# QUESTIONNAIRE DESIGNERS VERSUS INSTRUMENT AUTHORS: BOTTLENECKS IN THE DEVELOPMENT OF COMPUTER-ADMINISTERED QUESTIONNAIRES

**Irvin Katz, George Mason University, Linda Stinson, Frederick Conrad, US Bureau of Labor Statistics**
**Irvin R. Katz, Department of Psychology, George Mason University, Fairfax, VA 22030-4444**

**Keywords: CAI, questionnaire design, software**

Development of computer-assisted interviewing (CAI) instruments is often resource-intensive and time-consuming (Nicholls & Appel, 1994), especially compared with developing paper questionnaires. In addition to many of the same activities needed to create a paper instrument, the computer instrument must be programmed. Thus, developing a CAI instrument involves both the staff typically associated with questionnaire design (e.g., content specialists, statisticians) as well as programming staff to create the computerized instrument (Mockovak, 1996). Typically, however, people who are expert in questionnaire design are not also expert in the programming skills needed to implement a questionnaire on computer. As a result, the actual designers of questionnaires must work through intermediaries (programmers), with concomitant delays and the potential for miscommunication.

Although CAI technologies have expanded, many procedures for designing CAI instruments are little more than the same procedures used for paper questionnaires. New technologies, such as graphical user interfaces and pen-based systems, provide the programmer with more flexibility in constructing sophisticated survey instruments. However, there has not been a similar increase in tools that aid questionnaire design tasks (see Table 2)—writing questions, specifying skip patterns, defining complex edit rules (constraints), and so forth.

This report examines the processes of CAI design and development with the goal of providing software support to facilitate activities. We first conducted interviews with staff involved in CAI design and development to investigate the general nature of the interactions between questionnaire designers and programmers. The interviews resulted in a preliminary task analysis of questionnaire design, which identifies some critical activities involved in the specification and initial implementation of CAI instruments. Based on the results of the interviews, we conducted a survey on questionnaire design to determine the specific tasks that cause trouble for designers and programmers. The survey results revealed a few key tasks that might benefit from software support, especially for questionnaire designers. The final section of this report briefly describes a software tool aimed at aiding nonprogramming questionnaire designers as they specify CAI instruments.

## INTERVIEWS

The purpose of the interviews was to collect preliminary data on the CAI design and development process. We focused our attention on structure of the CAI instrument—the questions, their ordering, the data to be collected, and so forth. Thus, interviews concerned the questionnaire design process as opposed to tasks related to programming. For example, what activities occurred and when did they occur? What other staff were involved in the projects and at what points? What methods had staff learned over the years to make the design process flow more smoothly?

Twenty staff members from the Census Bureau (Census) and Bureau of Labor Statistics (BLS) were interviewed—11 questionnaire designers (people who write specifications for CAI instruments) or managers of questionnaire designers and nine programmers or managers of programmers. Interviews typically lasted approximately one hour and were unstructured: although the questions listed above guided the overall content of the interview, the interviewer prompted for information about those areas of questionnaire design that the individual appeared to be particularly interested in discussing.

Some of these individuals provided examples of questionnaire specifications they helped design or were asked to implement on computer. Approximately 10 example specifications were collected (here, "specifications" refers to the documents used to communicate a questionnaire to someone else, such as a programmer); these specifications were for questionnaires covering various topics such as food security and health issues. The example specifications ranged from 10 to 80 pages of text.

### Results

At BLS and Census there is a clear separation between programming staff and questionnaire design staff. Although each CAI development project is unique in its details, most followed the high-level process depicted in Figure 1. Questionnaire designers develop specifications for the CAI instrument; these specifications are passed along to programmers, who implement the instrument using a questionnaire authoring system, such as CASES.

The interviews suggested that a bottleneck in the CAI development process may center on the formal "communication channel" between questionnaire designers and programmers—the instrument specifications. For example, questionnaire designers reported that programmers sometimes misinterpret the specifications and so produce instruments that do not work correctly. Programmers reported that specifications are often ambiguous or incomplete, requiring that they either (a) clarify the specifications by speaking with the questionnaire designers or (b) decide on their own. These miscommunications, in turn, reportedly lead to time-consuming iterations between the designers' review of the proposed instrument and the programmers' generation of another version of the instrument.

