Survey State Model (SSM)
Authoring language of electronic questionnaires

Jose Lloret - jose@pexel.co.uk

July 9, 2015

1 Introduction

Survey research is the systematic collection of information from individuals and organisations to address research and/or business objectives. This social science is formed by five stages (see Figure 1) and allows the identification of market demands, detection of opportunities or customer preferences understanding among others. This manual is in-

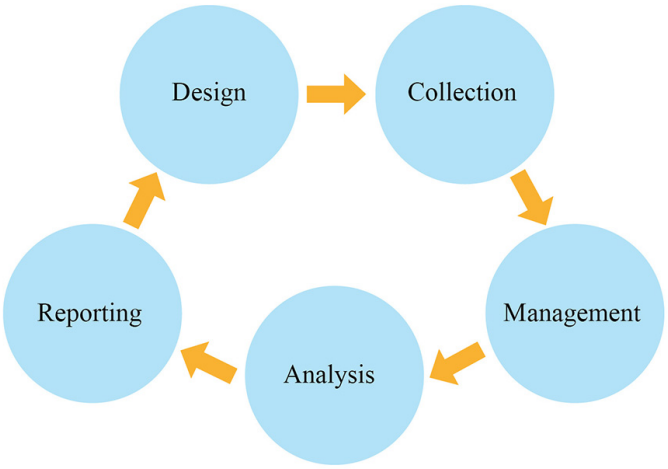

Figure 1: 5 stages process for surveys

tended to describe the most frequent features from surveys through Survey State Model (SSM). This language follows a state-transition approach to describe the flow of questionnaires and improves the existing Extensible Markup Language (XML) languages to describe questionnaire's constructs by adding personalisation features.

2 Survey metadata

Survey metadata section of SSM permits describing the name, description and date for a survey. Listings 1 defines an example of metadata according to SSM.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        <name>01</name>
        <description>Description of the survey</description>
        <date>2015-07-01</date>
    </survey>
    ...
</ssm>
```

Listing 1: Metadata for surveys

3 Survey content

Survey content is structured with sections. A section allow grouping questions and permits reusing it several times when it is referenced in a composite state (see Listing 2). There are five types of questions that can described in SSM: intro, single, multiple, open and grid, following subsections will explain them in detail.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            ...
        </section>
        <section id="...">
            ...
        </section>
        <section id="n">
            ...
        </section>
    </content>
    ...
</ssm>
```

Listing 2: Sections structure

3.1 Intro question

Intro question not only becomes useful to introduce the respondent to a new section but also can be used for ending a survey with a customised message. Listing 3 defines an intro question whose name is INF and there is a label defined in English to describe the content that would appear during the collection stage.

```
<intro name="INF">
  <label lang="en">Good morning/afternoon. My name is _ and I am
      calling from PEXEL, an independent
market research agency. We are currently conducting nation wide
    research concerning commercial/industrial carton
sealing needs.
  </label>
</intro>
```

Listing 3: Intro question example

3.2 Single question

Single question asks the respondent anything in which one and only one response from the set can be chosen. Listing 4 represents a single question whose name is Q0, contains one label in English and four possible responses, one closed and three open.

```
<single name="Q0">
    <label lang="en">Question to ask</label>
    <close code="1">
        <label lang="en">response 1</label>
    </close>
    <open code="2">
        <label lang="en">Other verbatim response 2</label>
        <string>
            <value></value>
        </string>
    </open>
    <open code="3">
        <label lang="en">Other integer response 3</label>
        <integer>
            <min>1</min>
            <max>6</max>
            <value>5</value>
        </integer>
    </open>
    <open code="4">
        <label lang="en">Other decimal response 4</label>
        <decimal>
            <value></value>
        </decimal>
    </open>
</single>
```

Listing 4: Single question example

A single question could be ”What is your favourite colour?, ”How many times do you read newspapers per week?” or ”What percentage do you expect to increase on sales this year? where verbatim responses from the respondent could help to get accurate results and respondent’s opinion, as such SSM offers the open response types string, integer or decimal (e.g. codes 2, 3 and 4 respectively) to satisfy this level of personalisation.

