CTT: CAI Testing Tool

Mary Dascola, Genise Pattullo and Jeff Smith, University of Michigan

1. Introduction

In 2006 Survey Research Operations (SRO), a unit within the University of Michigan's Institute for Social Research, initiated the development of a tool for testing Computer Assisted Interviewing (CAI) instruments. Without adequate tools for systematic testing of the instruments, a research study may experience production delays, interviewer frustration with instruments that don't function properly, or collection of imperfect or incomplete data that fails to meet the needs of the research project. Prior to the development of the tool, testing comments were recorded in an Excel spreadsheet or a Word document. After months of testing and iterative testing rounds these documents were cumbersome and time consuming to maintain.

To increase the overall quality of Computer Assisted Interviewing (CAI) through standardized testing procedures, reduce the cost of testing, increase access to information concerning CAI development and software de-bugging through preset and ad hoc performance metrics reports, the Computer Assisted Interviewing Testing Tool (CTT) was developed. The CTT is a comprehensive tool set that is used for testing and reviewing survey instruments and reporting on the testing process. The CTT application was developed to facilitate the testing of Blaise survey instruments by a variety of local and remote users. The CTT application automatically captures and consolidates testing comments, provides bug reports, assigns items to application developers and records testing outcomes.

This paper will discuss the major components of the CTT application which will include the Preload builder, Instrument testing, bug recording, random case generator and reporting on testing notes. The CTT Administration system will be reviewed as well.

2. Objective

In the past, research staff developing CAI applications did not have adequate tools for systematic testing of the instruments. As a result, projects would often have to send out revised data models during the production phase which added to the cost of administering each interview.

After careful review of current products on the market, we made an internal decision to develop our own CAI testing application. We wanted an application that would minimize the costs of modifying data models and sending updates to the field staff, and the costs of missing data callbacks to respondents. Secondly, standardization of testing across projects. Our testing components are generic and can be used by all projects. Each project prepares study specific preload and scripts that run through CTT to ensure testing of critical path scenarios. In addition, standard quality assurance checks are performed on the resulting data and ad hoc testing performance metric reports are available and used to guide future testing efforts.

CTT has increased the overall quality of CAI instruments through standardized testing procedures and reduced the cost of testing through automated regression testing and

random case generation. We were also able to facilitate development of risk-based testing plans and increase access to information concerning CAI development and software de-bugging through preset and ad hoc performance metrics reports.

CTT will facilitate testing at various stages of development (individual modules, sets of modules, the complete instrument), with or without a sample management system. Our system is adaptable for testing on multiple platforms, i.e., on the network, desktop, or interviewer laptop. Replaying previous test cases with new data models and creating preload strings for different testing scenarios will reveal past errors to project staff quickly. CTT automatically captures and consolidates testing comments and bug reports, reviewing and assigning items to programmers, and recording outcomes. Development of CTT was in C#.NET with Visual Basic .Net components with a SQL Server 2005 Database.

