the view for different screen sizes. Questionnaires can either be optimized for large screens (i.e. with several questions on one screen) or small screens (i.e. with only one question appearing at a time). However, CSProX is not touch-friendly and it does not have navigation buttons, command buttons, or other user-defined questionnaire features that aid computer interviews on devices without keyboards.

Questionnaire navigation: CSProX's navigation interface is limited. This is most likely due to its early life as a program for processing paper surveys. For example, to move forward, an interviewer can either answer the last question on a screen, press the Page Down button, or select Next Screen from the software menu. In addition, non-linear movement within a CSProX questionnaire is only allowed within areas that the interviewer has previously visited. An interviewer may not jump forward or otherwise bypass this strictly sequential progression. The only exception is if a designer has built skips or other enablements into the survey.

Questionnaire Review: CSProX does not feature a set-aside questionnaire review interface. Instead, the questionnaire review takes place in the same setup as data collection. This imposes a few notable limitations. First, the review interface suffers from the same navigation restrictions as the survey does in data collection mode – namely, the inability to move freely within the instrument and jump from one section to another.

Second, the review interface, just as the data collection interface, only shows the codes for answers to pre-coded questions. While this point may seem minor, its practical implications are major. Should a supervisor want to review a questionnaire, doing so would involve clicking on answered fields in order to see the non-numerical response. By clicking on the answered field, the supervisor risks accidentally changing valid survey answers.

Language Capabilities: CSProX allows multiple language versions of the survey – for Romance languages like French and for limited character-based languages like Arabic. Serpro is planning to convert to Unicode in order to expand language possibilities in the future. The developer codes multiple language versions on the same screen as the primary language. For the enumerator, language options are available only through menu options or through keyboard shortcuts. Unfortunately, CSProX does not offer any options for changing the language of the software interface.

Questionnaire implementation

Supported Question Types: CSProX offers six different answer types: radio button, check box, drop down, combo box, text box, and date. It also provides a complete array of question options to survey designers, such as filtering and cascading selection. Yet the full array of options is not always and everywhere available. Radio buttons and checkboxes are oddly only available for PDAs or smart phones that have a Windows Mobile operating system. Additionally, CSProX is not able to interface with and capture data from peripheral devices.

Question Support Features: CSProX offers an array of ways in which its questions can be supported. It allows designers to include interviewer instructions in the question text and to offer question-specific help files on demand. For question clarification, a survey can be programmed to display images. However, the package falls short in that it does not allow audio or video to be embedded in the instrument or different versions of the image to be displayed for the various translations.

Questionnaire navigation

From the field user's perspective, questionnaire navigation (apart from simply advancing forward within an instrument) can be clumsy. Navigation relies heavily on the keyboard (PageUp, PageDown), the program menus (Navigation menu, GoTo option), or programmed navigation fields that allow movement within the questionnaire. The enumerator can move backwards and forwards through a questionnaire and module, but will face problems if they desire to skip around the survey in a non-linear fashion. CSProX's logic will be interrupted if a question is not moved through in a sequential order, or will become harder for developers to draft if non-sequential completion is allowed. To alleviate the issue somewhat, CSProX can provide a visual display of all questions and corresponding answers. However, this visual display is done through an optional navigation pane that has some shortcomings in its implementation. Questions are arrayed in a case tree, which may not immediately intuitive for field users. Answers to these questions are listed as numbers rather than as their associate labels (in the case of pre-coded questions).

Case management

CSProX has two case management systems. One is in-built and limited in functionality. The other is a separate set of applications that attempts to replicate the case management systems of CAPI programs like Blaise, CASES, and MMIC.

The in-built one is well known – it is simply CSEntry, the CSProX program that captures and stores data. This system provides a simple dashboard of cases that enables interviewers to determine at a glance which cases are complete and which are incomplete.

Yet despite its seeming utility, the dashboard suffers two shortcomings that substantially limit its utility. The first is that the dashboard only provides a rough tag for each case, a concatenation of the case's key identifiers that is often difficult to parse in practice. This makes it hard for enumerators to quickly find interviews that need to be completed or reviewed.

The second is that the dashboard, as with other CAPI software, does not offer a more granular view of a questionnaire's level of completion. This makes it difficult for interviewers to evaluate how much work needs to be done to complete these cases at a later point.

The separate program is called CSProX Control, which is a set of applications that have been used for production surveys but have not yet been released as an off-the-shelf product. CSProX Control emulates the more extensive functionality of programs like Blaise, CASES, and MMIC. At the survey manager level, it defines which survey teams are allocated which clusters. At the supervisor level, it defines which interviewers are allocated which households. In this sense, it defines work assignments in a top-down structure.

However, the seemingly useful control structure of CSProX Control is currently rigid, and perhaps too much so for fast-moving surveys. CSProX Control relies on a strict assignment of work. It generates a file that restricts assignments. It must regenerate that work assignment file if any change in assignment takes place. As a result, CSProX Control works well when these assignments are strictly followed, but breaks down slightly when there are deviations.

Serpro, the developer of CSProX, is currently working on making this system more flexible.

Data transfer

Server Overview: CSProX offers an integrated server system with features that are appropriate for both the context and sensitive content of LSMS-ISA surveys.

However, CSProX does not appear to offer a dashboard that provides information on the success of data transfer, or any automated analytics (e.g. histograms) for data received from the field. This shortcoming is partially overcome through CSProX's tabulation features that allow users to execute highly customizable and complex tabulations of data gathered from the field.

*Data Security***:** CSProX's server system makes data transfer secure. It implements the Secure Socket Layer (SSL) protocol to encrypt data using a key that is only known by the server receiving field data.

Data Transmission Protocol: Data files can be transferred using the internet or any data storage devices such as a flash drive or CD.

Data Exporting

File Formats: Survey managers can elect to export only certain user-specified segments of data. Users can also easily define which type, or types, of data files will be exported (i.e. tab delimited text, comma-delimited values, CSPro, SPSS, SAS, or Stata). It cannot explicitly control the data layout (i.e. data in wide or long format).

Transfer Protocol: CSProX provides a simple and powerful data exporting facility. Using an intuitive menu-based interface, data managers can exercise considerable control over how, and how much, data are exported.

Features: CSProX generates program codes that apply variable and value labels.

Support and documentation

CSProX offers a wide range of support options. Through CSPro, CSProX offers a basic tutorial, a

comprehensive user manual, a free helpline with the US Census Bureau[17], and an array of well-commented example applications that come with the software installation including several that highlight CAPI functions. Through its licensed version, CSProX offers limited documentation of the additional functions (not available in the unlicensed CSPro) but excellent technical support through Serpro's software engineers (a service that is free for limited questions but fee-based for larger requests).

However, CSProX's support and documentation is slightly lacking in two dimensions. First, despite the broad menu of support options, there is one critical piece missing: a user community that supports the software and suggests extensions to it. This may be the case because most CSProX users choose to pay for Serpro's training services rather than to learn the software package from publicly available resources.

Second, the considerable breadth of support options at times belies a lack of depth. While the CSProX manuals and help files cover virtually all functions and commands, they often lack explanations and examples that flesh out concepts or demonstrate implementation. This is particularly true of functions that are unique to CSProX, and leads users who do not enlist Serpro's technical support to an often frustrating trial-and-error approach to programming.

Pricing and upgrades

The CSProX suite is composed of three packages: Developer, which governs application development and designs the CAPI questionnaire; Client, which runs applications on data collection machines; and Server (an optional component), which transfers data from the field to the home office. The pricing is as follows:

- Developer: $2000/machine
- Server: $5000/machine
- Client: $250/machine[18]

The licenses are perpetual for non-profit institutions and annual for commercial organizations. Pricing, particularly for non-profits, is quite competitive compared to other software service providers.

Upgrades are fee-based and irregular in frequency.

Technical assistance can be obtained on an as-needed basis for $800/day.

Extensibility

CSProX does not allow for user-written extensions to the software package. The programming language does not provide users with the extended language tools for creating user-defined functions that might

[17] This support is for CSPro only. However, the similarities between CSPro and CSProX may make the CSPro helpline useful for general questions that CSProX users may have. This helpline cannot, however, provide support on the features and commands specific to CSProX.

[18] Pricing is done on a sliding scale. The highest price is quoted above. For purchases of 1 to 99 client licenses, the unit price is $250; for 100-199, $200; for more than 200, $150 each.

extend CSProX's functionality beyond its in-built commands. The software is also proprietary, and Serpro does not have any current plans for opening CSProX's source code to the user community.

That being said, Serpro does make periodic updates to CSProX. These updates include both incremental improvements (e.g. extensions to existing commands) and major modifications (e.g. a browser-based data capture tool) to the software. Serpro is currently planning the following improvements:

- CSProX Light Client, a browser-based data capture application
- New programming elements to facilitate non-linear CAPI administration
- User functions library
- Full conversion to Unicode, to support a broader set of languages
- Tools for recording CAPI interviews as MP3 or OGG files[19]
- GPS capture in operating systems other than Windows CE[20]

All updates to CSProX are fee-based.

Hardware and Software needs

CSProX works on a wide array of devices, and notably on low-power UMPCs/MTPCs and on some smartphones/PDAs. CSProX works well with Windows OSes for PCs (Windows CE, XP, and 7) as well as Windows OSes for phones/PDAs (Windows Mobile 5 and 6).

Known software users: SEI-Consultores, Colombia; Universidad Alberto Hurtado, Chila; Macro International; World Bank; IRIS Center at the University of Maryland.

[19] Feature available in CSProX 4.1.
[20] Feature available in CSProX 4.1.

5.4 ENTRYWARE

Programming

Power of Programming Language: Entryware's programming language is powerful, but a bit unwieldy for more complex functions. The program is essentially a fusion between a scripting-intensive survey software language and a simple point-and-click program. Moreover, Entryware's scripting language is proprietary. Therefore, the user is essentially limited to the basic capabilities recognized by the program. There is only very limited scope to use some Visual Basic to provide functions that Entryware's proprietary language does not provide.

Functionality of Power: Entryware excels at simple tasks, such as skips, uniform range restrictions, and data validations. It has even made incorporating multi-media into questions and response lists a simple point-and-click process. However, complex functions, such as rosters, looping, dynamic range restrictions, as well as calling external databases,[21] are its downfall. Accomplishing these functions requires tedious workarounds and more lines of code as compared to other programs.

One rare feature that Entryware offers is the ability to randomize or rotate response lists for multiple choice questions to help prevent response bias.

Limitations of Power: There are three serious limitations to using Entryware for a large and complex survey. The first is that any survey conducted using the program will be very long, as the interface only allows one question at a time to be displayed on the screen. Unfortunately, there is no option to change this functionality using scripting.

Second, looping and rostering are not as dynamic as in some of the other programs. When a loop is created, a programmer sets the number of iterations of the loop, up to 99. In addition, the programmer can set a variable to count the number of iterations and create an argument that will terminate the loop once this number is reached. When this number is not known in advance, such as with a household roster, problems arise. The loop can be programmed to terminate when a blank answer is submitted for the first question in the loop by the enumerator. However, this method is not ideal, as it leaves behind a blank entry in the data set. In addition, creating a roster of captured data from a loop requires extensive programming that is prone to error.