3.3 Multiple question

Multiple question permits the respondent to choose one or more responses from the set. Listing 5 asks expectations from work and there is a minimum and maximum limit to choose (e.g. 1 and 3 respectively) from the set of 5 responses. Also, each response can be exclusive meaning that is not affected by the limit.

```
<multiple name="Q0">
    <label lang="en">What are you looking for from your work?</
        label>
    <limit>
        <min>1</min>
        <max>3</max>
    </limit>
    <close code="money">
        <label lang="en">Money</label>
    </close>
    <close code="motivation">
        <label lang="en">Motivation</label>
    </close>
    <close code="development">
        <label lang="en">Development</label>
    </close>
    <close code="leadership">
        <label lang="en">Leadership</label>
    </close>
    <close code="dontknow">
        <label lang="en">Don't know</label>
        <exclusive value="true"/>
    </close>
</multiple>
```

Listing 5: Multiple question example

Exclusive response feature becomes useful for responses such as don't know, refuse to ask or prefer not reveal. Similarly to the single question, open response types can be defined in multiple question type too.

3.4 Open question

Open question allows the respondent answering the question freely. There are three types defined in SSM to address survey requirements. One is string or verbatim question addressed to obtain respondent views and becoming extremely useful for sentiment analysis techniques of computing. Listing 6 asks for feedback regarding a training session where respondent's opinion may lead to improve things.

```
<open name="Q12">
    <label lang="en">We'd love to hear any feedback you have on
        the training and application
        processes as a whole, and if there are any improvements
            you can think of.</label>
    <string>
        <value></value>
```

```
    </string>
</open>
```

Listing 6: Verbatim question example

Other type is numeric integer, useful for survey designers if they do not know the possible closed responses. Listing 7 asks the number of people living in a property. Not only min and max limits could be imposed to avoid unexpected responses but also default value response can help when collecting data survey stage takes place.

```
<open name="Q2">
    <label lang="en">How many people live in the main property? (
        including yourself)</label>
    <integer>
        <min></min>
        <max></max>
        <value></value>
    </integer>
</open>
```

Listing 7: Numeric integer question example

Finally, numeric decimal, rarely used, can help to get accurate numbers. Listing 8 represents a decimal open question.

```
<open name="Q20">
    <label lang="en">What was the occupancy rate overall for last
        year (2009)?</label>
    <decimal>
        <value></value>
    </decimal>
</open>
```

Listing 8: Numeric decimal question example

3.5 Grid question

Grid question allows asking more than one thing in the same question. There are three types of grid supported in SSM permitting the respondent to choose a single option in each row, allowing them to tick multiple for each row or text boxes for entering numeric responses.
Grid single (Listing 9) asks three statements where only one response per row is permitted (Strongly disagree, disagree, neither, agree and strongly agree).

```
<grid name="Q1">
    <label lang="en">Please state to what extend do you agree with
         these statements about nutritional
        supplements:</label>
    <single>
        <rows>
            <close code="b">
```

```
                <label lang="en">Nutritional supplements are
                    relevant to maintain well-being in hectic
                    lifestyles.</label>
            </close>
            <close code="c">
                <label lang="en">Nutritional supplements can be
                    effective in weight management.</label>
            </close>
            <close code="d">
                <label lang="en">Nutritional supplements help to
                    achieve better sport performance.</label>
            </close>
        </rows>
        <columns>
            <close code="1">
                <label lang="en">Strongly disagree</label>
            </close>
            <close code="2">
                <label lang="en">Disagree</label>
            </close>
            <close code="3">
                <label lang="en">Neither</label>
            </close>
            <close code="4">
                <label lang="en">Agree</label>
            </close>
            <close code="5">
                <label lang="en">Strongly agree</label>
            </close>
        </columns>
    </single>
</grid>
```

