SLang: A Domain-specific Language for Survey Questionnaires

Luciane C. Araújo[1], Marco A. Casanova[2][a], Luiz André P. P. Leme[3][b] and Antonio L. Furtado[2]

[1]Brazilian Institute of Geography and Statistics, Rio de Janeiro, Brazil
[2]Dept. Informatics, PUC-Rio, Rua Marquês de São Vicente, 225, Rio de Janeiro, Brazil
[3]Institute of Computing, Federal Fluminense University, Niterói, Brazil

Keywords: Survey Questionnaires, Domain-specific Languages, Statistical Surveys.

Abstract: The use of surveys permeates the economy, ranging from customer satisfaction measurement to tracking global economic trends. At the core of the survey process lies the codification of questionnaires, which vary from simple lists of questions to complex forms that include validations, computation of derived data, use of triggers to guarantee consistency, and dynamic creation of objects of interest. Questionnaire specification is part of what is called survey metadata and is a key factor for the quality of the data collected and of the survey itself. In this context, the paper first introduces a comprehensive complex questionnaire model. Then, based on the model, it proposes a prototype domain-specific language (DSL) for modeling complex questionnaires, called SLang. The paper also describes a prototype implementation of SLang and an evaluation of the usefulness of the language in practical scenarios.

1 INTRODUCTION

A survey is a systematic method for collecting data about (a sample of) entities to construct quantitative descriptors of the attributes of a larger population of which the entities are members. The usage of a questionnaire is by far the most common data collection strategy (Saris and Galhofer, 2014).

A superficial look at what is a survey questionnaire might lead to the conclusion that this is a simple problem. However, underestimating questionnaire design complexity is a common flaw that can have a direct impact on survey quality (Saris and Galhofer, 2014).For instance, large surveys, such as the Brazilian Agricultural Census (IBGE, 2017), have questionnaires that involve hundreds of quantitative measures, with intricate rules that define which questions should be part of the questionnaire, in which order those questions should be presented, as well as complex validations to be performed as each answer is registered.

This paper addresses the *complex questionnaire design problem*, that is, the problem of designing questionnaires that involve the computation of derived data, the use of rules and triggers to guarantee

consistency, and the dynamic creation of objects of interest. A solution to this problem should reduce the gap between survey domain experts and software developers, improve reuse, eliminate redundancy, and minimize rework.

To solve this problem, the paper first introduces a comprehensive complex questionnaire model. The model defines the questionnaire structure, as well as structures for defining consistency and integrity rules that will be applied during data collection. The model also covers the behavioral aspects of the questionnaire, that is, how to navigate through the questions.

Next, based on the model, it proposes a prototype domain-specific language (DSL) for modeling complex questionnaires, called SLang. SLang is a textual DSL, developed using the MPS projectional language workbench, that addresses questionnaire metadata modeling. SLang does not contemplate the questionnaire presentation and the user interface, as well as general aspects related to data collection systems, such as security and data analysis. The paper also describes a prototype implementation of a data collection application that is compatible with Slang,

[a] https://orcid.org/0000-0003-0765-9636
[b] https://orcid.org/0000-0001-0014-7256

and an evaluation of the usefulness of Slang in practical scenarios.

The abstract syntax of SLang reflects the questionnaire model. The concrete syntax defines how SLang will appear to the users and was influenced by interviews with domain experts. In this paper, to facilitate reading, the concrete syntax is presented with the help of examples.

Experiments with two real-world questionnaires were also conducted to evaluate the usefulness of SLang. The questionnaire specifications were encoded in SLang, using a prototype implementation of the language. This part of the experiments evaluated the expressiveness of SLang. Then, mock-up surveys were run, using the encoded questionnaires, on top of a survey environment, called SInterviewer. This second part of the experiments validated the behavioral aspects of SLang. The full details can be found in (Araújo, 2019).

The rest of this paper is structured as follows. Section 2 discusses related work. Section 3 introduces the questionnaire model. Section 4 covers the design of SLang. Section 5 describes the implementation details and experiments with SLang. Section 6 presents conclusions and suggestions for future work.

2 RELATED WORK

Survey Models. Survey models have been proposed, some of which are part of standards, developed and maintained by entities related to the statistical survey community. Among standards we have: the Generic Statistical Information Model (GSIM), the Data Documentation Initiative (DDI), and the Statistical Data and Metadata eXchange (SDMX) initiative.

GSIM is the first internationally endorsed reference framework of information objects, which enables generic descriptions of the definition, management, and use of data and metadata throughout the statistical production process (GSIM, 2019). DDI is an international standard for describing survey metadata, including questionnaires, statistical data files, and social sciences study-level information (Data Documentation Initiative Alliance, 2019). SDMX aims at standardizing the mechanisms and processes for the exchange of statistical data and metadata among international organizations and their member countries (Statistical Data and Metadata Exchange, 2019).

Since the goal of this paper is to provide a DSL-based approach for questionnaire design, GISM and DDI are relevant inputs for the domain analysis.

Borodin and Zavyalova (2016) described an ontology for survey questionnaires. The proposed ontology models questionnaires with simple navigation rules, but it lacks support for grouping questions by theme, validating answers and creating objects of interest, among other requirements for complex questionnaires. Still, the ontology is a useful reference for modeling questionnaires.

Questionnaire Design Tools. Questionnaire design and data collection tools can be divided into Web-based survey tools and frameworks. Web-based survey tools work by allowing the user to create a questionnaire that will be distributed and answered through the Web. Google Forms (GOOGLE, 2020), SurveyMonkey (SURVEYMONKEY, 2019), Zoho (ZOHO, 2019), and Qualtrics (QUALTRICS, 2019) are examples of Web-based survey tools. CSPro (UNITED STATED CENSUS BUREAU, 2019), developed by the US Census Bureau, Blaise (STATISTICS NETHERLANDS, 2019), developed by Netherlands Statistics and Open Data Kit (OPEN DATA KIT, 2019) are examples of frameworks. They usually include at least a questionnaire design tool and a data collection tool.

