BlaiseIS Sample Management

Hueichun Peng, Lisa Wood and Gina-Qian Cheung - Survey Research Center, University of Michigan

1. Introduction

The University of Michigan's Survey Research Center (SRC/SRO) has been involved in research projects using web surveys as a means of data collection for years. We have used a variety of web survey software packages such as Inquisite and DatStat Illume. Some of the software comes with limited technical systems for sample management, such as tracking survey status, sending out emails and data report download. As these research projects with web surveys are getting dynamic in a variety of manners (e.g., mixed mode data collection and diary types of web surveys), we get ever increasing complex technical specifications and challenges. As a result, we have developed various systems for sample management for web survey projects.

In the last 3-4 years, SRO has initiated exploration of web surveys using BlaiseIS. In August, 2009, we started to do extensive testing and prototyping with BlaiseIS 4.7 for a large scale project, the Army Study to Assess Risk and Resilience among Service Members (STARRS). The original work scope for the Army STARRS project involved using web surveys on a very large scale (400,000+ respondents) and as part of mixed mode data collection in different waves. We made the technical decision to use BlaiseIS as the web survey data collection tool.

In May 2010, we used an internal SRO project to launch key components with BlaiseIS as a pilot project as well as proof of concept for our design and programming work with the Army STARRS Project. Since BlaiseIS does not have a sample management system (SMS), we included programming BlaiseIS sample management functions in our development. The result was very successful.

SRO's implementation of BlaiseIS involves incorporating the web SMS modules within our current Blaise SMS (SurveyTrak) system. We will explain how we designed and managed the process flow of running a BlaiseIS survey, starting from loading sample lines and key preload information, designing and scheduling email jobs, logging survey status, reporting, and closing out the survey.

2. Key Components of BlaiseIS Sample Management

Using the knowledge we have gained from working with web survey projects, we defined five key components we needed for managing sample with BlaiseIS.

1. Link the SMS database with BlaiseIS
2. Create email functions
   - Send email invitations and reminders with customized templates
   - Track bounced-back emails
   - Automatic resend efforts
3. Monitor/control sample
4. Manage/integrate web contact attempts with other contact efforts such as call records
5. Manage survey status and store key BlaiseIS paradata variables in SMS for production monitoring

2.1 Link the SMS Database with BlaiseIS

Linking sample management data with BlaiseIS was first accomplished by parallel preload of sample lines in BlaiseIS and our SMS (SurveyTrak) database. Specifically, the project ID, sampleline ID and login credentials were preloaded into both BlaiseIS and our SMS.

The remainder of this section outlines the process used to allow the respondent to access the BlaiseIS survey and to update our SMS database with key data from BlaiseIS (see figure on next page).

2.1.1 Authentication using our existing SMS/SurveyTrak database

Authentication using our existing SMS/SurveyTrak database involved multiple steps:

1) The Respondent (R) is sent an email directing them to a survey portal-login page. The respondent logs onto a portal page outside of BlaiseIS. This portal-logon page validates credentials in our SMS database.
2) If authentication passes, the R is automatically routed to the BlaiseIS start-up asp page.
3) If the authentication fails (e.g., credentials are not valid, the case has already been completed, or the sample line status is closed out), the R will be routed to customized warning pages or other processing module.