Listing 9: Grid single question example

Multiple grid (Listing 10) asks for each type of soft drink what features the respondent feels where there are no restrictions to pick up (Fun, Sexy and Masculine) for each soft drink.

```
<grid name="Q1">
    <label lang="en">Which of these are</label>
    <multiple>
        <rows>
            <close code="01">
                <label lang="en">Coke</label>
            </close>
            <close code="02">
                <label lang="en">Pepsi</label>
            </close>
            <close code="03">
                <label lang="en">Fanta</label>
            </close>
        </rows>
        <columns>
            <close code="01">
                <label lang="en">Fun</label>
```

```
            </close>
            <close code="02">
                <label lang="en">Sexy</label>
            </close>
            <close code="03">
                <label lang="en">Masculine</label>
            </close>
            <close code="99">
                <label lang="en">Do not know</label>
                <exclusive value="true"/>
            </close>
        </columns>
    </multiple>
</grid>
```

Listing 10: Grid multiple question example

Numeric grid (Listing 11) asks rating the performance for three statements in each company X, Y, Z provided. As the reader may note, it is needed to fill every row and column with a number from 1 to 5. This type of question should be shown to the respondent as a text box for each row/column.

```
<grid name="Q9">
    <label lang="en">How would you rate the performance of the
        following nutritional supplements
        providers in the following areas? (Scale of 1-5, where 1=
            very poor performance and 5=very good
        performance+Pass)</label>
    <numeric>
        <rows>
            <close code="1">
                <label lang="en">They offer good-quality products.
                    </label>
            </close>
            <close code="2">
                <label lang="en">They are a leading producer of
                    nutritional supplements</label>
            </close>
            <close code="3">
                <label lang="en">They are a scientific company</
                    label>
            </close>
        </rows>
        <columns>
            <close code="1">
                <label lang="en">X</label>
            </close>
            <close code="2">
                <label lang="en">Y</label>
            </close>
            <close code="3">
                <label lang="en">Z</label>
            </close>
        </columns>
    </numeric>
</grid>
```

Listing 11: Grid numeric question example

Note that each row of a grid question can be defined as closed or open (string, integer or decimal) whereas the columns are only closed. Although this could constitute a limit for the language thanks to the transposed feature present in grid questions (rows seen as columns and columns seen as rows) allows the respondent to see open responses as they were in the columns but really they would not be.

4 Field

Field section defines the variables that are shared across sections. Fiels are widely used for loop and computation states in the routing section of SSM. Listings 12 describes the four types of variables available in SSM.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        ...
    </content>
    <field>
        <decimal id="decimalVariable" value="0.0"/>
        <integer id="integerVariable" value="0"/>
        <string id="stringVariable" value="Hello World!"/>
        <iterator id="p2_iterator"/>
    </field>
    <routing>
        ...
    </routing>
</ssm>
```

Listing 12: Field examples

Every variable defined requires an id and value. The value must be according to the type declared, as such the example values are 0.0, 0, ”Hello World! matching with the types decimal, integer and string respectively. *Note that iterator variable does not have a value defined since its value is modified through for states of SSM routing.*

5 Survey routing

In order to describe routing for surveys it is used a state-transition approach. As such, the routing part is composed by statemodels which must refer to sections defined in the content part of SSM and an entrypoint marking the starting point of execution, see Listings 13. Every statemodel contains states that can be *simple*, *composite* or *pseudo*.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            ...
        </section>
        <section id="...">

        </section>
        <section id="n">
            ...
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        <statemodel ref="1">
            ...
        </statemodel>
        <statemodel ref="...">
            ...
        </statemodel>
        <statemodel ref="n">
            ...
        </statemodel>
        <entrypoint><!-- marks the beginning for the routing -->
            ...
        </entrypoint>
    </routing>
</ssm>
```

Listing 13: Routing example

Below, there are explained the different states available to describe routing features:

5.1 Simple

Simple states are those addressed to interact with the respondent because they are able to retrieve questions that are defined in the content part. For instance, Listings 14