Domain-Specific Languages (DSLs). Mernik et al. (2005) define DSLs as languages tailored to a specific application domain that offer substantial gains in expressiveness and ease of use in their domain of application when compared with general-purpose programming languages.

DSLs have been applied to many domains ranging from bioinformatics through robotics, including Web applications, embedded systems, low-level software, control systems, parallel computing, simulation, data-intensive apps, real-time systems, security, education, and networks (NASCIMENTO et al., 2012).

The use of DSLs for questionnaire design is not new. Kim et al. (2015) developed the Survey Design Language (SDL) and its supporting tool, SDLTool. SDL consists of a set of domain-specific visual languages. Each language in the set is designed to model a specific aspect of statistical surveys, providing high-level and low-level modeling facilities capable of matching expert cognitive models for statistical surveys. The SDLTool was the environment that tied together with the different DSLs aspects through visual modeling of survey resources, the design of statistical survey elements, running modelled surveys on target population datasets, and providing visualization support features.

The 2013 Language Workbench Challenge (LWC) assignment consisted of developing a DSL for questionnaires, which had to be rendered in an interactive GUI that reacted to user input and had to store the answers of each question. The questionnaire definition was expected to be validated, detecting errors such as unresolved names and type errors. In addition to basic editor support, participants were expected to modularly develop a styling DSL that could be used to configure the rendering of a questionnaire. The proposed languages offered basic questionnaire functionality but lacked primordial features, such as the possibility of specifying questionnaire navigation, complex validation rules and triggers, among others (ERDWEG et al., 2015).

Zhou, Goto and Cheng (2014) presented QSL, a language to specify questionnaire systems that includes some aspects of questionnaire design. Still, it is not clear whether QSL supports a questionnaire complex logic for navigation flow and question presentation. Also, the dynamic creation of objects of interest is not mentioned. Finally, QSL presents a tight coupling between questionnaire presentation and modeling.

None of the aforementioned DSLs were designed having questionnaire specification as its primary goal. As such, each of them has its shortcomings. SDL completely avoids questionnaire specification having its focus on survey methodology and data analysis. 2013 LWC languages were focused on demonstrating language workbenches potential and provided limited support for questionnaire specification. Finaly, QSL had its focus on e-questionnaire systems with limited support for questionnaire specification.

3 QUESTIONNAIRE MODEL

This section presents the questionnaire model adopted for the DSL design. The model was created after analyzing information about six large scale survey projects. The information included questionnaire specifications, source codes, survey data dictionaries, expert interviews, and survey standards. The resulting questionnaire model is organized into three categories: metadata (data about the collected data), data (actual collected data), and paradata (data about the data collection process). The notation used was adapted from UML class diagrams. A dotted line represents an additional type of relationship that exists only by convention. That means that there is no formal relationship between the two concepts, and different applications use specific conventions when expressing those relationships. The

name of a concept (or class) will be denoted in italics, starting with an upper-case letter, and an instance of a class in lower-case letters, in italics, when it appears for the first time, and in regular font style, otherwise.

3.1 Metadata Structural Model

Metadata concepts define the survey structure, as well as the consistency and integrity rules that will be applied during data collection. Figure 1 presents the metadata structural model.

Survey and *Dictionary* are the root concepts that tie together the metadata concepts structure.

When creating a survey, one first defines one or more o*bjects of interest*, which are the sets of entities that the survey will collect data. For example, demographic censuses usually have at least two objects of interest: houses and people. In most cases, people are further divided into house inhabitants, deceased, and emigrated. Objects of interest are closely related to *entity generators*.

A *theme* helps to organize questions in terms of knowledge clusters. For example, the 2015 PeNSE Survey (PeNSE, 2015) had questions organized in themes such as student socio-economical aspects, family context, eating habits, physical exercising.

A *question set* adds another layer of organization by grouping questions inside a theme. The reason to create a new question set might be, for example, to create a validation rule that involves multiple measurements, or to hold interview instructions that are pertinent to a theme subset.

Questions are linked to a *question set* and are the core of the survey questionnaire. Each question has a group of *question items*.

A *measurement* is linked in a one-to-one relationship with a question item, allowing to create questions with multiple answers and to create derived measurements, which are not related to a question (Saris and Galhofer, 2014). For example, consider a question about telephones in households. In step one, the person answers if the household has a telephone line. In step two, the person answers the number of telephones (if the answer in step one was positive).

Measurements are part of a generalization hierarchy. The first level of this hierarchy consists of three possible data types for measurements: *text* (*TextMeasurement*), numbers (*NumberMeasurement*) and *dates* (*DataMeasurement*). Numbers can be further specialized to support domain and calculated measurements. *DomainMeasurement* extends *NumberMeasurement* by allowing the user to associate a *domain list*, which is a map, where an integer is the key, and a string is the value. A

calculated measurement has a rule that specifies what the value of the measurement should be, based on questionnaire parameters and other measurement values.

Filters allow controlling domain measurements through the use of expressions to define if a domain item should be available to be selected as a measurement value.

A *dictionary* provides a unified view of survey measurements.

Three other concepts pertain to the domain structure through generalizations: *Identifiable*, *EntityGenerator*, and *NavigationItem*.

Navigation items are responsible for specifying the sequence that should be followed when presenting questionnaire questions to the respondent. It is closely related to the questionnaire questions hierarchy. *Navigation items* also allow specifying when to skip a question through the *visibility* attribute. From a structural point of view, they also allow associating questions to validations and triggers, which are used to guarantee measurement consistency.

Validations allow the questionnaire designer to define when an answer is acceptable by specifying expressions. Expressions are evaluated to a Boolean value and used when defining navigation or as survey measurement values, constants, and literals. The types of validations are *informational validations*, *alert validations*, and *error validations*.

Triggers allow the execution of specific actions according to the result of a Boolean expression. One example of action is resetting previous answers.

