A Case against the Skip Statement

Samuel Spencer*- Senior Information Analyst
Australian Bureau of Statistics

February 29, 2012

Abstract

Despite huge growth in computing power, questionnaire design remains a predominantly manual field. With statistical agencies facing shrinking budgets, and a greater desire for evidence-based policy, statistical surveys must become more agile. One possible way to improve productivity and responsiveness is through the automation of questionnaire design, reducing the time to produce data and improve data quality. But despite computer enhancements to many facets of survey research and data collection, questionnaire design has not realised the same levels of improvement as other areas. It must then be asked why, in spite of such benefits, is such automation so difficult?

This paper suggests that the stalling point of automation within questionnaire design is the 'skip statement'. An artifact of paper questionnaires, skip statements complicate the understanding of questionnaires and impede their transition to computer systems. This is due to the structural similarity of skip instructions and the unstructured and deprecated `goto` statement in computer programming languages. By examining this similarity and changing how questionnaire logic is constructed we can lay the foundations for more structured questionnaire creation, thereby realising the benefits of automation.

1 Introduction

It is now common for government organisations to publicly acknowledge the financial constraints and the changing environment of national statistics [13]. These acknowledgements in turn lead to the organisational requirement to address costs and identify efficiencies. With the bulk of official statistics still reliant on questionnaires for data collection, the ability to find efficiencies in their creation and validation stands to have a significant impact on the operating budgets of official statistical organisations. However, such automation has historically proven difficult due to the complex nature of questionnaires.

*samuel.spencer[at]abs.gov.au

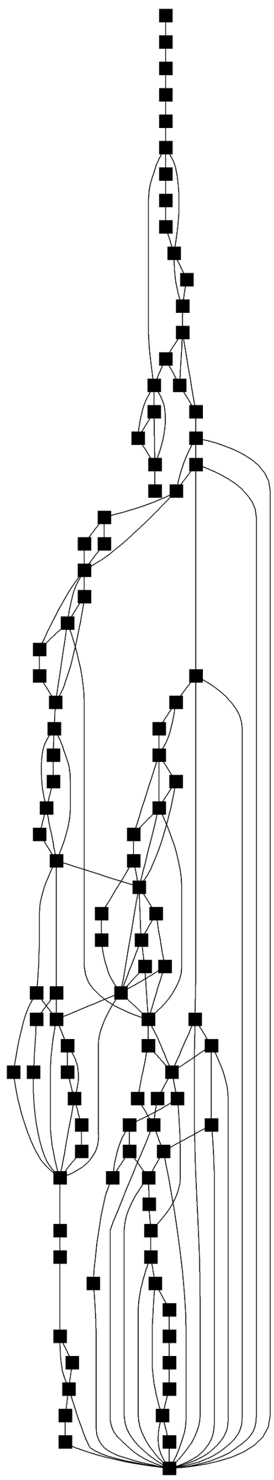
2 The logic of questionnaires

Defining the term questionnaire is a loaded and contentious issue, and within the field of survey research there is no widely agreed or fixed definition for the term “questionnaire” [12]. Oppenheim attempts to bring together a set of common criteria that many practitioners would agree with, suggesting that a questionnaire, regardless of mode, is “not just a list of questions or a form to be filled in”, but also a “tool for data collection”. But this description puts too little emphasis on the logical flow of a questionnaire. For the purposes of this paper, we can propose a generic definition that covers all survey instruments - a questionnaire is a collection of questions with a given order and logical flow, that is presented with a given format to collect specific data. With this definition in mind, this paper will focus on the logical flow of a questionnaire - i.e., the design and analysis of the potential branches and loops over questions that a respondent may take.

While a questionnaire may consist of a number of sequentially ordered questions, a respondent may end up answering only a small percentage based on responses to earlier questions. Often this flow is described through the use of filter or contingency questions [5] and skip instructions. However, when dealing only with these questions and skips individually, the logical structure of a questionnaire can become difficult to visualise, and the complexity becomes obscured.

While there has been ample study into the importance of question wording and the visual presentation of a questionnaire to a respondent, study into the management of this logical structure is surprisingly lacking. But the challenge of managing such complicated structures is not a technological issue, but a conceptual one.

