Blaise AT Report System

Youhong Liu, Eduardo Galvan, Gina Cheung
Institute for Social Research
University of Michigan

1. Introduction

During an interview or data entry session, Blaise records interviewer actions in an Audit Trail/Keystroke (ADK) file. This text file not only records the values of each field before and after the cursor enters a field, it also captures the actions that brought the cursor to the field and when exiting the field. Timestamps, keystrokes, remarks, the use of Help, switching languages, and other functions are recorded for each field visited.

In the last few years, the Institute for Survey Research at the University of Michigan (UM) implemented the Blaise Audit Trail for all projects and developed tools to analyze the data produced by them. In this paper, we will present the most recent tool developed, the Blaise Audit Trail Report System.

2. History

UM had an Audit Trail analysis program named ADK_Report.exe. It had been used extensively by the UM data processing and quality control team and methodology researchers. However, there were some problems and limitations with this program. The program also was written in Delphi, an older programming language no one in our group was familiar with. After using BCP for several years, we decided to rewrite the Audit Trial analysis program using BCP.

3. Programming Language and Backend Database

The programming language we used for the system is Visual Basic (VB). It is a high level language that we have much experience using with BCP. The system produces online reports that require a backend database such as Microsoft Access, Microsoft SQL or Oracle. MS SQL and Oracle seemed overkill for this system, while Access is not only more portable but does not have to be connected to a network to perform an analysis.

4. Reporting Tool

The built-in reporting tool found in VB did not produce the desired results. By switching to Crystal Reports and its many functions, the data could be presented in various ways. This will be discussed later in the paper.

5. Two Subsystems

The system contains two subsystems:

a. Standard and Customer Report Creation

This loads Audit Trail timing data to an Access database and produces various reports through the Crystal Reporting tool.

b. Audit Trail Data Generation

This loads data into a text file that can be used to feed to other software programs such as SAS, MS Access and Excel for further analysis. This option produces more robust information from the Audit Trail. The General Audit Trail option provides information on functions used, language switching and back up patterns that are not available with the Standard and Custom Report Creation option.

6. Standard and Customer Report Creation

a. Database Selection – Figure 1

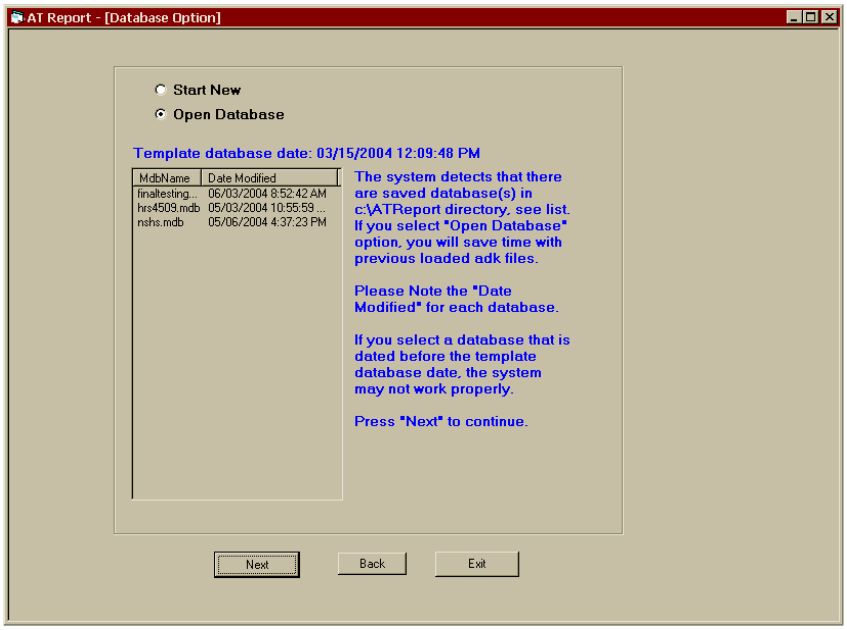

Figure 1 – Database Selection

The backbone of this portion of the system is an MS Access database – ATReport.mdb. The empty ATReort.mdb is located in the \support subdirectory where the AT report executable is located. E.g., if ATReport.exe is under C:\temp, then the Access file will be in C:\temp\support directory.

