Survey state model (SSM): XML authoring of electronic questionnaires.

LLORET, J. and WIRATUNGA, N.

2015

Survey State Model (SSM)

XML Authoring of electronic questionnaires

Jose Lloret
The Robert Gordon University
<j.m.lloret-perez@rgu.ac.uk>
Nirmalie Wiratunga
The Robert Gordon University
<n.wiratunga@rgu.ac.uk>

Abstract

Computer Assisted Interviewing (CAI) systems use questionnaires as the instruments to conduct survey research. XML constitutes a formal way to represent the features of questionnaires which include content coverage, personalisation aspects and importantly routing functionalities. In this paper we conduct a comparative analysis on different XML approaches to questionnaire modelling. Our findings suggest that existing language formalism are more likely to cover content coverage but often fail to model routing aspects.

In particular the popular hierarchical approach to modelling routing functionality has one or more draw backs along the lines of ability to facilitate questionnaire logic validation, ease of understanding by domain experts and flexibility to enable refinements to questionnaires.

Accordingly we introduce the Survey State Model (SSM) XML language based on a state-transition model to address these shortcomings. We present our results from testing SSM on a sample of real-world surveys from Pexel Research Services in the UK. We use the distribution of SSM's vocabulary on this sample to demonstrate SSM's applicability and its coverage of questionnaire constructs and effective routing support.

Keywords: XML, XSD, SCH, authoring, survey, questionnaire, CAI, hierarchical model, state-transition model

1. Introduction

Surveys are the systematic collection of information from individuals or organizations to address research and business objectives [1]. Questionnaires are one of the instruments of surveys utilised to collect data considered as structured interviews.

The Computer Assisted Interviewing (CAI) systems allow the design, collection, management, analysis and reporting of surveys through computers and they should

have a specification language addressed to describe every feature presented in questionnaires.

XML was designed to represent structured documents and as such may be used to formally represent every requirement from questionnaires. These requirements are imposed by the designers of surveys who decide the order in which the questions are shown to the respondent as well as the possibility to be asked or not based on respondent previous answers. The use of XML to describe questionnaire's specification may reduce the need of programmers to implement questions order as well as complicated logical decisions since the creation of intuitive interfaces can generate questionnaire's requirements easily in XML. Also, the exchangeability inherent in XML allows the design of questionnaires be circulated among different Computer Assisted Interviewing (CAI) systems without hardware or software restrictions.

Pexel Research Services carries out a large number of surveys through telephone every year based on client's specification. As such they use a Computer Assisted Interviewing (CAI) solution to conduct surveys but they are keen to consider alternative solutions which may offer commercial advantages over existing systems.

The rest of this paper is structured as follows: Section 2 explains what questionnaires are and the features that may appear on them. Section 3 reviews the different approaches based on XML aimed to represent questionnaires. Section 4 present the desired criteria when modelling routing of questionnaires and explains the most popular approach used to model the routing of questionnaires. Section 5 presents our alternative solution to describe surveys based on state-transition model in which XML examples are provided. Finally, Section 6 summarizes the results obtained after testing our approach based on real surveys provided by Pexel Research services.

2. Electronic questionnaires

A questionnaire is one of the instruments of surveys to collect information from people or organizations. A paper questionnaire contains a set of *questions* addressed to the interviewees and *instructions* for interviewers which allow skipping over questions or even directly jumping to the end of the questionnaire [4] based on interviewee responses.

Table 1. Paper questionnaire

INF1. This is an example survey to demonstrate the features that can appear in electronic questionnaires.
Q1. How often do you use your car?

01. Never **GOTO END**

02. Almost never **GOTO END**

03. Occasionally/Sometimes **GOTO END**

04. Amost every time

05. Every time

Q2. Which brands are you aware of? **[FIRST SPONTANEOUS MENTION]**

01. A

02. B

03. C

04. D

05. E

06. F

07. G

08. H

99. Don't know **GOTO END**

Q3. Which brands are you aware of? **[OTHER SPONTANEOUS MENTIONS Q2]**

Q4. Using a scale 1 to 5 where; 5=essential, 4=very important, 3=quite important, 2=relatively unimportant and 1=not at all important. How important are the following safety features when you want to buy a car?

01. Stable body shell

02. Pre-tensioned and load-limited seatbelts

03. Good head restraints

04. Seat-mounted side airbags

05. Side curtain airbags

06. Knee airbags

[IF 'F' IS SELECTED IN Q2 OR Q3 OTHERWISE GOTO END]

Q5. How many cars have you had or have of F brand?

[REPEAT Q6a FOR EACH CAR]

Q6a. Were you satisfied of the safety features?

01. Yes

02. No

03. Don't remember

[IF SATISFIED FOR EVERY CAR OF 'F']

INF2. We are happy that you always like our safety features and we hope you consider us again for future purchases.

END. THANKS AND CLOSE

Generally a questionnaire is divided into *sections* where the presence of *intro* sentences to introduce or end a section becomes important to locate the respondent. This example includes an outer section for INF1, Q1, Q2, Q3, Q4, Q5, INF2 and an inner section for Q6a.