An *entity generator* is responsible for creating objects of interest. There are two modes for the creation of an object of interest: *theme-based creation* and *question-item-based creation*. An example of the first case is registering household members in a demographic survey. The theme questions work as the specification of the attributes that the generated entity must have, providing the basis for the creation of entity management (creating, updating, and deleting) in a questionnaire. An example of the second case is creating one object of interest for each permanent culture an agricultural establishment keeps. In this example, the question asks the
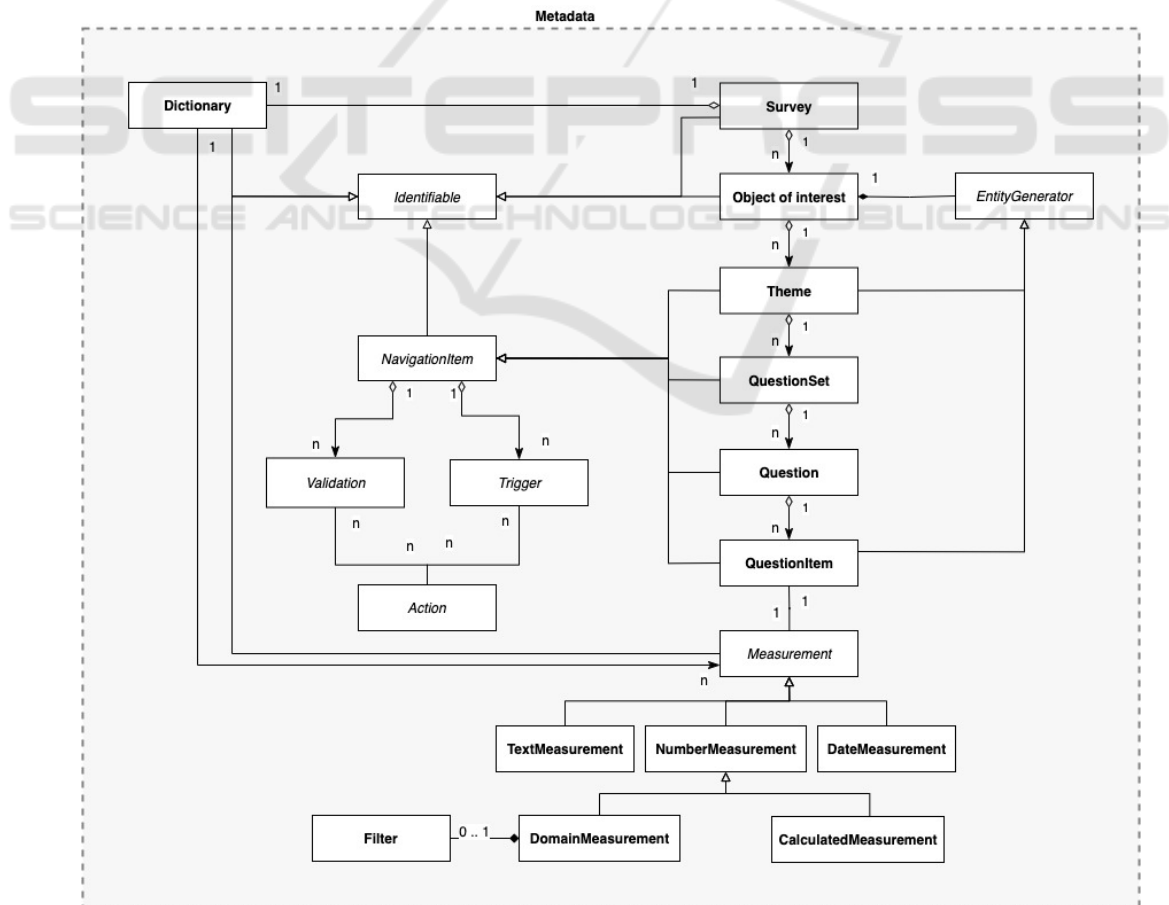

Figure 1: Metadata structural model.

respondent to select from a list of crops (for example, apples, avocados, grapes, oranges, peaches, and pears) that are grown in his establishment. If the respondent selects apples and peaches, two permanent objects of interest will be created: one for apples and another one for peaches. *Entity generators* are designed using different approaches for distinct survey solutions. The most common is the use of generalization, which is the strategy adopted by the questionnaire structural model presented here. This modeling strategy implies multiple inheritance for *Theme* and *QuestionItem* concepts.

Finally, each survey metadata object is an *identifiable*, which means that the object can be uniquely identified in a survey metadata environment. Each identifiable holds a code and a parent code and might be uniquely identified by the combination of its code with its parent code.

3.2 Data and Paradata Structural Model

The questionnaire data follows a very simple structural model, presented on the left side of Figure 2. *Interview* is the root concept and is a container for entities. It also keeps operational information about the questionnaire, such as the last viewed item and how far the respondent has navigated through the questionnaire. An *Entity* is synonymous with an object of interest and holds a set of answers.

An *Answer* holds the value from a specific measurement. The relationship between an answer and a measurement is, in general, made by convention. Hence, the option was to connect those two concepts through a dotted line. *Answer* is specialized into *TextAnswer*, *NumberAnswer*, and *DateAnswer*.

Questionnaire paradata, as shown on the right side of Figure 2, holds two independent concepts: *Summary* and *Log*. A *summary* contains a map of aggregates that can be specified by the user, such as

the average of a group of variables. Aggregates are specified using rules.

Logs are specialized into six classes. A *datetime log* holds specific information about dates, for example, the date on which an interview was initiated; a *time log* keeps track of time spans, such as the time a respondent spent answering a question; a *location log* registers latitude and longitude values; a *status log* keeps track of status changes as respondent is answering the questionnaire; a *value log* registers changes in answers; and a *navigation log* information about the sequence themes, question sets and questions that are presented to the respondent.

3.3 Domain Model Transversal Concepts

Two concepts are transversal in the sense that they are related to many concepts: *Expression* and *Rule*.

Expressions support Boolean, arithmetic, and comparison operators, as well as set and aggregation operators. Valid operands are Boolean values, strings, numbers, measurements, and survey constants.

Rules can be used to generate computed values or in a rule-based *actions*. Computation rules always evaluate to a number and use only arithmetic and aggregation operators. Operands might be numbers, measurements and survey constants.

