Automated Testing of Blaise Questionnaires

Jay R. Levinsohn, Research Triangle Institute
Gilbert Rodriguez, Research Triangle Institute

INTRODUCTION

Social science data collection has seen a significant movement toward the use of automated data collection procedures, using increasingly sophisticated tools. The Blaise programming language and data collection system is one example of this kind of data collection tool. While we see increasing numbers of social scientists designing large and complicated data collection instruments under Blaise we have not seen the parallel development of tools to aid in the testing and certification of these instruments. As the specification for a given question grows in size and complexity its implementation into a Blaise program grows into a significant software development project. While there are tools to help in the software development process in Blaise we do not see the tools that support quality assurance and validation.

At the Research Triangle Institute (RTI) the National Household Survey of Drug Abuse project uses the Blaise system to collect computer assisted personnel interview data from almost 70,000 individuals each year. This data collection effort is an annual effort with quarterly data collection waves each year. The questionnaire is large and increases in programming complexity each year. The typical quality assurance (QA) methods of manually testing the instrument for adherence to the designers specifications, accurate program logic and flow, correct data storage and error free transmission of the data is an increasingly time consuming, expensive, slow, and error prone set of procedures. The QA process becomes even more problematic as one repeats it over the inevitable questionnaire revisions; each with tighter timelines that occur as a given questionnaire approaches its production deadlines.

Over the past year RTI has been developing a set of tools to allow the Blaise developer to automatically test some components of the questionnaire implementation. RTI has developed a prototype system that allows the questionnaire developer to create “test scripts” that can be fed into a working Blaise questionnaire implementation, in the actual data collection configuration, as a keystroke stream. The test scripts are developed to exercise the flow, variable calculation, sub-sampling, and all major paths through the questionnaire. As the script stream is fed into Blaise, the program flow is monitored with respect to the correct question sequence (as defined by the script) and the correct exit point from the questionnaire. Testing logs are generated for each script documenting time and date of testing, screen by screen progress through the questionnaire, and the data entered on each screen. The data output from a given script goes into the Blaise database as a simulated questionnaire completion. The Blaise database can then be compared to the script input for validation or as the case at RTI we actually transmit the data, simulating field procedures, through the laptop transmission system and then validate the data received against that expected from the script.

This relatively simple process provides several benefits:

- Complete end to end testing of keystroke input against received data output
- Documentation and repeatability of the testing process
- Validation of specific paths through the instrument, one per test script
- Rapid re-testing as one steps through the final iterations of the development process (regression testing)

This process still leaves significant elements of a questionnaire untested:

- Screen text validation
- Answer text validation
- Sound file validation
- Paths and calculations not covered by the range of test scripts

RTI is continuing to investigate adding new elements to the testing system. We are currently working on a component that would allow “Random Walks” through the questionnaire to test for logic errors, flow errors, and range errors. In addition we are looking at development of a more formal process for creating questionnaire specifications that might facilitate testing screen and answer text.

THE METHODOLOGY

The goal of this testing process is to insure that a given Blaise program faithfully implements the specifications of the questionnaire designer. Figure 1, shown below, depicts the process of designing, developing, testing and then rolling out a questionnaire. The figure shows a process in which a design leads to written specifications that are then prototyped. The prototype is reviewed and tested. The design specification stage may be repeated, multiple times, until the designers, testers, and reviewers feel that the questionnaire is correct and ready for production testing. At this stage a round(s) of coding, test and review is done until all parties feel the instrument is correct. In terms of the work we are doing here being correct implies that the program implements the written specifications. In all cases, if errors or the need for change are found the design documents and written specifications must be changed, followed by another round of testing.