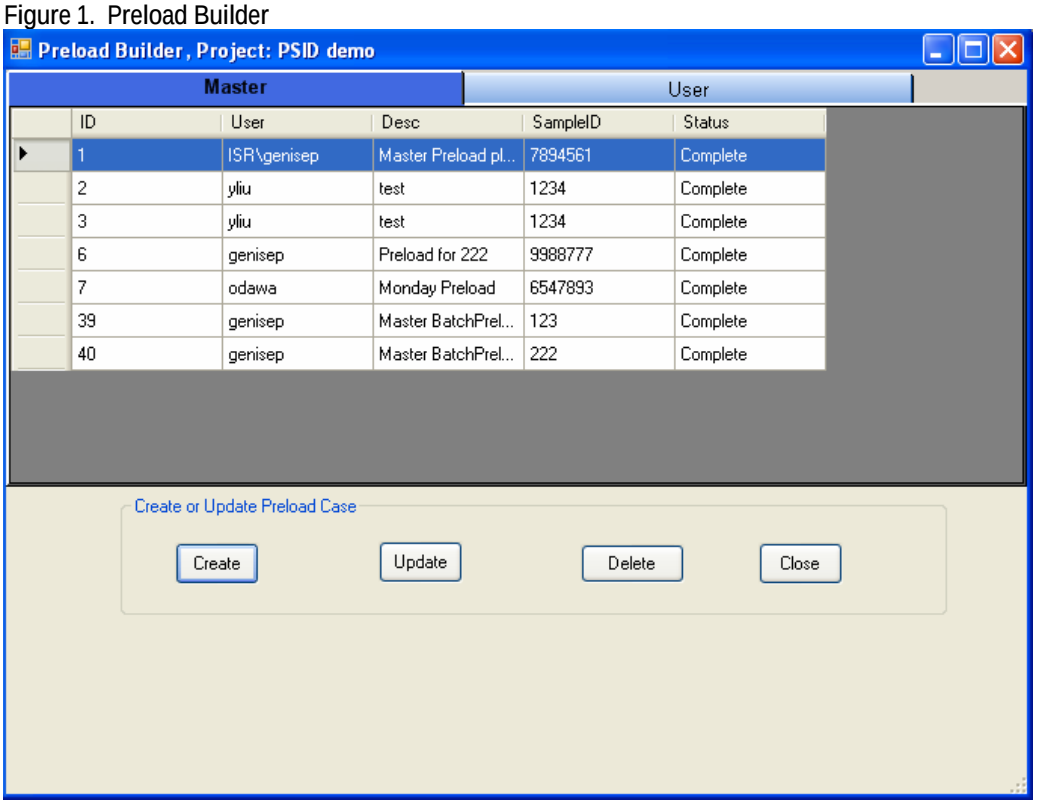
3. Components

The CAI Testing Tool includes six components for implementing and managing the testing process at various stages of instrument development. The Preload Builder and Instrument Testing modules are the main tools for actual testing of the instrument. Managing Problems and Reports facilitate maintenance of testing bugs found, fixed, or needing additional attention from the Testing Coordinator. The Random Case Generator tests skip patterns using data randomly generated by the system. The final module, Admin, manages administration of the entire testing tool, including adding projects for testing, assignment of testers to projects, and access rights.

3.1. Preload Builder

To begin the testing process, preload lines are needed that contain data appropriate to the instrument being tested; that is, data that closely resemble the actual sample that will be used for production interviewing. The Preload Builder allows a tester to enter data and create preload lines without having to wait for another staff member to add different scenarios. A bit of “intelligence” about the production sample and questionnaire is necessary when entering preload data. Following is the first screen you come to when you choose to create preload.