2.1.2 Routing to BlaiseIS start-up page

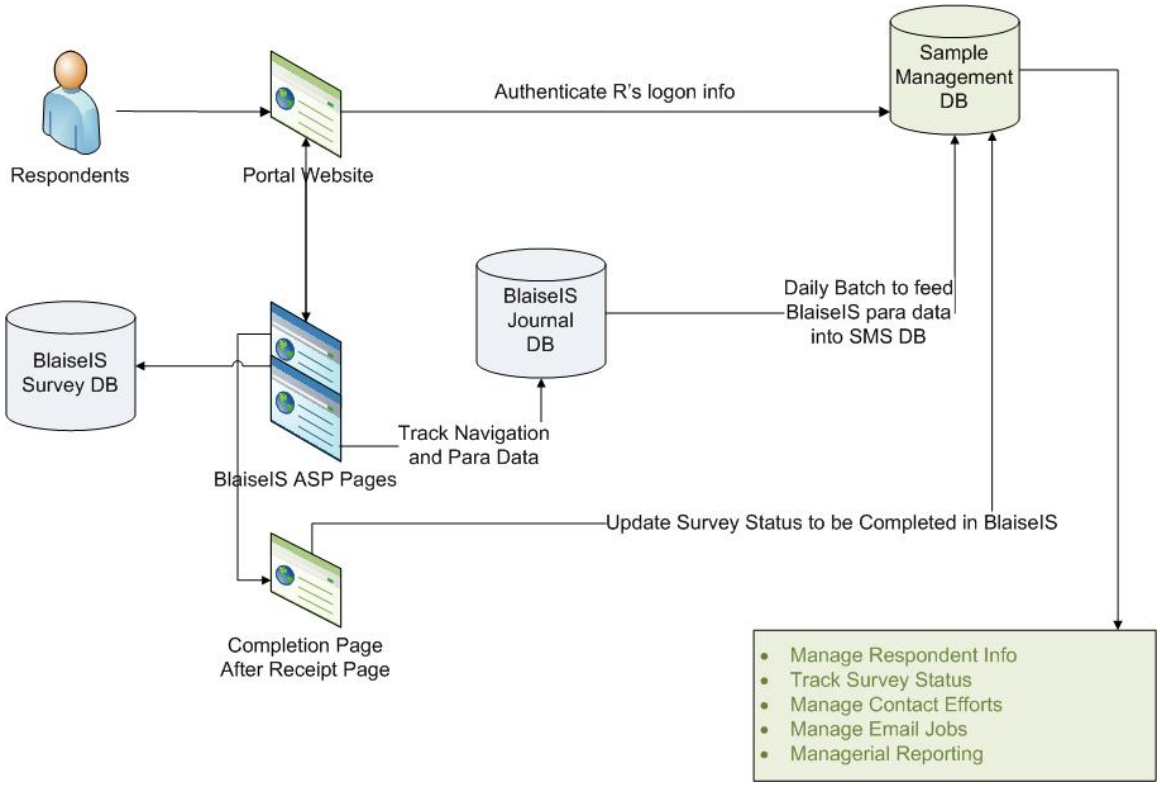
If R passes the authentication at the Portal page, it gets into the BlaiseIS start-up page. For our pilot study, the stored procedure at authentication passes in the logon credentials and the project ID and outputs sample line ID and the location (i.e., URL) of the BlaiseIS start-up page. Then we automatically redirect R into BlaiseIS. The sample line ID is the unique key in the BlaiseIS database.

2.1.3 Determining survey completion

We captured survey completion by adding a second page after the receipt.asp. When R gets to this last page, the page calls a stored procedure in our SMS/SurveyTrak that updates the sample line as the survey is completed. In addition, we update the SMS/SurveyTrak result code to a finalized result code and add a final contact record.

2.1.4 Capturing BlaiseIS paradata

We added modules to capture detailed paradata and navigation data and save these data in a BlaiseIS Journal (see paper by Ostergren and Liu, IBUC 2010). Then we synchronize these paradata (on an aggregate/sample line level) to our SMS as a scheduled job (data was linked using the sample line ID). This information is used for to create managerial reports and for production monitoring.



2.2 Create Email Functionality

We added two modules to (1) automate the process of sending email invitations and reminders, and (2) to capture and log bounced back emails. We developed a suite of C#.Net applications to handle these two processes. For the Send Email module, we used the SMTP Client object of the .Net Mail Namespace; for the Bounce Back Checker module, we used IMAP programming with .NET Sockets Namespace.