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            <label lang="en">Default Section</label>
            <single name="Q0">
                <label lang="en">Can we ask a set of questions? It
                    only takes 5 minutes or even less.</label>
```

```
                <close code="yes">
                    <label lang="en">Yes</label>
                </close>
                <close code="no">
                    <label lang="en">No</label>
                </close>
            </single>
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        <statemodel ref="1">
            <state id="s0">
                <variable ref="Q0" />
                <transition target="p0"/>
            </state>
        </statemodel>
    </routing>
</ssm>
```

Listing 14: Simple state example

contains a simple state (s0) which contains one variable whose reference is a single question defined in the section 1 of the content. When this state is left, that is the respondent press forward or next, the state reached should be p0.

5.2 Composite

Composite states are able to reuse other statemodel. Listings 15 defines the state c0 as composite which embeds the statemodel 1 in the statemodel 0, that is the statemodel 0 executes the behaviour from the statemodel 1 when c0 is reached.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="0">
            ...
        </section>
        <section id="1">
            ...
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        <statemodel ref="0">
            ...
            <state id="c0"><!-- COMPOSITE STATE -->
                <include statemodel="1"/>
```

```
                <transition target="sink0"/>
            </state>
            ...
        </statemodel>
        <statemodel ref="1">
            ...
        </statemodel>
    </routing>
</ssm>
```

Listing 15: Composite state example

Composite states become useful when they are combined with For states (see Section 5.5).

5.3 If

If states are used to describe flow through the survey, in other words, what path to take whether the expression is satisfied or not. Listings 16 describes a filter to decide whether going to sink0 state or s2. The logical expression (written in postfix mode) would be true if the respondent selected ”no” from Q0. Bear in mind that not only every variable referenced in the expression must be defined in the section pointed by the statemodel but also the target transitions have to be states specified in the statemodel, otherwise the XML file is not valid according to SSM.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            <label lang="en">Default Section</label>
            <single name="Q0">
                <label lang="en">Can we ask a set of questions? It
                     only takes 5 minutes or even less.</label>
                <close code="yes">
                    <label lang="en">Yes</label>
                </close>
                <close code="no">
                    <label lang="en">No</label>
                </close>
            </single>
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        <statemodel ref="1">
            ...
            <state id="p0"><!-- PSEUDO STATE IF -->
                <if>
                    <condition>
```

```
                        <variable ref="Q0"/>
                        <constant type="string" value="no"/>
                        <operator name="IS_SEL"/>
                    </condition>
                    <then>
                        <transition target="sink0"/>
                    </then>
                    <else>
                        <transition target="s2"/>
                    </else>
                </if>
            </state>
            <state id="sink0">
                ...
            </state>
            <state id="s2">
                ...
            </state>
        </statemodel>
    </routing>
</ssm>
```

Listing 16: If state example

5.4 Check

Check states are addressed to validate any inconsistence stopping the flow if the logical expression is not valid. Check state can be either *warning* or *error* giving the possibility to continue responding the survey or stopping until the inconsistence is solved respectively. Listings 17 defines an error check which validates whether never was selected in Q3a or not.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="PERSON">
            <label lang="en">Person</label>
            <single name="Q3a"><!-- Marital status question -->
                ...
            </single>
            <open name="Q3b"><!-- Age question -->
                ...
            </open>
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        <statemodel ref="PERSON">
            <state id="p0"><!-- PSEUDO STATE IF-->
                <if>
```

```
                    <condition><!-- Q3b < 18 -->
                        <variable ref="Q3b"/>
                        <constant type="integer" value="18"/>
                        <operator name="LT"/>
                    </condition>
                    <then>
                        <transition target="p1"/>
                    </then>
                    <else>
                        <transition target="s2"/>
                    </else>
                </if>
            </state>
            <state id="p1"><!-- PSEUDO STATE, CHECK -->
                <check type="error">
                    <condition>
                        <variable ref="Q3a"/>
                        <constant type="string" value="never"/>
                        <operator name="IS_SEL"/>
                    </condition>
                    <label lang="en">People younger than 18 have
                        never been married.</label>
                </check>
                <transition target="s2"/>
            </state>
            <state id="s2">
                ...
            </state>
        </statemodel>
    </routing>
</ssm>
```

Listing 17: Check state example

The above check only validates that never is selected in Q3a but it is needed a previous if state to decide whether reaching the check or not according to the respondent age, otherwise the check state would become false for any response different to never in Q3 without taking into account respondent's age.