3.4 Behavioral Aspects

The behavioral aspects of the questionnaire model are closely related to questionnaire navigation and to states of navigation items, which are controlled by validations, triggers, and expressions.

Figure 3 indicates the specialization hierarchy of the behavioral aspects. A *question item* has two *expression* attributes: *mandatory* and *visibility*. Visibility controls if a navigation item should be presented to the user. Mandatory is used during the validation cycle and is a shortcut to an actual
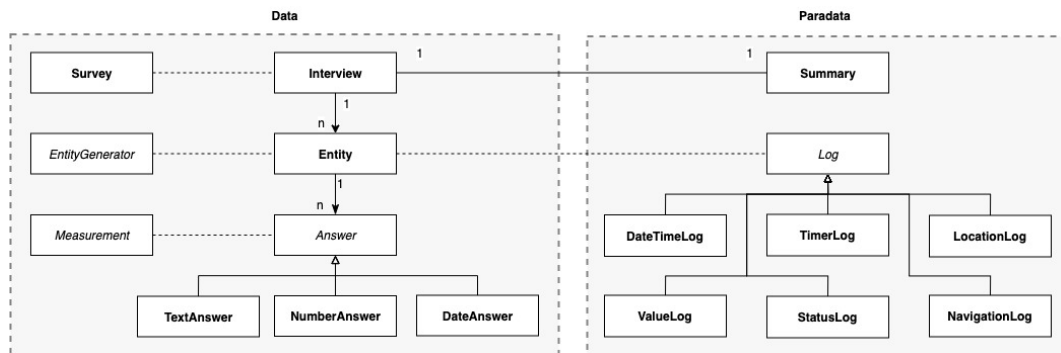

Figure 2: Data and paradata structural model.

validation. The evaluation of the expression associated with the mandatory attribute defines whether the question item should be answered.

Navigation behavior is responsible for the evolution of question and questionnaire states. During an interview, navigation can be *vertical* or *horizontal*, from a starting point. *Horizontal navigation* goes through all nested navigation items, one by one (Figure 4). *Vertical navigation* starts when horizontal navigation reached its end. Accordingly, there are two types of events: horizontal navigation events and vertical navigation events.
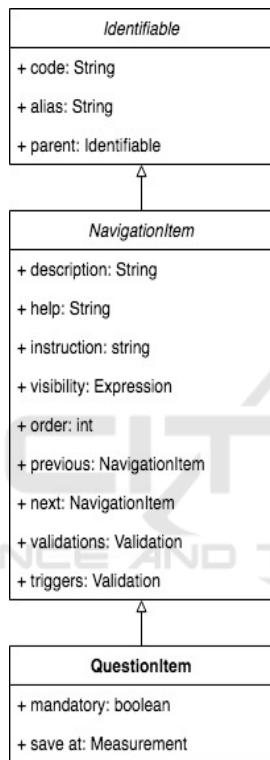
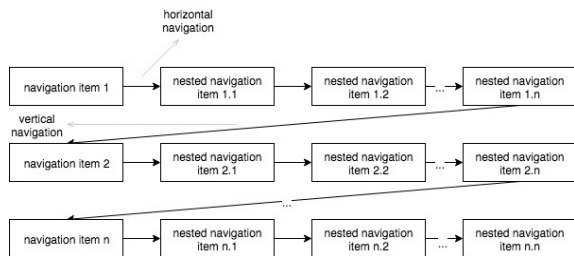

Figure 3: Navigation item generalization.

Figure 4: Navigation sequence through navigations items hierarchy.

As presented in Figure 5, a navigation item can be in four states: NOT ANSWERED, VIEWED, SKIPPED, and ANSWERED. The initial state is always NOT ANSWERED. A not-answered navigation item can either move to the state VIEWED any time it is the next navigation item and its visibility evaluates to true. A not-answered navigation item can also move to the state SKIPPED when it is not visible, and it becomes the next navigation item. A viewed item stays on the VIEWED state until it is valid, when it changes to the ANSWERED state. An answered navigation item can, during navigation, become not visible. In that case, it moves to the SKIPPED state. A skipped navigation item, according to questionnaire navigation logic, become visible and change to the VIEWED state.

Transitions are a consequence of the actions that are executed when a navigation event occurs. When there is a navigation event, the first action to be executed is to run navigation item validations. If the item is not valid, the navigation is aborted. If it is valid, triggers are run, the item is marked as ANSWERED, and the next navigation item is retrieved. If the next navigation item is not visible, it is marked as SKIPPED and the next navigation item is retrieved. When the next navigation item is visible, a check is made to verify if it is answered. If it is, the navigation is completed. If it is not, it is marked as VIEWED, and the navigation is completed.

The *state* of an interview indicates how far a questionnaire has been answered (see Figure 6). The state can be: NOT STARTED, PENDING, and FINALIZED. Interview states are computed based on the states of the navigation item. An interview is marked as NOT ANSWERED, when all its navigation items are marked as NOT ANSWERED, as FINALIZED, when all its navigation items are marked either as SKIPPED or ANSWERED, and PENDING, otherwise.
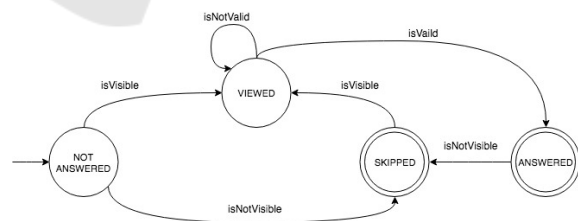