Lastly, as calling external databases requires the use of Visual Basic or Javascript, it is only possible in Windows operated devices. This limits survey deployment to devices running this operating platform if this functionality is desired. In addition, the programmer is limited to calling one data point at a time, exluding complex dynamic sorts and cascading selection from the range of possibilities.

Familiarity: Entryware's programming language resembles common programming languages, such as Javascript and Visual Basic. While persons familiar with those programs will have an easier time with Entryware, the 'User Guide' gives a thorough overview of programming and questionnaire design and there are several examples provided to help novice programmers understand more complex features.

[21] Soda, a smart-phone only survey software product by Techneos with many of the same features as Entryware but not reviewed for this paper, does support the pre-population of data from previous rounds of a survey.

Ease of Use: Questionnaire design and programming logic (called 'scripting') in Entryware are both simple and easy to use. Questions are created through an intuitive point-and-click interface located directly above the area that logic and skip patterns are written. A programmer must first insert a question, define the variable name, define the question text, apply uniform range restrictions, formatting, etc. Then, in a box below the questionnaire design section, they can add programming logic that can accomplish dynamic range restrictions, skips, data validation, etc.

The novice programmer can use the built-in 'script-builder', which allows one to choose the variable and the Boolean logic from a drop down list. The written script is then inserted into the script editor. More experienced programmers would likely bypass the script-builder and program directly into the script editor, as Entryware's language is similar to many of the common coding languages. Hand-coding allows for more complex functions, such as filtering and setting up a roster, that extend beyond options available in the script builder.

Development Environment

Integration: Entryware's development environment is fully integrated and extremely intuitive. Survey creation, logic programming, dictionary creation and application all occur within the Design tab on the Entryware Designer software interface.

Survey Creation: The first step of designing a survey is to insert a question, which prompts a screen to appear that allows the designer to define the answer type and name the question and its 'alias' (variable name). Once completed, the designer is presented with a screen with space to write out the question prompt and define other aspects of the question, such as the response list, font type, size, etc. Once this is completed, scripting can be added to the bottom and the question properties can be changed (e.g. by limiting the range of answers) on the side.

Logic Programming: The programmer writes the accompanying logic on the same screen that the questionnaire is created. Logic can be implemented before the question or after the question. In addition, simple range restrictions, filtering options, and formatting options can be selected in the 'Question Properties Window.'

Dictionary Creation: The program automatically creates the associated variables, called the data dictionary, as the survey is created. The programmer has the option of customizing variable names in the process.

Application Testing: Checking for syntax errors in Entryware is easy to do. There is a button that can be tapped for a syntax check of a single question. Entryware will also check the logic of the current question before it will allow you to proceed to a different question. Moreover, it always checks the syntax of the whole survey before it allows you to proceed to the survey simulator. The syntax checker acts by highlighting the error and providing a brief explanation of the problem, but does not link the programmer to help files for further assistance.

Interface for Field Users

Interface Aesthetics: The Entryware Mobile interface is very professional looking and is extremely navigable. Whether viewed on a computer, Palm, or Windows Mobile device, it always produces uncluttered surveys because the survey rescales automatically to accommodate different screen sizes.

Entryware can even take advantage of devices equipped with touch screens. The only major drawback to the interface is that the program only allows one question to be displayed at a time, with no option to change this functionality.

Questionnaire Navigation: The questionnaire navigation interface allows the enumerator to either tap the forward and backwards button to move through the survey. Entryware also facilitates non-linear completion by allowing the enumerator to use the pop-up navigation pane to skip to specific questions or modules. Unfortunately, the ability to move across different instances in a loop is not available in Entryware Mobile.

Questionnaire Review: The enumerator can check whether a survey is complete on the main screen. However, checking the level of completion or the parts of the survey still remaining to be completed is not an option. The enumerator would have to skip, one-by-one, through all of the questions to check information or fill in missing answers. For this reason, non-linear completion, though easy to use, creates more work because there is no way to see what information is missing.

Language Capabilities: Entryware Designer features the ability to create questionnaires with multiple language options. To add a language, the designer must select the language and export the questionnaire, fill in the translation, then import the survey back into Designer. The program will recognize any language supported by Microsoft Windows. Switching the survey language is easy – the enumerator need only change the language in the menu options.

Additionally, Entryware includes the option of having the menu options and system messages in Chinese (Simplified—China), English, French, German, Portuguese, and Spanish. The ability to change system messages is in-built, limiting the user to what is already available from Techneos.

Questionnaire Implementation

Supported Question Types: Designer, Entryware's design software, offers several different answer types to choose from, which include open-ended numeric and text, categorical list and dropdown/grid, multiple response, ranking, date and time selector, signature capture, barcode scanning, and sliding scale. In addition, Entryware offers non-traditional answer types. The questionnaire can easily be programmed to allow the enumerator to capture images and audio as part of the interview process. Entryware can even capture a signature if you have a touch device with a stylus. However, it is not possible to capture video,[22] call external devices, or capture GPS readings using Entryware.

Question Support Features: Entryware provides many options for helping the enumerator conduct a high quality survey that surpasses the ability of paper surveys. To help inform respondents, questions with an image, video, or even an audio clip can easily be embedded in the question prompt or in the multiple choice question options. However, there is no option to change the audio and video files for different language versions of the survey.

Entryware does not hold up when compared to paper interviews in terms of the ability to make notes in the survey. While it is possible to allow interviewer comments to be added to the questionnaire or to the individual fields, it would require creating a new question (for each question) to accommodate comments. As a result, the questionnaire would double in size.

[22] Soda, mentioned in the previous footnote, can capture video if the survey is implemented using an iPhone.

Questionnaire Navigation

Entryware's questionnaire navigation interface handles basic navigation extremely well, allowing one to move backwards and forwards through, within, and across modules and screens with ease. Navigation is achieved by clicking on icons and menus on the survey screen. The user can either point and click or touch, if the survey device is touch-enabled. The survey can also be programmed to provide a navigation pane for jumping to different questions or different sections of the survey without disturbing the programmed question logic.

While simple navigation is completed with ease in Entryware, more complex navigation is lacking. For instance, the ability to jump to different questions and sections allows the enumerator to complete an interview in a non-linear fashion. However, non-linear completion is not a suitable method to complete a survey in Entryware because the enumerator interface does not list unanswered questions within a survey. In order to complete a non-linear interview, the enumerator would have to tap through the entire survey, question-by-question, to locate any unanswered questions. In addition, Entryware falls short in terms of moving across looped instances. The only way to navigate through a set of looped questions is to tap through them one-by-one. As such, Entryware is best suited for flat, linear surveys.

Case Management

Enumerators: The case management features of Entryware are very basic. Enumerators can see whether the interview is complete or incomplete, as well as change completed and incomplete interviews at any time. There is no option to view the progress of specific parts of the questionnaire, such as a roster or module.

Supervisors: The case management interface for project supervisors is cloud-hosted via the server provided by Techneos. After data are synced back to the server by the enumerator(s), supervisors can view the data, but the data cannot be changed through the online server interface.[23]

Survey Managers: Survey managers will face the same limitations as the supervisors in terms of data monitoring. However, it is possible for the survey manager to set reminders by time of day or a time relative to when the survey is being completed using scripting. The only other exception is that the Entryware server allows the survey manager to set and monitor quotas for quota sampling. The server does not however provide the survey manager with the option to monitor the completion of assigned tasks, completion of assigned enumeration areas, or to track the location of individual enumerators or teams.

Other features of the cloud-hosted data management platform include the ability to create very simple reports (e.g. bar charts for answer frequency) and to provide enumerators access to projects virtually.

Data transfer

Server Overview: Transferring data from the survey devices is both safe and easy using the cloud-based server provided with the Entryware Design user license. There is no set-up required, the user need only log-in and upload the survey. However, to allow enumerators access to the server, each device has to be

[23] Soda, Techneos' other survey product, does offer more extensive management options, such as messaging and increased flexibility with the assigning and monitoring of tasks.

synced and access has to be granted through the server.

Data Security: Data are always safe using Entryware. The files are stored using a proprietary storage format and data are encrypted upon transfer using a 56-bit DES encoding algorithm.

Data Transmission Protocol: If the customer chooses to use their own server, the data can be transferred through syncing, Bluetooth, email, flash drive, or through the data transmission cable included with the handheld device.

Data Exporting

File Formats: Entryware allows data to be exported in SPSS and CSV formats. From these formats the data can easily be transferred to SAS or Stata. The data can also be downloaded in Entryware's proprietary format, MPR, which makes transferring data back to Entryware Designer very easy; other options include Access and ASCII. Reports created via the cloud-hosted server can be exported to PDF, Word (RTF), HTML and Excel. Regardless of the file format chosen, Entryware can only be exported as a wide file.

Transfer Protocol: Data can either be transferred from the survey device by clicking the sync button on the main interface or by using a flashdrive. It is possible to program the survey to automatically sync upon completion of a survey, but this must be hand-coded in the design phase.

Features: Data transferred to SPSS will contain both value labels and variable labels. Surveys with multiple language versions will export the data in the language used during the survey, thus allowing for multilingual value labels included in the same dataset.

A somewhat rare feature among the packages reviewed, Entryware provides an internal identifier for each variable. This allows the questionnaire designer to rename, delete, and add variables at any point in the survey implementation process. The software always knows how to match up variables based on the internal identifier. This means that old surveys are still usable, as they will have a missing value for new questions, and new surveys will have a missing value for deleted questions.

Support and Documentation

Techneos offers free technical support via email or phone during an active contract period. They are available weekdays during normal business hours using Pacific Standard Time. The Techneos support staff is very prompt, professional, and helpful.

Entryware's documentation is quite comprehensive. Techneos has even made several mini-tutorials/examples with sample code available for some of the more complex survey programming options. In addition, there is a section on the Techneos website where popular help topics have been posted with the option to ask a help question. Entryware does not have a user group at this juncture.

Unfortunately, Entryware's documentation and help features are provided solely in English, though arrangements can be made for assistance in Spanish. Emails can be answered in different languages, but this will be facilitated using an online translator.

Pricing and Upgrades

Entryware's price tag is one of the most negative aspects of this software. The designer software costs $4,500 annually per single-user license and includes use of the Entryware server and online data interface. It also includes unlimited technical support via phone or email and two hours of web-based training tailored to the client's needs. Additional training can be purchased at $150 per hour (2-hour minimum) for web-based and $1,750 per 8 hour day, which includes a 1hr lunch, plus travel expenses for onsite training.

The interviewer software is $585 annually per single-user license.

It is possible to negotiate lower per unit prices for multiple single-user licenses, as well as multi-year contracts. There is also a discretionary discount available for nonprofit and academic institutions.

Entryware does perform minor updates to the software, which are free and unnoticeable to the client. Major updates, such as those that warrant a new version, require a new licensing agreement.

Extensibility

The functions of Entryware can be extended by attaching an external script file to the questionnaire (limited to Windows operated devices).This file allows one to perform several different lookup functions, such as in a database or on the web. However, database look-ups are limited, as it only allows you to call one piece of data at a time. The extensible script file also facilitates the use of custom calculations, advanced scripting, and creating standard functions.