Database Structure:

Tables

- Case_Standard – For Blaise case level data
  Fields
  - CaseID
  - CaseTime
  - NumVisit
  - Iwer
  - IwerDate
  - FileIndex
- Field_Standard – For Blaise field level data
  Fields
  - CaseID
  - FieldName
  - FieldTime
  - Visits
  - BlockName
  - FileIndex
  - IwerName
- FileName – For file name (keeping track of the file name to save loading time)
  Fields
  - FileNamePath
  - FilePath
  - FileIndex
  - TSTAMP
  - FieldUpdate
  - CaseUpdate
- Field_Cust – For customer section reports
  Fields
  - FieldBlockName
- Section_Cust – For customer section reports
  Fields
  - SectionName
  - FieldBlockName

As shown in Figure 1, the user can perform one of two things:

- Start New:
  This will copy an empty ATReport database to the C:\ATReport directory. Upon exiting program, you will have a chance to rename the database to a meaningful name.

- Open Databases
  This will allow you to open a previously processed database. In this case, the data is already in the database, and it will eliminate time for loading data to the tables.

b. ADK File Selection (Figure 2)

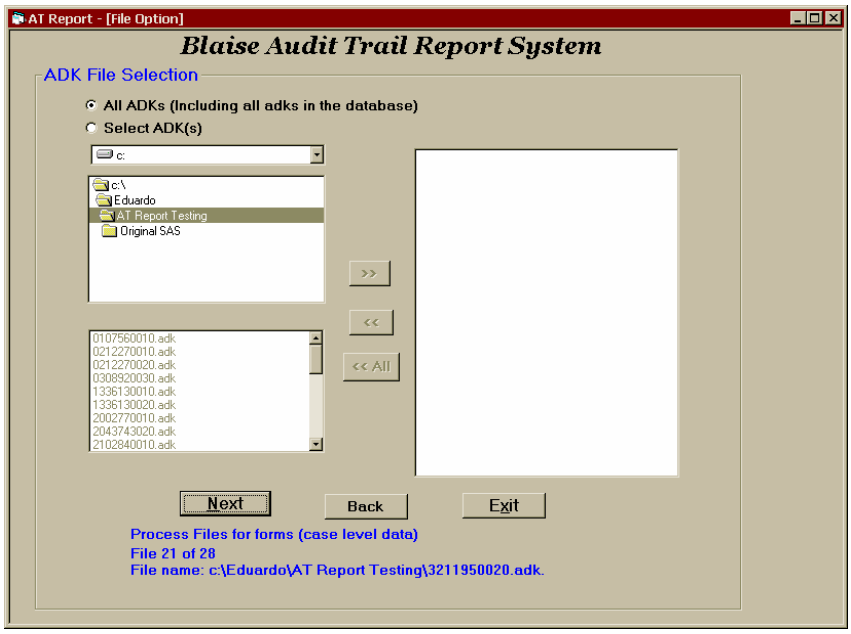

Figure 2 ADK File Selection

At this point, users can navigate to the directory containing Audit Trail files that are to be processed. At the University of Michigan, the extension of the Audit Trail file is ADK. The system offers two options. You can either select the whole directory of files or a subset of files in the directory.

c. Loading Data to Tables

After clicking "next" button shown in Figure 2, the system will start to load data to the Access tables; this part is the heart of the program. It processes Audit Trail files in two passes: case level and field level.