Table 1 presents a small questionnaire that nevertheless demonstrates a variety of common constructs ranging from simple to complex semantics. The most common types of questions are *single-response*, *multiple-response* and *open-ended* (e.g. Q1, Q3, Q5 respectively). Whilst grid (e.g. Q4), unlike the common constructs differs in the manner in which a respondent chooses the responses and remains more cognitively demanding on the interviewee [6].

The instructions, in bold font, are normally needed to manage questionnaire routing according to responses from the interviewee. There are three such routing constructs in this example:

- *skip* feature attached over responses in Q1 or Q2, known as unconditional skips, as well as conditional skips linked in the Q5.
- *filter* constructs, based on a logical expression involving the responses to one or more questions. They are represented using if-then-else statement, for instance the instructions attached over Q5, INF2.
- *loop* feature, allowing the execution of a part of the questionnaire a number of times. The instruction over Q6a permits the execution of that question as many times as the respondent had/has cars of F brand.

In addition to the constructs discussed above there are several additional features in *electronic questionnaires* that are not present in paper questionnaires:

- *Piping* which allows the retrieval of an answer from a previous question as part of the text for another or the automatic generation of responses based on a expression (e.g. Q3 responses are generated automatically according to the responses non-selected in Q2).
- *Computation* that constitutes the execution of an arithmetical expression and its assignment over a variable referenced. Usually it is used to communicate data among sections. For example, after Q6a, if the respondent was satisfied with the safety features, the addition and assignament over a variable could be used in the logical expression preceding INF2.
- *Check* which involves the satisfaction of a logical expression notifying the respondent to solve the inconsistency if it is not fulfilled. For instance, imagine a

scenario in which it was asked: "Qx. do you use a car to go working?" and the respondent replied yes, after it was asked "Qy. how much do spend on petrol?" and the respondent answered zero. This construct might create a logical expression "check Qy is greather that cero if Qx is yes" *warning* or *stopping* the flow of the questionnaire until the inconsistency was solved.

3. Related work

A Computer Assisted Interviewing (CAI) system should have a specification language addressed to describe the features of questionnaires at a much higher level than a programming language in which these constructs will eventually be manipulated. As such, the authoring languages Computer-Assisted Survey Execution System (CASES) [7] and BLAISE [8] provide this level of abstract language specification covering a substantial number of questionnaire constructs. However they each use a proprietary representation language which constrains its adoption and wide scale usage.

In more recent work XML based formal representation of questionnaires has increased in popularity. Table 2 summarizes for each language in the literature the set of questionnaire features that are being addressed. First, *content* which contains the possible types of questions that may appear over questionnaires as well as how these may be grouped. Secondly, *routing* which eliminates the need to follow questionnaire intructions manually. Usually, the routing is described through boolean expressions (e.g. skip, filter, loop and check) but may use arithmetical expressions (e.g. computation) to guide the respondent through the questionnaire and finally *personalise* to adapt the survey to the respondent and to create dynamic adaptation of content at run-time. In this category, protecting a respondent's privacy through the *randomising* of responses to a question to reduce bias evasive responses [15], or *rotating* constructs are examples widely used in surveys.

The content category is well represented in each language explored because they cover the three type of questions most used (single, multiple and open) as well as section. However, Survey Interchange Standard (Triple-S) [2] and Questionnaire Definition Language (QDL) [9] do not contain intro questions and none of them provide the possibility to represent grid questions. Despite the fact that grid could be replaced through several single or multiple questions, we have decided to include this construct as a new question type because it is widely used and covered by Azzara in the design of questionnaires [1].

The routing is partially covered in Survey Interchange Standard (Triple-S) in the form of simple filters based on logical question (e.g. question with Yes/No answers). Simple Survey System (SSS) and Structured Questionnaire Building Language (SQBL) offer filter and loop constructs but they do not implement the skip feature. This could be because this feature can be reversed and use filters instead [3] or because a questionnaire designed without skip constructs is easier to modify, share

Table 2. Features of electronic surveys

Category	Feature	Triple-S	SSS	SQBL	QDL
Content	Section	✓	✓	✓	✓
	Intro	✗	✓	✓	✗
	Single-response	✓	✓	✓	✓
	Multiple-response	✓	✓	✓	✓
	Open-ended	✓	✓	✓	✓
	Grid	✗	✗	✗	✗
Routing	Skip	✗	✗	✗	✓
	Filter	✗	✓	✓	✓
	Loop	✗	✓	✓	✓
	Check	✗	✗	✗	✓
	Computation	✗	✓	✗	✓
Personalisation	Piping	✗	✗	Partial	✗
	Randomising	✗	✗	Partial	✗
	Rotating	✗	✗	✗	✗

and understand [10]. In regard to Questionnaire Definition Language (QDL), it is the only candidate able to represent every routing feature, however its expressions are typed in infix mode involving the use of parenthesis that result in complicated expressions that are prone to error and harder to process.

The personalisation features are only partially covered by Structured Questionnaire Building Language (SQBL) although it is less able to generate automatic responses from previous answers to single/ multiple questions and does not offer random order or rotating of responses to a question.

There are different ways to model the routing that questionnaires follow like flowcharts or the use of graph theory principles but the hierarchical model is the most popular among the languages that we have explored.