Figure 1: A flowchart showing the routing graph of the 2004 Labour Force Survey.



3 Flowcharts as survey documentation

The most important piece of research into question flow documentation was conducted by Thomas Jabine in 1985 [8]. In this paper, Jabine examines the role of flowcharts as a questionnaire documentation tool. Examining both paper-based and computer-aided questionnaires, he concluded that flowcharts were a “valuable tool for use in developing, reviewing and understanding questionnaires”.

For example, figure 1 presents a visual representation of the potential routing of the 2004 Labour Force Survey conducted by the Australian Bureau of Statistics [1]. The graph representing the logical structure, contains 94 questions (nodes) and the 164 possible paths between them. This image illustrates the complexity of a standard official statistical survey, and highlights how this structure can be quite complex, while still being able to be visualised in a compact setting.

Despite this perceived value, Jabine's analysis of the literature at the time showed little mention of flowcharts in survey research and a more recent analysis suggests little has changed. Similarly, De Vaus recommends the use of flowcharts as an aid to validate existing questionnaire skip logic [5]. But this reasoning stops short of the use of flowcharts to document logical structure, or to assist in question ordering.

Similarly, there is little analysis on the documentation of the logical flow of a questionnaire, with many books glossing over a structured approach and assuming if a questionnaire designer had an appropriate order of questions they would be able to provide adequate logic and branching between question segments through the liberal use of skips and filter questions.

One of the few other papers to look at role of flowcharts was in Katz's examination of the interplay between survey designers and computer programmers when designing computer-aided instruments [9]. Katz found that while flowcharts were deemed to be useful in the development of instruments, when they were used they "were largely for [personal] use and rarely became part of the documentation for a CAI instrument."

What all of these papers do agree on, however, is that the logical structure of a questionnaire can be documented using flowcharts.

4 Questionnaires as algorithms

Flowcharts have a greater use than just documentation, as posited by influential early computer scientists Böhm and Jacopini who wrote that "flow diagrams [are] a two-dimensional programming language", suitable for representing algorithms or programs [3]. As such, if we are able to document the logical structure of a questionnaire as a flowchart, we must also be able to construct a programmatic or algorithmic representation.

This is confirmed when we examine the criteria set out in *The Art of Computer Programming*, where Knuth outlines the five basic criteria for an algorithm [11]:

Finiteness An algorithm must always terminate after a finite number of steps.

Definiteness Each step of an algorithm must be precisely defined; the actions to be carried out must be rigorously and unambiguously specified for each case.

Input An algorithm has zero or more inputs, i.e., quantities which are given to it initially before the algorithm begins. These inputs are taken from specified sets of objects.

Output An algorithm has one or more outputs, i.e., quantities which have a specified relation to the inputs.

Effectiveness All of the operations to be performed in the algorithm must be sufficiently basic that they can in principle be done exactly and in a finite length of time by a man using pencil and paper.

Looking at these criteria, there is a distinct overlap of what is an algorithm and what is a questionnaire. Statistical surveys have a history dating back to the late 1800's [4], and until recently questionnaires had to be designed and completed solely using a pen and paper (criterion 5). Being wholly reliant on manual means and human understanding, questionnaires needed to be quick to complete with clear end points (criterion 1) and unambiguous (criterion 2) to ensure data consistency [5, 14].

The output of a questionnaire is the data it aims to capture, and traditionally is quite well defined. Even if the data that respondents put is in some way invalid, the final data matrix is usually defined well in advance [5](criterion 3). In the algorithmic sence, the input to a questionnaire is data that is entered before a respondent has begun to complete the form. In many cases questionnaires are left blank, although some may pre-fill information, such as names or addresses, to reduce respondent burden. In either case, a questionnaire meets the criteria of having zero or more inputs into the process (criterion 4).

Therefore, it is possible to assert that all questionnaires are specialised data collection algorithms, designed for producing a data matrix. While there are additional restraints on what makes a questionnaire "good", any given questionnaire will still meet the criteria of a "good" algorithm.

