# An Ontology for Surveys

Mark S. Fox and Megan Katsumi

## Abstract

This paper defines an ontology for the representation of Surveys. We have two goals, where the first is to provide an ontology for the representation of a survey's structure, questions and answers. This ontology provides a different approach than typical survey representations by combining content and logic into simple to understand and reuse classes. The second goal is to extend the ontology of questions to represent their meaning. We achieve this by defining a mapping from the survey ontology onto a target, domain specific ontology whose semantics is well defined. Examples are provided from the health and transportation domains.

## Keywords

Survey Ontology, Question Semantics, Linked Data.

## 1. Introduction

This paper defines an ontology for the representation of Surveys. We have two goals, where the first is to provide an ontology for the representation of a survey's structure, questions and answers. This ontology provides a different approach than typical survey representations by combining content and logic into simple to understand and reuse classes. The second goal is to extend the ontology of questions to represent their meaning. For example, are the questions related to disorders that the respondent has (e.g., medical), or past events they participated in (e.g., attendance at a class). We achieve this by defining a mapping from the survey ontology onto a target, domain specific ontology whose semantics is well defined. In addition, by using this ontology it is possible to:

1. Represent a survey and its contents in a manner that can be communicated over the Semantic Web,
2. Analyse the questions and answers using data mining techniques because the meaning of the questions and answers are represented, and
3. Integrate a survey's results with other sources of data for enhanced analysis.

This ontology is based on an analysis of a survey developed for the SHINESeniors Project (Bai et al., 2015), and the Transportation Tomorrow Survey[1]. SHINESeniors, or Smart Homes and Intelligent Neighbours to Enable Seniors, is a Singapore Management University initiated effort to make community care services effective through innovations in care delivery by leveraging on Information and Communications Technology (ICT). The survey gathers information about seniors participating in

[1] http://dmg.utoronto.ca/transportation-tomorrow-survey/tts-introduction

the study. It ascertains their physical and mental health, care they receive, activities they perform, adherence to treatments, family interactions, social interactions, service needs and more.

The Transportation Tomorrow Survey (TTS) is a comprehensive travel survey conducted in the Greater Toronto and Hamilton Area (GTHA) once every five years. It samples 5% of the households in the survey area and gathers information on households, number of vehicles, employment, dwelling type, trip purpose, mode and length, etc.

In the following, we provide background on existing approaches to representing surveys, review relevant vocabularies, and identify issues. The next section defines an ontology for the definitions of a survey's content (i.e., questions), and structure (i.e., sequencing, grouping, contingencies). The section that follows defines an extension to support the mapping of survey responses onto a domain specific ontology in order to support semantic interoperability. Examples are then provided to illustrate how the ontology might be applied in practice.

## 2. Background and Motivation

Huq & Karras' (2003) analysis of healthcare surveys distinguishes Survey Content (questions), Survey Display (visualization of questions) and Survey Logic (route taken through the questions). Within Survey Content, the types of questions are partitioned into: Comment Questions, Open-Ended Questions, and Close-Ended Questions, and Close-Ended Questions are divided into Nominal, Ordinal and Interval Variable answers. No ontology is provided.

Early work in capturing survey content and logic has focused on the use of XML to provide basic tags for questions, answers and logic (Barclay et al, 2002). Similarly Moodle[2], an open source learning environment, provides an XML format for the representation of questions, as does QueXML[3] (McNeil, 2015). These latter XML formats are more expressive. Popular survey web sites, such as Survey Monkey do not offer an XML format or ontology for defining and uploading questions[4]. Regardless of the approach, XML is being used to upload/download survey questions to/from a software application that will then pose the questions and document the answers.

Of particular note is the Data Documentation Initiative (DDI) that has developed standards for the representation of social sciences datasets (www.ddialliance.org). The representation of survey content and logic is just one of many aspects of DDI, and was greatly extended in version 3 of the standard. DDI provides a rich vocabulary for survey content and logic represented as XML. It explicitly separates survey content from logic in order to enhance reusability[5]. But this richness results in a complicated representation as demonstrated in the simple questionnaire example[6] that contains over

[2] https://docs.moodle.org/30/en/Moodle_XML_format

[3] https://quexml.acspri.org.au/

[4] http://help.surveymonkey.com/articles/en_US/kb/Can-I-upload-a-survey-created-in-another-file-or-import-responses

[5] Documentation of DDI 3.2 is limited to a Technical Specification (Thomas et al., 2014) which does not cover the majority of survey elements, and an online xml schema specificationthat is complete but lacking in explanation/documentation.

[6] http://www.ddialliance.org/sites/default/files/ddi3/SimpleQuestionnaire.zip

300 lines of XML to represent five questions including a single "Go to" statement. Secondly, DDI's approach to reusability, where all components are separately defined, e.g., logic from questions, makes it difficult to specify a mapping of a survey's answers onto an existing ontology.