Overall, the design of the two Email Modules was to try to minimize human effort in managing the email contacts for web surveys.

2.2.1 Send Email Module

The sending email function is triggered to run at scheduled times that are pre-determined and specified by the project. The email template can be customized with pre-filled information from the sample line. For the pilot project we conducted in May, the sample was divided into two types. For each sample type, there were three templates:

1. Invitation
2. Reminder 1
3. Reminder 2

Criteria for which R should get an email of each type were based on data for the R that was included into our SMS database and the pre-determined dates. For example, if the R has already completed the survey, reminders would not be sent.

When the send email module is triggered, it calls a stored procedure in the SMS database by passing in project ID, type of email, and the time the job is triggered. The stored procedure outputs a record set (i.e., a list of sample lines that should receive the email); our C# .Net program processes the record set and sends out emails one by one. The email has the following information and functions.

- Prefilled information specific to R (such as names)
- Credential data to log onto the BlaiseIS survey
- Portal page URL
- The email content and subject line are customized by pre-defined logic (different email templates for different sample types)
- After the email is sent, the program calls a second stored procedure in our SMS to record a contact attempt record for the R
- Email can be scheduled to run at different times with pre-defined logic
- We use ASCII encoding format for the email body
- Email is sent from a specific email box. The Email Bounced Back Checker (BBC) module checks emails in this email inbox to identify and process bounced back emails.

2.2.2 Bounced Back Checker (BBC) Module

The second part of email program is to track and handle bounced backed emails. The purpose of this module is multi-functional, including:

- Catch the bounced back emails that result from our email sending program
- Record the bounce back in our SMS database
- Inform project and production managers of the bounced-back email
- If possible, launch resend efforts

Bounced back emails will come back to the email inbox used by the Send Email Module. Due to logging by the BBC, this specific email box should only be used by Email Modules and should not to be used for managerial communication purposes with the R. We set up a second study email box, and directed respondents to use this second email box for any replies back to project staff.

The following outlines the process established for the Bounce Back Checker (BBC) module.

- 1st Step: Create a Bounced Back Definition table that specifies the bounced back text to search for and the error codes assigned to each kind of text. This can be configured and customized by project. The table below lists the definitions we used for our Pilot Project.

lvBouncedMessage	vBouncedCode	iErrorCode
The message could not be delivered because the recipient's mailbox is full.	MailBox is full	2006
The destination server for this recipient would not be found in Domain Name Service (DNS). Please verify the email address and retry.	Problem with recipient's server	2006
I'm afraid I had problems forwarding your message.	Could not deliver at time sent	2006
Invalid email address	Insufficient address	2006
MDaemon had identified your message as spam. It will not be delivered.	Return as spam	2006
This Message was undeliverable due to the following reason: The recipient(s) indicated below did not receive this message because their mailbox size limit would have been exceeded.	MailBox is full	2006
Could not deliver the message in the time limit specified. Please retry or contact your administrator.	Could not deliver at time sent	4901
The e-mail system was unable to deliver the message, but did not report a specific reason.	Not deliverable as addressed	2006

- 2nd Step: Logging emails that appear in the study email send inbox. The BBC module opens emails one by one at the root folder of the study's specified email box. The program parses the content (including header) of the email messages in this email box and tries to match the text with bounced back definition texts we defined in step #1.

  For example, we defined that if the program finds "The message could not be delivered because the recipient's mailbox is full", we categorize it as a "Mailbox is full" bounced back and code it with the "2006" bounced back code. Or, if we find the text "Could not deliver the message in the time limit specified. Please retry or contact your administrator", we treat it as a "Could not deliver at time sent" bounced back and code it with the "4901" bounced back code.

  If the parsing finds a match, it moves the email to a processed sub-folder, and logs the email in a Bounced Back Log table with a bounced back code. If the parsing cannot identify the email message, the program either leaves the message in the root folder for managerial checking or moves it to a pending sub-folder. Review of the bounce backs that were not logged could result in new bounced back messages being added to the Bounced Back Definition table.