The hierarchical model is the approach used to describe the flow of questionnaires in Simple Survey System (SSS) and Structured Questionnaire Building Language (SQBL) and we have studied them in order to address the following questions:

- Does the hierarchical model lend itself to questionnaire design?

- How can features in a questionnaire be represented in an XML authoring language?

Other language approaches such as Survey Interchange Standard (Triple-S) is not considered in our comparative analysis because it does not offer a modelling approach or in the case of Questionnaire Definition Language (QDL) which is not supported any longer.

4. Hierarchical routing model

A language for questionnaire should provide a means to express both content features as well as vocabulary to express the routing of questionnaires. The latter is particularly challenging as it requires the modelling of sophisticated logic which dynamically impacts the relevance of questions given one or more responses to previous questions.

So what are the criteria that must be considered when creating a representation to model question routing?

- *pre-test*: A pre-test is a formal review of a questionnaire [5] aimed at discovering problematic questions and rewriting them to improve understanding. This is expected to lead to improved collection of responses. For instance in a pre-test one would expect to discover any logical inconsistencies in the routing. Thus, a model in which testing does not become cumbersome is beneficial.
- *matching design-model*: The designers of questionnaires specify routing through skip constructs or filters and having a model to support both features should be useful in many questionnaire design projects [13].
- *adaptability*: It is frequent to introduce changes after the questionnaire implementation in the Computer Assisted Interviewing (CAI) system, so a model in which the changes are easy to make would be desirable.

There are variety of modelling approaches that have been proposed for questionnaire routing management. Flow-charts which are used in programming languages not surprisingly has also been used to develop and understand questionnaires [11]. Accordingly the applicability of graph theoretic relationships has been studied by Fagan and Greenberg [14] in the context of questionnaires with particular focus on modelling skip logic that allows skipping over questions based on responses to previous questions. Generally questions, can be modelled as vertices, and skip constructs, modelled as edges that direct the source and destination questions. However, to the best of our knowledge such approaches to modelling routing behaviour have not been developed in to a formal XML representation.

A more promising approach to routing behaviour modelling can be seen in the hierarchical model, used in Simple Survey System (SSS) and Structured Questionnaire Building Language (SQBL) (see Figure 1). Here boxes represent questions and filters are diamonds. The logic behind this approach is a tree that presents advantages

like: each question has one and only one path to determine its reachability. The tracing through directed edges between parent-child relationships also allows to determine all circumstances under which a question can be executed [10].

In particular the hierarchical model enables easy pre-testing of a questionnaire as it allows exploration of one or more paths from the start node to any selected end node. For instance, in order to test Q4 isolated, four previous questions must be reached (INF1, Q1, Q2, Q3) and three filters have to be true.

In regard to matching design-model, this alternative does not offer skip features which requires the negation of the logic when questionnaires are described using these constructs. Despite the fact that questionnaires could be described using filters only, the designers of questionnaires are usually non-programmers and they continue to use skips. This means that although the hierarchical model can be used to model routing behaviour; expecting designers to interpret any skip logic as filters is likely to be demanding. For example, the unconditional skips attached over the responses 01, 02, 03 of Q1 represented with a filter would be NOT(Q1 EQ '01' OR Q1 EQ '02' OR Q1 EQ '03) then Q2 else END.

Concerning the adaptability to changes, imagine a simple modification of the survey example in the Table 1 to introduce a new skip over the response 04 (Almost every time) in Q1. This scenario would require another filter to determine whether this response was selected or not as well as the duplication of the question Q4 since the other Q4 modelled is not directly reachable due to the logic inherited from a tree.

5. Survey State Model (SSM) solution

We propose a state-transition based modelling approach to better address the routing requirements involving pre-testing, designing and adaptability criteria.

The state-transition model, depicted in Figure 2, describes the questionnaire presented in Table 1. This model contains various types of states that are linked through transitions to form state-models. This approach has a set of initial states addressed to decide what state is executed first (e.g. filled circle pointing to INF1 or Q6a) as well as a group of states indicating the ending and represented by filled circles with a white outline. Each state-model includes a set of variables representing different types of question (e.g. Q1, Q2, Q3, Q4, Q5, Q6a are variables which describe questions and their values stored) or fields used for computations constructs (e.g. SATISFIED describing an integer number of cars in which the respondent is happy with the safety features). Every transition connects a source with a target state and allows the description of every possible route through the questionnaire. Transtions with decision states involving a source is depicted as diamond. Such nodes are selected when a boolean expression is satisfied. We propose the use of Reverse Polish Notation (RPN) [16] for all expressions involving variables and any constants that appear in our state model. This approach is considered faster than infix or prefix