Hardware and Software Needs

Hardware and software requirements for Entryware Designer are very standard. Entryware Mobile operates on devices running Windows 98 and higher, devices using Palm™ Desktop software 3.1 or higher, and on Windows Mobile 5 & 6 PDAs and Smartphones.

Future Developments

At this time, Techneos does not have plans to significantly alter the functionality of Entryware. The company is tentatively planning to release a version 7.0 within the next year. However, the details on the new functionality that will be added to Entryware 7.0 are not available at this time.[24]

Known software users: e-academy Inc; Whistler Tourism; GfK Indicator; Mindshare; Synovate; Institute for Fiscal Studies; Canada's Wonderland; Ipsos Reid; The Planning Edge; King Brown Partners; Luminosity Marketing; Canadian Sport Tourism Alliance.

[24] Based on email exchange with Nicolette Chin-Shue, Entryware.

5.5 MMIC

Programming

Power of Programming Language: MMIC is an incredibly powerful and flexible survey software program. MMIC's relative strength is the result of two key features. First, it is open-source. Second, in addition to its own native language, MMIC accepts and performs the functions of several web programming languages: Javascript, PHP and HTML. The result of MMIC's open-source structure and its ability to co-opt other programming languages is a program with a plethora of possibilities for customization.

Functionality of Power: MMIC's native language can be used to accomplish simpler functions, such as simple skips and validation. It can also be used to achieve more complicated functionalities, such as incorporating reference responses from early in the survey into later parts. However, a questionnaire programmed exclusively using the native language would be limited by its inability to incorporate any of the more sophisticated functionalities that would be desired in a complex questionnaire, such as calling to data in an Excel file.

Developers can employ the three web programming languages to create questionnaires that contain complex functionalities, such as calling external databases, performing complex skip patterns and checks, validating data and assisting in filling out the questionnaire, integrating third party devices and creating a navigation panel.

Limitations of Power: MMIC is fully capable of accomplishing all of the requirements of a survey with the magnitude of LSMS-style surveys. Any limitations beyond these capabilities were not readily detected in the review process.

Familiarity: MMIC's native language is simple and reflects the programming language of Blaise, another CAPI software under review. Javascript, PHP and HTML are the three web programming languages recognized by MMIC and can be used to perform more complex functions.

Ease of Use: MMIC's native language is simple to learn for the inexperienced programmer. A developer with a background in other programming languages would be able to program in MMIC with great ease. However, the consequence of MMIC's flexibility is its relative complexity. The successful creation of a questionnaire with complex capabilities requires the use of the three web programming languages accepted by MMIC, thus requiring the skills of a programmer with expertise in these languages.

Development Environment

Integration: MMIC has a fully integrated development environment that allows both the programming and testing of the questionnaire within the same interface. The development environment uses a simple side menu bar to make navigation between different components easy. However, some of the components on the side menu bar lack clarity and the documentation does not have any information on their function, or on the development environment in general.

Survey Creation: The creation of questions, and their corresponding variables, is done through a menu where the prompt, question type, layout and other features are also defined. Once created, these variables can then be referred to when scripting the questionnaire.

Some inputs in the menu also accept the aforementioned web programming languages, so these too can be used to customize the features of questions. However, the information in the documentation about the question menu is extremely limited so it is unclear what is accepted in the various inputs in the menu.

Logic Programming: MMIC does not offer a drag-and-drop interface or a point-and-click menu to access program-created syntax. Every part of the questionnaire has to be scripted, including the question order, skips, and restrictions.

Data Dictionary Creation: The data dictionary is automatically created as the questionnaire is being developed because the variable is created once its corresponding question is created.

Application Testing: To assist the programmer, the programming environment highlights the syntax of the code as it is being written. Additionally, it is possible to check the syntax of the code for errors without having to run the questionnaire. Error messages state where the error is contained in the code and briefly describe the cause of the error. However, the error message does not link errors to documentation that would assist the programmer in correcting the error.

MMIC also has a unique test mode where the developer can run through the questionnaire to check for consistency. It allows the developer to take notes on errors in the questionnaire and to change the questions while still in test mode. This is a beneficial feature that allows developers to immediately fix errors in questions as they are testing its performance under survey conditions through the questionnaire.

Interface for Field Users

Interface Aesthetics: The default interface for field users is reminiscent of many online questionnaire tools, such as SurveyMonkey, but it is highly customizable – from something as simple as font and background color to something more complex like the addition of a menu bar. This customizable interface allows the programmer to change the questionnaire to best suit the desired medium for data collection. It is also possible to simultaneously display multiple questions on the same screen.

MMIC is able to implement questionnaires on most computers, PDAs, and mobile phones.

Questionnaire Navigation: Backward and forward buttons are the main method of navigation through a MMIC survey, making the interface touch-friendly and accommodating for PDAs and mobile phones. If additional navigation capabilities are desired, a pane could be created for movement across modules and between non-sequential questions. This type of programming requires a deeper understanding of MMIC and its compatibility with HTML, Javascript, and PHP.

Questionnaire Review: Through complex programming, the survey developer can create an interface that allows the enumerator to see the level of completion of the modules. However, to review individual questions, the enumerator would have to navigate through the questions one-by-one in order to alter answers. By default, MMIC does not offer a more efficient method to review the questionnaire.

Language Capabilities: MMIC allows enumerators to change languages both at the start and during questionnaire implementation by simply clicking on a button, assuming translations have been added beforehand. Adding the language switch button to the questionnaire interface is a preset function in

MMIC so it is a simple task. In addition, the questionnaire interface for enumerators can be designed to be in many different languages. MMIC has been used to create and implement questionnaires in languages such as Arabic, Chinese and Dutch.

Questionnaire Implementation

Supported Question Types: MMIC allows many types of answers: radio buttons, checkboxes, open-ended text inputs, and numerical inputs. Questions can be answered by simply clicking on the checkbox or radio button.

MMIC also has the flexibility to accept data from third party devices, such as GPS, microphones, heart rate monitors, etc. However, the integration of third party devices can only be achieved through complicated programming using the three web programming languages mentioned previously. Capturing multimedia during the interview, such as audio, video, or photo, also requires complicated programming using the web programming languages.

Question Support Features: MMIC provides all of the multimedia question support features desired for a LSMS survey. Pictures, audio and video can easily be added to support a question. They can be added either in the prompt itself or on the previous or next screen. It is possible to provide different language versions for these multimedia add-ins.

Furthermore, MMIC provides enumerators with the ability to append comments to any question or to the questionnaire as a whole. This greatly enhances the enumerator's ability to relay important information to the supervisor.

Questionnaire Navigation

The default mode of navigation through a questionnaire is by clicking the forward and backward buttons. While the forward and backward buttons provide simple navigation, it would be a tedious and time consuming method to move across modules, or even within modules containing a lot of questions. For this reason, MMIC allows a navigation pane or menu to be programmed into the survey to allow for swifter navigation to any question within a module or across modules. This non-linear mode of navigation does not interrupt the implementation of the survey logic; the logic corresponding to a question will be triggered regardless of whether the question is reached through forwards or backwards motion. Additionally, all of the modes of navigation through a questionnaire are touch-friendly.

Case Management

MMIC offers an excellent case management system for users at all levels of the survey operations hierarchy.

Enumerators: For enumerators, the software offers a dashboard of assigned survey cases. The enumerator can start any interview or add notes about a case on an ongoing basis. The case management system is thus a dashboard for launching interviews and recording information about them.

Supervisors: For supervisors, the software allows the easy assignment and careful tracking of survey cases. Supervisors can assign interviewers survey cases passed to them by headquarters. Supervisors

can also generate reports on survey progress at the aggregate level of their enumeration area or at the granular level of individual interviewers.

Survey Managers: For survey managers, MMIC offers an excellent suite for assigning interviews and tracking them across teams. MMIC is particularly distinguished by its ability to generate reports on survey progress at several different levels of the study, e.g. the non-response rate for a given survey team during a given week.

Data transfer

Server Overview: The MMIC package does not offer an integrated server option. However, arrangements can be made with RAND to set up and host a server through them.

Data Security: MMIC provides strong lines of security for data. Data are encrypted when they are stored in the database. Upon transfer, the data are then further encrypted using the Rijndael 256 protocol, an encryption protocol also used by the US Government for classified information.

Data Transmission Protocol: Data can be transferred to central servers through the Internet. The data can also be cloud-hosted, meaning that they can be accessed from different locations through the Internet. These functions are not preset and would require more sophisticated programming and hardware.

Data files can also be transferred using any data storage devices such as a flash drive or CD.

Data Exporting

File Formats: MMIC can export data to SPSS, Stata and CSV format.

Transfer Protocol: The point-and-click interface for exporting data is very user-friendly. Information on data exports is stored in logs that state the time of export, the output file type and whether the export was successful.

Features: MMIC is able to export data in multiple languages and allows the user to choose which language to export the data into. It also offers users the option to export only certain subsets of the entire data. Data variable and value labels are maintained, unless the data are exported into CSV file format.

Support and Documentation

MMIC's documentation is available only in English and has many shortcomings. The documentation is extremely sparse. Explanations of syntax, features, and functions are briefly and ineffectively explained, and certain sections are completely empty. Sample code and examples are rarely present in the documentation, and tutorials are nonexistent. The poor quality of the documentation combined with the complete absence of a user community make learning MMIC a time-intensive and arduous endeavor.

RAND provides help desk services for MMIC, at an additional cost. The MMIC help desk is prompt and very helpful in addressing questions and concerns.

Pricing and Upgrades

MMIC is a free and open-source program, but the support, design, and development services of RAND's MMIC team are quoted at the following consulting prices:

- $5,000 for programming of a simple survey with <30 questions and <1000 respondents, including administration page set up
- $7,000 programming of a moderately complex survey (some routing)
- $10,000 programming for a complex survey (routing throughout, versions for different respondents, graphics, videos, etc.)
- $2,000/month server costs for hosting (including hardware, license and bandwidth)
- $500/month help desk services by phone and email
- $2,500/day for senior programming
- $1,750/day for mid-level consultant programming
- $1,000/day for junior consultant programming
- $1,500/day additional programming for each day required beyond initial quote

Extensibility

MMIC is a completely customizable program because it is an open-source program. HTML, Javascript, and PHP can be used to extend questionnaire functionality. This ability provides developers with substantial flexibility to design highly customized questionnaires and incorporate readings from peripheral instruments. However, this functionality requires a programmer with extensive knowledge of web programming to accomplish such a high level of customization.

Hardware and Software Needs

MMIC's hardware and software requirements are very low, so it can work on UMPCs, PDAs and smart phones. It will operate on either Linux, or Windows 2000 and later. Prior to installation, MMIC requires the following software to be pre-installed: PHP version 5.2 and above, mySQL version 5.0 and above, and Zend Optimizer. MMIC can be implemented in any web browser (i.e. Internet Explorer, Chrome, Firefox, etc.) but it operates best on Mozilla Firefox.

Future Developments
There are no scheduled updates for MMIC in the near future.

Known software users: University College Dublin; NatCen London; Korea Labor Institute Seoul; Survey of Health, Aging and Retirement in Europe; Health and Retirement Study US; CentERdata Tilburg; China Health and Retirement Longitudinal Survey Beijing; Federal Reserve Bank of Boston; Federal Reserve Bank of New York; University of Michigan, ISR; Netspar, The Netherlands; Ohio State University; *Internal (Within RAND)*: Labor and Population; Health; Arroyo Center; RAND Cambridge Office.