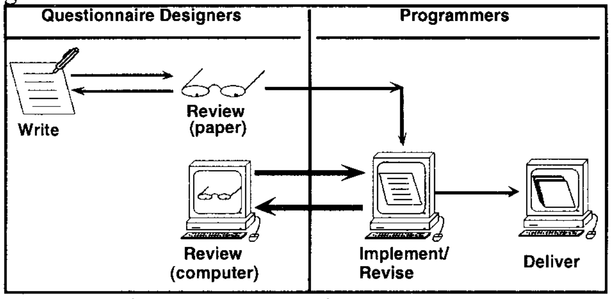


Figure 1: CAI development process

Thus, there are at least two sources of errors that lead to the bottleneck between designers and implementers. First, the instrument produced by the programmers may not meet the specifications—a *communication error*, in which the programmer misinterprets the specifications. Second, the instrument may meet the specifications, but upon review, designers decide that the instrument does not meet their expectations—a *visualization error*. In other words, the designers mis-specified the instrument, perhaps because of difficulties in imagining from the text specifications how the actual instrument would behave (e.g., imagining the implications for a skip pattern of deleting one or more questions).

What is the source of the iteration that occurs between designers and implementers? Which aspects of questionnaire design are particularly difficult and what parts of a questionnaire are particularly difficult to specify (if any)? To expand on the informal information collected through the interviews, a survey on questionnaire design was conducted. The purpose of this survey was to provide guidance on the specific aspects of questionnaire design that could potentially benefit from software support.

## SURVEY ON QUESTIONNAIRE DESIGN

A new software tool will have the greatest influence on questionnaire design if using it specifically addresses problems in the current design process. To identify those places in the design process where software support would add the most value, a questionnaire was administered to questionnaire designers and programmers. The purpose of the survey was to identify the "trouble spots" in designing a questionnaire. That is, what tasks do questionnaire designers find most difficult? What do they spend their time doing? Thus, our goal was to gather information to inform a software tool that would (a) facilitate tasks that designers find difficult or time-consuming when using current tools (e.g., a word processor, paper and pencil) and (b) promote comprehension of those aspects of specifications that programmers find hard to implement or understand.

The interviews with questionnaire designers and programmers uncovered several critical elements of questionnaire specifications. These elements are defined in Table 1. Based on these elements, a list of questionnaire design tasks (see Table 2) was constructed. This list of tasks represents one of the few explications of questionnaire design activities, and is therefore a contribution on its own.

It is recognized that these elements do not occur in all specifications (e.g., many questionnaires do not employ rosters) nor do all designers conduct all associated tasks. Furthermore, there are specification elements and design tasks not represented in these lists. However, during the interviews, these elements and tasks were mentioned most often as relevant specifically to the specification of CAI instruments.

### Method

*Population*

The instrument was geared toward staff who write specifications for, or program, computer instruments. We compiled a list of 74 individuals from BLS and Census—21 CAI programmers and 53 questionnaire designers. Staff were visited at work by the first author, who described the purpose of the survey and handed the individual the questionnaire, if he or she was willing to participate (all staff evidenced an interest in the project). After two weeks, a follow-up letter was sent to those who had not yet responded.

*Respondents*

Response rates were fairly high for both groups (programmers: 86%; questionnaire designers: 77%). Of the questionnaire designers, 27 indicated that they had most recently worked on a CAI instrument as opposed to a paper questionnaire. Because the focus of this report is on CAI instruments, the analyses presented below will be from the data of the 27 CAI questionnaire designers and 18 CAI programmers.

*Instrument*[1]

Two forms of the instrument were constructed, one for questionnaire designers and the other for programmers. There were only slight wording differences between the two forms (e.g., the questionnaire designer form referred to communicating with a programmer while the programmer form referred to communicating with a questionnaire designer).

The focus of the instrument was on the design of questionnaire structure rather than tasks associated with design of questionnaire content, such as determining an analysis plan or conducting pretests. Questions focused on questionnaire specifications—the documents that represent the current version of a questionnaire and are

[1]The instrument is available from the first author.

used to communicate the questionnaire among project staff (e.g., sponsors, questionnaire designers, programmers). Respondents were asked about the differentelements that make up a set of specifications (see Table 1), the ways those specifications are created, and the changes that occur to the specifications in the process of designing a questionnaire.