5.5 For

For state is used to indicate that a statemodel has to be repeated a number of times, in other words, to repeat the questions of a section as many times as the expression is satisfied. There are three types of iterations: *range*, *expression_list* or *list*. The range iterator accepts integer operands coming from constants, variables or expressions returning an integer operand. Listings 18 describes a for state whose iterator is range.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
```

```
            <label lang="en">Default Section</label>
            ...
            <open name="Q2">
                <label lang="en">How many people live in the main
                    property? (including yourself)</label>
                <integer>
                    <min></min>
                    <max></max>
                    <value></value>
                </integer>
            </open>
            ...
        </section>
        <section id="2">
            ...
        </section>
    </content>
    <field>
        <iterator id="p2_iterator"/>
    </field>
    <routing>
        <statemodel ref="1">
            ...
            <state id="p2"><!-- PSEUDO FOR RANGE STATE -->
                <for>
                    <field ref="p2_iterator"/>
                    <in>
                        <range>
                            <start><constant type="integer" value=
                                "0"/></start>
                            <end><variable ref="Q2"/></end>
                            <step><constant type="integer" value="
                                1"/></step>
                        </range>
                    </in>
                    <transition target="c0"/>
                </for>
                <transition target="p3"/>
            </state>
            <state id="c0"><!-- COMPOSITE STATE -->
                <include statemodel="2"/>
            </state>
            ...
        </statemodel>
        <statemodel ref="2">
            ...
        </statemodel>
    </routing>
</ssm>
```

Listing 18: Range for state example

In order to describe a range iterator it is needed:

- *field* referring to an iterator variable defined in the field section. This variable can be used for text piping for questions since the same question can be repeated

many times so knowing in what position is the iterator permits its unique identification.

- *range start* marks the beginning of the iterator. Usually the start element is a constant integer 0.
- *range end* sets the ending of the iterator. The example provided refers to an open integer question.
- *range step* indicates the increments to perform in each iteration. Usually is a constant integer 1.
- *true transition* requires a composite state without transition to avoid cycle loops. The for state will create a new statemodel for each iteration of the loop.
- *false transition* is the successor state when the iteration loop ends.

The expression_list iterator accepts expressions which return a list such as selected/unselected/all choices from single/multiple question(s). Listings 19 describes a for state whose iterator is an expr_list.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            <label lang="en">Default Section</label>
            ...
            <multiple name="Q7">
                <label lang="en">Which of the following do you
                    like to watch?</label>
                <close code="1">
                    <label lang="en">Business / financial</label>
                </close>
                <close code="2">
                    <label lang="en">Comedy</label>
                </close>
                <close code="3">
                    <label lang="en">Current affairs</label>
                </close>
                ...
                <open code="13">
                    <label lang="en">Other (please specify):</
                        label>
                    <string>
                        <value></value>
                    </string>
                </open>
                <close code="14">
                    <label lang="en">None</label>
                    <exclusive value="true"/>
                </close>
            </multiple>
```

```
            ...
        </section>
        <section id="2">
            ...
        </section>
    </content>
    <field>
        <iterator id="p7_iterator"/>
    </field>
    <routing>
        <statemodel ref="1">
            ...
            <state id="p7">
                <for>
                    <field ref="p7_iterator"/>
                    <in>
                        <expr_list>
                            <variable ref="Q7"/>
                            <operator name="SEL"/>
                        </expr_list>
                        <randomising>
                            <all present="4"/>
                        </randomising>
                    </in>
                    <transition target="c2"/>
                </for>
                <transition target="sink0"/>
            </state>
            <state id="c2">
                <include statemodel="2"/>
            </state>
            ...
        </statemodel>
        <statemodel ref="2">
            ...
        </statemodel>
    </routing>
</ssm>
```