5.6 OPEN DATA KIT

Programming

Power of Programming Language: Open Data Kit is a moderately powerful survey software with great potential. There are three reasons why ODK may someday surpass the other CAPI software programs currently on the market. First, ODK is open-source, allowing the flexibility to expand on its current features. Second, ODK's survey programming is accomplished through the use of XML, a popular and powerful extensible scripting language. Third, the software is developed in a collaborative format with input from both implementers and software developers alike.

Functionality of Power: In some ways ODK has already eclipsed the other survey software programs under review, such as the ease of adding multi-media into questions or capturing GPS and multi-media. It can handle complex skips, some data validation, and non-linear completion with ease. The ODK developers have created a very easy and streamlined way to implement a loop of unknown size. The whole process of programming a loop and adding a set of questions only requires a few extra lines of code. The enumerator can add elements to the roster and, when finished, simply indicate that there are no other elements to add.

Limitations of Power: ODK lags behind other comparable software in some areas. For example, loops of known size and elements are not possible due to the software's inability to turn a roster into a list that can be looped through with a series of questions. The fact that looped data are stored in a subform that is exported separately from the main dataset is another glaring issue. ODK also lacks the ability to test multiple data validations. In general, complex functionalities often require creativity and a proficient knowledge of XML. Evaluating ODK's current programming functionality leads to the conclusion that, while it has serious potential, it is best suited for constructing simple surveys without rosters.

It is helpful to put these shortfalls into perspective. ODK's core relies on the opensource JavaRosa platform. Therefore, it cannot add functionality that does not yet exist in JavaRosa. At this point, XML has capabilities far surpassing those of ODK and the ODK team has plans to add these capabilities once JavaRosa catches up. When all of the promised updates are made, such as making it easier to call external databases, allowing multiple questions on a screen, the addition of multiple constraints, and the resolution of existing issues with looping, ODK should be given another thorough evaluation.

Familiarity: ODK uses XForms to specify a series of prompts and input fields in a form. XForms is not a programming language, but rather a model for XML that is popularly used to build forms on the Internet. XML is a language used to structure, store, and transport data. It is the most common tool for data transmission both on the Internet and in office productivity applications, such as Microsoft Office. The implication of all this is that a user must be proficient in Java and XForms programming to be able to fully customize and add desired functionalities to forms in ODK.

Ease of Use: A developer using ODK could completely design an entire form through hand coding. XForm structures the code in a straightforward and intuitive way so even an inexperienced programmer would be able to code a simple survey with little trouble or time commitment. There is no centralized documentation for ODK to help users learn XForms and XML, but there are general tutorials provided by other sources readily available on the Internet.

One of the best ways to learn ODK's programming structure and nomenclature is to study the sample

code found on the ODK website. In addition, the online user community is a great resource for advice and assistance on coding problems and how to add various functionalities. The members of the community are very helpful and prompt about responding to questions, usually replying within the same day. However, the responses are usually very technical so an experienced programmer would be better placed to implement more complex functionalities in ODK. Another area of concern is that these are all voluntary forms of assistance and will thus only continue to be helpful as long as ODK is relevant to its community.

An alternative to coding a form is to use the survey design tools, like ODK Build. ODK Build allows the programming novice an easy way to create a simple survey. The point-and-click interface is intuitive and can accomplish several simple and some complex survey functions. However, some functions, such as adding images, video, and audio, can only be accomplished through hand-coding at this time.

Development Environment

Integration: ODK currently has a very fragmented development environment with different tools for different functions. ODK Build is used for developing questionnaires, ODK Aggregate is the server that collects and disseminates questionnaires and survey data. ODK Collect is the Android application that launches the survey, collects the data, and syncs it back to Aggregate. ODK Manage is a tool that can assign tasks and manage data collection. Moving among these different components is predominantly done through exporting and importing the appropriate XML file.

To exacerbate the fragmentation of the toolkit, there are also third party options for each development tool. For example, there are two widely used alternative applications for designing a form: Purcforms and KoBo.

Survey Creation: ODK Build is the official ODK survey form design tool that allows a user to design a form through a point-and-click interface. The designer essentially inserts questions into the survey form, writes out the question text and defines the variable name, then writes in constraints and skip patterns. Once completed, the form can be exported to an XML file.

ODK Build is well-suited to create simple forms, but more complicated forms are currently beyond its capabilities. Additional functionalities can be achieved through hand coding. Surveys can be coded by hand using any standard text editor.

Logic Programming: Hand-coding part or all of the form can be done in a text editing software, such as Notepad++. There are a few sample surveys available on the ODK website that can be used as a XForms template. There is also a very responsive user group that is available to answer questions about programming and to provide sample code. Unfortunately, without prior knowledge of XML programming, the options for extending ODK functionality beyond what is available in Build is limited.

Dictionary Creation: The data dictionary is automatically coded in ODK Build. It can also be coded by hand, following the XForms format.

Application Testing: A separate application, ODK Validate, can be used to check the syntax of any hand-coded form to ensure that the code is compatible with ODK Collect. Although it is simple to use, ODK Validate is not very useful because its error messages are very technical and its assessments are not always accurate.

None of the available form designers contain a test mode. Consequently, the only available method to test the form is to upload it to Aggregate and to then open it in ODK Collect using an Android phone or an Android simulator. Testing the form on an Android phone is a simpler method, as setting up an Android simulator on a computer is a complicated process that requires the installation of several programs.

Interface for Field Users

Interface Aesthetics: The enumerator interface, ODK Collect, is very simple and sleek looking and takes advantage of the progress made in smart phone technology. However, the customization of the interface is completely limited, as the size and color of fonts cannot be altered and only one question at a time can be displayed on the screen. These enhanced features are promised in later releases of the software.

Another limitation of ODK Collect is that it is only available for devices using the Android operating system and the software developers have no intention of expanding its compatibility to other devices. It is worth mentioning that Honeycomb, the upcoming release of the Android operating system, will be compatible with some tablet PCs in the future. One can imagine this development leading to new customizability features.

Questionnaire Navigation: Through a simple and intuitive method, ODK Collect allows enumerators a significant amount of freedom to navigate throughout a questionnaire. The enumerators need only slide their finger across the screen from right to left to go forward, left to right to go backwards. It is also possible to navigate to a different part of the survey by selecting 'Go to Prompt' option from the easily accessible ODK menu. This brings up a list of all of questions and their respective answers. To navigate to a different part of the survey, the enumerator need only touch the question of choice.

Questionnaire Review: Reviewing and changing answers on a complete or incomplete questionnaire is easy to do as well. From ODK Collect's main screen, the enumerator can select 'Continue Saved Form,' which will display a list of surveys. Once a survey is selected, the enumerator is brought to a list of the questions and their respective answers. This allows the enumerator to review answers and find where an answer was left blank. One caveat of this navigation feature is that it does not organize questions into modules, making locating unanswered questions a chore in longer surveys. ODK Collect is equipped with the ability to pause a survey midway through and return to it later. By pressing the back button, the enumerator will be prompted to save or discard the data already collected. Saved, sent and unsent interviews can be accessed again from the main menu by selecting 'Continue Saved Form.' The enumerator will then be taken to a list of all saved, sent and unsent surveys organized by time of last save. There is no way to sort through the list to find a particular interview, other than scrolling through them one-by-one, as the ordering is a bit unclear. The ability to remove sent questionnaires from the Android phone after it is submitted will be added in the next release of Collect.

Language Capabilities: The survey can be offered in numerous languages, as long as translations have been programmed into the survey alongside the main survey language beforehand. It is very easy to change languages, even mid-interview. While several languages can be added into the questionnaire, a questionnaire with two languages will require twice the amount of coding and three languages will triple the lines of code. Every additional language added into the questionnaire will significantly increase the load time for the questionnaire and will make testing the finished survey a time-consuming task.

Questionnaire Implementation

Supported Question Types: ODK contains all of the standard survey question types expected of a survey software program. These include: text, number (integer or decimal), multiple choice, multiple select, date, time, and date & time. While not available now, ODK will soon offer drop-down capabilities. In addition, ODK offers several media options for question implementation that are either not offered in other programs or are hard to use. It can capture video, audio, and images, as well as GPS capture and barcode scans. The ability to capture data from peripheral devices, such as those connected via Bluetooth or USB, will also be added in the future.

Question Support Features: ODK allows you to add video, audio, and images to question prompts. Multiple choice questions can integrate images or audio to support the answer choices. You can even provide different versions of the embedded media, in question or in multiple choice options, for each language.

An obvious limitation to ODK questionnaire implementation is that it currently only allows one question to be displayed at a time. The ability to display multiple questions at a time is planned for release by mid-June 2011.[25] Another drawback to the program is that it is not possible to add comments associated with questions or with the questionnaire, unless they are programmed in as questions.

Questionnaire Navigation

There are few limitations to where the enumerator can navigate to while implementing a questionnaire. The enumerator can move backwards, forwards, and skip non-sequentially to other questions. ODK remembers skip patterns, and does not show questions that have been skipped as navigable options.

Navigation in ODK is incredibly simple. The ODK Collect interface executes questionnaire navigation by utilizing the standard Android phone gestures and touch icons. ODK's layout is inherently suited for smaller screens, is touch-friendly, and changes its layout between portrait and landscape format based on the orientation of the phone.

ODK does not offer a navigation pane in the classical sense, but the enumerator need only select the menu button, then choose 'Go to Prompt' to be directed to a screen with a navigable list of questions. From this screen they can simply touch the question they would like to update or complete. However, because the screen displays the list of all the questions without organizing them into sections, the list can easily become unwieldy for longer questionnaires.

Case Management

Enumerators: Using ODK Collect on an Android device, an enumerator can access their associated ODK Aggregate server. From there they can download different questionnaires and well as sync collected data to the ODK Aggregate server. ODK Collect offers a few case management features for enumerators. Incomplete questionnaires can be saved and resumed from where the questionnaire was paused, from the first question or from the last question of the questionnaire. Enumerators are also allowed to review the questions and alter the answers in both complete and incomplete questionnaires.

[25] http://code.google.com/p/opendatakit/wiki/Roadmap

Supervisors: For supervisors, the only tool ODK offers is ODK Manage, which allows supervisors to assign tasks to the phones and track the status of these tasks. However, ODK Manage is in its very early stages of development so there are still a number of bugs that need to be worked out; as a result, the software reviewers were unable to get this application to properly function.

Both supervisors and survey managers can also use ODK Aggregate to review data from complete questionnaires that have been synced back to the server. However, ODK Aggregate is a very simple application and does not have any additional information, such as reports on collection errors. Another major limitation of ODK Aggregate is that it does not allow for the review of questionnaires that have not been synced back to the server.

Survey Managers: ODK currently does not offer any tools specifically for survey managers. However, some of the features of ODK Aggregate and ODK Manage could be used to view data and assign tasks.