- 3rd step: The BBC module is triggered to run at scheduled times to process the logged bounced back email messages as a batch. At the end of each batch, the program calls a stored procedure in our SMS, which processes these bounced back logs with pre-defined business rules. For the pilot project, the stored procedure compares the bounced back logs to see if there is any email that matches with the email sent out from our sending email program. If we find one, the procedure inserts a contact record for R and then codes the sample line for R with the specified result code.

- 4th step: For the pilot study, we did not implement re-send efforts if the module finds a bounced back email sent from our program. However, with the Army STARRS Study, we considered very comprehensive business rules regarding whether we should trigger another email from our Send Email module if we have a second email address for the R (and the first email bounced back). Manual review of cases that did bounce back was also an option, which would include allowing the production management staff to update email addresses and flag a case for resend of the invites/reminders.

2.3 Monitor and Control Sample Status

The basic concept is to monitor and track the current status (result code) of the sample at any point in time so that managers can make proper managerial decisions. Since our SMS has full functionalities in this regard, we do not need to make extensive revisions for addition of BlaiseIS sample. What is specific to BlaiseIS is to define another layer of business rules that will drive the BlaiseIS survey, for example, what result codes can be linked with what actions. For instance, if we code out certain sample, the portal will block logins for those sample. Or if we have a diary or journal kind of project, we will track the progress and then route R to correct sequenced journal from the portal.

2.4 Manage and Integrate Contact Efforts on the Sample Line

A lot of web survey projects will still involve phone calls and human tracking efforts like in-person visits or mail communication. Respondents might call in to report problems, opt out of the study, inquire about project details, or even update contact information such as emails, contact address, etc. Our SMS has full functionalities for this part. What we added for BlaiseIS is the logging of the email contact attempts (both outbound attempts such as invites and reminders, and inbound attempts from bounced back emails). These contact attempts can be integrated with contact attempts from the phone or in-person follow-ups. For some projects, the data collected from these contacts might get (pre-) loaded into the web survey itself (such as confirming if the contact information is correct).