Listing 19: Expression_list for state example

In order to describe a expr_list iterator it is needed:

- *field* referring to an iterator variable defined in the field section. This variable can be used for text piping for questions since the same question can be repeated many times so knowing in what position is the iterator permits its unique identification.
- *expr_list* permits defining any kind of expression which returns a list. The example provided is retrieving the responses selected in Q7 to create the iterator but other combinations such as unselected in Q7, all choices in Q7, selected in Qx union unselected in Qy and so on are possible. *Note, every variable referred in the expression list must be a defined single or multiple question in the section pointed by the statemodel.*

- *true transition* requires a composite state without transition to avoid cycle loops. The for state will create a new statemodel for each iteration of the loop.
- *false transition* is the successor state when the iteration ends.

Finally the list iterator refers to a list variable defined in the field section. *Note that this type of iterator is still not implemented in the collection stage of surveys.*

5.6 Computation

Computation state can be used to specify intermediate operations that are carried out during the execution of the survey. Basically, this state performs an arithmetical operation over a variable defined in the Field section. Listings 20 adds one to the working variable.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            ...
        </section>
        ...
    </content>
    <field>
        ...
    </field>
    <routing>
        <statemodel ref="1">
            ...
            <state id="p3"><!-- PSEUDO STATE, COMPUTATION -->
                <computation ref="working">
                    <assignment>
                        <variable ref="working"/>
                        <constant type="integer" value="1"/>
                        <operator name="ADD"/>
                    </assignment>
                </computation>
                <transition target="sink0"/>
            </state>
            ...
        </statemodel>
        ...
        <entrypoint>
            ...
        </entrypoint>
    </routing>
</ssm>
```

Listing 20: Computation state example

Note, the variable to refer only can be a variable from the Field section. In this case, working is an integer variable.

5.7 Sink

Sink states are aimed to describe the ending of a statemodel, that is the ending of a section. For instance, Listings 21 contains an if state which switches to sink0 if the expression becomes true meaning that sink0 would be the last state to execute in the statemodel 1.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            ...
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        <statemodel ref="1">
            ...
            <state id="p0">
                <if>
                    <condition>
                        <variable ref="Q0"/>
                        <constant type="string" value="no"/>
                        <operator name="IS_SEL"/>
                    </condition>
                    <then>
                        <transition target="sink0"/>
                    </then>
                    <else>
                        <transition target="s2"/>
                    </else>
                </if>
            </state>
            <state id="sink0"><!-- SINK STATE -->
                <sink/>
            </state>
            <state id="s2">
                ...
            </state>
        </statemodel>
    </routing>
</ssm>
```

Listing 21: Sink state example

5.8 Terminate

Terminate states are used to describe the ending of the routing, that is the ending of a survey. They become useful for sections that have screening questions to filter whether

the respondent is appropiate to continue or not. For example, Listings 22 terminates the survey (see line 54) if don't know is selected in Q4, otherwise finishes the statemodel 1 and start executing statemodel 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            <label lang="en">Current bonding use</label>
            <single name="Q4">
                <label lang="en">Where do you currently source
                    bonds from?</label>
                <close code="1">
                    <label lang="en">Your bank</label>
                </close>
                <open code="2">
                    <label lang="en">Other source</label>
                    <string>
                        <value></value>
                    </string>
                </open>
                <close code="99">
                    <label lang="en">Do not know</label>
                </close>
            </single>
        </section>
        <section id="2">
            ...
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        <statemodel ref="1">
            <source id="s4" /><!-- First state to execute -->
            <state id="s4">
                <variable ref="Q4" />
                <transition target="p4" />
            </state>
            <state id="p0">
                <if>
                    <condition>
                        <variable ref="Q4"/>
                        <constant type="string" value="99"/>
                        <operator name="IS_SEL"/>
                    </condition>
                    <then>
                        <transition target="terminate"/>
                    </then>
                    <else>
                        <transition target="sink0"/>
                    </else>
                </if>
```