Data transfer

Server Overview: ODK Aggregate is the application released by ODK to facilitate the data transferring process. ODK Aggregate operates on a server and the server can either be hosted locally or in a cloud. In general, ODK Aggregate appears to have a tendency to operate incorrectly. This flaw is likely a symptom of the application being in its early stages of development.

In spite of its unreliable performance, the actual data transfer procedure is quite simple using ODK, although perhaps too simple. The enumerator first specifies which data to sync in ODK Collect. The data are then transferred to the ODK Aggregate server, using GPRS or the internet. A message will display indicating whether the transfer was successful or not. Unfortunately, the error message will only state that the transfer failed. It does not elaborate on the reason behind the failure, making it difficult to troubleshoot the error. Data transfer can also become problematic when the internet connection is weak or the questionnaire is lengthy, rendering it slow to impossible to transfer data between the phone and server.

It is important to mention that the application does not have an integrated installation file, so setting up ODK Aggregate requires a fair level of technical knowledge.

Data Security: ODK does not currently have a data encryption option, which is very disconcerting when surveys involving sensitive data are being implemented. In order to encrypt data in ODK, the codes for ODK Collect and Aggregate must both be modified, which is a task that would be extremely difficult for someone who is not a competent programmer.

Data Transmission Protocol: Data collected in ODK Collect can either be transferred using the internet or GPRS using Aggregate. An alternative to ODK Aggregate is KoBo Post Processing (KoboPP) developed for the Kobo Project by UC Berkeley's Human Rights Center. KoBoPP allows users to transfer data directly from the phone to the computer through a USB connection. Transferring data in this way makes working in areas with limited connectivity or with sensitive data possible.

Data Exporting

File Formats: ODK does not export to any of the popular statistical software packages. The only exporting options are CSV, XML, and KML files. The inconvenient consequence of this is that the exported data will be flat, which means that variable labels and value labels will be lost in the process. In addition, ODK only exports using wide format and it is not possible to export a codebook for the data collected.

Transfer Protocol: Data can be exported through a simple point-and-click interface in ODK Aggregate or ODK Briefcase. While data export is possible through the ODK Aggregate interface, ODK Briefcase is the official applet to download the contents of ODK Aggregate server. While both have a simple interface, ODK Briefcase performs better for the export of data. ODK Briefcase has a message that informs the user on the status of the data export and if it was successful or not, which is a feature that ODK Aggregate lacks. ODK Briefcase also performs slightly better with respect to exporting data from loops, called child questionnaires, in the questionnaire. In Briefcase, the child questionnaires are exported into separate CSV files. In ODK Aggregate, they are exported into the data set as hyperlinks to a web page with the table for the loop. While neither program is ideal, at least the user can join the child questionnaire files from ODK Briefcase with the main questionnaire by merging the two questionnaires using a statistical analysis software program in order to create a long dataset.

Features: One unique and innovative feature of ODK is that a user can export data into a KML file that can then be read by Google Earth or Google Maps. In ODK Aggregate, exporting the data into a KML file is as simple as exporting it into a CSV file, in that it only requires clicking on a button. Using the GPS data in the KML file, Google Earth or Google Maps will be able to map the different GPS locations from the questionnaires and associate it with the responses to the questions. Using this technology, it would be possible to map the different households in a survey and a user would be able to view the data from their questionnaire by clicking on the pin marking the household. Multimedia captured as part of the questionnaire would also be displayed when clicking on the pin. Taking advantage of this technology is very simple in ODK and it provides a very powerful tool for visualizing data from a survey.

Support and Documentation

The learning environment for ODK is a bit less formalized than the other software packages under review. No formal training is available and there is no tutorial or user guide. The documentation for this software consists of the very limited amount of brief pages providing information on such things as installation of the various ODK components, logic in XML, how to create an XForm, and frequently asked questions. Fortunately, however, the language used to create questionnaires in ODK, XML, is commonly used so there are several more extensive tutorials offered by other unassociated websites. Likewise, XForms is a commonly used format to create questionnaires so there are multiple websites with informative tutorial for it as well. ODK also offers about six sample questionnaires that might offer some guidance for a user trying to learn how to create a questionnaire in XForms.

ODK offers an extremely active free online user community where questions are promptly answered by proficient users of ODK, including the software developers themselves. The online community is a useful medium for assistance on such topics as troubleshooting problems, adding functionalities to ODK, debugging code, and asking general questions about the capabilities of ODK.

The ODK developers do not offer fee-based services but they recommend a Boston-based firm, Dimagi,

for fee-based technical assistance. The ODK developers and Dimagi have also worked with several firms in developing countries to help build their capacity and strongly encourage hiring those in-country firms for technical assistance.

Pricing and Upgrades

ODK is a completely free and open-source program. The main medium for technical support is through the ODK online community,[26] which is both active and free.

The frequency of updates to the various ODK components varies. For example, ODK Collect, the most advanced component, is updated every month. However, the other components, such as ODK Aggregate, have not benefited from such frequency. A roadmap of future releases is available on the Open Data Kit website.

Extensibility

ODK is an open-source program, making it possible for proficient programmers to be able to augment ODK by adding additional functions to suit their needs. One notable example is the KoBo Project, designed by the Human Rights Center and the University of California at Berkely.

It is easy to integrate different media such as audio, video, GPS, or a barcode scanner into questionnaires using ODK. ODK does not incorporate multimedia into the dataset when exported into CSV format, but rather exports it into a separate folder. Integrating other devices with ODK, through USB or Bluetooth, is possible but would require the user to change the source code of ODK. The difficulty of integration would depend on the device, but the task would range from difficult to extremely difficult for a user that is not a proficient programmer. The ODK developers are only now beginning to work on the integration of certain USB and Bluetooth devices, so it is not a feature that is expected to become available in the near future.

Hardware and Software Needs

ODK Collect currently only operates on Android phones, version 1.6 or higher. Upcoming versions of the Android OS will be fully compatible with tablet PCs, so there is potential for ODK to be used on tablets.

Future Developments

Because ODK is a relatively new program, the list of future functionality is quite extensive. For the sake of brevity, we will highlight the features most relevant to the needs of the LSMS Team.[27]

Aggregate: The ODK team plans to release a new version of Aggregate v. 1.0 in January of 2011. The list of promising features includes automatic deletion from the enumerator device of submitted interviews and the ability to handle larger multimedia files.

Collect: A new version of Collect v. 1.1.6 is planned for a January 2011 release. Updates include the ability to display multiple questions per screen and some survey branding options. In addition, there are

[26] http://groups.google.com/group/opendatakit
[27] For a more extensive list, please visit http://code.google.com/p/opendatakit/wiki/Roadmap

tentative plans for Collect v.1.1.7. This version would allow selections from maps, selections from a grid of icons, drop-down lists, ordering, selections from a slider, and character-in-choice filtering.

Known software users: USAID-AMPATH (Kenya); Google (Tanzania, Cambodia, Brazil); Grameen Foundation Application Laboratory (Uganda); Episurveyor at DataDyne (Kenya); Change at the University of Washington (Seattle); Human Rights Center at UC Berkeley (Central African Republic); Wharton School of Business at University of Pennsylvania (South Africa); Information School and Haas School at UC Berkeley (Ethiopia, Uganda, India, Mexico); Vetaid (Zanzibar); D-Tree (Tanzania); Johns Hopkins Center for Clinical Global Health Education (Uganda); Johns Hopkins University Global Water Program (Ghana); Villagereach; Person Finder Mobile (Haiti); Lumana (Ghana); Small Meadows Farm (Virginia); Columbia University (Mali); Transit, Technology, & Public Participation Project (North Dakota).

5.7 PENDRAGON

Programming

Power of Programming Language: Pendragon Forms VI is a computer-assisted interviewing program with a low level of programming power. Survey design and creation is predominantly accomplished through the easy-to-use point-and-click development features. Advanced routing and validation require the use of scripting. However, Pendragon's needlessly confusing proprietary programming language often makes even simple scripting tasks a formidable endeavor.

Functionality of Power: Pendragon can accomplish many simple and complex features, such as non-linear completion, simple and complex skips, uniform range restrictions, and even allows the inclusion of pictures into questions for clarification. However, Pendragon does not excel in any one of these areas.

Limitations of Power: Pendragon lacks many of the basic features offered by the standard survey software programs on the market. It is unable to perform some complex data validations, such as dynamic range restrictions. Other notable shortfalls include the inability to integrate audio and video into question prompts and the inability to customize question prompts based on previous responses. However the most glaring limitation is that Pendragon handles rosters in a very sloppy manner. In order to fill in a roster the program has to launch a new form. While this form is still linked in some way to the original form, meaning that data can still be validated across forms, the data management process is made unduly complicated.

Familiarity: Pendragon has its own proprietary programming language so it does not substantially resemble any of the other available programming languages. One would, therefore, have to familiarize themselves with the documentation provided by the makers of Pendragon.

Ease of Use: As Pendragon's programming language is proprietary, one would need to read the documentation thoroughly and often reread it to understand how to code a survey with elementary questions. The documentation gives very basic examples with easy-to-follow instructions, making the software look deceptively simple. However in practice, when trying to program surveys with multiple routing and referencing similar to those used by the LSMS team, it ends up being difficult to use. For this reason, a higher level of programming experience would be beneficial. Furthermore, the language itself makes simple programming tasks unduly difficult. Even simple scripting requires more effort than one might reasonably expect.

Development Environment

Integration: Pendragon Forms VI has an integrated interface. Creating a questionnaire involves a point-and-click interface that has some scripting for logic commands.

Survey Creation: Being in a Microsoft Access environment, a programmer needs to open Access, launch Pendragon Forms Manager, then enable the corresponding Macro. Once the Macro is enabled, the Pendragon Forms Manager window will be shown. From there, the programmer will need to click the *NEW* button to open the Form Designer window and create a new form. In the *FORM NAME field,* the programmer will need to type a name for the form. In order to add fields, the programmer needs to type a *FIELD NAME* and select a *FIELD TYPE*. The field name is the text that a user would see on the

questionnaire screen. Therefore, the question text is usually put on the field name. The field type is the answer type that is allowed for that specific field/question. To add other fields in a form, a programmer must click the "+" button on the Form Designer.

Logic Programming: In Form Designer, the developer would select the *SCRIPT* tab to add logic commands to a field. Scripts can be used to perform skips, calculations, and reduce handheld data- entry by pre-filling certain fields. By default, Pendragon refers to specific fields by their position in the questionnaire. So if address is field number 7, it is referenced simply as '7' by Pendragon. There is an option, and it is recommended, to give fields a *FIELD LABEL.*

Dictionary Creation: Dictionary creation is automatically done when one fills in the *FIELD NAME*. This means that the full question automatically becomes the variable name, i.e. "What is your name?" becomes the "WhatIsYourName" variable. If a different name for the field is desired to be viewed on a database, Pendragon allows one to change it under the 'Column Name' field in the DATA Tab.

Application Testing: Once the electronic survey is created, the developer can tap the *CHECK SCRIPT* button to compile and check the syntax of the form. When the compiler finds an error in the code, it alerts the user, indicates the field in the form with the problem, and provides an explanation for the error. However, the compiler is not very thorough and, as such, allows some problematic codes to compile.

Once the form is compiled, the developer can test the form in the client device. However, one first needs to save and 'freeze' the form, and then send it to the client device. To edit the form after it has been frozen, one has to make a copy of the frozen form, then delete it. This process is tedious if the survey requires extensive testing.