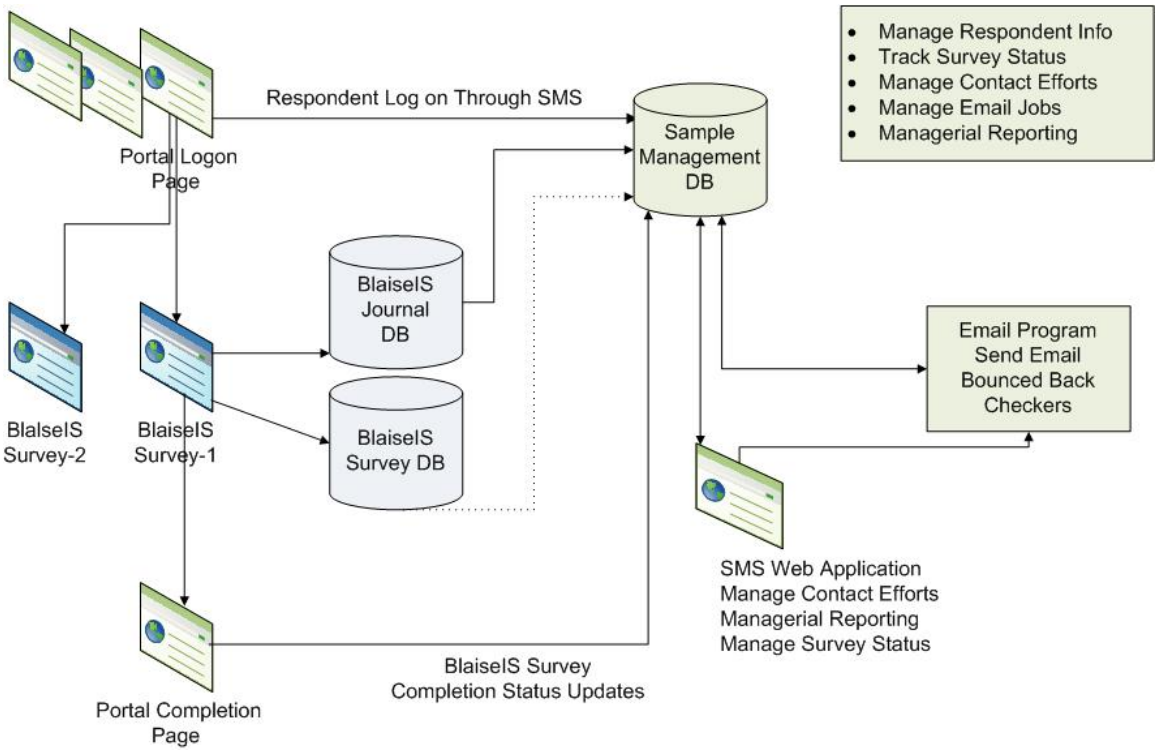
2.5 Reporting

Since our SMS database has integrated the up-to-date tracking of the survey status, contact records, and other BlaiseIS paradata, we are able to set up reporting for the following.

- The cases that have been completed.
- The cases that have been started and aborted.
- Reports showing more detailed information about our sample from the BlaiseIS paradata such as the last question answered, total time in survey, the number of survey logins and break-offs, the browser used, etc.
- The contact efforts for the sample including counts of how many emails were sent out, emails that bounced back, human calling efforts, paper communication, etc.

3. Vision

We propose to use a Portal framework for BlaiseIS surveys. Every R will go into the BlaiseIS world through a common or specific portal. The logon action will interact with the SMS database, route R to correct BlaiseIS Survey Page and track completion status. The integration of BlaiseIS paradata into SMS will facilitate very efficient managerial monitoring and reporting. A lot of managerial functions such as sending emails, checking for bounced back emails and tracking of contact efforts can be integrated into the SMS database.



4. Challenges

Although our pilot using SMS with BlaiseIS was implemented with much success, some challenges were considered as we consider further use.

4.1 Portal concept

The portal concept/framework is not just one or a few web pages; it is a framework that is to wrap the BlaiseIS surveys for sample management purpose. How to design this portal will depend on the complexity of the relationship between the sample and the surveys.

In most cases, we will have one survey for one sample line. We might have multiple surveys activated for the same sample line. Or we could have multiple system entry points that will need access one survey. For example, we invite R to take a web survey, then for follow-up efforts, we might call R and help R to take a survey. This involves mode change on one survey. Ideally this portal concept should be able to handle the variety of operation.

4.2 Cross browser compatibility and consistency

The second challenge is a technical issue. It seems to us that BlaiseIS needs to be tested with a variety of browsers. For instance, we found the behaviors of users pressing browser back/forward buttons are different in different browsers. Since it is extremely hard to control and dictate the browser type when we launch a web survey, to have consistency and compatibility across major browsers will be crucial. As we have learned in a few projects with web surveys, browsers on PDA device have been ever increasingly brought up by R.

4.3 Mixed Mode Sample Management

As we began development of sample management for BlaiseIS, it was with the vision of using as part of mixed mode data collection. As the scope of work changed, our pilot did not include mixed mode sample data collection. Initially we were planning mixed mode data that would be a sequential mode switch from web to phone, not simultaneous. Issues considered included whether we would use a modified version of the BlaiseIS survey or Blaise for decentralized phone follow-up after non-response, how to modify our sample management system to provide feedback to interviewers about the status of the web data collection, and how a mode switch would work when R provided a partial web survey. Developed to be expandable, one of our next steps will be to pilot our BlaiseIS sample management system for a mixed mode data collection project.