Case level:

Case ID – Primary Key
CaseTime – Total interview times of all forms for a case
NumVisit – Number of visits to a case
Iwer – Window user name in an Audit Trail
IwerDate – Interview date

Field level

Case ID
FieldName – Blaise field name
FieldTime – Total times for a field
Visits – Number of time the fields is accessed
BlockName – Top Block name of the field.
IwerName – Windows user name in an Audit Trail

d. Standard Reports (Figure 3)

Figure 3 – Standard Report Selection

Once the data are loaded into the database, the Crystal Reporting tool is used to generate reports. Unlike the Custom Reports that will be discussed later in this paper, the Standard Reports portion of the application does not allow a user to select which field or blocks to include in the timing reports that are generated. Although this may seem inflexible, the tradeoff is that these reports are simpler for the user to generate and also take less time to generate.

- Total Interview Time By Case ID—This report lists (for each ADK processed) the caseid, interviewer id, interview date, the number of times the case was accessed by the interviewer, and the total amount of time it took to complete the case.

  Basic SQL statement: *Select * from case_standard*

- Average Interview Time by Interviewer ID—This report lists average interview length and number of cases completed by each interviewer.
  Basic SQL statement: *Select * from (Select Iwer, Avg (CaseTime), count(caseID) from Case_standard Group by Iwer)*

- Average Interview Time for All Cases—This report provides the average interview length for all the cases that were selected for processing by a user. The total number of cases is also provided.

  Basic SQL statement:*Select Avg(CaseTime) From Case_Standard;*

- Average Field Time by Field Name—This report lists the Blaise instrument field name, the number of interviews in which the field was visited, and average time spent in the field.

  Basic SQL statement: *Select FieldName, avg(FieldTime), count(visits) from field_standard group by fieldname;*

- Average Block Time by Block Name—This report lists Blaise block names, the number of cases having timing data for the block, and average time spent in the

block. The average is based on the number of cases that visited the particular block and is not based on the total number of ADKs processed.

Basic SQL statement:*Select Blockname, avg (sumTime) from (Select CaseID, Blockname, sum(fieldtime) as sumTime from field_standard group by caseID, Blockname Having BlockName Is Not Null)*
Group by Blockname

e. Custom Reports – Selected Fields/Blocks

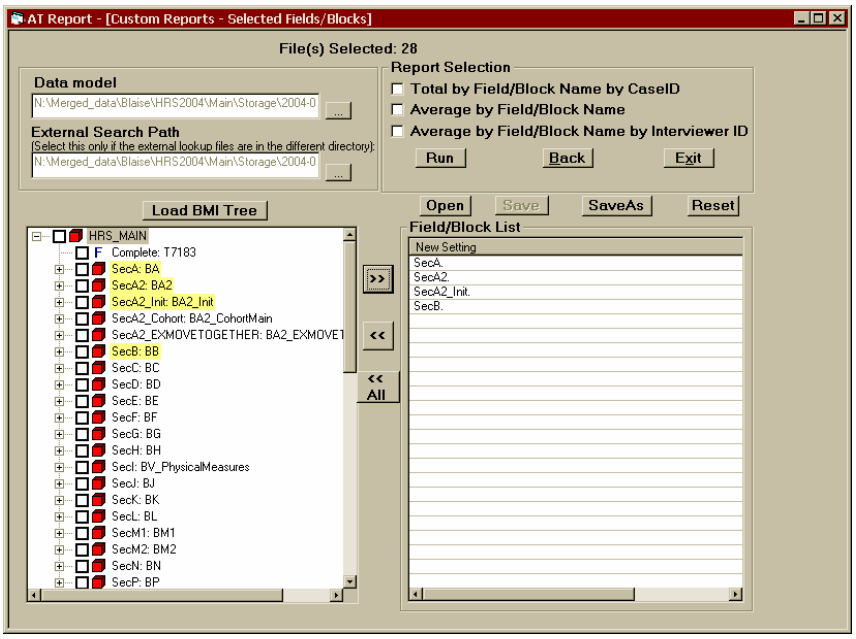

Figure 4 – Custom Report- Fields/Blocks Selection

The Custom Reports – Selected Fields/Blocks portion of the application gives a user more flexibility in determining which fields/blocks to use in running reports.