Figure 1. Preload Builder

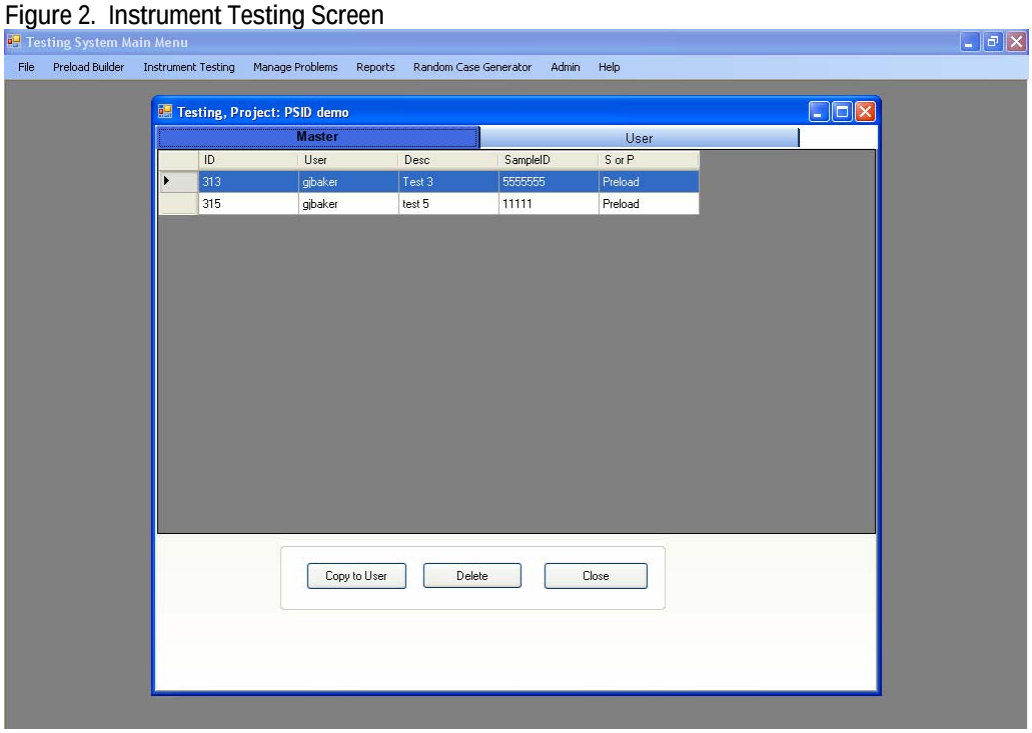


Once a preload line is completed, it can be updated any number of times. It can also be updated and saved as new preload, thus creating additional lines representing different sample situations. Preload lines can be saved to the User tab so that only the tester has access to the lines; or lines can be saved to the Master tab, making the lines available to all testers assigned to the project.

3.2. Instrument Testing

Preload lines created in the Preload Builder are presented for selection in the Instrument Testing module. A preload line is not altered in any way when it is selected for testing the instrument. It is possible, though, to create a "scenario" that captures all data a tester entered while testing the instrument. Using a preload line to create a scenario is akin to accessing a sample line and conducting an interview. It is not a requirement that a scenario is saved from every testing session. A tester can access a preload line, test the instrument, enter testing notes, and exit the instrument without saving. Following is the first screen you are presented when you select Instrument Testing.

Figure 2. Instrument Testing Screen



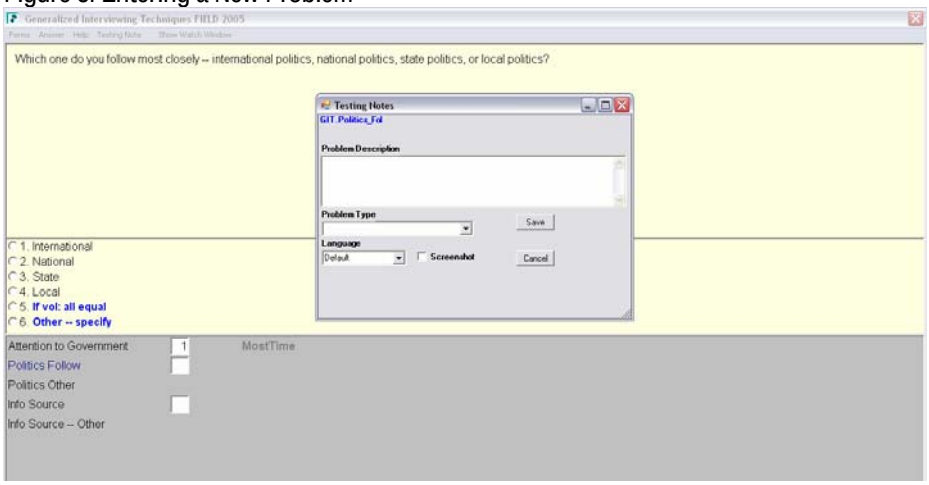
The Master tab allows you to view scenarios and preloads that have been determined to be useful, thus eliminating the need for each tester to make their own preload. If you wish to test a scenario or preload on the Master tab, you simply select the button "Copy to User" and that scenario/preload will be copied to the User tab.

Scenarios can be updated as many times as needed. A scenario can also be updated and saved as a new scenario, creating additional lines representing different interview situations. Scenarios are significantly useful when re-testing problems that were recorded in a previous testing session and fixed by the programmer. Scenarios are also important for testing revised data models. In both situations, using a scenario prevents a tester from having to re-answer questions in the instrument. A tester only needs to use the Jump to Field (shift F9) feature to get to the desired field for re-testing.

During testing, all problems are saved to a master database that records important information about each problem, such as the problem ID, the tester who created the Testing Note, the data model used, the section and field indicator, question text, and the description of the problem. Prior to CTT, if the tester made an error recording the question the problem occurred on, it would become difficult for the programmer to find the question the tester was talking about. This tool eliminates the confusion and wasted time.