5 The building blocks of questionnaires

Given that questionnaires can be expressed as an algorithm, then for each questionnaire there exists a set of instructions that control the logical flow. This is reinforced by Knuth who described the relationship between flowcharts and flow outlines [10] – the latter described as flow outlines as a simple English, sequential, one-dimensional expression of a two-dimensional flowchart. These flow outlines then form an easily understood list of directives used to control flow and describe the execution of an algorithm.

While it is possible to assume that each questionnaire is uniquely built using specialised questions and logic, the challenge is to create a generic, minimum set of unambiguous instructions capable of describing the logic all questionnaires. For this, we must examine the logical flow of questionnaire free of its expression, such as its question text, modal issues and presentation elements. If we do this, we can break a questionnaire down into the four generic and reusable instructions - (shown below as instructions with a short name in `fixed-width font`, each of which take some input to be acted upon, shown as bracketed ellipses `(...)`):

Asking the respondent a question - `ASK (...)`

The most important action on any questionnaire, this action requests information from a respondent and stores this data - in a computer system as a piece of data in a database, or a response field on a paper form.

Giving the respondent information - `TELL (...)`

The basic action of giving a respondent a piece of information, that elicits any action on the part of the respondent. Unlike the previous action, no data is collected through an instruction.

Instructing the respondent to move to another location - `SKIP-TO (...)`

The directive for the respondents to move to a new location in the questionnaire. Often the ordering of questions in a questionnaire will require users to skip over certain questions to ensure redundant or unrelated questions are not answered.

Instructing the respondent to move to another location, based on prior response - `IF (...) SKIP-TO (...)`

Skipping questions based on prior responses is the primary way that branching is performed within questionnaires. Additionally, if the location to skip to is an instruction that precedes the branch, simple looping can be achieved.

Although the exact presentation of these elements may differ depending on the mode of survey, they can be thought of as abstract representations of the most basic instructions used to drive the logic of a questionnaire, be it paper or electronic in presentation. For example, in practice it is common to see skip logic tightly embedded within questions on traditional paper forms, but, this is a presentational simplicity. When examining the logic alone, for simplicity we can separate this presentational concept into two separate logical entities - i.e. an `ASK` and a `SKIP`.

Figure 2 shows a possible paper-based presentation of the questionnaire described by the flowchart in figure 5. Looking at the simple presentation of a questionnaire in figure 2, we can see how these generic actions may be presented to a respondent. In contrast, figure 3 demonstrates how these actions can be chained together in a pseudo-specification similar to a flow outline as described by Knuth. Although there are more subtleties when presenting a completed survey to a respondent, when describing and documenting the logical flow of the questionnaire, these four actions are sufficient.

6 Questionnaire logic and computer programming

In this context, a questionnaire can then be expressed as a linear collection of instructions which are processed sequentially during data collection. As such, the four basic actions given in section 5 constitute a generic, low-level, domain-specific programming

Figure 2: A simple employment questionnaire.

This is a survey about employment.

Question 1: Are you employed? *(tick one)*

Yes ☐ ... *Skip to Question 3*

No ☐

Question 2: How long have you been unemployed?

__________ years ... *Skip to Question 4*

Question 3: What is your current salary?

$ __________

Question 4: How long did you hold your last job?

__________ years

End of questionnaire.

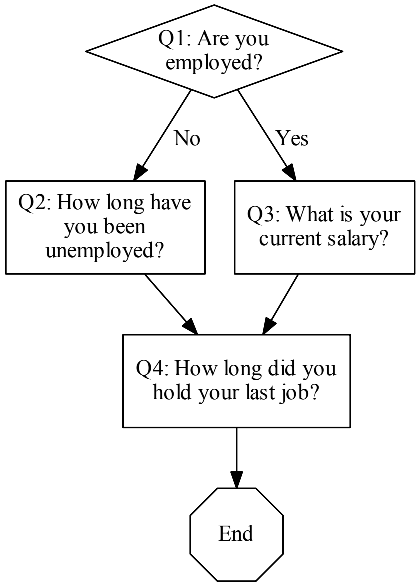
Thank you for taking the time to fill out this survey.

language for questionnaire based data collection. However, this assertion holds not only for electronic questionnaires, but any questionnaire - including paper-based surveys.