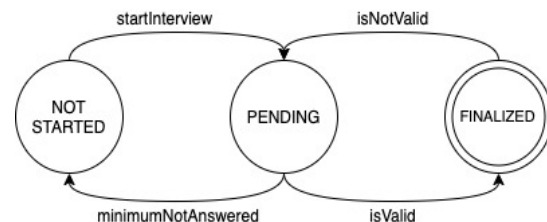Figure 5: Navigation item state diagram.
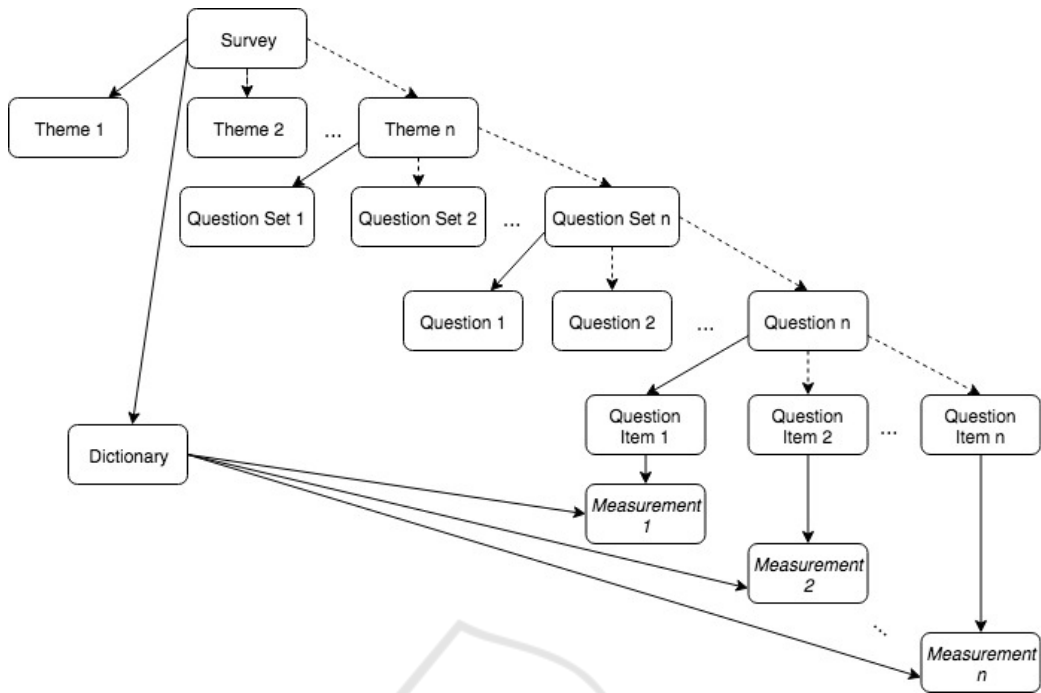

Figure 6: Interview state diagram.

Figure 7: A simplification the Survey Abstract Syntax Tree.
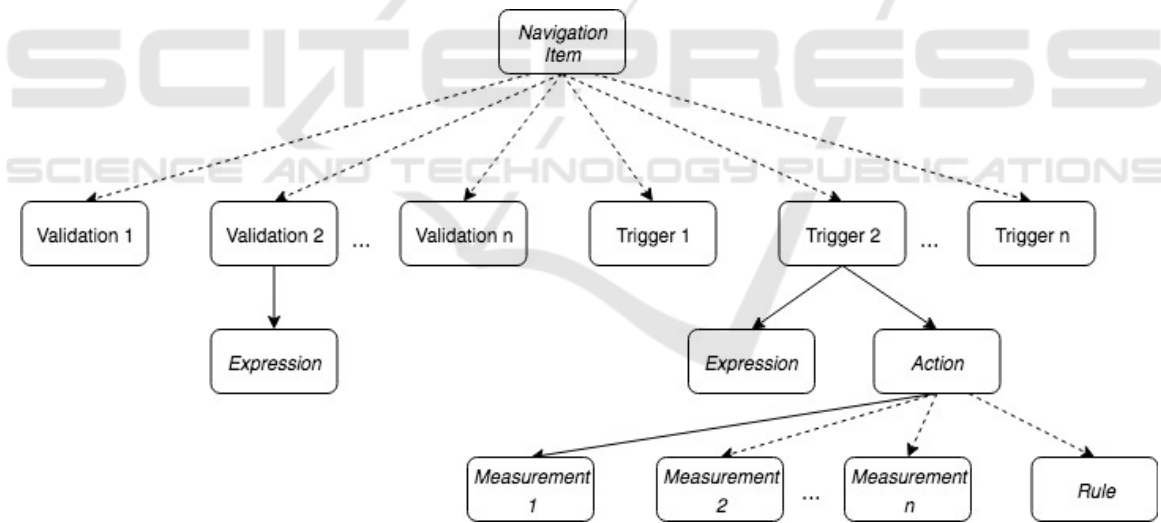
Figure 8: Navigation Item Abstract Syntax Tree.

4 SLANG DESIGN

This section describes the process of transforming the model created in Section 3 into a DSL for survey questionnaire specification, called SLang. It covers the abstract and the concrete syntaxes of SLang, and the model transformations that SLang offers.

4.1 Abstract Syntax

An *abstract syntax tree* (AST) is a data structure that represents the abstract syntax used by compilers to represent and manipulate programs. In the diagrams that follow, rectangles represent nodes or concepts, arrows link parent and child concepts, forming the

AST branches, and dotted arrows specify non-mandatory parent/child relationships.

The SLang abstract syntax reflects the questionnaire model introduced in Section 3, as shown in Figure 7. The root concept is *Survey*. *Dictionary* and *Survey* are related in a one-to-one composition, which is slightly different from the questionnaire model. Initially, SLang was designed to have *Dictionary* as a second root concept. The decision to model it as a child of *Survey* was taken after interviewing domain experts, who pointed out that dictionaries are survey-specific.

Figure 8 represents the navigation item ASTs. *NavigationItem* replicates the questionnaire model by providing the generic type that *Theme*, *QuestionSet*, *Question*, and *QuestionItem* specialize, which parallels the specialization hierarchy in Figure 3.

Each *navigation item* can have multiple validations and triggers, which form branches of the AST. *Validation* has one child concept, which is an *Expression*. *Trigger* has two children concepts: one *Expression* and one *Action*. *Action* has one child *Rule* and can have multiple children measurements. Actions are part of a generalization structure similar to domain model actions design. *Rule* is a specialization of *Expression*.