Interface Aesthetics: The interface for field users looks unattractive and has limited customizability. Aside from adjusting the font size and font color, there is no other way to customize the form to make it look more elegant. Since Pendragon was intentionally created for small client devices such as iPhone, iTouch, iPad and Android, questions and answers are displayed on a limited screen size. Space is limited to 10 lines per screen, even when Pendragon Forms VI is run on an HTML5 browser like Chrome or Safari. As Pendragon Forms VI was mainly established for Apple and Android products, the software is touch-friendly.

Questionnaire Navigation: It is possible to move forward in a questionnaire by clicking the next question or by pressing the "tab" button. It is also possible to move backward, move between modules (which are in the form of subforms/child forms) and finish a survey in a non-linear fashion. However, the developer must exercise caution when field users are allowed to finish a survey in this way. The developer must program a very thorough (and complicated) check to make sure that an interviewer was able to complete all the required fields.

Questionnaire Review: Once a questionnaire is completed, Pendragon saves the answers in the client device's memory. Working within the environment, an enumerator can review the answers on the database. It is not possible to check the level of completion of a survey, but one can try to bypass this by having a field checkbox that can be manually checked once a module is completed. Pausing a survey and starting from the last position visited is not an option with this software.

Language Capabilities: Pendragon is a very basic survey tool, so it is not possible to program a survey that hosts multiple languages.

Questionnaire Implementation

Supported Question Types: There are 20 answer types supported by Pendragon Forms VI. These are: freeform text, numeric, currency, date & time, time, date, option 1 to 5, yes/ no, time checkbox, completion checkbox, popup list, lookup list, jump to section, section, subform list, single subform list, signature, multi-selection list, button and custom control. Looking at the list, it may seem that some field types are redundant (e.g. date, time, and date & time) but because of the limited fields allowed in one form, it makes sense to have these fields as separate field types.

Pendragon supports Custom Controls, separate program modules that can be installed on the handheld in order to extend Pendragon's capabilities. The software is shipped with the GPS Custom Control. Handheld devices must support the geolocation feature of HTML5 in order to use GPS with Pendragon Forms VI.

Depending on the developer, other customization is possible through the field type custom control. Some users have been able to create a camera custom control where it saves the taken picture in a JPEG format.

Question Support Features: In Pendragon, pictures can be attached to questions as long as the picture meets the image size requirements. When JPEG files (*.jpeg) are used, the handheld screen area is 320 pixels wide and when bitmap files (*.bmp) are used, images should be 320x192 pixels in size and not more than 256 colors. However, it is not possible to use images to aid multiple choice questions and it is not possible to add audio and video clips. Answers can only be assisted through a static prompt and by attaching pictures to questions.

There are no features for assisting the enumerator. It is not possible to display notes or comments for the enumerator and Pendragon does not allow the enumerator to append comments to questions or to the whole survey.

Questionnaire Navigation

Navigating in a survey with Pendragon Forms VI is intuitive. Pendragon supports different methods of questionnaire navigation. It is possible to go forward and backwards within and across a screen and module. To go to the next field, the *tab* button should be entered or a specific field can be clicked with a mouse button. To access a subform or pull up a reference form, a clickable subform/reference form (which has its own form name) button can be clicked. Once the subform button is clicked, it will pull up a new form. In the case of clicking the reference form button, a reference list will be shown.

Since Pendragon is still relatively new, bugs are still persistent. As a result, while in theory pulling up subforms and reference forms should be easy, in practice, errors will pop up even when the survey form is well-designed.

Case Management

Enumerators: It is possible to monitor the completeness of a module and a questionnaire by using the

Yes/No checkbox and some programming logic. For example, at the end of a module, a programmer can insert a Yes/No checkbox to signal if a form is completed. Through logic programming, one can program in a hard check to count the number of completed fields and disallow moving on to the next form until the number of completed fields is satisfied.

Pendragon allows the programmer the ability to create a menu to review existing records. Once created, a user can tap on a record on the review screen to go to that record.

Supervisors: Pendragon does not offer any additional features for a supervisor in terms of case management.

Survey Managers: Pendragon does not offer any additional features for a survey manager in terms of case management.

Data transfer

Server Overview: Pendragon Forms VI comes with a *Staging Server* – a web server and a database that hosts the form designs and data while they are exchanged with remote users. Typically, a desktop or laptop can serve as the *Staging Server*.

Data transfer is a simple two-level synchronization. From the Administrative PC, forms created are synchronized with the Staging Server so that client devices can pick up the forms from there. Devices synchronize with the Staging Server to pick up forms and data, and to send back data that has been collected from the field.

The *Pendragon Transfer Agent* is used to administer the Staging Server and to transfer data between the Administrative PC and the Staging Server. It is also used to add users and to deploy forms to users.

Data Security: The Staging Server is password protected but if one is working in a non-secure, public network, information transferred is not protected. To make communications more secure, it is recommended to use a SSL certificate for the Staging Server. This serves to encrypt conversations via SSL security.

Data Transmission Protocol: Transferring data using the Staging Server is usually accomplished through WiFi, 3G or 4G wireless connection.

Data Exporting

File Formats: Pendragon does not offer much by way of file formats. It can only export data in a long format to ASCII and Excel.

Transfer Protocol: Through a point-and-click interface, after the forms have been synced to the main database, data can be exported to an ASCII (*.csv and *.txt) or Excel (*.xls) format. Variable labels are retained. Since one cannot create value labels using Pendragon Forms VI, value labels will need to be generated manually after the data have been exported.

From a user's perspective, the point-and-click interface is simple and the job gets done well. However, users do not have a lot of control with the formatting of the data.

Features: Pendragon does not provide any notable data export features. It also does not allow multilingual variable labels at this time.

Support and Documentation

Pendragon Forms VI comes with a basic manual with a quick start guide. The manual not only provides directions on how to use Pendragon but also provides sample codes.

There is an online community for users to exchange ideas and solve problems. One important shortfall is that there is a low level of user activity in the community. Few users actually answer questions. Furthermore, even though tech support monitors the board usage, they often leave questions unanswered. If help is needed beyond the user community, one must rely on reaching tech support via phone or email. The tech support team, which offers support free to any users, makes a strong effort to answer questions and provide assistance.

Training is not provided by Pendragon.

Pricing and Upgrades

Pendragon Forms VI Starter Kit comes with a license to use the software on one handheld device. In order to use more handheld devices, more licenses must be acquired. The fees are as follows:

- Developing software: $299/year. The license is valid for one server and one handheld device.
- Client software: $54-$69/year per user. Bulk pricing is available if requested.

There is no basic user training that is provided by the company but technical support is provided for free. Upgrades do exist; the last major overhaul of the system was in 2007. Users who would like to update to the new version are subject to a new license agreement. Smaller updates for system improvement are released as soon as they are developed and are provided free of charge.

Extensibility

It is possible to extend Pendragon Forms VI through the custom control feature. Custom controls are separate program modules that can be installed on the handheld in order to extend Pendragon Forms' capabilities. The software comes pre-packaged with a GPS Custom Control.

It is important to note that handheld devices must support the geolocation feature of HTML5 in order to use GPS with Pendragon Forms VI. Some devices like iPod, iTouch, and HTML5-based browsers only support the geolocation through the WiFi hotspot location. These devices would not be able to produce a GPS reading when one is not actively connected to a WiFi hotspot.

The software does not have a multi-language interface.

Hardware and Software Needs

In order to use Pendragon Forms VI, one would need an Administrative PC – a computer to host the Staging Server and Client Browser.

The Administrative PC is typically a Windows desktop or laptop running Windows XP, Windows Vista or Windows 7. The Administrative PC could also be a Windows server 2003/ 2008. The Administrative PC contains the Pendragon Forms Manager database and Pendragon Forms Manager. The Pendragon Forms Manager database is a Microsoft Access database that acts as the survey's back office. The Pendragon Forms Manager is where forms are designed, users and user groups are managed, and data are viewed.

Administrative PC Compatibility and System Requirements include Windows XP SP3, Windows Vista, Windows 7, Windows Server 2003/2008; approximately 200MB Hard disk space; and MS Access 2003 and up.

The Staging Server is a web server and a database that hosts your form designs and data while they are exchanged with remote servers. In a typical installation, the desktop or laptop that serves as the Administrative PC is also the Staging Server. However, the Staging software will run on any system, that supports the Apache web server, the MySQL database server, and the PHP programming language. This means that the staging server software can run on a hosted server elsewhere on the Internet, or on one of Pendragon's public server.

Staging Server System Requirements: The Staging Server software can be run on the Administrative PC. If the Staging Server will not be run on the Administrative PC, any server that runs Apache 2.2, MySQL5, and PHP5 can be used. Furthermore, PHP MySQL and PDO libraries must be available. There should be an allotted 2MB storage space and database storage for the data and form designs. The server does not need to run Windows.

The Client Browser is an HTML5 web browser that allows users to fill in forms whether online or offline. Typically the client browser is run on mobile devices such as iPhone, iPad, iTouch and Android. It can also be run on a desktop or laptop using Safari or Chrome.

Client Devices: iOS3 or better, iPhone, iPod touch 2G or better, iPad, or Android 2.2. Any device with an HTML5 web browser and the following HTML5 features: Web SQL API, Offline Application Cache, and Canvas API (for signature capture).

Though UMPC/MTPCs can be used with Pendragon, they are not particularly suitable. UMPC/ MTPC's are much more powerful than an Android, iPhone iPad, etc., so when Pendragon is used with them, the full potential of these machines are not utilized.

Known software users: Baltimore National Aquarium; City of Rancho Cordova (California, USA); Interplast; Kaiser Permanente; Government Computer News; Security Solutions.

5.8 SURVEYBE

Programming

Power of Programming Language: Surveybe offers the best of both worlds: a programming toolkit that fulfills the functional needs of today while anticipating those of the future. Surveybe does this by utilizing powerful external programming languages in such a way that the costs of their use are minimized. For this reason, programming in Surveybe is simple enough for entry-level users yet powerful enough for advanced audiences.

Functionality of Power: Surveybe's programming tools can accommodate all essential needs of LSMS-type multi-topic surveys. For example, Surveybe provides effortless non-linear navigation, effective use of pictures to assist the interviewer, and easy appeal to external databases features. Many of these functions are accomplished through a drag-and-drop interface, whereas the other features are coded using external programming languages.

Surveybe's drag-and-drop capabilities can accomplish many tasks typically reserved for hand-coding. For example, creating rosters, which typically requires complex programming tools like loops or arrays, can be accomplished using the questionnaire builder. In fact, rosters-within-rosters are created in the same simple way as regular rosters. Additionally, non-linear navigation, which often requires special commands or workarounds in other packages, is offered by default and without any additional programming required in Surveybe.

Most of the programming in Surveybe is accomplished using SQL, a database programming language. Programming in SQL is mainly concentrated in the following three areas: giving skip instructions, calling external databases, performing data validations, drawing in past rounds of survey data and comparing current survey answers against them. Enabling these functionalities typically requires little more than a short line of simple code, although data validations tend to require more. Developers also have the option of employing HTML to format question text and of including media to support question text. Both SQL and HTML offer a broader array of functions than those mentioned, thus leaving room for extensibility not yet envisioned.