```
            </state>
            <state id="terminate">  <!-- TERMINATE STATE-->
                <terminate></terminate>
            </state>
            <state id="sink0">
                <sink/>
            </state>
        </statemodel>
        <statemodel ref="2">
            ...
        </statemodel>
        <entrypoint>
            <source id="c1"/><!-- First state to execute -->
            <state id="c1">
                <include statemodel="1"/>
                <transition target="c2"/>
            </state>
            <state id="c1">
                <include statemodel="2"/>
                <transition target="sink0"/>
            </state>
            <state id="sink0">
                <sink/>
            </state>
        </entrypoint>
    </routing>
</ssm>
```

Listing 22: Terminate state example

Note in the above example the flow of the survey is defined through the entrypoint where first is executed c1, after c2 (if terminate is not reached in statemodel 1) and finally sink0.

6 Survey personalisation

Personalisation section defines the dynamic behaviour of surveys such as Piping, randomising or rotating. These features make feel the respondent as the survey was created exclusively for her/him because the content features like the question text is changing according to her/his previous responses. Below there are explained the constructs that SSM offers to customise the survey for each respondent.

6.1 Piping

Piping allows the retrieval of an answer from a previous question as part of the text for another (*text piping*) or the automatic generation of responses for single/multiple questions based on an expression returning a list (*response piping*). Listings 23 contains an example using text piping. In this instance there is a piping element referring the section 1 and two pipes. The pipe0 retrieves the responses selected in Q0 and the pipe4 gets all the responses (selected/unselected) from Q1. These pipes are referred in the Q1 text so the content will vary according to the answers obtained from each respondent.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            <label lang="en">Section one</label>
            <multiple name="Q0">
                <label lang="en">What colours do you like?</label>
                <close code="red">
                    <label lang="en">Red</label>
                </close>
                <close code="white">
                    <label lang="en">White</label>
                </close>
                <close code="green">
                    <label lang="en">Green</label>
                </close>
                <close code="yellow">
                    <label lang="en">Yellow</label>
                </close>
            </multiple>
            <open name="Q1">
                <label lang="en">Why do you like the colours <pipe
                     ref="pipe0"/> of <pipe ref="pipe4"/>?</label>
                <string>
                    <value></value>
                </string>
            </open>
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        ...
    </routing>
    <personalisation>
        <piping ref="1">
            <pipe id="pipe0">
                <variable ref="Q0"/>
                <operator name="SEL"/>
            </pipe>
            <pipe id="pipe4">
                <variable ref="Q0"/>
                <operator name="VALUEOF"/>
            </pipe>
        </piping>
    </personalisation>
</ssm>
```

Listing 23: Text piping example

Listings 24 has an instance using response piping for multiple questions Q7b and Q7c by referring to pipe0 which gets the responses unselected from the single question

Q7a and pipe1 that joins the selected responses from Q7a and Q7b respectively. As the reader may note, the set of responses for Q7b and Q7c will vary according to the selected choices from Q7a.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            <label lang="en">Section one</label>
            <single name="Q7a">
                <label lang="en">Which fruits are you aware of? [
                    FIRST SPONTANEOUS MENTION]</label>
                <randomising>
                    <all/>
                </randomising>
                <close code="apple">
                    <label lang="en">Apple</label>
                </close>
                <close code="banana">
                    <label lang="en">Banana</label>
                </close>
                <close code="blueberry">
                    <label lang="en">Blueberry</label>
                </close>
                <close code="grapefruit">
                    <label lang="en">Grapefruit</label>
                </close>
                <close code="melon">
                    <label lang="en">Melon</label>
                </close>
                <close code="orange">
                    <label lang="en">Orange</label>
                </close>
                <close code="pear">
                    <label lang="en">Pear</label>
                </close>
                <close code="pineapple">
                    <label lang="en">Pineapple</label>
                    <label lang="es">Pi a</label>
                </close>
            </single>
            <multiple name="Q7b">
                <label lang="en">Which fruits are you aware of? [
                    OTHER SPONTANEOUS MENTIONS]</label>
                <pipe ref="pipe0"/>
            </multiple>
            <multiple name="Q7c">
                <label lang="en">Which of the following fruits do
                    you eat? [MUST BE SELECTED AT Q7a/Q7b]</label>
                <pipe ref="pipe1"/>
            </multiple>
        </section>
    </content>
    <field>
```