In order to pick Fields/Blocks, the corresponding Data Model has to be displayed for the user.

See Figure 4, after picking up the Blocks/Fields and "Run" button is clicked, the selected Blocks/Fields is inserted into Cust_field table.

The insert SQL Statement: *Insert into cust_field values ('" & ListView1.ListItems(i).Text & "')"*

- Total by Field/Block Name by CaseID—This report provides the total amount of time (in seconds) spent and the number of visits made by an interviewer into each block/field specified by a user. The data are sorted in CaseID order for each of the specified blocks/fields.

  Basic SQL Statement: *Select cust_field.fieldblockname, caseID, sum(fieldtime/1000), sum(Visits) from (select * from field_standard) as T1, cust_field where mid(T1.Fieldname, 1, len (cust_field.fieldblockname)) =cust_field.fieldblockname group by fieldblockname, caseID*

- Average by Field/Block Name—For each block/field specified the report lists the average time (in seconds) needed to complete the particular

block/field. The report also lists the number of cases that had the timing data for the particular block/field.

Basic SQL Statement: *Select fieldblockname, avg(sumfieldtime), count(CaseID) from (Select cust_field.fieldblockname, caseID, sum(fieldtime/1000) as sumFieldtime from (select * from field_standard) as T1, cust_field where mid(T1.Fieldname, 1, len (cust_field.fieldblockname)) = cust_field.fieldblockname group by fieldblockname, caseID) group by fieldblockname*

- Average by Field/Block Name by Interviewer ID--For each block/field specified, the report lists the average time (in seconds) needed by each interviewer to complete the particular block/field. A count of the number of cases completed by each interviewer is provided. The data are sorted in interviewer id order for each of the specified blocks/fields

  Basic SQL Statement: *Select IwerName, fieldblockname, count(CaseID), avg(sumfieldtime) from (Select cust_field.fieldblockname,Iwername, caseID, sum(fieldtime/1000) as sumFieldtime from (select * from field_standard) as T1, cust_field where mid(T1.Fieldname, 1, len (cust_field.fieldblockname)) = cust_field.fieldblockname group by fieldblockname, caseID, IwerName) group by fieldblockname, IwerName*