If we accept this assertion, we can then begin to use the research of computer science to understand how to better structure and analyse the logic frameworks of questionnaire design. For example, the introduction of control flow analysis and graph theory allows the computational analysis of respondent completion paths. Computer programming principles can be applied to determine the best and simplest instructions to better create questionnaires and allow for automated validation and optimisation of linear questionnaire paths.

Once these structured specifications are created, program compiler theory can help to better understand how to automatically generate efficient linear skip-based expressions of questionnaires to reduce respondent burden. Lastly, by the Church-Turing conjecture, if we accept the computability of questionnaire logic we can posit that any questionnaire, regardless of mode or complexity, must be describable within computer logic.

Figure 5: A flowchart showing the routing graph of a fake employment survey.



7 The logical and the presentational skip statement

The remainder of this paper focuses on the single change to questionnaire design that has the greatest potential for impact - the elimination of skip statements at a purely logical level. Firstly, it is important to differentiate between the use of a skip statement as a presentation element to support respondents, and as a logical construct for managing instruments. There is sufficient research into how the presentation of movement instructions, such as skip statements, impacts response. However, what we are discussing is the elimination of skips when describing the logical structure of the form.

For instance, figure 2 shows a short example questionnaire, asking about employment

Figure 3: A simple skip-based questionnaire specification
Here each action is preceded by a label to support the skips between sections.

```
     TELL "This is a survey about employment."
Q1:  ASK "Are you employed?"
     IF Employed: SKIP-TO Q3
Q2:  ASK "How long have you been unemployed?"
     SKIP-TO Q4
Q3:  ASK "What is your current salary?"
Q4:  ASK "How long did you hold your last job?"
END: TELL "Thank you for taking the time to fill out this survey."
```

or unemployment. The specification in figure 3, shows how this may be designed in a traditional way ready for a programmer to be enter into Computer Aided Interview software or other data collection tool.

The goal of this paper isn't to recommend for or against changes to the visual expression of a questionnaire as presented to a respondent or interviewer. Instead, the goal is to recommend changing the underlying specification of questionnaire logic to use a structured approach. Thus while the questionnaire presented in figure 2 would remain the same, we would opt to use a specification like that in shown in figure 4.

8 The elimination of the logical skip

If we accept the fact that questionnaire design, or at the least the logical component of questionnaire design is a domain-specific programming language, then the skip statement become analogous to the computer language `goto` command. The `goto` is a jump within the logic of a computer program to divert flow through a program and in the early years of computer programming was quite prevalent. However, in modern programming languages the `goto` statement is virtually non-existent. This change can be traced back to a single article - Edsger Djisktra's 1968 paper *Go To Statement Considered Harmful* [6]. In this paper Dijsktra explains how the use of `goto` statements prevented programmers from reliably predicting the paths taken to arrive at a specific instruction. Similarly, in questionnaire design, the use of skip statements prevents us from reliably predicting the paths a respondent would take to arrive at a given question. It is this unpredictability in the logic behind questionnaires that has hampered the shift towards more structured approaches to the design of CATI-scripts or online forms.

As a replacement for the `goto`, Böhm and Jacopini demonstrated that any program written using `goto`s could be rewritten using three basic constructs [3] - loops, conditional branches and sequential subprograms[1]. These constructs in programming languages allow modern programmers to write in high-level languages, which are then compiled into a low-

[1] In some cases it is necessary to include state variables to help maintain equivalence between an unstructured and structured program. At present the role of state variables within questionnaire design has not been fully explored. It is the authors hypothesis that these could allow for more formal description of sub-populations targeted by specific branches within a questionnaire.

Figure 4: A simple if-then-else-based questionnaire specification

```
TELL "This is a survey about employment."
TELL "Thank you for taking the time to fill out this survey."
ASK "Are you employed?"
IF Employed:
    THEN:
        ASK "How long have you been unemployed?"
    ELSE:
        ASK "What is your current salary?"
ASK "How long did you hold your last job?"
TELL "Thank you for taking the time to fill out this survey."
```

level machine languages, where structured logic is converted to simpler, but equivalent unstructured jumps during execution.