This process if conducted without error should provide a Blaise questionnaire that treats a given sequence of interviewer inputs according to design specifications. The output from a specific set of interviewer inputs should produce a predictable set of data outputs in the Blaise database. Using this idea we have developed software and a set of procedures that tests Blaise questionnaires with the goal of verifying that a specific set of inputs (as codified in a script) produces a predictable set of Blaise outputs. Figure 2 highlights the points at which the system tests. As shown in Figure 2 we are trying to verify that the specification developed and that data produced in the database, given a known set of inputs, are in concert. In order to do this we developed the idea of creating “Test scripts” that are derived from using only the questionnaire specifications. These scripts would then be feed into the Blaise program. If the program is correct we can predict exactly which fields and which data values should be in the Blaise database. This is essentially the model of testing that is done manually when a designer walks through a Blaise program to test the program flow and data validation. The tester would enter data and using their knowledge of the questionnaire specifications, then review and confirm the data entry and routing through the questionnaire.

This testing process is implemented in a collection of software tools that we have named RoboCAI. RoboCAI uses the following tools:

- WinBatch - a batch language interpreter for Windows. Batch files are written using WIL, the Windows Interface Language. A batch file can contain commands to run programs, read and write directly to files, send keystrokes to Windows applications, and perform a number of other tasks.

Using the WinBatch compiler, an executable file can be created from a batch file. WinBatch is a proprietary product developed by Wilson WindowWare, Inc.

- Blaise 4.x – CAI instruments created with different versions of Blaise 4.x have been used with RoboCAI. Currently, Blaise 4.5 is being used.
- A modified audit trail DLL, audit.dll – This DLL, written in Delphi, writes out the name of the current field in an instrument to a text file where it can be read by a WinBatch batch file or executable.

When executed, the RoboCAI program file prompts the user for the name of a text file containing response data corresponding to an interview case (which is called a script file) and then prompts the user for the name of a log file to which diagnostic data is written. The RoboCAI program starts CAI instrument, reads the response data file, and then sends the response data keystrokes to each screen of the CAI instrument. The audit trail DLL writes the name of the current screen name of the CAI instrument to a text file (currentfield.txt) so that the RoboCAI batch file can compare the script file routing to where the CAI instrument reports as its current location. Figure 3 presents a flow diagram of these steps.

The flow and logic of RoboCAI is simple with the one complication of coding to allow for synchronization of the activity between the Blaise audit.dll and the Winbatch code. These two processes, RoboCAI and Blaise, run in parallel under the Windows system and the RoboCAI code must wait for the Blaise code to complete. The RoboCAI code must allow the Blaise audit.dll enough time to open, write into, and close the screenfile.txt file before it can check to see if it is still tracking the flow through the questionnaire correctly.

An additional software module was written, in Visual Basic, that completes the last part of the test sequence. This code compares the extracted text file from Blaise that contains the data stored during the RoboCAI phase to the data input from the script. The comparison process documents the test and highlights any differences between script data and Blaise data.

EXAMPLE

The following sections present a short RoboCAI example. In Tables 1 through 4 we present the information that serves as input into the RoboCAI test process and the outputs. In this example scenario the "respondent" is a 21 year old male from Wyoming who is on active duty in the military. Hence, for this example survey, he will be routed out of the interview very early since he ineligible to participate in the survey. QUESTID is an identifier for the interview case and is the primary key for the datamodel.

A. Sample Questionnaire

The CAI questionnaire screens that appear for this scenario are shown in Table 1.

Table 1: A Sample CAI Questionnaire

CAI SCREENS FOR QUESTID 2000024

****
Question Name: startup

INTERVIEWER: SELECT THE LANGUAGE TO BE USED FOR THIS INTERVIEW.

1 ENGLISH

2 SPANISH
3 MULTIMEDIA LANGUAGE

Response: ENGLISH
****
Question Name: note1

FI: DO NOT READ ALOUD UNLESS RESPONDENT QUESTIONS THE BURDEN (OR TIME) ASSOCIATED WITH THIS INTERVIEW.

NOTICE: Public reporting burden for this collection of information is estimated to average 60 minutes per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information ………

PRESS [ENTER] TO CONTINUE.

Response: [Return] to continue

****
Question Name: remindfi

INTERVIEWER: IF YOU HAVE NOT READ THE "INTRO TO CAI" IN YOUR SHOWCARD BOOKLET ALOUD TO THIS RESPONDENT, DO SO NOW. WHEN RESPONDENT IS FULLY INFORMED, CONTINUE WITH THE INTERVIEW.

PRESS [ENTER] TO CONTINUE.