SLang supports three primitive types: number, Boolean, and text. Primitive types serve as wrappers for measurements, constants, and literals, which are the concepts that can be part of expressions. Operators wrap arithmetical, logic and comparison operators. *Function* specializes *Expression* and allows the implementation of set and aggregation operators. *Expressions*, *Operators*, and *Primitives* are used to check if expressions are valid by matching types.

4.2 Concrete Syntax

The *concrete syntax* of a language defines the notation with which users can express programs. To facilitate reading, the concrete syntax of SLang will be presented with the help of examples.

When starting the design of SLang concrete syntax, the first question asked was what users would expect. From interviews with domain experts, it became clear how hard that question is. The way questionnaires are expressed varies considerably, depending on the user background in terms of IT tools. Some use spreadsheets, while others adopt text documents. As a result, it becomes difficult to decide what would be the most adequate. Still, while designing the concrete syntax, an effort was made to define it as close as possible to the textual writing of

```
Survey: A demographic survey (S0001)
    description: A demonstration survey
    version: 1  !! allows controlling changes in the
    survey specification
    constants:
        min_wage: number 7.0
        reference_date: string 08/01/2019
    Objects of interest:
        alias: resident
        description: a person living in a household
        alias: household
        description: a building inhabited by people in a
    known location
        default: true
```

Listing 1: Survey coding example.

```
Dictionary for: A demographic survey (D0001)
    Measurements:
    household_proprietor
    code: V0003
    type: text
    size: 100
    resident_income
    code:V0060
    type: number
    precision: 11
    scale: 2
    household_income
    code: V0060t
    type: number
    precision: 14
    scale: 2
    rule: sum(resident_income)
```

Listing 2: SLang dictionary code.

a questionnaire by using sensible defaults, consistent choice of style (colors, bold, italic, font), indentation and conventions. The principal concrete syntax concepts include *Survey*, *ObjectOfInterest*, *Themes*, *QuestionSet*, *Question*, *Item*, *Trigger*, and *Validation*. A survey can also have multiple objects of interest, each of which has an alias and a description as mandatory properties. In a survey with multiple objects of interest, one of them must be marked as default by setting the default attribute as *True*.

Listing 1 presents a survey definition, including survey constants and multiple objects of interest (the orange color indicates the constant types). Each survey has an associated dictionary.

Listing 2 presents the *Dictionary* concrete syntax. A *dictionary* is a list of measurements. *Measurement* attributes vary according to their type and the only non-mandatory attribute is alias. Measurements

replicate the exact hierarchy from the domain model, including number, text, date, domain, and computed measurements.

Surveys have four concepts that specialize *NavigationItem*: *Theme*, *QuestionSet*, *Question*, and *QuestionItem*. For each of these concepts, attributes with sensible default values are kept hidden. The language designer changes those on demand according to the expected questionnaire behavior.

Survey children concepts are hierarchically organized in the following order: *Theme* is the parent of *QuestionSet*, which is the parent of *Question*. This hierarchy is created inside a survey through indentation, as shown in Listing 3. Attribute visibility is not shown for any of the navigation items because it is set to its default value, which is the Boolean literal "true".

QuestionItems are children of *Question* and specify measurement consistency criteria through the definition of validations and triggers. Listing 4 shows question items with different configurations of triggers and validations. The *Expression* typed attribute visibility is used to control which question items will be presented to the questionnaire respondent. Once visible, the mandatory attribute *Expression* makes sure that all these items will be answered.

The concrete syntax contemplates two action types: *clear* and *input*. The *Trigger* property expression defines when the trigger should run (the trigger expression default value is *True*), and the *measurements* property specifies to which measurements the trigger action should be applied.

```
Theme: Inhabitants information - T01
  object of interest: household
  instructions: Here go instructions for a potential
  interviewer
  help: Here goes an explanation of this theme
  Question Set:
    Question: 1.1.1 - How many people lived in this
  household on {reference date}?
      Item:
        save at: number qtd_people_household
    Question: 1.1.2 - How many children with ages
  between zero and nine (including newborns) lived in
  this household on {reference date}?
      Item:
        save at: number qtd_children_household
```

Listing 3: SLang theme, question set and question hierarchy.

```
Question: 8.1.1 - How many sons and daughters born
  alive until {reference date}?
  visibility: sex == 2 && age >= 10
  mandatory: false
  Item:
    save at: domain had_children_born_alive
      1: Had Children
      2: Didn't have children
    Triggers:
      action: clear
      measurements: V0802, V0803
      expression: had_children_born_alive == 2
  Item: How many man?
    visibility: V0801 == 1
    save at: number V0802
    Triggers:
      action: input(V0802 + V0803)
      measurements: qtd_children_born_alive
     Validations:
      type: ERROR
      expression: qtd_children_born_alive > 1 &&
               qtd_children_born_alive 30
      message: "The number of children born alive is
invalid."
    Item: How many women?
  visibility: V0801 == 1
  save at: number V0803
  Triggers:
    action: input(V0802 + V0803)
    measurements: qtd_children_born_alive
```

Listing 4: Question Items with Triggers and Validations.

4.3 Slang Model Transformations

The design of SLang includes two models to text (M2T) transformations: questionnaire to SQL schema and questionnaire to data collection software metadata (using Json notation).

The questionnaire to SQL schema transformation aims at creating a relational scheme to store collected data. Here the focus is on collected data structure and not on metadata or paradata structure and content.

Survey and object of interest mappings are straightforwardly mapped. Table columns are created for each measurement associated with an object of interest, with the appropriate type, according to the measurement type attribute. Validations and triggers present a challenge because of the nature of expressions. They are respectively mapped to CHECK and CREATE TRIGGER attributes.

The questionnaire to data collection software metadata transformation generates questionnaire specification in Json format, which can be submitted to SInterviewer (see Section 5.2).

5 SLang IMPLEMENTATION

Experiments with real-world questionnaires were conducted to validate SLang. First, two real-world questionnaire specifications were encoded in SLang, using a prototype implementation. This part of the experiments validated the expressiveness of SLang. Then, mock-up surveys were run, using the encoded questionnaires, on top of a survey environment, called SInterviewer. This second part of the experiments validated the behavioral aspects of SLang. This section very briefly covers all these aspects.