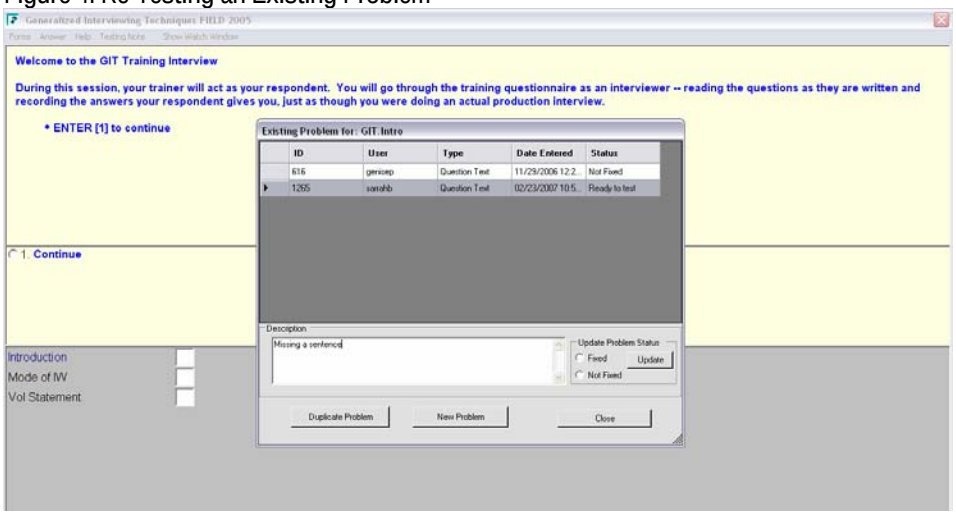
Pressing a single key, F8, will bring up the Testing Notes screen and allow the tester to easily record their comments and resume testing.

Figure 3. Entering a New Problem



If a bug or change has been requested already for a question, a list of the reported issues will be displayed for the tester. This feature has proved to be a big time saver since the tester can review previously entered comments, thus reducing the number of duplicate entries.

Figure 4. Re-Testing an Existing Problem



3.3. Reports

Reports are available to assist in the management of the testing process. When the testing round has been completed, the programmer will generate the "Not Fixed by Programmer" Detail Report from the Reports menu.

Figure 5. Available Reports

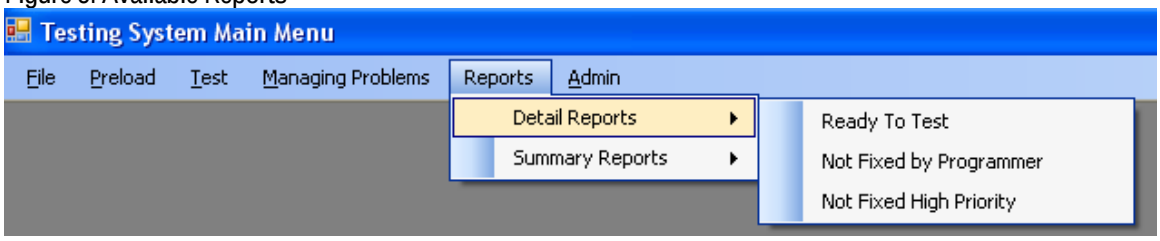
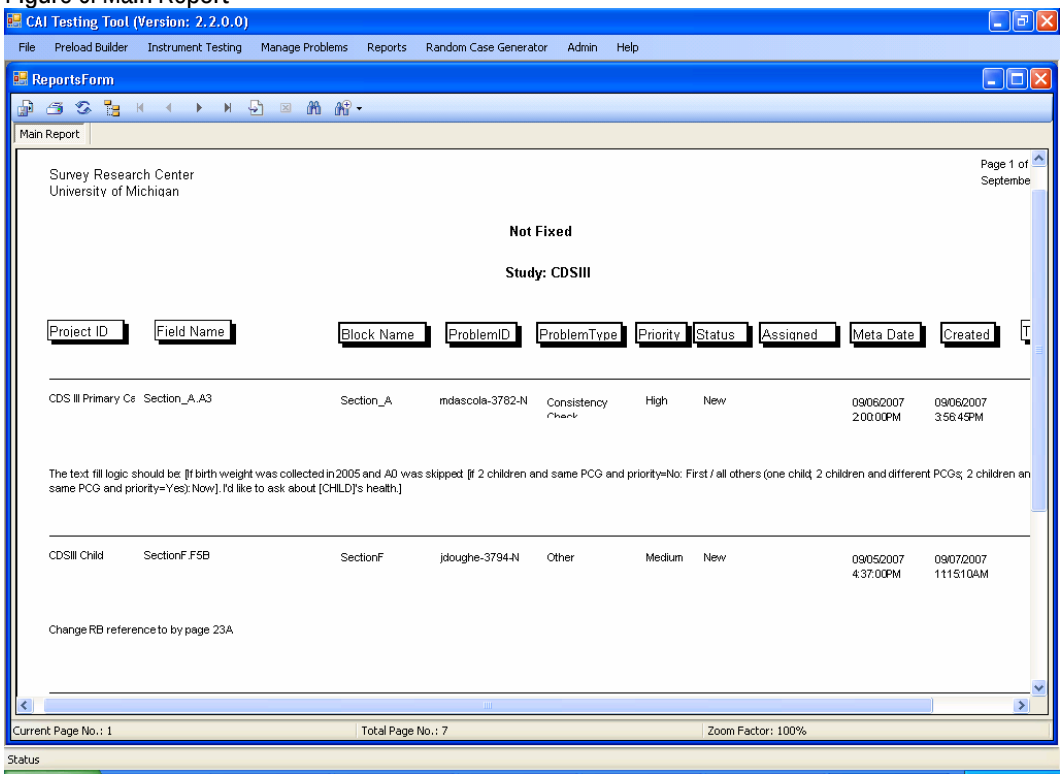