Response: [Return] to continue
****
Question Name: age1

What is your date of birth?

ENTER MM-DD-YYYY.

Response: 7-24-1980
****
Question Name: confirm

That would make you 21 years old. Is this correct?

1 YES
2 NO

Response: YES
****

Question Name: FIPE1

INTERVIEWER: WERE 2 PERSONS SELECTED FOR AN INTERVIEW AT THIS SDU?

1 YES
2 NO

Response: NO
****
Question Name: FIPE4

INTERVIEWER: IN WHAT STATE IS THIS SAMPLE DWELLING UNIT (SDU) LOCATED?

1 ALABAMA
2 ALASKA
3 ARIZONA
4 ARKANSAS
5 CALIFORNIA
6 COLORADO
7 CONNECTICUT
8 DELAWARE
9 THE DISTRICT OF COLUMBIA (WASHINGTON, DC)
10 FLORIDA
11 GEORGIA

27 MONTANA
28 NEBRASKA
29 NEVADA
30 NEW HAMPSHIRE
31 NEW JERSEY
32 NEW MEXICO
33 NEW YORK
34 NORTH CAROLINA
35 NORTH DAKOTA
36 OHIO
37 OKLAHOMA

12 HAWAII
13 IDAHO
14 ILLINOIS
15 INDIANA
16 IOWA
17 KANSAS
18 KENTUCKY
19 LOUISIANA
20 MAINE
21 MARYLAND
22 MASSACHUSETTS
23 MICHIGAN
24 MINNESOTA
25 MISSISSIPPI
26 MISSOURI

38 OREGON
39 PENNSYLVANIA
40 RHODE ISLAND
41 SOUTH CAROLINA
42 SOUTH DAKOTA
43 TENNESSEE
44 TEXAS
45 UTAH
46 VERMONT
47 VIRGINIA
48 WASHINGTON
49 WEST VIRGINIA
50 WISCONSIN
51 WYOMING

Response: WYOMING
****

Question Name: FIPE5

INTERVIEWER: THE STATE YOU ENTERED IS:

WYOMING

IS THIS CORRECT?

1 YES
2 NO

Response: YES
****
Question Name: QD01

The first few questions are for statistical purposes only, to help us analyze the results of the study.

INTERVIEWER: RECORD RESPONDENT'S GENDER

5 MALE
9 FEMALE

Response: FEMALE
****
Question Name: QD03

Are you of Hispanic, Latino or Spanish origin or descent?

1 YES
2 NO

Response: YES
****
Question Name: QD04

HAND R SHOWCARD 1. Which of these Hispanic, Latino, or Spanish groups best describes you? Just give me the number or numbers from the card.

TO SELECT MORE THAN ONE CATEGORY, PRESS THE SPACE BAR BETWEEN EACH CATEGORY YOU SELECT.

1 MEXICAN / MEXICAN AMERICAN / MEXICANO / CHICANO
2 PUERTO RICAN
3 CENTRAL OR SOUTH AMERICAN
4 CUBAN / CUBAN AMERICAN
5 OTHER (SPECIFY)

Response: CUBAN / CUBAN AMERICAN
****
Question Name: QD05

HAND R SHOWCARD 2. Which of these groups describes you? Just give me a number or numbers from the card.

TO SELECT MORE THAN ONE CATEGORY, PRESS THE SPACE BAR BETWEEN EACH CATEGORY YOU SELECT.

1 WHITE
2 BLACK/AFRICAN AMERICAN
3 AMERICAN INDIAN OR ALASKA NATIVE (AMERICAN INDIAN
INCLUDES NORTH AMERICAN, CENTRAL AMERICAN, AND SOUTH AMERICAN INDIANS)
4 NATIVE HAWAIIAN
5 OTHER PACIFIC ISLANDER
6 ASIAN (FOR EXAMPLE: ASIAN INDIAN, CHINESE, FILIPINO,
JAPANESE, KOREAN, AND VIETNAMESE)
7 OTHER (SPECIFY)

Response: WHITE
****
Question Name: QD07

Are you now married, widowed, divorced or separated, or have you never married?