5.1 SLang Implementation Highlights

SLang was constructed using the JetBrains Meta Programming System (MPS). Five workbenches were first considered for the implementation of SLang: Spoofax, XText, Rascal, MetaEdit+, and MPS. The decision to adopt MPS was influenced by the fact that only MPS had a projectional editor, that is, an editor that makes it possible to create, edit and interact with one or more ASTs, avoiding the need to use parser tools (CAMPAGNE, 2016).

The concepts of the questionnaire model were first mapped into MPS concepts, defining the basic rules for AST creation. Then, the concrete syntax was created using the MPS editors. Finally, behavior and static semantics were reinforced using the MPS behavior, constraint and type system.

Language structural behavior was constructed on top of MPS Behavior Aspects. Behavior aspects made possible, for example, to attribute default values to questionnaire model properties, as well as to create and manipulate child nodes and references using MPS concept constructors and MPS concept methods.

Static semantics was established through MPS Constraint Aspects and Type System Aspects. Constraint aspects provided control of where concepts are allowed, validation for properties values, answer options control, among others. Type system aspects were used for semantic aspects that could not be modeled using MPS base concepts, behavior aspects, and constraint aspects. For example, preventing nodes with the same name to exist in a specific scope could not be done using concept structure or constraint aspects. Constraint and type systems aspects provided hooks used by MPS to implement context assistance and error reporting, in the final language IDE generated using MPS.

The two model transformations described in Section 4.3 were implemented using MPS TextGen aspect. MPS TextGen allowed the conversion of a SLang questionnaire model into a SQL script for schema creation and into questionnaire Json metadata files used by SInterviewer to enable interviews.

Finally, SLang IDE was generated, using the MPS build language feature.

5.2 SInterviewer Highlights

SInterviewer is a data-intensive mobile application, built on top of the Android platform, to collect questionnaire data and paradata. Its architecture is an evolution of the data collection software developed for five surveys conducted by the Brazilian Institute of Geography and Statistics, including the 2017 Agricultural Census and the 2020 Demographic Census.

The SInterviewer functionalities are organized in a four layers architecture, built on top of the Android SDK, following the MVC architectural pattern (see Figure 9). The top layer is responsible for presentation and governs all UI aspects. The controller layer uses the services layer to provide business rules, to access the deserialization services, to parse expressions, and to persist data.
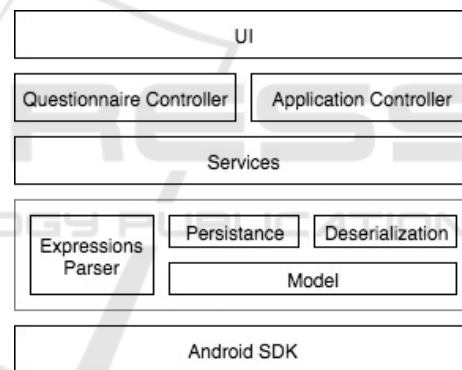

Figure 9: SInterviewer architecture.

SInterviewer accepts questionnaires specified in Json format. The deserialization of a questionnaire is triggered once, when the application is started, and is used through the services layer to control questionnaire navigation and persistence of the answers, as well as general application functionality, such as closing questionnaires and registering interview observations.

The status of a question is updated after the validations are executed. The validation of a question is executed on the navigation flow and also when the user closes the questionnaire. Questionnaire status control is derived from the question statuses and is updated when the questionnaire is closed.

Finally, SInterviewer supports two ways to export data: CSV files and Json format. These formats can

be used to import interviews (collected data from the questionnaire) to databases or data analysis tools.

The integration between SInterviewer and Slang is made through json files generated through the M2T transformation mentioned in Section 4.3. Presentation aspects are not customizable through Slang syntax; currently, this is done through sensible defaults, which can be manually adjusted after the questionnaire Json specification is generated. As future work, the development of a questionnaire presentation DSL seems to be the ideal approach to tackle that limitation.

5.3 SLang in Use

SLang was evaluated by specifying the questionnaires of two surveys and running mock interviews with the specified questionnaires on SInterviewer. Survey 1 was about K-12 students health aspects and included themes such as student's socio-economical aspects, family context, eating habits, physical activity, among others (students' questionnaire for PeNSE 2015). Survey 2 was a preliminary version of the 2017 Brazilian Agricultural Census.

Both survey questionnaires were completely encoded in SLang and successfully ran on SInterviewer. Table 1 summarizes the characteristics of the tested survey questionnaires.

Table 1: Characteristics of the Test Survey Questionnaires.

Characteristic	Survey 1	Survey 2
#Objects of Interest	1	8
#Themes	18	41
#Question Sets	18	41
#Questions	102	185
#Question Items	102	893
#Measurements	107	957
#Validations	34	134
#Triggers	28	102

The number of Themes and Question Sets were the same for the surveys since both had no distinct question organization level inside a specific Theme. Survey 1 was composed of Single choice multiple answer questions, which implies one *QuestionItem* per *Question*. Each *QuestionItem* was connected to a domain Measurement that defined the answer domain.

The use of SLang to specify questionnaires has several benefits in the context of data collection applications. SInterviewer is based on data collection applications developed at the Brazilian Institute of Geography and Statistics, which supports the ad hoc

definition of questionnaires in Json notation. What was noticed is that for each new survey, software developers adopted different conventions on questionnaire metadata. This lack of uniformity made reusing questionnaire specifications and data collection software difficult. This called for the definition of a questionnaire editor, which is, to some extent, what SLang provides.

Other potential productivity gains would be achieved by the model transformations SLang offers. Model transformations simplify tracking changes to a questionnaire and thereby facilitate verifying the compatibility between the version of the questionnaire used during data collection and the version of the questionnaire as originally specified. They also help achieve systems integration since they allow generating both Json and relational schemas for data collection storage systems from a given SLang questionnaire.

6 CONCLUSIONS

This paper presented a practical case of DSL development in the domain of questionnaire-based surveys. Although creating a DSLs is not a novel idea, the survey domain is still just starting to realize the power of DSLs as tools to tame questionnaire complexity.