Figure 6. Main Report

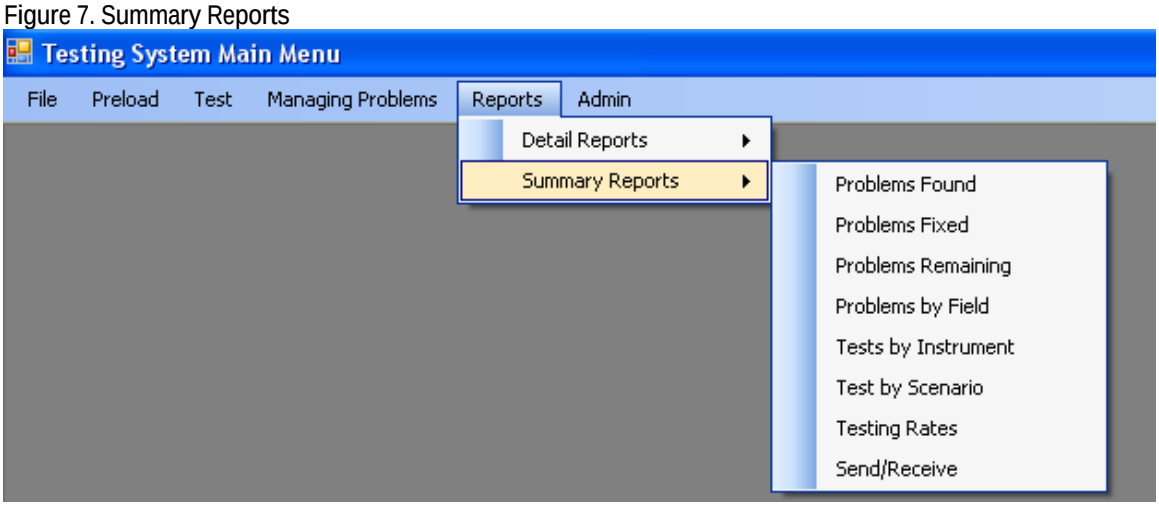


The report option allows the programmer to easily view the problems that have been reported. Previously, the programmer would have to pick out the problems from a large document that could get up to 30 or more pages in length.

The tester was not forgotten either. They have a report in the Details Report menu called "Ready to Test." As the name indicates, this report lists all the items that the programmer has fixed and the tester needs to review.

Several summary reports are available that provide information on test performance metrics such as number of problems found, fixed, and remaining for each data model tested.