INTERVIEWER NOTE:
If the respondent is divorced but currently remarried, code as married.
By 'divorce' we mean a legal cancellation or annulment of a marriage.
By 'separated' we mean legally or informally separating due to marital discord.

1 MARRIED
2 WIDOWED
3 DIVORCED OR SEPARATED
4 HAVE NEVER MARRIED

Response: MARRIED
****
Question Name: QD08

How many times have you been married?

Response: 2
****
Question Name: QD09

Have you ever been in the United States' armed forces?

1 YES
2 NO

Response: YES
****
Question Name: QD10

Are you currently on active duty in the armed forces, in a reserves component, or now separated or retired from either reserves or active duty?

1 ON ACTIVE DUTY IN THE ARMED FORCES
2 IN A RESERVES COMPONENT
3 NOW SEPARATED OR RETIRED FROM EITHER RESERVES OR
ACTIVE DUTY

Response: ON ACTIVE DUTY IN THE ARMED FORCES
****

Question Name: MILTERM1

I need to verify what I just entered into the computer. You said you are currently on active duty in the armed forces. Is that correct?

1 YES
2 NO

Response: YES
****
Question Name: MILTERM2

```
People who are currently on active duty in the armed forces are not eligible to be interviewed in this study. I appreciate you taking the time to
speak with me. Thank you.

PRESS [ENTER] TO CONTINUE.

Response:     [Return] to continue
**********************************************************************

Question Name:     fiexit

End of interview reached.

PRESS 1 TO EXIT.
Response:     1

**********************************************************************
```

B. SAMPLE SCRIPT

Table 2 displays the corresponding RoboCAI script. When the RoboCAI program file is executed, it starts the CAI instrument, reads in this script and sends the appropriate keystrokes to each screen of the CAI instrument. The script file syntax allows for a totally blank line which is ignored, a line starting with a ";" which indicates a comment line and a line containing three fields each separated by a tab character. The three fields are: screen name, data value, and an optional comment (denoted by starting with a ";").

Table 2: A Sample RoboCAI Script

```
QuestID         2000024
; 10/24/2000    short one only military
;
; call startup and intro demographics
;
startup  1
note1
remindfi
age1     07-24-1980
confirm  1
fipe1    2
fipe4    51
fipe5    1
;
QD01     9        ;sex of R 9=female
QD03     1        ;Hispanic? 1=Yes, 2=No
QD04     4        ;kind of Hispanic (1-4, 5=other)
QD05     1
QD07     1
QD08     2
QD09     1
QD10     1
Milterm1          1
Milterm2
Fiexit   1
```

C. SAMPLE LOGS AND COMPARISON OUTPUT

The text displayed in Table 3 is the output log file generated by the RoboCAI program file. The log file contains the data that were used as input and lists any error or information messages from the test run. RoboCAI compares the screen names appearing in the script file to the actual screens encountered and writes out error messages to this file if any discrepancies occur. It also reports the data supplied to Blaise from the script.

Table 3 RoboCai Logs

```
RoboCAI Script Testing
Mon  5-07-2001 11:32:31 AM


     questID      2000024

      Blaise         Script               Script
      Screen         Screen               Data
      --------------  ---------            -------------------
1      STARTUP        STARTUP              1
2      NOTE1          NOTE1
3      REMINDFI       REMINDFI
4      AGE1           AGE1                 07-24-1980
5      CONFIRM        CONFIRM              1
6      FIPE1          FIPE1                2
7      FIPE4          FIPE4                51
8      FIPE5          FIPE5                1
9      QD01           QD01                 9     ;sex of R 9=female
10     QD03           QD03                 1     ;Hispanic? 1=Yes, 2=No
11     QD04           QD04                 4     ;kind of Hispanic (1-4, 5=other)
12     QD05           QD05                 1
13     QD07           QD07                 1
14     QD08           QD08                 2
15     QD09           QD09                 1
16     QD10           QD10                 1
17     MILTERM1       MILTERM1             1
18     MILTERM2       MILTERM2
19     FIEXIT         FIEXIT               1

============================================
End of Script
Mon  5-07-2001 11:33:00 AM
```

As mentioned in a previous section, an additional Visual Basic program was written that compares the extracted text file from Blaise to the data input from the script. Any differences found between script data and Blaise data are highlighted in a log. Table 4 displays the contents of the output log file from this program for this example. In Table 4 one difference is highlighted with "****". This difference stems from the difference in the date storage, Blaise converts input dates for storage as Day-Month-Year but the data was input as month-day-year.