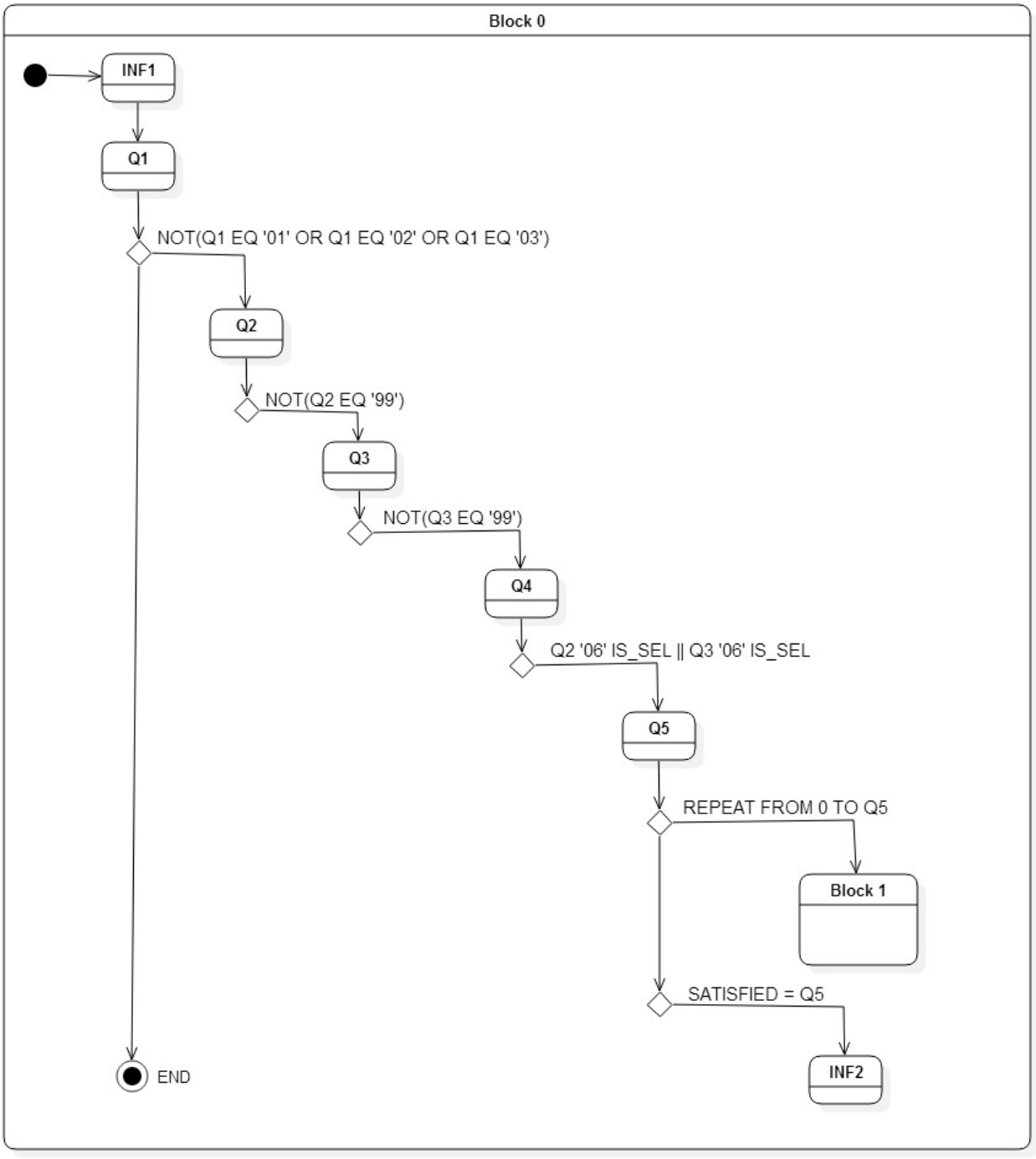

Figure 1. Hierarchical model

notations because the expressions do not need parenthesis and fewer operations are required to solve the expressions. There are several expressions implemented in the example like [Q1, '01', IS_SEL, Q1, '02', IS_SEL, OR, Q1, '03', IS_SEL, OR] addressed to decide whether ending the questionnaire or continuining with Q2.