To facilitate completion of the survey, respondents were asked to answer each question with respect to their most recent questionnaire project. During the interviews, we found that people had trouble answering general questions about their design experiences, but found it easier to answer detailed questions about a recently completed questionnaire. Thus, the first 10 questions "calibrated" respondents by asking about their most recently completed questionnaire.[2]

The next three questions asked respondents to rate several questionnaire design tasks (see Table 2) along different dimensions. The purpose of these questions was to uncover the "trouble spots" in the questionnaire design process—the tasks for which staff spend the most time and effort. In this report, we will focus on two of the questions: "Compared with the other tasks, about how much time, if any, did you spend working on each tasks?" and "Compared with the other tasks, how easy or difficult was each task?".

In developing a CAI instrument, the questionnaire designer must not only create the questionnaire, but also write out the specifications for the programmer on how that questionnaire should be implemented. These specification documents are often the primary channel along which questionnaire designers and programmers communicate. However, sometimes parts of a specification are ambiguous or incomplete, and so the programmer must ask the questionnaire designer for clarification. As was discussed earlier, this clarification process may be one of the primary bottlenecks in the overall CAI development process. Our instrument included a question that asked designers to rate how frequently programmers ask for clarification of particular elements of the specifications. Programmers were asked how frequently they asked the questionnaire designers for clarification of particular elements of the specifications.

Another set of questions focused on alternative representations of questionnaires. Several interviewed staff said that they created a visual representation of their questionnaire to ease questionnaire design. A "visual formalism" was defined as any type of symbolic representation of a questionnaire, including outlines, tables, flowcharts, and other diagrams. Most often, the questionnaire designer or programmer created a flowchart. A flowchart provides an overview of a questionnaire—it succinctly shows the connections among various questions that can be obscured when viewing a linear list of questions. However, staff indicated that these visual formalisms were largely for their own use and rarely became part of the documentation for a CAI instrument. We were interested in whether use of these visual formalisms is common among designers and programmers, for what purposes they use these formalisms, and whether they receive formalisms from others.

A final set of questions gauged each respondent's level of experience. Respondents were asked for their years of experience in questionnaire design (or CAI programming), number of questionnaires on which they worked, and their government salary level.

## Results

*Respondent background*

Respondents had worked a mean of 4.8 years in their current role (designers: 5.1 years; programmers: 4.2 years) and had worked on a mean of 6.1 questionnaire products (designers: 6.3 products; programmers: 5.7 products). Respondents had most recently worked on a wide variety of questionnaires: the number of questions ranged from 7 to 4500, with a median of 100 questions. These questionnaires took their respondents anywhere from 2 to 120 minutes to complete (median 15 minutes).

*Isolation of programmers*

Several results suggested that programmers may be less aware of the entire questionnaire project compared with designers. This isolation may be a source of communication difficulties between designers and programmers. The programmers generally appeared to be insulated from the overall project, working instead largely on their own and interacting with just one questionnaire designer (who perhaps served as a representative for an entire questionnaire project). For example, while all questionnaire designers knew the approximate respondent burden for their questionnaires, five of the 18 programmers were unable to answer this question. Seventeen of the designers worked with others on their project, while only six programmers did so, a marginally significant different ($\chi^2(1)=3.79$, $p<.06$). Programmers instead tended to work on their own. When asked to estimate how many questionnaire designers worked on the entire questionnaire project ($M=2.8$), only three (of 27) designers could not answer compared with seven programmers, a significant different ($\chi^2(1)=3.8$, $p<.03$). However, both groups were equally able to estimate the number of programmers working on the entire questionnaire project ($M=1.9$)—only two designers and no programmers indicated that they did not know.

[2]Or their most recently completed portion of a questionnaire. Collectively, either a portion of a questionnaire or a full questionnaire was referredto as a *product*. Respondents were asked to complete the questionnaire with respect to their most recently completed product.

*Critical tasks in the questionnaire design process*

Because our interest is in the bottlenecks in the CAI development process, for each question we identified the three or four tasks rated most "troublesome." Troublesome tasks include those rated as (a) taking a "Great Deal" of time and (b) "Very Difficult" or "Moderately Difficult." Additionally, elements of questionnaire specifications were considered troublesome if rated by many respondents as requiring clarification "Very Often" or "Often." These categories represent the top ends of their respective scales.