Table 4: Comparison of Script Data and Blaise Data

```
05-07-2001    5-7-2001 11:49:20 AM       c:\roboCai 2001\24out.txt

Seq #        Field Name    Script           Data File
 2          STARTUP      {1}                [1]
 5 ****   AGE1          {07-24-1980}    [24071980]
 6          CONFIRM      {1}               [1]
 7          FIPE1          {2}               [2]
 8          FIPE4         {51}              [51]
 9          FIPE5         {1}               [1]
 10         QD01         {9}               [9]
 11         QD03         {1}               [1]
 12         QD04         {4}               [4]
 13         QD05         {1}               [1]
 14         QD07         {1}               [1]
 15         QD08         {2}               [2]
 16         QD09         {1}               [1]
 17         QD10         {1}               [1]
 18         MILTERM1   {1}              [1]
 20         FIEXIT        {1}               [1]
```

SUMMARY, CONCLUSIONS, FUTURE WORK

For our purposes, RoboCAI has proved to be effective. RoboCAI becomes very useful as the size and complexity of an instrument increase. While not intended as a means of checking the screen and answer text of an instrument or sound file validation, it does provide a means for checking the correctness of routing and the resulting data for an instrument. In particular, we have used it to perform regression testing on new versions of CAI instruments prior to deploying them for field use and for acceptance testing when upgrading to newer versions of Blaise.

Some possible topics for future work include,

- Work on automated script generation
- Develop a random walk application
- Investigate the use of Blaise API objects to extend testing capability of RoboCai
- Allow specification of date and time of test environment

Regarding the first item, a Visual Basic application has been developed that uses the Blaise API objects to write out RoboCAI scripts using Blaise data files as input. Thus, a user can test an interview case and if the results are acceptable, a corresponding RoboCAI script can be generated. The script can then be used as part of a test library for regression testing and for instrument certification.