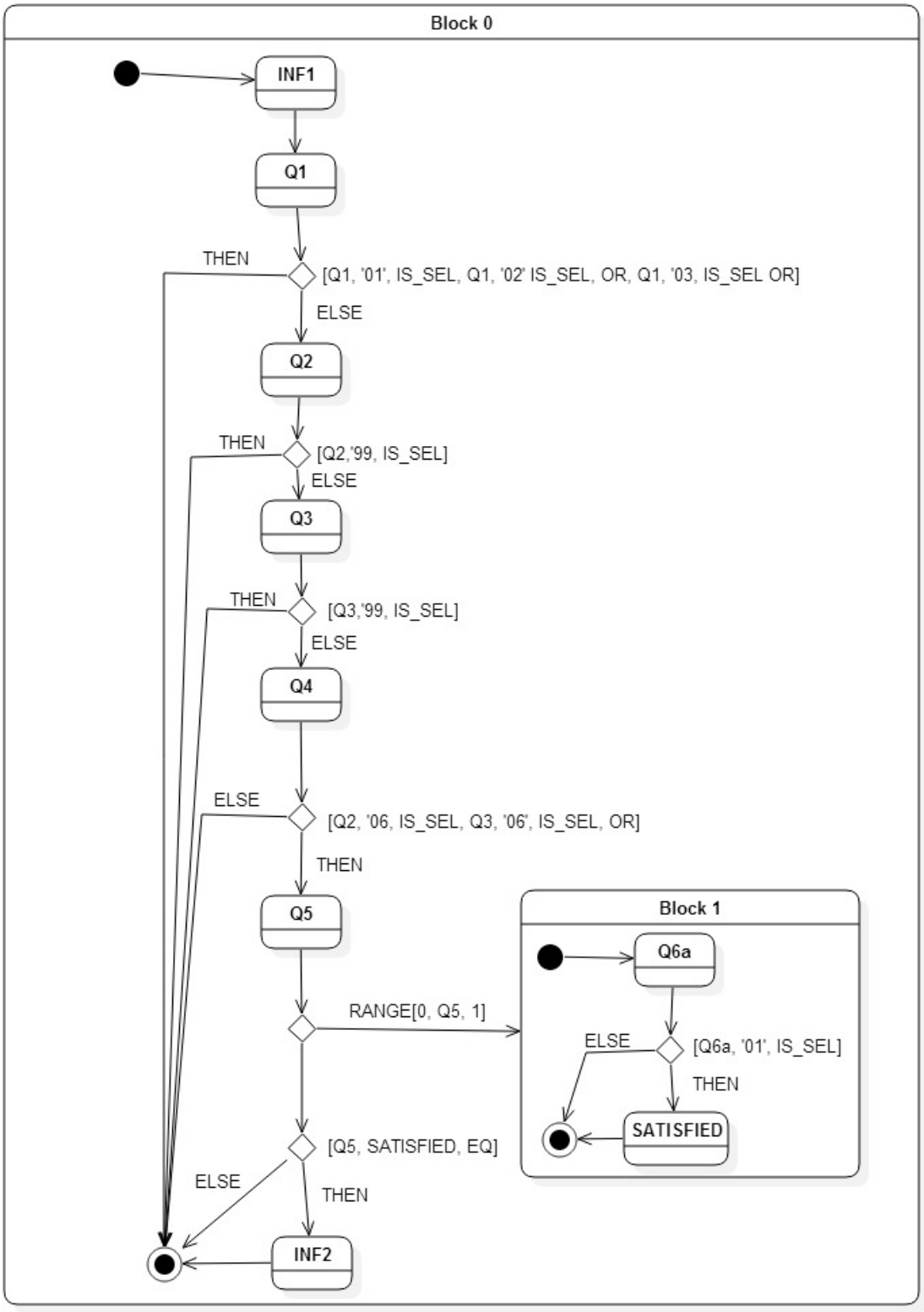

Figure 2. State-transition model

Accordingly we formalise the state-transtion model that is applied to questionnaire routing as follows:

$$M = \langle Q, V, T, I, E \rangle \quad (1)$$

where,

1. $Q \neq \varnothing$ is a finite set of states. These states are: simple, composite, if-then-else, for, check, computation and sink.
2. V is the set of variables. Every variable V may be accessed at every non-sink state $q \in Q$.
3. T is a finite set of transitions.

   A transition $t \in T$ is represented as $q \xrightarrow{[c]} q'$, where $\{q, q'\} \in Q$ and c is a boolean expression involving variables of V and/or constants. The absence of c is interpreted to *true*.
4. $I \subset Q$ is the set of initial or source states. These states determine which state is the first to be reached for a state-model defined.
5. $E \subset Q$ is the set of end states. These states determine which state is the last to be reached for a state-model defined.

5.1. States of Survey State Model (SSM)

In addition to meeting the routing criteria discussed in the previous section, a questionnaire modelling language should also have a good coverage of the different types of questionnaire routing logic. In the proposed Survey State Model (SSM) this translates to the different types of states that can be modelled. These are discussed next with reference to the question types illustrated in Figure 2. Additionally, each state explained ends with an XML code according to SSM XML authoring language.

- *Simple* state is responsible for retrieving the content of the variable (i.e. the definition of the question as well as the response stored, if any). This state allows retrieving one or more variables simultaneously which means the Computer Assisted Interviewing (CAI) system should present on the screen every variable referenced. For instance, Q1 state references the single-response variable Q1.

```
<state id="Q1">
    <variable ref="Q1"/>
    <transition target="p0"/> <!-- p0 is the decision state
after Q1 -->
</state>
```

- *Composite* state permits to switch and reuse the state-model referenced. It could be seen as functions in structured programming in which the code under that function is executed. For instance, the block 1 is called as many times as the range defined after Q5 is true.

```
<state id="c1"> <!-- c1 is the state pointed by the
                transition RANGE[0, Q5, 1]-->
    <include statemodel="block1"/>
</state>
```

- *If-then-else* state represents the filter and skip features of questionnaires. It is composed of a boolean expression in Reverse Polish Notation (RPN) and permits describing two transitions *then* and *else* for true and false result of the expression respectively. For instance, the diamond between Q1 and Q2 implements the logic for the unconditional skips attached over the responses of Q1. Due to the transition principle of this approach there is no difference between skip or filter. The next XML example code decides whether skipping to the END of the questionnaire or continuing with the Q2. The expression in Reverse Polish Notation (RPN) is [Q1, '01', IS_SEL, Q1, '02', IS_SEL, OR, Q1, '03', IS_SEL, OR] witch is translated to five binary expressions because the operators IS_SEL and OR expect two operands.

```
<state id="p0">
    <if>
        <condition>
            <binary>
                <binary>
                    <binary>
                        <variable ref="Q1"/>
                        <constant type="string" value="01"/>
                        <operator name="IS_SEL"/>
                    </binary>
                <binary>
                     <variable ref="Q1"/>
                     <constant type="string" value="02"/>
                     <operator name="IS_SEL"/>
                </binary>
                <operator name="OR"/>
            </binary>
            <binary>
                <variable ref="Q1"/>
                <constant type="string" value="03"/>
                <operator name="IS_SEL"/>
            </binary>
            <operator name="OR"/>
        </binary>
```

```
      </condition>
      <then>
         <transition target="sink0"/>
      </then>
      <else>
         <transition target="Q2"/>
      </else>
   </if>
</state>
```