Limitations of Power: There are two limitations to Surveybe's power, the first of which is that Surveybe does not resume the questionnaire in the place where the enumerator last exited the application. However, the software's seamless non-linear navigation ability allows users to return to that position, thus making this issue easy to overcome.

The second, more structural issue, is that many programming tools, such as skip and enablement conditions, cannot currently be extended to rosters, screens, or other groups of questions. Skip and enablement conditions currently only apply to single questions or groups of questions. However programmers can obtain provide this missing functionality through a fairly straight-forward work-around: adding comparable skip instructions to "hidden" questions, whose effect is roughly identical to the more elegant solution outlined above. EDI plans to make available screen enablement functionality sometime in the latter half of 2011

Familiarity: With no proprietary programming language of its own, Surveybe draws its power from two external programming languages: HTML and SQL. HTML enables designers to insert survey-enhancing

images and other media into question prompts. Its current usage in Surveybe is limited in scope and very simple to understand, even for novice programmers.

SQL, which constitutes Surveybe's core programming package, can be used to perform a wide array of tasks, from skips to cascading answer selections, with great power and limited difficulty. However SQL can be a burden, as some SQL commands can become long, unnatural, and unwieldy, particularly for data validations. It can also become formidable if the task is complex. However, because of Surveybe's superlative documentation (see support and documentation section below for further details), these problems are minimized. The expansion of the user community can reduce this burden in the future.

Ease of use: By design, Surveybe's programming suite is easy to use. Many complex tasks that are typically handled with program code in other CAPI packages are handled with a drag-and-drop interface in Surveybe. If coding is necessary, it is either guided by a code builder or executed through relatively simple commands. Skip instructions, for example, are constructed by first clicking a button to add a skip rule, then defining the condition, and lastly specifying the skip destination. The end result is a suite designed to suit developers with varying levels of technical expertise.

Development Environment

Integration: Surveybe offers a development environment that is mostly complete, integrated, and relatively easy to use. It brings development tools, with a few noteworthy exceptions, into a unified interface for questionnaire developers. The development toolkit contains three pieces: the Questionnaire Designer tab, which defines questions and controls questionnaire layout; the Data tab, which controls the structure of captured survey data and determines the use of external databases; and the Design Alerts tab and Implementer program, which together check the completeness, syntax, and proper functioning of the questionnaire.

Survey creation: The Questionnaire Designer tab provides an integrated and complete palette for designing the layout, content, and flow of surveys. In a single and unified interface, the Questionnaire Designer tab allows users to outline questionnaires into screens, sub-screens, and rosters, create questions on the fly, place them on the screen, assign them skip instructions, and simultaneously see a (non-functional) visual preview of the resulting questionnaire.

In order to obtain this unified interface, Surveybe divides the questionnaire design studio into four panes – questionnaire components, questionnaire tree, questionnaire preview, and question properties – that each offer contextually relevant tools for manipulating appropriate aspects of questionnaire design. Within the questionnaire properties pane, the designer can access, define, and edit all attributes of a question: the question text, answer type, and skip instructions. Multiple choice answer options can also be added here, as long as they have been previously defined as a list in the Data tab.

Logic programming: The Data tab offers an integrated, if somewhat less coherent, toolkit for everything related to data: data tables, data attributes, and data validations. Each of these sets of controls are available to the user by clicking through tabs along the left-hand side of the Data tab. Despite the common theme of control over data, the tools in the data tab seem like an awkward mishmash of disparate design controls. The tools concerning how the data are stored and how external databases are called are the most tightly related, thematically. However, the other tools seem better suited for the questionnaire designer tab, as they deal more with questions than with data. Equally perplexing is the fact that skip logic and enablement conditions are housed in the interface for drafting and defining a

survey question. The placement of the logically, or at least functionally, related control features into different interfaces is confusing.

Dictionary creation: Variable names and labels are created in the Questionnaire Designer interface as survey questions are scripted and placed into the instrument. The Data tab contains the other tools for determining how data are stored and described in the database. Within the Data tab, the response list tab details answer options and their respective numerical codes and value labels; the item list tab provides the elements of fixed-dimension rosters and their numerical codes and string descriptions; and the table tab determines the size and contents of data tables, as well as their identifier variables.

Application testing: The questionnaire testing and debugging suite is the least integrated aspect of the development environment. While the design alerts are cleverly integrated as error flags into the Questionnaire Design tab, the questionnaire testing application is entirely separate. The only questionnaire testing features within Designer are those that arise from design errors, which are simple error flags that an expected design element is missing or malformed. Questionnaire testing, strictly speaking, occurs entirely within the separate Implementer package.

Limitations: Unlike many other menu-driven programs, Surveybe does provide users a useful--but, ultimately, somewhat limited--tool for re-using pieces of questionnaires across surveys. While survey designers can import and reuse previously developed questionnaires in their entirety, the inability to integrate various components of previously developed questionnaires as part of a new application— such as questions, rosters, or modules— remains.

Another implication of this limitation is that survey development with Surveybe--as with most menu-based CAPI programs--is both centralized and sequential. The current development environment does not allow for simultaneous work among several survey programmers which can later be consolidated into one application. Instead, development work is either done by a single programmer or is done sequentially by a set of programmers. This may hamper the ability of survey developers to construct CAPI applications quickly and effectively, particularly as the complexity of a questionnaire rises.

Interface for Field Users

Interface Aesthetics: Surveybe offers field users a handsome and intuitive interface which is clearly the product of a new generation of software development. With sleek lines and minimalist coloring, the look is clean, modern, and appealing. Tabbed questionnaire screens make for a graphical user interface that is not only appealing but also practical, particularly for interviewers whose computer savvy may make this design choice intuitive.

Surveybe was explicitly designed for UMPCs/MTPCs (with 7 to 10 inch screens). It has large, clickable icons for everything from answer options to navigation tabs. It has small sub-screens and pop-up windows that house follow-up questions to minimize the use of precious screen real estate. Surveybe is also touch-friendly throughout**.** However, it is unclear that that software would fare well on smaller screen devices (such as those with 3.5 to 4.5 inch screens), as there are no in-built features, as exist in CSProX, to modify the layout.

Questionnaire navigation: Surveybe makes its navigation extremely intuitive by emulating paper. While paper has pages, Surveybe has screens; while paper advances by flipping pages, Surveybe does so by clicking on clearly labeled tabs. In addition, navigation, typically driven by keyboard commands in other

packages, is done uniquely and seamlessly through the touch interface. Other features accomplished through touch interface include validating the questionnaire, selecting and opening a survey case, and loading and using a questionnaire. Surveybe's only shortcoming in navigation is its lack of visual cues indicating where the enumerator has been and which sections (or sub-screens) have been completed.

Questionnaire review: Surveybe allows the developer to program the survey to display any language supported by HTML. At this time, the program lacks the ability to switch between different translations of a survey, thus requiring a different language version for each language, if multiple languages are desired. EDI plans to make available the functionality for switching between different translations in Surveybe Version 2.3 that is planned to be released in October 2011. Surveybe currently lacks the ability to change the language of the software interface.

Questionnaire Implementation

Supported question types: Surveybe provides a relatively small but comprehensive set of answer types. Beyond the standard numerical and text input, Surveybe offers drop-down menus, radio buttons, and single-select checkboxes for open-ended questions. Multi-select checkboxes are in the development pipeline, but are currently unavailable. Additionally, Surveybe's utilization of SQL allows technically savvy users to create unanticipated modifications to its stock of question types. One notable extension is the ability to randomize multiple-choice answer options, which prevents the common tendency of respondents to select answers higher in the list.

A major shortcoming of Surveybe is its current inability to capture and store multi-media files and data from peripheral devices. EDI plans to make available GPS capture functionality in Surveybe Version 2.3 that is planned to be released in October 2011.

Question Support Features: Surveybe does not offer much by way of question support for either the respondent or the enumerator. Media-heavy modes of answer elicitation or answer capture are entirely absent from the current version. This limitation includes the inability to embed a video or audio clip into a question for clarification, or to provide a touchable image for multiple-choice answer selection. Surveybe's utilization of HTML for embedding static pictures could provide a portal for this to happen in the future. Moreover, while Surveybe does allow the enumerator to append comments to questions or to the survey, it does not allow the developer to embed links to clarifications of question meaning.

One unique and useful feature of Surveybe is that it is designed to update intelligently as answers are added or modified. For example, adding a new household member automatically includes data points for that person in other sections' questions concerning household members. These standard controls, also found in more traditional CAPI software, are thoughtfully modified to accommodate non-linear navigation.

Questionnaire Navigation

In terms of questionnaire navigation, Surveybe outperforms the other software packages included in this review. The field user can move freely throughout the questionnaire – forward and backwards within a given screen, forwards and backwards across screens, and from any given section to another. This navigation can be accomplished by simply clicking on questionnaire elements: clicking on a question to answer it, clicking on a button to open a sub-screen, or clicking on a tab in the navigation pane to move to a module. The software also implements skip instructions, by enabling or disabling questions on

the basis of prior answers, in a way that works well with non-linear navigation. However, Surveybe does not provide the ability to visualize which segments of the survey are complete, thus relying on the validation errors to indicate incomplete sections and to aid non-linear completion.

Case Management

Surveybe does not currently feature case management functionality that would in an integrated environment allow (i) survey managers to assign households or enumeration areas to the teams, track field work progress, and review the quality of output, (ii) team leaders to receive assignments from survey managers, assign cases to enumerators, track field work progress and review the quality of output, and (iii) enumerators to receive assignments from team leaders and monitor their progress with information on the degree of completion for each interview.

The current rendition of the case management dashboard in Surveybe Implementer is no more than a list of interview files that are manually generated in the field, as opposed to being allocated from headquarters or via team leaders, often in accordance with the geographical information entered as part of Surveybe Implementer's main tab. Since Surveybe Implementer does not allow for separate modes for enumerators, team leaders, and survey managers, with differentiated functionality under each mode, all layers of the survey hierarchy have to rely on the same case management dashboard. EDI plans to make available case management functionality in Surveybe Version 2.2 that is planned to be released in July 2011. It remains to be seen whether and to what extent these developments will put the software on more equal footing with programs like Blaise, CASES, and MMIC.

Data Transfer

Server Overview: Surveybe does not have an in-built data transfer system, but works well with external systems. It can easily be integrated with open systems like SQL, free systems like DropBox, and a variety of FTP services. In the current system, users need to look elsewhere for file transfer, synchronization, and warehousing services.

Data Security: Surveybe does, however, offer an end-to-end data security system. Surveybe automatically encrypts data files during the interview. It then decrypts data with a key that is unique to the study. Thus, data are encrypted from capture in the field, through transfer over the internet, to export in headquarters. Though the transfer system may not be, the contents of the files are secure at every stage of the process.

Data Transmission Protocol: Surveybe data can be transmitted over the internet, using an FTP server, or through GPRS. Surveybe clearly takes sides on the issue of avoiding lapses in data transmission in low-connectivity environments. Unlike other CAPI packages, which tend to record all survey cases in the same file, Surveybe places each survey record in its own file. Beyond the obvious upside for case management, this approach also has the virtue of breaking data into small and eminently transferable packets, whose maximum file sizes typically only approach 120K.