Figure 1 -- Production Cycle

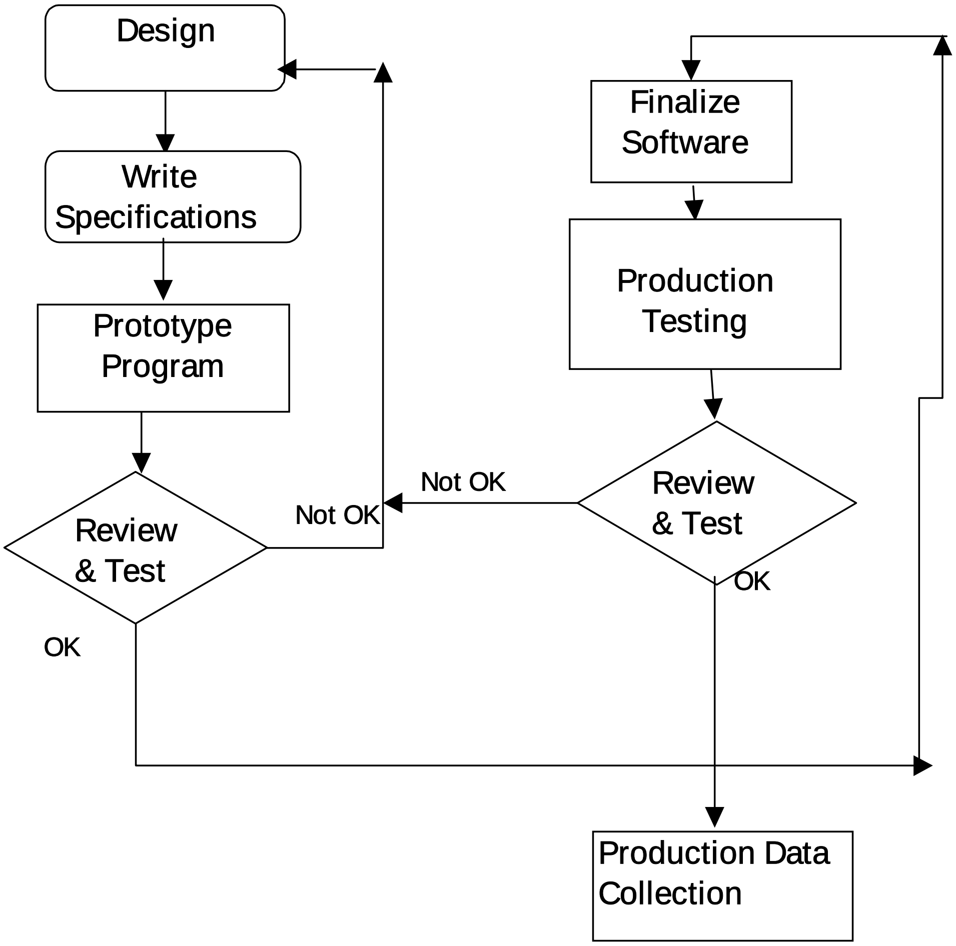


Figure 2 -- Test Goals

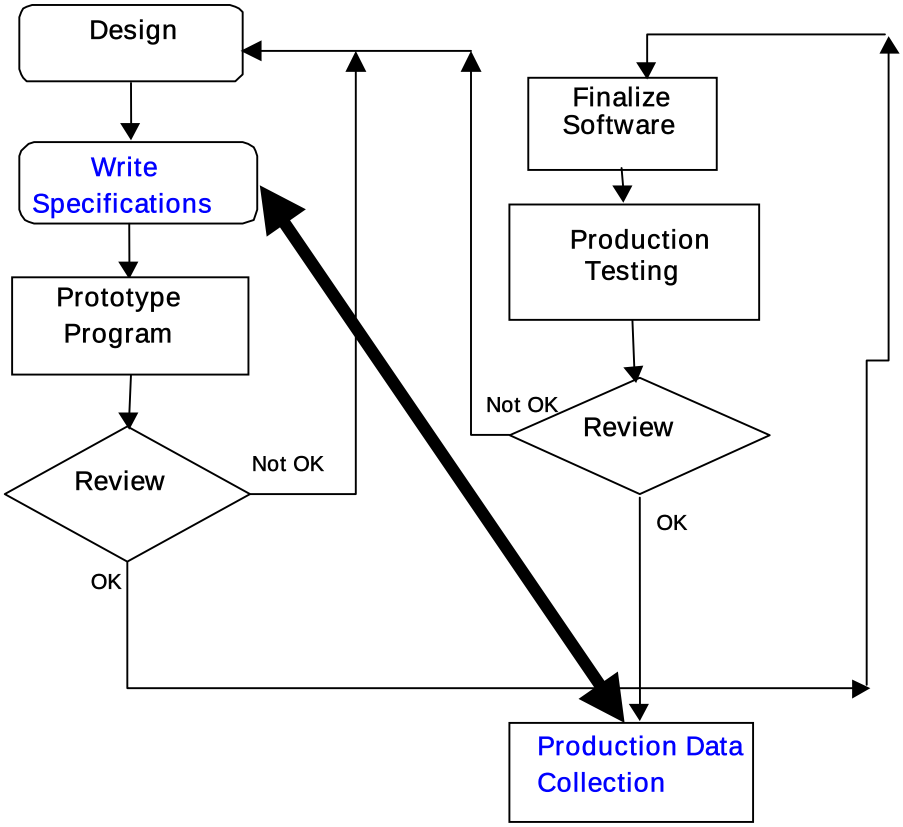
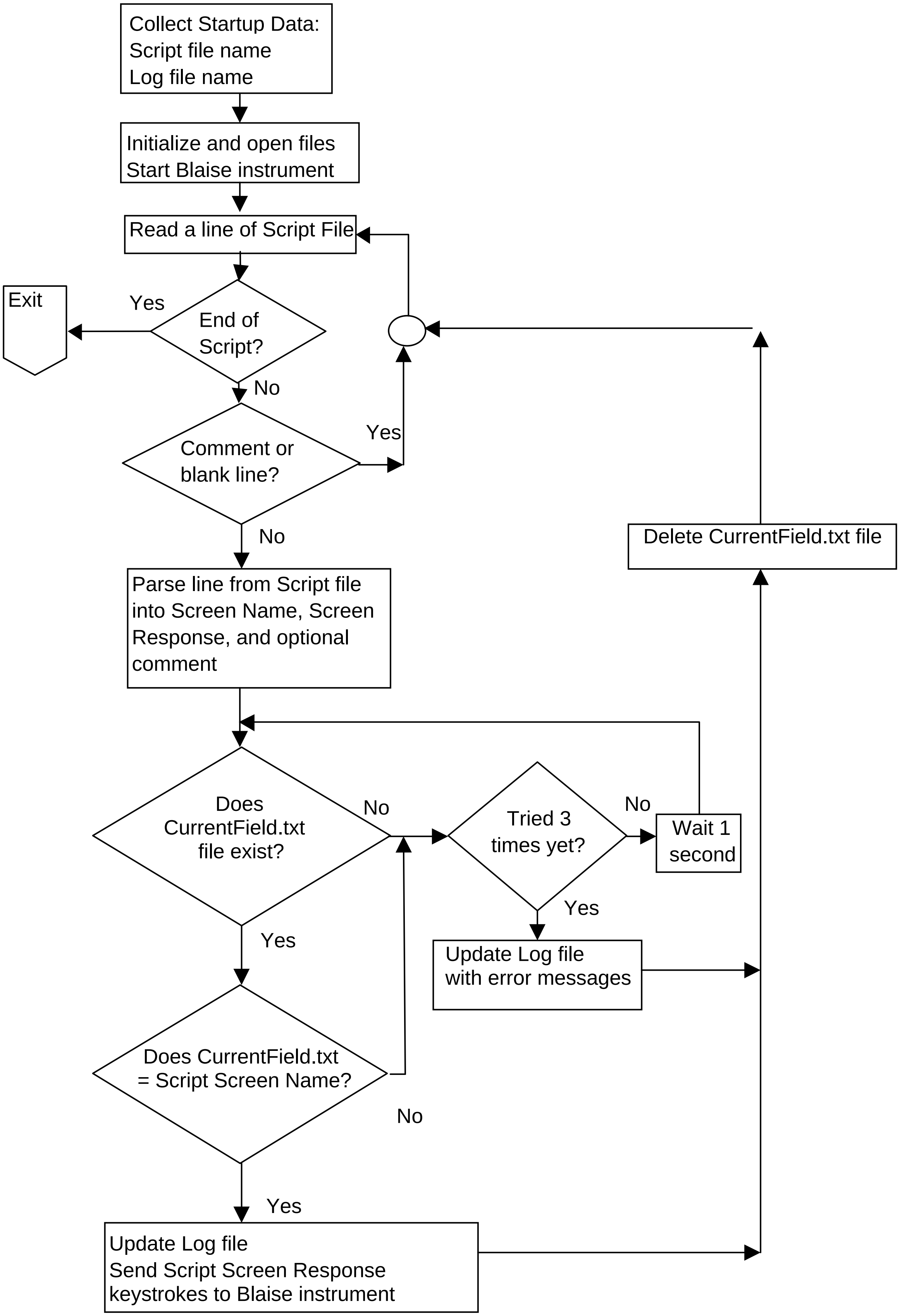


Figure 3 -- RoboCAI


Collect Startup Data:
Script file name
Log file name
Initialize and open files
Start Blaise instrument
Read a line of Script File
Exit
Yes
End of Script?
No
Comment or blank line?
Yes
No
Delete CurrentField.txt file
Parse line from Script file into Screen Name, Screen Response, and optional comment
Does CurrentField.txt file exist?
No
Tried 3 times yet?
No
Wait 1 second
Yes
Update Log file with error messages
Yes
Does CurrentField.txt = Script Screen Name?
No
Yes
Update Log file
Send Script Screen Response keystrokes to Blaise instrument