- *For* state matches with the loop feature of surveys and has two transitions, one for the true and another for the false result of the boolean expression. This state has start, end and step elements to define the boundaries of the range expression. For example, the diamond after Q5 starts at 0, ends at the number stored in Q5 and increments in steps of 1. A true result switches to the second state-model (e.g. block 1) whereas the false jumps to other state (e.g. the if-then-else state pointing to INF2).

```
<state id="p4">
    <for>
        <field ref="p4_iterator"/>
        <in>
            <range>
                <start><constant type="string" value="0"/></start>
                <end><variable ref="Q5"/></end>
                <step><constant type="string" value="1"/></step>
            </range>
        </in>
        <transition target="c1"/><!-- c1 is the composite
                                      state which includes
                                      block1 -->
    </for>
    <transition target="p5"/><!-- p5 is the decision state
                                  [Q5, SATISFIED, EQ] -->
</state>
```

- *Check* state defines a boolean expression in which the true result shows a message (warning or error) notifying to the respondent the trigger of an inconsistency. The error message is aimed to stop the execution of the state-model until the conflict is solved. The next XML code describes a expression to test whether Qy is greather than zero or not. A true result should show the label *"A car cannot run without fuel"* in warning mode.

```
<state id="py">
    <check type="warning">
        <condition>
            <binary>
```

```
                    <variable ref="Qy"/>
                    <constant type="integer" value="0"/>
                    <operator name="GT"/>
                </binary>
            </condition>
            <label lang="en">A car cannot run without fuel</label>
        </check>
        <transition target="pz"/>
    </state>
```

- *Computation* state permits defining an arithmetical expression. For instance, the state after Q6a, named as SATISFIED, describes an arithmetical operation of incrementing the variable by one. The use of this state allows referencing variables out of the scope of an state-model like SATISFIED variable referenced in the if-then-else state pointing to INF2 in the block 0.

```
<state id="p1">
    <computation ref="SATISFIED">
        <assignment>
            <unary>
                <variable ref="SATISFIED"/>
                <operator name="INC"/>
            </unary>
        </assignment>
    </computation>
    <transition target="sink0"/> <!-- sink0 is the sink state
                                  (filled circle with a white
                                   outline)
</state>
```

- *Sink* state is aimed to describe the ending of the state-model. For example, the two state-models, Block 0 and Block 1, contain sink states indicating that no more states are reached after them. As the reader may appreciate, there are no transitions going out from them.

```
<state id="sink0">
    <sink/>
</state>
```

Finally, the *start* state is a property which determines the first state to execute in the set of states from a state-model. As such, the following XML code describes the source state for block 0 and block 1 respectively.

```
<statemodel id="block0">
    <start id="INF1"/>
    <state id="INF">...</state>
    ...
    ...
```

```
    ...
</statemodel>
<statemodel id="block1">
    <start id="Q6a"/>
    <state id="Q6a">...</state>
    ...
    ...
    ...
</statemodel>
```

5.2. Validation of questionnaires

Survey State Model (SSM) XML language implements a two-steps validation process to determine whether an XML document describing a questionnaire is correct or not (see Figure 3). Both steps take as input the *XML document* to be validated against *a set of rules* defined in a formal way to express syntax and/or semantics.

The first step checks the structure, form and syntax in XML Schema Definition (XSD). For instance, *"A section contains a set of questions and these may be intro, single-response, multiple-response, open or grid."*. This process may finish with "yes" involving that the document should be passed to the second step or "no" which reports a JavaScript Object Notation (JSON) document with the errors. If the errors are not fatal the process carries on until the end of the file is detected, otherwise the process stops checking at the line where the error was raised.

The second step examines the relationships among elements through Schematron (SCH). For example, *"Every transition's target must be a defined state in the statemodel"*. Similarly, when a non valid XML file is encountered the process ends by communicating the errors through a JavaScript Object Notation (JSON) file. This process is not able to differentiate between fatal and non-fatal so every error raised is reported.

6. Results

In order to test Survey State Model (SSM) XML authoring language coverage and to know the relevance of every feature of questionnaires we used it to model a set of 15 questionnaire that were sampled by Pexel Research Services Ltd. In particular we studied the frequency distribution of SSM vocabulary over this sample of questionnaires. Figure 4 shows the distribution of SSM vocabulary applied to a sample of 14 test questionnaires sorted by decreasing order of frequency. The graph includes SSM constructs relating to content constructs (section, intro, single, multiple, open, grid), routing behaviour (skip, filter, loop, check, computation) and personalisation constructs (piping, randomise, rotate). It was encouraging to us that all questionnaires in the sample were represented using SSM. Apart from the check and computation constructs, every other feature was included in them.