f. Custom Reports – User Defined Sections

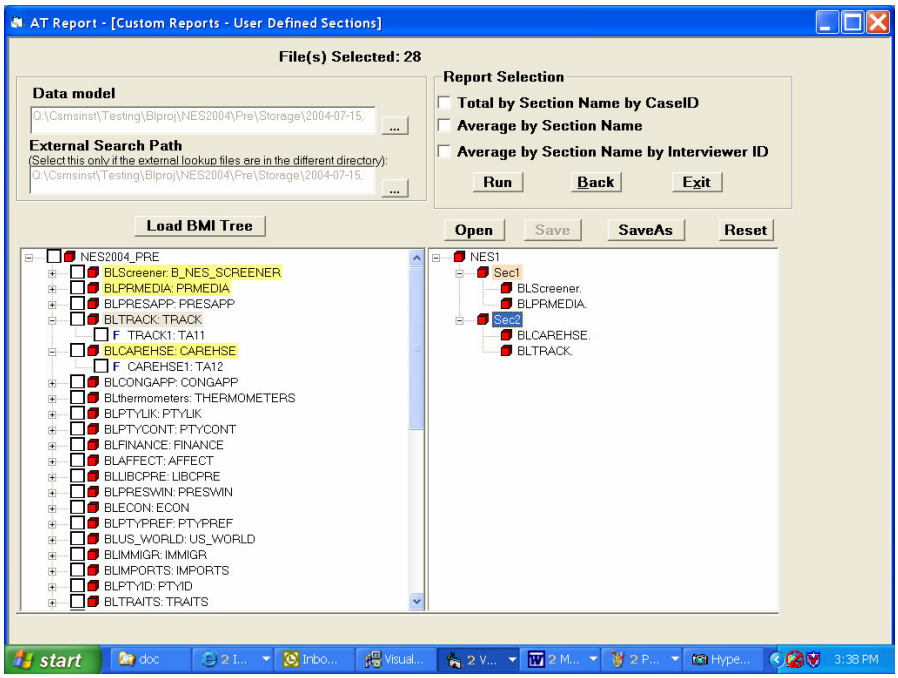

Figure 5 – Custom Report- Section Selection

The Custom Reports – User Defined Sections portion of the application looks and functions in a somewhat similar manner to the Custom Reports – Selected Fields/Blocks component described in the previous pages. In addition to allowing a user to specify which fields and/or blocks to include in a timing report, this portion of the application allows a user to group fields and/or blocks into sections defined by the user. For example, if a user is interested in generating timing data for a portion of a block rather than the entire block, this portion of the application will allow that. Likewise, a user can define a section to include items from more than one block.

See Figure 5. After setting up the sections, it is ready to run reports. First it will insert Section name and Blocks/Fields name into Cust_Section table.

The insert SQL Statement: *Iinsert into cust_Section values ('" & tempnode1.Text & "', '" & tempnode2.Text & "')*

- Total by SectionName by CaseID—This report provides the total amount of time (in seconds) spent and the number of visits made by an interviewer in each user-defined section. The data are sorted in CaseID order for each of the specified blocks/fields
  Basic SQL Statement: *Select Cust_section.SectionName as sectionname, temp.caseID as caseID, sum(temp.Fieldtime/1000) as sumoffieldtime from (Select * from field_standard) as temp, cust_section where mid (temp.fieldname, 1, len(cust_section.fieldblockname)) = cust_section.fieldblockname group by sectionname, caseID*

- Average by Section Name—The report lists the average time (in seconds) needed to complete each of the user-defined sections. The report also lists the number of cases that had timing data for the section.

  Basic SQL Statement: *Select SectionName, avg (sumofFieldtime), count (caseID) from (select Cust_section.SectionName as sectionname, temp.caseID as caseID, sum(temp.Fieldtime/1000) as sumoffieldtime from (Select * from field_standard) as temp, cust_section where mid (temp.fieldname, 1, len(cust_section.fieldblockname)) = cust_section.fieldblockname group by sectionname, caseID) group by sectionname*

- Average by Section Name by Interviewer ID-- For each section specified the report lists the average time (in seconds) needed by each interviewer to complete the particular section. A count of the number of cases completed by each interviewer is provided. The data are sorted in interviewer id order for each of the specified sections.

  Basic SQL Statement: *Select IwerName, Sectionname, avg(sumoffieldtime), count(CaseID) from (select Cust_section.SectionName as sectionname, temp.caseID as caseID, IwerName, sum(temp.Fieldtime/1000) as sumoffieldtime from (Select * from field_standard) as temp, cust_section where mid (temp.fieldname, 1, len(cust_section.fieldblockname)) = cust_section.fieldblockname group by sectionname, caseID, IwerName) group by sectionname,Iwername*

g. Other Report Features

There are several other useful report features:

- A user can then choose to print the output or export the output to a Text (*.txt), Excel (*.xls), or PDF (*.pdf) file.

- A user can save his customer report settings to xml files that can be opened later by the same user other users

- A user can choose different time format that are presented on the reports.

- A log file is produced if there are problems with the Audit Trail files.