The following two points summarize the results:

• For both groups of respondents, two tasks were consistently rated highest in terms of difficulty, time-intensiveness, and a source of iteration between designers and programmers: defining complex branching and defining multi-item constraints (sometimes called "complex edits"). Both of these tasks may involve the creation of complex logical statements, which are implemented (and often specified) as a long series of conditional (or, "if-then") statements. For complex branching, determination of which question should be asked next is based on several previous responses. For multi-item constraints, the decision of whether to accept a particular response is again based on several previous responses. Questionnaire designers, who typically do not have a programming background, may have trouble creating the conditional statements from their more informal understanding of the desired routing or constraints. For programmers, sets of conditional statements are notoriously difficult to design, test, and debug (Sime, Arblaster, & Green, 1977). Given the difficulties in specifying such conditions and the difficulties in correctly implementing them, it is no wonder that complex branching and multi-item constraints are identified by both groups as troublesome to design and often needing clarification.

• For the questionnaire designers, creating the overall flow of the questionnaire (respondent routing or skip patterns) is considered difficult and time-consuming. Also, questionnaire designers reported that respondent routing is an aspect of the specifications that programmers often ask for clarification about. Programmers tended not to identify the creation of respondent routing as troublesome; the task was rated as relatively easy, taking little time. Programmers agreed that respondent routing was a source of iteration, but did not rate it as highly as did designers.

*Visual formalisms*

There were few strong differences between designers and programmers in terms of their use of visual formalisms. Consequently, only the results from the total population of respondents will be reported, except as noted below.

Approximately half of the respondents (22) indicated that they had created some type of visual formalism during their most recent questionnaire project. Most of these people indicated that the formalism contained information concerning the questionnaire's skip pattern. In a "check all that apply" question, 91% indicated their formalism contained information about specific respondent routing, 77% included complex branching (branching based on the responses to more than one prior question), and 64% included information about the overall flow of the product. These were the three most frequent responses.

Of the respondents who created a visual formalism, 61% had created a flowchart. Respondents indicated that they created this flowchart most frequently: "to help me remember the structure of the product" (93%), "to help me design some aspect of the product" (86%), and "to serve as an aid in testing the programmed product against the specifications" (86%).

The data also suggested that more experienced designers were more likely to have created visual formalisms on their most recent project. Questionnaire designers who report having created a visual formalism on their most recent project had more years of experience ($M$=6.8 years) compared with those that reported not having created a visual formalism ($M$=3.3; one-tailed $t(25)=2.8$, $p<.01$). Those who did not create visual formalisms were almost all at the same government salary level, while those that create visual formalisms were relatively evenly distributed among that level and the next two higher levels ($\chi^2(3) = 9.1$, $p<.03$). Finally, those who created visual formalisms tended to have worked on a greater number of projects (7.8) compared with designers who did not create a visual formalism (4.7), although this difference is not significant. For the programmers, there was no corresponding relationship between experience and creation of visual formalisms.

Only 33% (15) of all respondents indicated that they had received a visual formalism from someone else, but, when received, a flowchart was by far the most frequently received formalism (67%). Of the 15 respondents who received a formalism from someone else, 12 indicated that the formalism was either "very useful" or "useful."

*Summary.* Flowcharts appear to be useful in the design and development of CAI instruments. Unfortunately, only the more experienced questionnaire designers use flowcharts, and even then, the flowcharts are typically not used to communicate with others. Rather, staff use flowcharts to help only their own thinking.

## Discussion

Some of the communication difficulties in the CAI development process may be due to differences in priorities and perceptions of designers and programmers. Perhaps because of their differing roles on a project, each group provided different ratings of what they spend time doing and what tasks they perceive to be most difficult. However, the groups agreed that certain tasks—defining complex branching and multi-item constraints—are among the most troublesome for CAI design.

For the questionnaire designers, the results of the survey (and interviews) suggest that specification of respondent routing (skip patterns) is a common yet troublesome task. Designers reported that designing skip patterns (i.e., respondent routing) is difficult and may often be the source of time-consuming iteration between themselves and programmers. Most questionnaires contain some form of a skip pattern, even if they do not contain some of the more complex (and therefore, more difficult to design) elements of a questionnaire, such as rosters and constraints. Thus, software to support the design of skip patterns should be useful in many questionnaire design projects.

To design skip patterns, more experienced questionnaire designers tend to use flowcharts. Flowcharts, as the name suggests, are useful for capturing the overall flow of a questionnaire—they provide the questionnaire designer with an overview that is difficult to infer from text specifications.