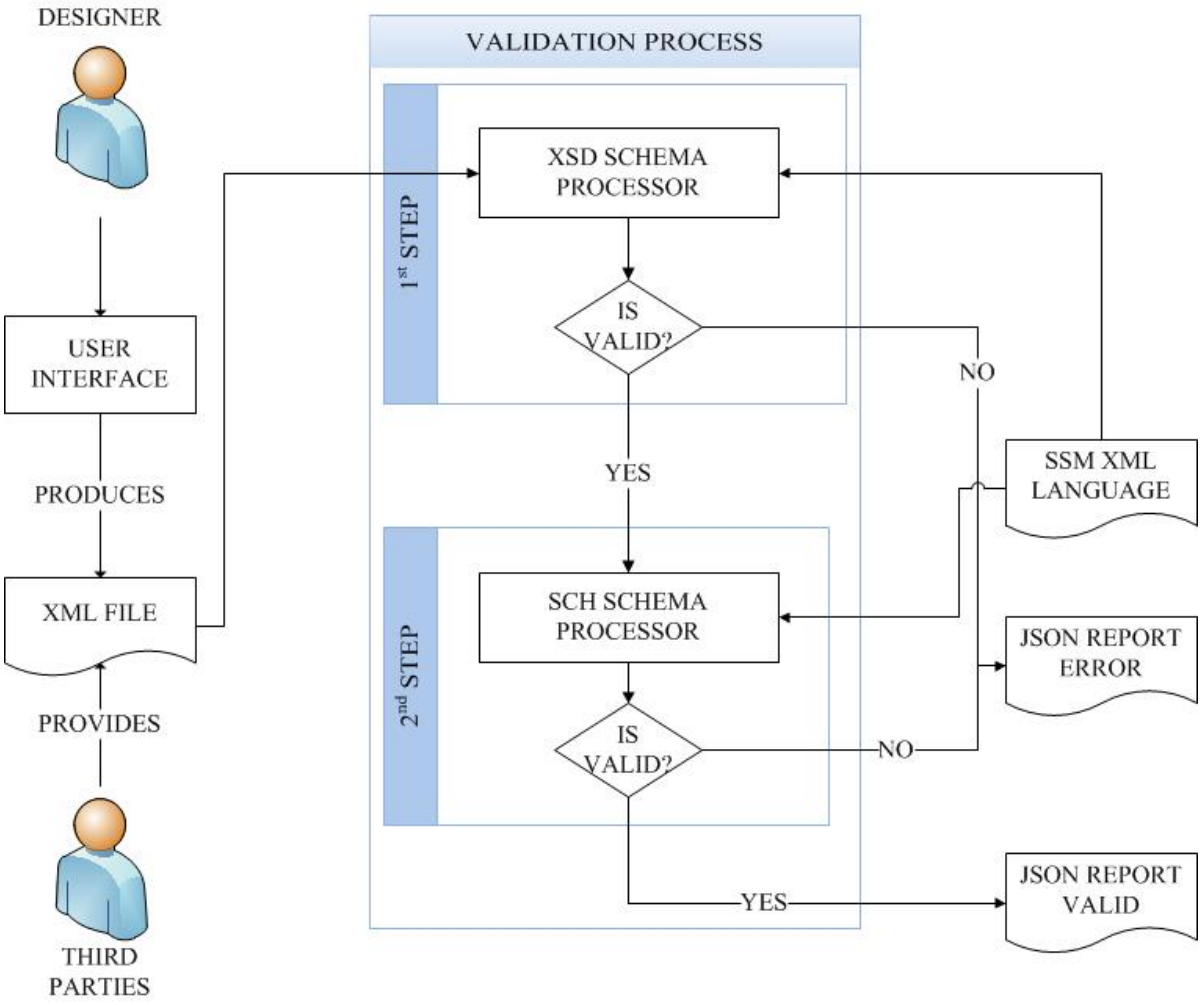

Figure 3. CAI validation process for SSM

Firstly, for the content category, may be affirmed that single-response has the most significant frequency whereas grid question, which unexpectedly has resulted in the lowest important in terms of question types.

Secondly, for the routing classification, the skip feature is frequently most used in our sample of questionnaires even though it is normally best avoided in conventional programming languages [12]. Whilst it is true that programmatically it can lead to non-elegant code constructs, here we conclude that it remains an important feature of questionnaires and so should ideally be facilitated by the underlying language of questionnaires. Computation feature, without any frequency at all, was expected some significance since it becomes essential when a section requires data from other section, however the absence of check does not constitute any surprise since it is a rare feature over questionnaires.

Finally, for the personalising grouping, the three features are presented in the sample tested and piping constitutes the most used specially for generating responses whereas randomising or rotating were less popular.