7. Audit Trail data Generation

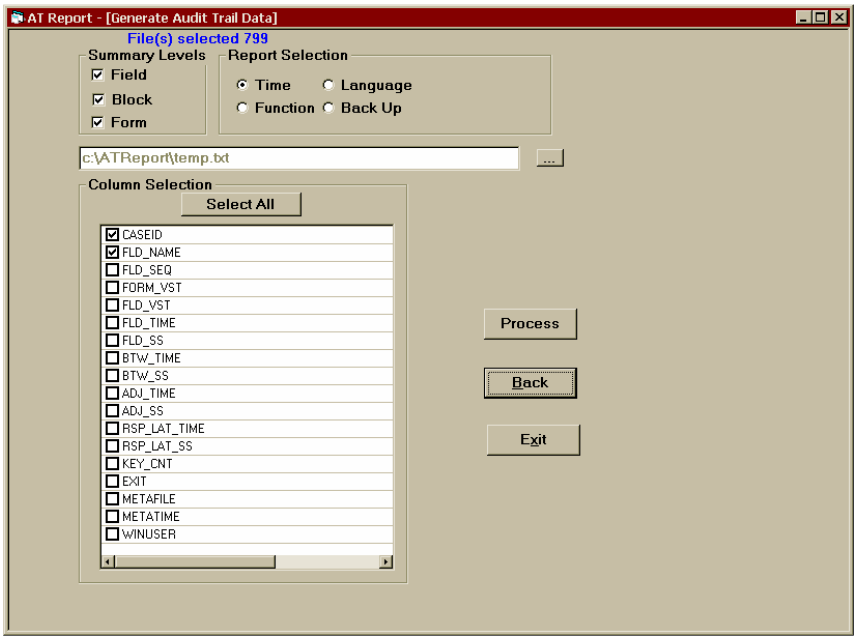

Figure 6 – Audit Trail Data Selecton

This part of program can generate four Audit Trail Data files:

a. Time File

This file summarizes times and key counts.

Columns:

CASEID! FLD_NAME! FLD_SEQ! FORM_VST! FLD_VST! FLD_TIME! FLD_SS! BTW_TIME! BTW_SS! ADJ_TIME! ADJ_SS! RSP_LAT_TIME! RSP_LAT_SS! KEY_CNT! EXIT! METAFILE! METATIME! WINUSER

b. Function File

This file summarizes important functions captured in the audit trail and key stroke files.

This file has a unique record for each CASEID and FLD_NAME combination. If there is more than one record for a field for a CASEID, then the data is collapsed/summarized by FLD_NAME.

Columns:

CASEID! FLD_NAME! VISITS! SUSP! ABSUSP! TOTSUSP! C_X_SUSP! EXIT_ACTION! REMCLK! REMCHNG! QHELP! BLAISEHELP! ERROR_ESC! ERROR_SUPP! ERROR_JUMP! MEDIA_START! MOUSE_CLICK! F1! F2! F3! F4! F5! F6! F7! F8! F9! F10! F11! F12

c. Language File

This file summarizes important language related actions in the audit trail and keystroke files. The development of this file is not complete. The fields we have to date as well as the incomplete fields are described below.

The file has a record for each visit to a field.

Columns:

CASEID! FLD_NAME! FLD_VST! NEXTLANG! PREVLANG! SETLANG! C_L_SETLANG! ENTER_LANG! EXIT_LANG

d. Back Up File

This file summarizes important information about movement up and down in the instrument.

The file has a record for each visit to a field.

Columns:

CASEID! FLD_NAME! FLD_VST! ENTER_VALUE! LEAVE_VALUE! LEAVE_CAUSE

e. Reading the output files

The files produced in this part of the program can be read by most spreadsheet programs (e.g. Excel), database programs (e.g. Access), SAS and SPSS. We are not going discuss this further in this paper.

8. Conclusion

This is the first version of the program. Many types of users at the University of Michigan are using this tool frequently for various purposes. After collecting feedback from these users, the UM programming group will make improvements to the program. Minor changes may be needed for the program to work with the output generated by the Blaise 4.7 Audit Trail DLL.

9. References

Statistics of Northlands Blaise Guide
Hansen, S. and Marvin T., (2001) "Reporting on item times and key strokes from Blaise Audit Trails", 7th International Blaise Users Conference