```
        ...
    </field>
    <routing>
        ...
    </routing>
    <personalisation>
        <piping ref="1">
            <pipe id="pipe0">
                <variable ref="Q7a"/>
                <operator name="UNSEL"/>
            </pipe>
            <pipe id="pipe1">
                <variable ref="Q7a"/>
                <operator name="SEL"/>
                <variable ref="Q7b"/>
                <operator name="SEL"/>
                <operator name="UNION"/>
            </pipe>
        </piping>
    </personalisation>
</ssm>
```

Listing 24: Response piping example

6.2 Randomising - Rotating

Randomising and Rotating features are available for single and multiple questions allowing the responses reorder during the collection stage of surveys. There are two modes of reordering for randomising/rotating approaches:

- *all* mode reorders all the responses for a question. This mode contains an optional attribute *present* which describes the number of responses to show to the respondent after randomising/rotating. Listings 25 will reorder the responses of Qx randomly and after will present just 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            <label lang="en">Section one</label>
            <multiple name="Qx">
                <label lang="en">What are your favourite items
                    ?</label>
                <randomising>
                    <all present="3"/>
                </randomising>
                <close code="1">
                    <label lang="en">Response 1</label>
                </close>
                <close code="2">
                    <label lang="en">Response 2</label>
```

```
                </close>
                <close code="3">
                    <label lang="en">Response 3</label>
                </close>
                <close code="4">
                    <label lang="en">Response 4</label>
                </close>
                <close code="5">
                    <label lang="en">Response 5</label>
                </close>
                <close code="6">
                    <label lang="en">Response 6</label>
                </close>
                <close code="99">
                    <label lang="en">None of these</label>
                </close>
            </multiple>
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        ...
    </routing>
    <personalisation>
        ...
    </personalisation>
</ssm>
```

Listing 25: All mode example for randomising responses

- *subset* mode reorders the responses specified by code and keeps those not mentioned. This mode becomes useful when there are responses such as don't know, none, refuse to respond since their order is not usually altered. Listings 26 will reorder the responses 1, 2, 3, 4, 5 and 6 of Qx through rotation but will keep the order defined for the response 99.

```
<?xml version="1.0" encoding="UTF-8"?>
<ssm>
    <survey>
        ...
    </survey>
    <content>
        <section id="1">
            <label lang="en">Section one</label>
            <multiple name="Qx">
                <label lang="en">What are your favourite items
                    ?</label>
                <rotating>
                    <subset>
                        <code ref="1"/>
                        <code ref="2"/>
                        <code ref="3"/>
                        <code ref="4"/>
```

```
                        <code ref="5"/>
                        <code ref="6"/>
                    </subset>
                </rotating>
                <close code="1">
                    <label lang="en">Response 1</label>
                </close>
                <close code="2">
                    <label lang="en">Response 2</label>
                </close>
                <close code="3">
                    <label lang="en">Response 3</label>
                </close>
                <close code="4">
                    <label lang="en">Response 4</label>
                </close>
                <close code="5">
                    <label lang="en">Response 5</label>
                </close>
                <close code="6">
                    <label lang="en">Response 6</label>
                </close>
                <close code="99">
                    <label lang="en">None of these</label>
                </close>
            </multiple>
        </section>
    </content>
    <field>
        ...
    </field>
    <routing>
        ...
    </routing>
    <personalisation>
        ...
    </personalisation>
</ssm>
```

Listing 26: Subset mode example for rotating responses

Note that randomising and rotating constructs are only available for single and multiple questions.