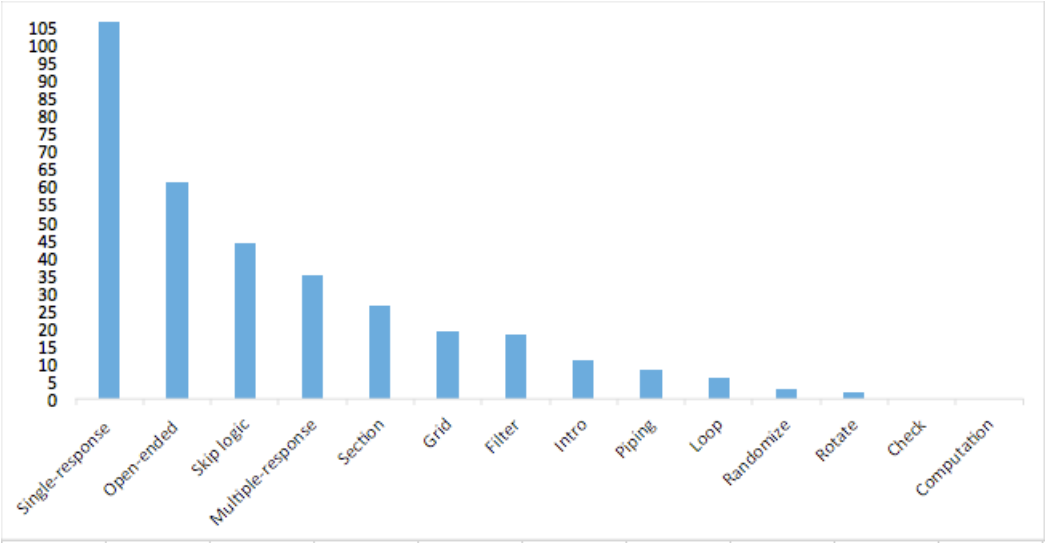

Figure 4. Real surveys frequencies

7. Conclusions

The use of XML to manage the representation of questionnaires for Computer Assisted Interviewing (CAI) systems is not new. We have conducted a detailed comparison of commonly used XML languages in terms of coverage of questionnaire constructs to facilitate representation of content and routing functionality.

Our findings suggest that all existing formalisms to modelling the routing task falls short in terms of one or more of the three key criteria: pre-test, matching design-model and adaptability to changes; similarly none of them covers all constructs that are to be expected in questionnaires.

Accordingly to address this problem we have introduced Survey State Model (SSM) XML language which uses a state-transition model to represent routing. Finally, our results from testing SSM on a set of real-world large-scale surveys suggest that the state-transition model is not only able to represent a wider range of questionnaire constructs but also lends itself to addressing the challenges of the routing task.

The authors would like to thank Bruce Leslie, technical Director of Pexel Research Services, for giving his expertise at every stage of this survey research as well as for selecting the variety of real-world survey samples. Also, we wish to thank to Pexel Research Services because without the funding provided by the company, the research paper would not have been possible.

Bibliography

[1] Carey V. Azzara. *Questionnaire design for business research*. Tate Publishing & Enterprises, LLC.

[2] Laurance Gerrard, Keith Hughes, Steve Jenkins, Ed Ross, and Geoff Wright. *Triple-S XML, The Survey Interchange Standard*. ASC.

[3] Albert D. Bethke. *Representing Procedural Logic in XML*. Journal of Software, Vol. 3, No. 2. February 2008.

[4] Jelke Bethlehem. *The routing structure of questionnaires*. Internation Journal of Market Research, Vol. 42, No. 1. 2000.

[5] IBM Corporation Software Group. *The hows and whys of survey research*. October 2012.

[6] Tim Bock. *Market Research*. http://mktresearch.org/wiki/Main_Page .

[7] University of California at Berkeley. *Computer-Assisted Survey Execution System*. http://cases.berkeley.edu/ .

[8] Statistics Netherlands. *Blaise, Survey software for professionals*. http://www.blaise.com/ .

[9] Jelke Bethlehem and Anco Hundepool. *On the Documentation and Analysis of Electronic Questionnaires*. Statistics Netherlands. 2002.

[10] Samuel Spencer. *A case against the Skip Statement*. 2012.

[11] Thomas B. Jabine. *Flow Charts: A Tool for Developing and Understanding Survey Questionnaires*. Journal of Official Statistics. Vol. 1, No. 2. 1985.

[12] Edsger W. Dijkstra. *A Case against the GO TO Statement*. ACM 11 (1968), 3: 147-148.. 1968.

[13] Irvin Katz, George Mason, Linda Stinson, and Frederick Conrad. *Questionnaire designers versus instrument authors: Bottlenecks in the development of computer-administered questionnaires*.

[14] Jim Fagan and Brian V. Greenberg. *Using graph theory to analyze skip patterns in questionnaires*. Statistical Research Division Bureau of the Census Washington, D.C.. 1988.

[15] Stanley L. Warner. *Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias*. Journal of the American Statistical Association. Vol. 60, No. 309. 1965.

[16] Bob Brown. *Postfix Notation Mini-Lecture*. http://bbrown.spsu.edu/web_lectures/postfix/ .

Glossary

Computer Assisted Interviewing (CAI)

Survey Interchange Standard (Triple-S)

Questionnaire Definition Language (QDL)

Simple Survey System (SSS)

Structured Questionnaire Building Language (SQBL)

Survey State Model (SSM)

Computer-Assisted Survey Execution System (CASES)

XML Schema Definition (XSD)

Schematron (SCH)

JavaScript Object Notation (JSON)

Reverse Polish Notation (RPN)