The results of the survey suggest that a software tool that aids the design of skip patterns through facilities for creating flowcharts might meet some of the needs of questionnaire designers. In fact, because less experienced designers tend not to create flowcharts, such a system may help less experienced questionnaire designers by providing an easier means for creating and manipulating flowcharts compared with drawing them by hand.

The next section describes a software tool created for questionnaire designers. The goal of the system is to facilitate communication between questionnaire designers and programmers by helping designers more accurately visualize and specify their instruments.

## SYSTEM DESCRIPTION

Questionnaire Designer (QD) is a software tool aimed at facilitating the design of CAI instruments (Katz & Conrad, 1997). QD was designed for use by questionnaire designers who might not have any programming experience. The goal of QD is to provide a questionnaire designer's "sketchpad"—a way for the questionnaire designer to quickly put an approximation of their questionnaire onto computer, evaluate their current work, and to revise the questionnaire. Figure 2 shows the main QD screen along with descriptive text.

QD was designed to help the early phases of questionnaire design, when the general needs of the questionnaire have been worked out, but not all of the questions nor all of the logic has been finalized. The goal of QD is to help designers work out their ideas for skip patterns by providing an overview of the evolving questionnaire in the form of a flowchart. A flowchart allows the questionnaire designer to see more of the questionnaire at one time (e.g., greater number of questions, all the links leading out of and into each question), which may ease consideration and comparison of alternative orderings of questions and questionnaire logic. Because the flowchart is on computer, the questionnaire designer can easily rearrange portions of the questionnaire to try out their alternative ideas about the questionnaire's logic. In other words, the questionnaire designer can ask "what if" questions such as "what if I arranged the question this way; would the questions flow together smoothly? Could I eliminate redundant questions?".

Preliminary evaluations have indicated that nonprogramming questionnaire designers can, after brief training, use QD to design questionnaires of moderate complexity (Katz & Conrad, 1997).

## CONCLUSIONS

The complexity of CAI instruments demands involvement of staff with differing expertise and backgrounds. We have presented potential sources of communication difficulties between designers and programmers, and have identified specific tasks in the CAI design and development process that cause the most difficulties for these groups.

QD, and the research presented in this report, represent first steps toward improved CAI development. QD attempts to supplement the existing CAI development process by allowing questionnaire designers methods for experimenting with alternative designs for their questionnaires and for evaluating those alternatives. Design tools, such as QD, should facilitate the CAI development process by reducing the bottlenecks that plague the current process.

## ACKNOWLEDGMENTS

We thank Sarah Jerstad for assistance in data collection and the staff of BLS and Census who participated in the research. This work was funded by the American Statistical Association, Educational Testing Service, National Science Foundation, US Bureau of Labor Statistics, and US Census Bureau.

## REFERENCES

Katz, I. R., & Conrad, F. G. (1997, May). *Questionnaire designer: A software tool for specification of computer-administered questionnaires.* Paper presented at the 1997 meeting of the American Association for Public Opinion Research, Norfolk, VA.

Mockovak, W. P. (1996). Issues and steps involved in designing a questionnaire for computer-assisted interviewing. To appear in *Proceedings of InterCASIC '96.*

Nicholls, W. L., & Appel, M. V. (1994). New CASIC technologies at the US Bureau of the Census. In *Proceedings of the Section on Survey Research Methods, Volume II* (pp. 757-762). Alexandria, VA: American Statistical Association.

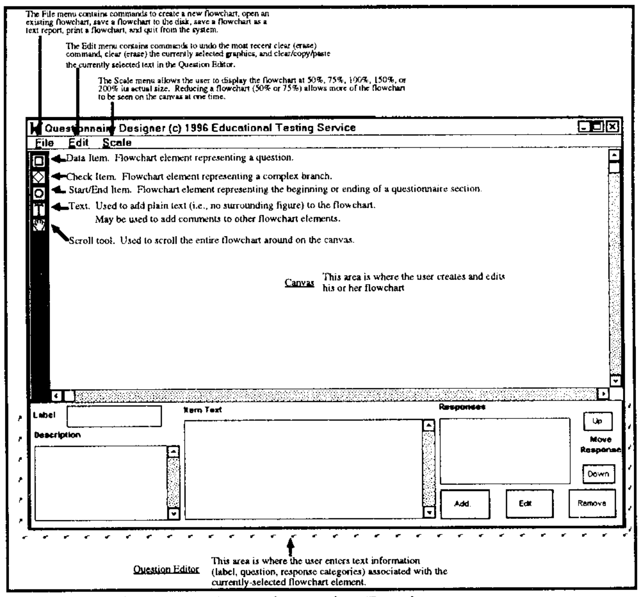
Sime, M. E., Arblaster, A. T., & Green, T. R. G. (1977). Reducing programming errors in nested conditionals by prescribing a writing procedure. *International Journal of Man-Machine Studies, 9*(1), 119-126.