Data Exporting

File Formats: Surveybe currently only exports to CSV and Stata, in long format. Surveybe automatically creates a Stata .do file for transforming the CSV file into Stata format. The .do file helpfully provides data managers an "audit trail" of how data were exported to Stata and programmers with a means of

making any appropriate adjustments.

Transfer Protocol: Another strong suite in Surveybe, the data export facility gets superlative marks for combining the complementary virtues of power and simplicity. Surveybe transforms all of a survey's modules, even the most complex, into convenient datasets for each distinct module. Through a few clicks of the mouse, Surveybe generates intermediate data files (CSV files) and the executable program code (Stata .do file) for creating a set of properly formatted datasets for analysis. One limitation to Surveybe's transfer protocol is that it does not allow the export of user-defined segments of the data. Instead, Surveybe automatically exports all data into as many files as there are modules.

Features: Surveybe accommodates variable and value labels for data exported to Stata. Surveybe exports the labels and values as code in a separate .do file that must be run subsequent to opening the dataset in Stata.

A strong feature of Surveybe's data export facility flows from the relatively unique way in which the program stores data. Many CAPI programs store survey data in fixed format files prior to export, where each variable occupies a well-delineated space. Surveybe, in contrast, stores its data before export in a flexible format, where variables may appear at any point in the data file but are identified by a unique identifier. Flexible storage formats, like Surveybe's, ensure that all data files can be exported in exactly the same way, since changes in variables in no way affect the unique identifier used for arranging the data in the raw data file. Variables with the same key will always be matched in one column, no matter where they are located in the actual survey. If new questions are added mid-survey, empty cells will appear in the column for those surveys conducted prior to the change.

Support and Documentation

Currently, Surveybe has manuals for enumerators and survey designers, assisted support for time-sensitive technical troubleshooting, and personalized packages for providing clients a wide array of support services. The manuals come with the software purchase, while the assistance packages are fee-based. For licensed users, Surveybe also has starter videos for using the designer, FAQ pages, and message boards that are user-driven in content but moderated by Surveybe. The vision for the message boards is to provide users a forum to ask questions, share solutions, and even share surveys and survey code.

For the Implementer and Designer packages alike, the manuals offer a thorough, well-structured, easy-to-follow walk-through environment. Filled with helpful screenshots and practical examples, new users can quickly familiarize themselves with extensive explanations of common SQL commands and extended examples of complex roster-within-roster designs, while the Designer manual satisfies the needs of more advanced users. Whether the additional pieces of the intended future support system live up to this example remains to be seen.

Pricing and Upgrades

Surveybe has a unique pricing structure. In the mold of other CAPI products, Surveybe has a per license cost for its development software, Developer ($3,000/license, with bulk pricing possible). On the other hand, Surveybe's data collection software, Implementer, is free. It can be used on as many machines as the implementing agency deems appropriate at no extra cost.

Unlike the other software packages under review, Surveybe's variable costs occur not for software but for the amount of data collected. Data are collected free of charge with Implementer, but once the data are imported back into the Developer for analysis and export, a fee is calculated and charged per data point in a pay-as-you-go type format. The pricing scale, with illustrative intervals, is currently as follows:

- 250,000 data points for $1,875
- 1,000,000 data points for $3,5003,000,000 data points for $9,500
- 5,000,000 data points for $15,500
- 10,000,000 data points for $25,000
- Price based on the application for 10,000,001 and more data points.

This pricing system provides a flexible way for Surveybe to bill for usage. It makes Suveybe an affordable CAPI package for small surveys, and yet still competitively priced for larger ones. Nevertheless, its disadvantages are equally clear. Survey planners may find budgeting surveys with Suveybe difficult, or at least confusing. Survey developers, under this pricing scheme, could find asking more questions – e.g. in order to avoid "double-barreled" questions – a costly strategy.

Surveybe's upgrade system is not unique. Updates to software that provide patches or fix bugs are free. Upgrades that offer new or extended functionality are fee-based. Alongside upgrades are fee-based add-ins, called extras, that provide limited new features. Three such extras are envisioned to be automated GPS coordinate capture, support for multilingual instruments, and extensions to question media support, all of which are slotted for release throughout 2011.

Extensibility

Although a proprietary software, Surveybe offers openness in two ways, through external programming languages and through APIs. With HTML and SQL, survey developers can realize features that Surveybe never explicitly planned. Exercising powerful control over data tables with SQL, technically astute users can program novel ways of asking questions. In a recent example, one user created a sophisticated choice experiment that randomized some choices while keeping others fixed. Other avenues for extensibility may exist for HTML if the tool for integrating images into a survey instrument is extended to audio and video clips.

Beyond its designed extensibility, Surveybe is a product that is positioned to extend itself, and to do so in lock step with user needs. Surveybe has a current software development roadmap for new features, but also employs an agile management system that allows its internal development team to change priorities as a function of user feedback.

Surveybe's next major development priorities include: (i) case management functionality; (ii) allowing simultaneous development of questionnaires; (iii) possibility of copying and reusing structural components of previously developed questionnaires; (iv) multi-language features, for question text and variable and value labels; and (v) GPS capture, which pulls coordinates directly from on-board GPS chips.

Hardware and Software Needs

Surveybe designed its software explicitly to work on UMPCs/MTPCs. The software works on modern Windows operating systems as well as Linux. Surveybe is currently exploring an expansion to Android as

well. The main software requirement is that the full Java Virtual Machine be available. This should not be a limiting condition for most survey hardware.

Known software users: Ethiopia Central Statistical Agency; Oxford University; Clinton Health Access Initiative; International Livestock Research Institute; International Food Policy Research Institute; Georgia National Statistics Bureau; World Bank.

APPENDIX A – COMPACT CHECKLIST

See CAPI.Software.Assessment.Appendix.A.pdf downloadable at:

http://siteresources.worldbank.org/INTSURAGRI/Resources/7420178-1294259038276/CAPI.Software.Assessment.Appendix.A.pdf

APPENDIX B – DETAILED CHECKLIST

See CAPI.Software.Assessment.Appendix.B.pdf downloadable at:

http://siteresources.worldbank.org/INTSURAGRI/Resources/7420178-1294259038276/CAPI.Software.Assessment.Appendix.B.pdf

APPENDIX C – META-EVALUATION

See CAPI.Software.Assessment.Appendix.C.pdf downloadable at:

http://siteresources.worldbank.org/INTSURAGRI/Resources/7420178-1294259038276/CAPI.Software.Assessment.Appendix.C.pdf

APPENDIX D – FEATURED USERS

See CAPI.Software.Assessment.Appendix.D.pdf downloadable at:

http://siteresources.worldbank.org/INTSURAGRI/Resources/7420178-1294259038276/CAPI.Software.Assessment.Appendix.D.pdf

APPENDIX E – SOFTWARE SCREENSHOTS

See CAPI.Software.Assessment.Appendix.E.pdf downloadable at:

http://siteresources.worldbank.org/INTSURAGRI/Resources/7420178-1294259038276/CAPI.Software.Assessment.Appendix.E.pdf

APPENDIX F – GLOSSARY

Algorithm Assisted Coding: typing in a word or set of words will search an database and return a similar or matching value that the user can select from. For example, typing in "Toyota" would return the values "Toyota Prius," "Toyota Camry," "Toyota Corolla," etc. The user would then be able to choose from the different makes of Toyota brand cars.

Cascading Selection: sometimes also referred to as dynamic selection list. Cascading selection usually works on two or more lists and the contents of each successive list is dependent on the selected value from the previous list. The content of the second list is dependent on the selection in the first list and the content of the third list is dependent on the selection in the second list, and so forth. For example, if a user selected Maryland in the first list of states, then the second list would contain the name of counties in Maryland and the third list would contain the names of the cities in the county selected in the second list.

Cloud-hosted: when data are stored in such a manner so that it allows users in different geographical positions to be able to access the data through the Internet. For example, if the data from a survey being performed in Ethiopia is cloud-hosted, then personnel at the Washington, DC headquarter would be able to use the Internet to access and view the survey data.

Complex Skip Patterns: skips that occur based on the responses to several questions that are usually located in many different parts of a questionnaire.

Data Dictionary: a collection of descriptions of variable names or items in the data to assist programmers or users who need to refer to them.

Data Element: a basic unit of information that has a unique meaning and distinct values. For example, a household member's name is a data element.

Dynamic Prompt: a question prompt where parts of the prompt are filled in based on information provided earlier in the questionnaire. For example, if the input for the household member's name is Jane Smith, then the prompt for a later question inquiring about age would be customized to state, "What is Jane Smith's age?"

Dynamic Range Restrictions: using previously collected information earlier in the questionnaire to determine the possible responses for a question. For example, if respondent reported having spent $100 last week, an error should be triggered if the sum of the amount of money that spent in each the sub-categories (i.e. food, clothing, housing, etc.) exceeds $100.

Data Validation: a method that specifies the logical relationship between data elements. If the logical relationship is not satisfied, a warning or an error will be invoked. For example, an error should appear when a household is reported to contain more than one head of household.

Enumerator: the person conducting the interview and filling in the questionnaire.

Flat data file: a file containing information that has no structured interrelationship or descriptive information of any kind. For example, a comma-separated value (csv) file is a flat data file because it

does not contain variable or value labels, nor does it contain any information about the level (i.e. household, person) at which the information was collected.

Hard error: an error that cannot be suppressed and must be resolved by changing one or more values. For example, a mother's reported age is less than the age of her daughter.

Hierarchical: the relationship between different parts of the data with respect to the level at which the data was collected. Examples of hierarchies include the household level and the person level. Hierarchical data maintains information about the relationships between different parts of the data.

Looping: a program that performs a series of instructions repeatedly until some specified condition is reached. This usually involves asking the same set of questions, such as asking demographic questions for all the members in a household.

Non-linear completion: filling out the questionnaire in a manner where the questions are not answered in the exact order that they appear on the questionnaire. CAPI programs that can facilitate non-linear completion will allow enumerators to move freely around the questionnaire, such as being able to skip over certain questions or entire modules. An example of when non-linear completion would be useful is when the respondent is the male head of the household and cannot accurately answer questions about child-feeding practices. In this situation, the enumerator would be able to skip questions relating to that topic until the mother could be interviewed.

Rostering: the presentation of questions in a matrix format.

Routing: skip functions that are defined as either simple or complex. Simple routing leads users to a path/question based on the answer of a given question. Complex routing leads users to a path/question based on a series of answers from multiple questions.

Soft error: an error that can be suppressed. This error is usually triggered by a response that is unlikely but not impossible, e.g. a household that reports having 20 members.

Syntax highlighting: a feature of text editors that adds formatting attributes to terms in different categories. This feature makes it easier for readers to understand code by visually distinguishing the code structure. For example, syntax highlighting would color code functions (i.e. generate, if/then statements) as green while variable names would be purple.

Trigram search: a method that allows for the search of "near matches" and does not require exact spelling. This type of search is useful for both long and hard-to-spell text strings (i.e. "medicine"). For example, typing in "wor" would return "World Bank," "WordPress," etc.