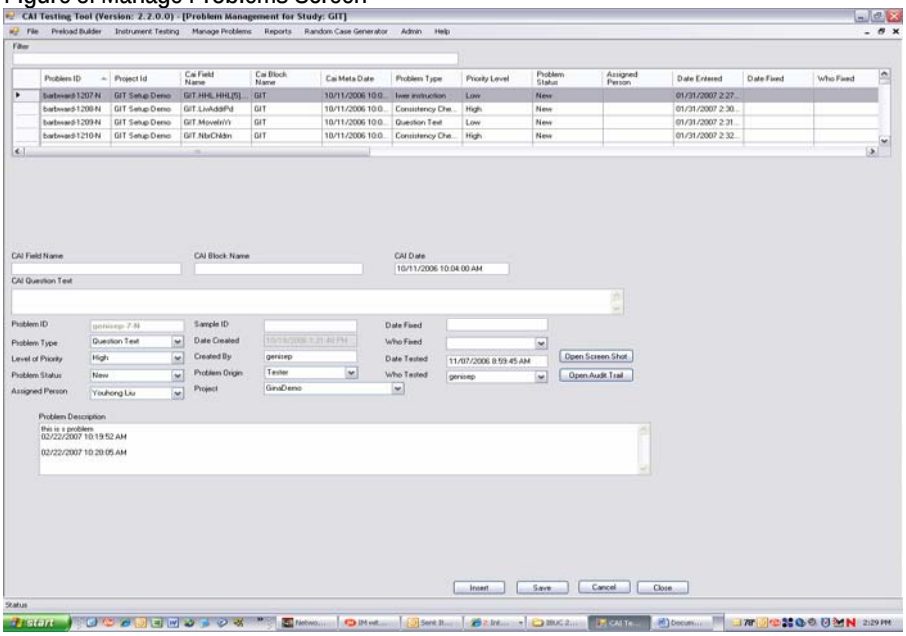
Figure 7. Summary Reports



3.4. Problem Management

The Manage Problems feature is used by project managers and testing coordinators to review and manage the bugs/enhancements that are reported during testing. For example, the testing coordinator can change the problem status or level of priority of a problem on the screen. The programmer uses the Manage Problems module to locate New and Not Fixed problems. When a problem is resolved and ready for re-testing, the programmer sets problem status to Ready to Test, and the Testing Note is made available to testers once again for re-testing.

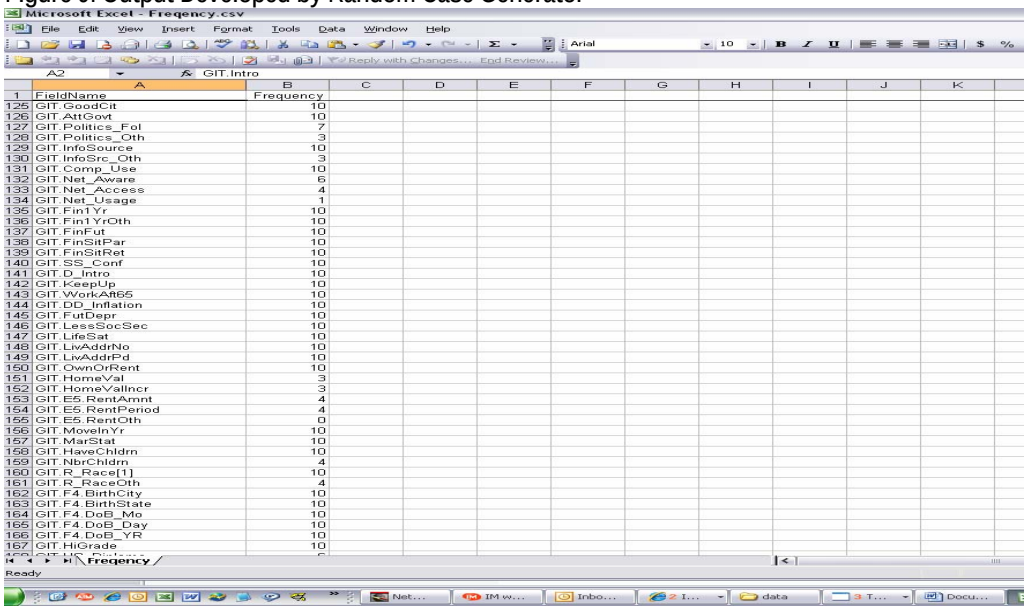
Figure 8. Manage Problems Screen



3.5. Random Case Generator

Automatically generating cases with random data in the Random Case Generator has the advantage of testing the entire instrument and every path within the instrument over many cases. It allows for focused generation of data and testing targeted skip patterns. The output generated, a file containing frequencies for all fields, is useful in determining whether or not all questions that should be on a path and answered actually have a response.