Table 1: Elements of a specification document

| | |
|---|---|
| **Constraint (multi-item)** | Any type of restriction on the answers to a set of questions (in the current questionnaire or a previous questionnaire). A multi-item constraint assures consistency among answers to related questions. Specifications sometimes include how the questionnaire should guide the interviewer to resolve any inconsistencies. **Example**: *An earlier answer indicated that the respondent is a nonsmoker, yet the respondent also indicates that he or she spent money on buying cigarettes.* |
| **Constraint (single-item)** | A specification for the acceptable answers to an individual question. In a computerized questionnaire, the system will not allow the interview to continue if an unacceptable answer is entered. **Example**: *An answer must be within a specific numeric range or contain a certain number of digits. The interview cannot continue until the respondent gives an answer that fits within the acceptable range or contains the correct number of digits.* |
| **Fill** | Any change to the wording of a question based on answers to prior questions. A fill generally consists of (a) the alternative wording to fill into the question and (b) the conditions for choosing alternative wording. Sometimes answers to previous questions are filled into the wording of later questions (e.g., a person's name). **Example**: *If the respondent is answering a question about him or herself, begin with "Do you...". If the respondent is answering about the household, begin with "Does anyone in your household...".* |
| **Rostering Scheme** | The method used to create a roster, to select elements from the roster for data collection, and to ask questions of each selected roster element. For example, a roster might first be collected that contains the names of everyone in a household. A series of questions might then be asked of a subset of people in the roster. |
| **Respondent Routing** | The method by which respondents are presented with different questions depending on answers to previous questions. Includes both complex branching and simple branching. |
| **Complex Branching** | A type of respondent routing in which jumping to alternative questions is based on answers to several questions. To define a complex branch, one must decide which questions (and which answers to those questions) should be used to route the respondent to the next question. **Example:** *A questionnaire needs to ask questions only of male smokers over the age of 50. The characteristics "male," "smoker," and "over 50" may represent answers to three separate questions* |
| **Simple Branching** | A type of respondent routing in which jumping to alternative questions is based on the answer to a single question. |

Table 2: Tasks involved in creating specifications

| | |
|---|---|
| a. | Composing introductions |
| b. | Writing questions |
| c. | Deciding labeling conventions for questions |
| d. | Selecting response categories |
| e. | Deciding order of response categories |
| f | Identifying respondent subsets |
| g. | Deciding phrases for fills |
| h. | Creating conditions for fills |
| i. | Designing respondent routing |
| j. | Defining complex branching |
| k. | Defining multi-item constraints |
| l. | Designing approach to verify and, if necessary, correct responses |
| m. | Defining single-item constraints |
| n. | Designing the layout of a question on the computer screen |
| o. | Designing rostering scheme |
| p. | Creating a flowchart or other diagram |
| q. | Creating scheme to record case status |


The File menu contains commands to create a new flowchart, open an existing flowchart, save a flowchart to the disk, save a flowchart as a text report, print a flowchart, and quit from the system.

The Edit menu contains commands to undo the most recent clear (erase) command, clear (erase) the currently selected graphics, and clear/copy/paste the currently selected text in the Question Editor.

The Scale menu allows the user to display the flowchart at 50%, 75%, 100%, 150%, or 200% its actual size. Reducing a flowchart (50% or 75%) allows more of the flowchart to be seen on the canvas at one time.

Data Item. Flowchart element representing a question.

Check Item. Flowchart element representing a complex branch.

Start/End Item. Flowchart element representing the beginning or ending of a questionnaire section.

Text. Used to add plain text (i.e., no surrounding figure) to the flowchart. May be used to add comments to other flowchart elements.

Scroll tool. Used to scroll the entire flowchart around on the canvas.

Canvas This area is where the user creates and edits his or her flowchart

Question Editor This area is where the user enters text information (label, question, response categories) associated with the currently-selected flowchart element.

Figure 2: Questionnaire Designer screen