Given the structural similarities between skip-based questionnaires to goto-based computer programs, we can suggest that skips could be replaced at a logical level with similar structured constructs. Similarly, it should be possible to transform these structured questionnaires specifications into more traditional skip-based logic like that in section 5. In fact, preliminary research from the Australian Bureau of Statistics has shown this is possible, where skip specifications for both electronic and paper-based questionnaires can be automatically generated from structured questionnaire specifications.[2] What is important to recognise, is that in each of these cases unstructured jumps are automatically introduced as an aid for the agent[3] executing the program or questionnaire to better understand and process the logical structure, rather than being directly managed by questionnaire designers. This ensures that the respondent burden regarding a complex hierarchical questionnaire is avoided, and the ample research into the cognitive load of skip instructions remains valid.

The structured program theorem proposed by Böhm and Jacopini does not preclude the creation of additional logical constructs – quite the contrary, as once we eliminate the skip it is possible to build more advanced questionnaire programming paradigms to support the complex needs of questionnaire designers. By chaining the simple `if-then-else` branch it is possible to create a higher level switch instructions to support multiple branching based on pattern matching. For example, this would support the common pattern in employment questionnaires where a respondent is sequenced on their hours worked. This allows respondents to be sequenced as full-time or part-time employed or unemployed persons within a single logical object, rather than across multiple questions or logical branches.

The benefits of this approach have not gone unnoticed, as tools such as Blaise already promote the design of skip-free specifications [2]. Likewise, the structured design behind the Data Documentation Initiative format allows for similar skip-free logic when describing questionnaire metadata [7]. However, these improvements have been based on technological convenience when designing for electronic documentation. This shift in design has not had the same impact when designing for paper instruments. Thus, while skip logic is being phased out as a technical solution to many of these issues, survey designers are being conceptually sheltered from these methodological changes.

9 Benefits of a structured approach to questionnaire design

An alternative approach is to encourage survey designers to think using these new logical patterns. By constructing and analysing questionnaires in this structured way we can take advantage of structured programming principles. Under a hierarchical model of questionnaires, like that shown in figures 4 and 6, we can easily predict the targeted respondents of a question by examining its parent constructs.

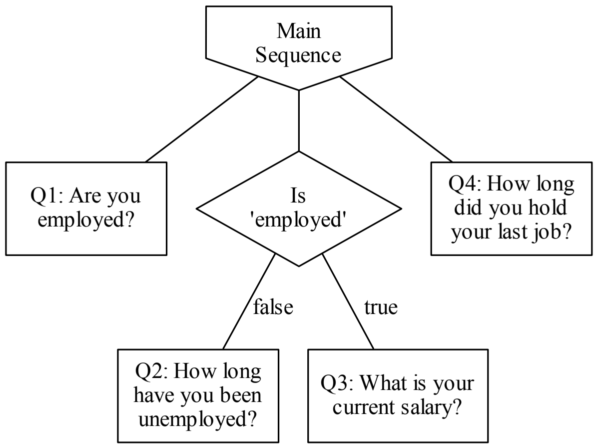
[2] In this case, the logical skips necessary for web-based questionnares (in XForms) and paper questionnaires (in LaTeX) were created using a single logical specification (in the DDI XML format).

[3] In a computer system, the executing agent of a logical program is the central processor, however when dealing with paper-based questionnaires the agent that executes and controls the logic is a human being!

For example, the structures of figures 5 and 6 each describe asking the same questions in the same order. If we wish to determine what respondents will answer question 4, in the traditional skip based structure of figure 5 we must trace both paths from question 4 back to the start. In a short question this is trivial, but in a questionnaire like that presented in figure 1 is is nearly impossible. In comparison, in figure 6, we need only trace back a nodes' parents to determine under what conditions it is answered. In this case, we can see that everyone who follows the main sequence will answer this question. However, it is also the case that if this were a small part of a larger tree, we can authoritatively say that everyone who meets the conditions to enter this sub-tree must also have to answer question 4.

Another great advantage of a hierarchical model for questionnaire logic, is the ease of modification and sharing it enables. As the tree model only couples a logical component to its parent. This allows sections to be added or removed free of side effects impacting other questions or logic. As such, it becomes possible to easily share standardised modules of questions across surveys or even across agencies - making shared international standards for official statistics a practical reality.