Figure 9. Output Developed by Random Case Generator

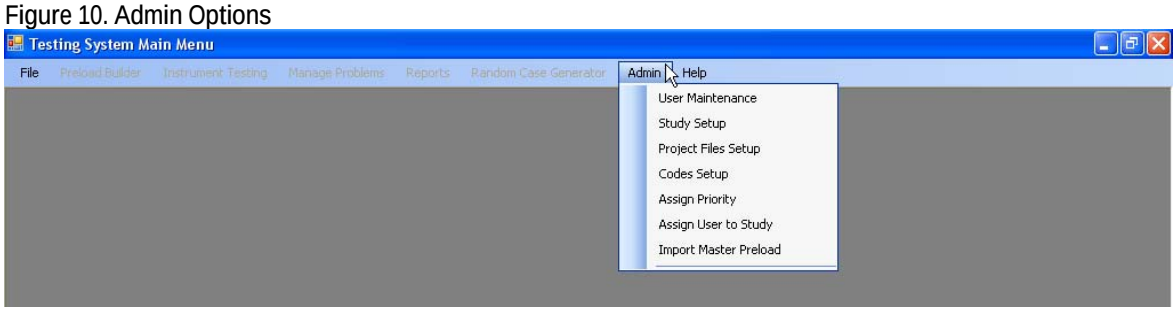


	FieldName	Frequency
125	GIT.GoodCit	10
126	GIT.AttGovt	10
127	GIT.Politics_Fol	7
128	GIT.Politics_Oth	3
129	GIT.InfoSource	10
130	GIT.InfoSrc_Oth	3
131	GIT.Comp_Use	10
132	GIT.Net_Aware	6
133	GIT.Net_Access	4
134	GIT.Net_Usage	1
135	GIT.Fin1Yr	10
136	GIT.Fin1YrOth	10
137	GIT.FinFut	10
138	GIT.FinSitPar	10
139	GIT.FinSitRet	10
140	GIT.SS_Conf	10
141	GIT.D_Intro	10
142	GIT.KeepUp	10
143	GIT.WorkAft65	10
144	GIT.DD_Inflation	10
145	GIT.FutDepr	10
146	GIT.LessSocSec	10
147	GIT.LifeSat	10
148	GIT.LivAddrNo	10
149	GIT.LivAddrPd	10
150	GIT.OwnOrRent	10
151	GIT.HomeVal	3
152	GIT.HomeValIncr	3
153	GIT.E5.RentAmnt	4
154	GIT.E5.RentPeriod	4
155	GIT.E5.RentOth	0
156	GIT.MoveInYr	10
157	GIT.MarStat	10
158	GIT.HaveChldrn	10
159	GIT.NbrChldrn	4
160	GIT.R_Race[1]	10
161	GIT.R_RaceOth	4
162	GIT.F4.BirthCity	10
163	GIT.F4.BirthState	10
164	GIT.F4.DoB_Mo	10
165	GIT.F4.DoB_Day	10
166	GIT.F4.DoB_YR	10
167	GIT.HiGrade	10

3.6. Administrative Features

CTT was designed to be a code driven system and contains an Admin module that allows CTT to be customized to meet different user needs without requiring changes from a programmer. You indicate in CTT the rights assigned to different users. This feature is only available for users with Admin rights. Some tasks completed in the module are adding users, setting up new studies, adding or modifying codes, assigning users to a study and importing preload files.

Figure 10. Admin Options



4. Summary

In summary, the CAI Testing Tool (CTT) is a very useful tool to manage the problem-reporting process for Blaise instrument testing. The system provides the testers an easy interface to record problems and produce reports to use during testing cycles. CTT provides the programmers with accurate details regarding where the problem is located and provides reports to help make the changes to the Blaise application.

During the last year of using CTT, we have been capturing end user's suggestions for improvements. These enhancements include:

- Adding the ability to copy all the Problem Type/Priority codes setup from one study to another one in the Admin menu item- Assign Priority. The studies are using the standard items and may change one or two. This enhancement would speed up the project setup and improve the usability of the tool.

- Enhancing CTT to be able to record problems/enhancements while testing in the Sample Management System SurveyTrak.

- Enhancing the Problem Management Window with several suggestions such as adding the ability to delete unwanted records; adding a fast way to change a group of problems to ready to test; adding the ability for the end user to create filters to control what displays for them in the Problem Mangemet window.

- Adding two new Detail reports: All Problems and Closed Problems. These reports would be useful to the testing coordinator to review all the problems regardless of status. Both reports are sorted by project, then Blaise field name.

- Adding the Synchronize Database feature to the CTT menu. Currently it is a shortcut on the desktop.

The addition of the above features or changes will make this system more robust and user friendly. We have found that the CTT has achieved what it set out to accomplish and that the system is greatly appreciated by the users.