Experiments with two real-world questionnaires were conducted to validate SLang. The encoding of the questionnaire specifications in SLang validated the expressiveness of the language, and the execution of mock-up surveys, using the encoded questionnaires, validated the behavioral aspects of SLang. The practical examples in which SLang was used showed the potential for using DSLs to solve practical problems, including the communication between domain experts and software developers.

SLang targets complex questionnaire design. However, it became clear that SLang might not be the best tool for simple questionnaires that involve only one type of entity as object of interest with straightforward questions, without much logic in the question navigation. Still, it would be important to conduct additional studies on language usability, as well as a comparison with tools, such as Google Forms, Zoho, Qualtrics, Survey Monkey, CSPro, Blaise, and Open Data Kit to better assess SLang's strengths and where it can be improved.

Further investigation on this research topic should include a formal user evaluation of SLang, the definition of a language evolution plan that can help deal with variabilities in questionnaire specification

practices, and the development of a DSL for questionnaire presentation.

Also, the problem of questionnaire evolution from one survey to the next should be addressed. A possible approach would be to create a mapping language that would allow users to express how questionnaires map to each-other, much in the same way that ontology or schema mapping has been addressed in the conceptual design field.

ACKNOWLEDGMENTS

REFERENCES

Araújo, L. C., 2019 Model-driven Questionnaires based on a Domain Specific Language. Master Dissertation presented to the Graduate Program in *Informatics, Pontifical Catholic University of Rio de Janeiro*.

Borodin, A. V.; Zavyalova, Y. V., 2016. Ontology-Based Semantic Design of Survey Questionnaires. *Proceeding of the 19th conference of open innovations association (FRUCT). Jyvaskyla: IEEE. 2016. p. 10-15*.

Campagne, F. The MPS Language Workbench. Third Edition. ed. [S.l.]: [s.n.].

Cotton, F.; Gillman, D. W., 2015. Modeling the Statistical Process with Linked Metadata. *Proceedings of the 3rd International Workshop in Semantic Statistics*. [S.l.]: [s.n.].

Couper, M. P., 1998. Measuring survey quality in a CASIC environment. *Proceedings of the Survey Research Methods Section: American Statistical Association. p. 41-49*.

Data Documentation Initiative Alliance, 2019. DDI Alliance Homepage. DDI Alliance. Available at: <https://www.ddialliance.org>

Erdweg, S. et al., 2015. Evaluating and comparing language workbenches: Existing results and benchmarks for the future. *Computer Languages, Systems & Structures, v. 44, n. A, p. 24-47*.

Google Forms. Google Forms Home Page. Google, 2020. Available at: <https://www.google.com/forms/about/>

Groves, R. M. et al., 2009. Survey Methodology. 2nd. Ed. New Jersey: *John Wiley & Sons*.

GSIM. GSIM and standards. UNECE Statiswiki, 2019. Available at: <https://statswiki.unece .org/display/gsim/GSIM+and+standards>.

IBGE. Census of Agriculture. Brazilian Institute of Geography and Statistics, 2017. Available at: <https://www.ibge.gov.br/en/statistics/economic/agriculture-forestry-and-fishing/21929-2017-2017-censo-agropecuario-en.html?=&t=o-que-e>

IBGE. National Students Health Survey. Brazilian Institute of Geography and Statistics, 2019. Available at: <https://www.ibge.gov.br/estatisticas/sociais/educacao/9134-pesquisa-nacional-de-saude-do-escolar.html?=&t=o-que-e>

Karsai, G. et al., 2009. Design guidelines for domain specific languages. *Proceedings of the 9th OOPSLA workshop on domain-specific modeling. Florida: [s.n.]*.

Kim, C. H.; Grundy, J.; Hosking, J., 2015. A suite of visual languages for model-driven development of statistical surveys and services. *Journal of Visual Languages and Computing, Orlando, 26, n. 99*.

Mernik, M.; Heering, J.; Sloane, A. M., 2005. When and how to develop domain-specific languages. *ACM Computing Surveys, New York, v. 37, n. 4, p. 316-344*.

Nascimento, L. M. D. et al., 2012. A systematic mapping study on domain specific languages. *Proceedings of the Seventh International Conference on Software Engineering Advances (ICSEA 2012). Lisboa: IARIA XPS Press, p. 179-187*.

Open Data Kit, 2019. Open Data Kit Home Page. Open Data Kit. Available at: <https://opendatakit.org>.

PeNSE, 2015. National Survey of School Health (PeNSE) survey. Available at: <https://www.ibge.gov.br/en/statistics/social/justice-and-security/16837-national-survey-of-school-health-editions.html?=&t=o-que-e>

Qualtrics, 2019. Qualtrics Home Page. Qualtrics. Available at: <https://www.qualtrics.com>

Saris, W. E.; Galhofer, I. N., 2014. Design, Evaluation, and Analysis of Questionnaires for Survey Research. *2nd Ed. Hoboken: John Wiley and Sons, Inc.*

Statistical Data and Metadata Exchange, 2019. SDMX. SMDX Community. Available at: <https://sdmx.org/>

Statistics Netherlands, 2019. Blaise Home Page. Blaise. Available at: <https://www.blaise.com>

SurveyMonkey, 2019. SurveyMonkey Home Page. SurveyMonkey. Available at: <https://pt.surveymonkey.com/>

Thibault, S. A.; Marlet, R.; Consel, C., 1999. Domain-specific languages: from design to implementation application to video device drivers generation. *Transactions on software engineering*, v. 25, n. 3, p. 363-377, May/June.

United States Census Bureau, 2019. Survey Processing System. US Census Bureau Website. Available at: <https://www.census.gov/data/software/cspro.html>

Zoho. ZoHo Home Page. ZoHo, 2019. Available at: https://www.zoho.com/survey/

Zhou, Y.; Goto, Y.; Cheng, J., 2014. QSL: A specification language for e-questionnaire systems. *IEEE 5th International Conference on Software Engineering and Service Science*.