Figure 6: A hierarchical syntax tree[4] based on the structured description of the fake employment survey.



In the traditional model of figure 5 in to remove question 3, we need to check what paths lead into this question, and what paths lead out and adjust these accordingly. Alternatively, to add a new question following question 3, we would need to check the conditions under which it should be answered, and alter the logical flow for questions 2, 3 and 4 depending on the circumstances.

In the hierarchical model, these become simple tasks; to remove question 3 we remove it from the tree. In this case, if the logic for the 'true' branch of the 'Is employed' node is triggered, as there is no child elements flow simply passes to the next right sibling - question 4. To add new question, we just need to determine for whom it is applicable, if the new question is applicable only to those who are employed, we add a new child to the 'true' branch of the conditional; if it is for all respondents, we insert a new right sibling to the conditional. In either case, the logic required for skips can be determined and there is no chance of faulty logic forcing respondents to answer invalid questions. However, it is important to realise that this approach scales to allow whole new sections to be reliably inserted into the tree, as the logical flow can be recalculated and expressed as necessary.

[4] A syntax tree can be read in a standard depth first, left to right order. All actions under the leftmost child item are executed before moving the the next right sibling.

10 Conclusion

This paper has drawn comparisons between the logical structure of questionnaires and computer programs, which have helped identify and clarify the main problems in questionnaire design. The issues preventing automation of paper form design, the creation of electronic surveys and the ability to share standard modules all stem from one root problem – the lack of rigid and logical structure within questionnaire specifications.

With the elimination of the logical skip statement and a shift to a structured approach to questionnaire design, we can explore new ways to resolve these issues. Thus far, we have only examined a few key papers in computer science, written when it was still an emerging discipline. While these were instrumental in advancing the field, there remains another forty years of widely accepted theory to build upon. Armed with this new knowledge, statistical survey methodology is poised to undergo a paradigm shift no less profound than that of structured programming within computer science, and stands to change everything about what we think of questionnaires: from creation and collection all the way through to final data dissemination.

References

[1] Australian Bureau of Statistics. Information paper: Questionnaires used in the labour force survey, 2004.

[2] Bethlehem, J. The routing structure of questionnaires. *International Journal of Market Research 42*, 1 (2000), 95–110.

[3] Böhm, C., and Jacopini, G. Flow diagrams, turing machines and languages with only two formation rules. *Commun. ACM 9* (May 1966), 366–371.

[4] Bulmer, M. *Francis Galton: pioneer of heredity and biometry.* Francis Galton. Johns Hopkins University Press, Baltimore, MD, USA, 2003.

[5] de Vaus, D. *Surveys in social research.* Social research today. Routledge, St. Leonards, NSW, Australia, 2002.

[6] Dijkstra, E. W. Letters to the editor: go to statement considered harmful. *Commun. ACM 11* (March 1968), 147–148.

[7] Iverson, J. Metadata-driven survey design. *IASSIST Quarterly Spring 1*, 3-4 (2009), 7–9.

[8] Jabine, T. B. Flow charts: A tool for developing and understanding survey questionnaires. *Journal of Official Statistics 1*, 2 (1985), 189–207.

[9] Katz, I., Stinson, L., and Conrad, F. Questionnaire designers versus instrument authors: Bottlenecks in the development of computer administered questionnaires. In *Fifty-Second Annual Conference of the American Association for Public Opinion Research* (Norfolk, VA., 1997), American Association for Public Opinion Research, pp. 1029–1034.

[10] Knuth, D. Computer-drawn flowcharts. *Commun. ACM 6* (May 1966), 555–563.

[11] Knuth, D. *The Art of Computer Programming: Fundamental algorithms.* The Art of Computer Programming. Addison-Wesley, 2008.

[12] Oppenheim, A. *Questionnaire design, interviewing and attitude measurement.* Continuum, London, UK, 2000.

[13] Pink, B., Borowik, J., and Lee, G. The case for an international statistical innovation program. *Statistical Journal of the IAOS 26*, 3-4 (2009), 125–133.

[14] Saris, W., and Gallhofer, I. *Design, evaluation, and analysis of questionnaires for survey research.* John Wiley & Sons, Inc., Hoboken, NJ, USA, 2007.