More recently, a draft specification of a subset DDI in RDF has been posted on the web (Bosch et al., 2015). It is a direct translation of a subset of DDI into RDF triples (object, property, value), but it excludes much of the survey content and logic defined in DDI 3. The translation does not attempt to redesign the elements to take advantage of web semantics such as the OWL version of Description Logic, nor does it attempt to link variables and concepts to any domain ontologies.

The Triple-S standard (http://triple-s.org) focuses on a different issue. It provides an interchange format for survey results data. Using XML, it specifies the meta-data for a survey data file, i.e., the survey's variables, possible values, and positional information of these values in the data files. MacKay (2014) has demonstrated the translation of Triple-S files into the Semantic Web RDF format. He also points out that if the survey community adopts a common vocabulary of variables, each having its unique URI, it would be possible to merge and analyse data across multiples surveys. Although it is an intent of DDI to achieve the reusability of concepts and variables, without a definition of the semantics of the concepts and variables, any reuse will be subject to ambiguity as it is dependent upon each user's interpretation of the documentation.

The work described above does not address a major component of every survey, namely what the meaning is of each question. Why is this important? Survey results analysis is limited to the responses to questions, e.g., "what percentage answered yes to question 5". But performing analysis that requires an understanding of the meaning of a question is not possible without human intervention. Consider the following question: "what percentage performed any type of exercise." If all of the survey's questions refer to specific exercises such as walking, jogging, yoga, etc., then there would be no way to answer the more general question without someone identifying the exercise related questions. In other words, you have to understand the meaning of the questions in order to perform the analysis. A second reason for representing the meaning of questions is interoperability. If we wish to compare questions across surveys, we have no way of mapping questions from one to another without human intervention. Why have all survey representation efforts to date not addressed this problem? Because it requires solving the Natural Language Understanding problem.

The ontology defined in this report addresses two major issues. The first is providing a representation of the content and logic of surveys that conforms to Semantic Web (Berners-Lee et al., 2001), Web Ontology (Hitzler et al., 2012) and Linked Data standards (Heath & Bizer, 2011). To our knowledge this has yet to be made within the field of Survey Computing. By making this transition, surveys and their results can be more readily published, linked and analysed with other sources using linked data techniques. The second issue addressed is to provide an ontology for representing the meaning of questions. Our approach is to support a mapping from the survey ontology into a target ontology where meaning is expressed. This target ontology makes it possible to merge and analyse data, as expressed by McKay (2014). The target ontology can change from one application to the next allowing for domain independence.

Before we continue, we need to cover one more topic: how does an ontology differ from vocabularies or simple XML tags? An ontology is an "explicit representation of shared understanding" (Gruber, 1993). It "consists of a representational vocabulary with precise definitions of the meanings of the terms of this vocabulary plus a set of formal axioms that constrain interpretation and well-formed use of these terms" (Campbell & Shapiro, 1995). What distinguishes ontologies from vocabularies is that

the ontologies add definitions of the terms, and constraints on their interpretation, using logic languages such as Description Logic (Nardi & Brachman, 2002) or First-Order Logic. The benefit of representing definitions and constraints on the terminology is that common sense inferences can be made automatically. For example, the system can infer that an analysis of exercise applies to any questions referring to jogging. Or that an analysis of medical treatments can include analyses of when it occurred and possibly where.

## 3. Survey Ontology

Our approach to the design of our survey ontology differs in design philosophy from DDI. Rather than separate logic from content, we integrate them to achieve what we believe is a more perspicuous and easier to use representation. We take advantage of classification hierarchies to integrate logic and questions, and to enable extensibility by allowing users to specialize upper level classes in the survey ontology to suit their application.

Our Survey Ontology has three main parts:

1. A representation of the logic of the Survey, including contingent questions and repeated sections,
2. A representation of questions and admissible responses, and
3. A representation of answers provided by a respondent.

In the following, we use the Manchester Syntax for Description Logic to describe the ontology and publish it in OWL (Horridge & Patel-Schneider, 2012) on the Internet.

### Survey Structure

The SHINESenior survey has a complex structure of descriptions, parts, questions within parts, and questions contingent upon the answer of other questions. The purpose of the Survey class is to provide the survey structure in which the questions exist.

The root of the survey structure is the Survey class. It specifies that a Survey is composed of one or more parts called Survey_Part, has a purpose hasPurpose which is a text string, and identifies the demographics for the Survey via the hasDemographic property whose range is a Population defined in the Global City Indicator Foundation ontology (Fox, 2015). Each Survey_Part has a number (specified by hasSequence) that defines its sequence in the survey. A Survey_Part is composed of one or more sub parts, recursively defined as Survey_Part via the hasPart property, and/or one or more Questions.

| Object | Property | Value |
|---|---|---|
| Survey | owl:subClassOf | SurveyThing |
| | hasPart | min 1 Survey_Part |
| | hasPurpose | exactly 1 xsd:string |
| Survey_Part | owl:subClassOf | SurveyThing |
| | hasPart | (min 1 Survey_Part) or (min 1 Question) |
| | hasSequence | exactly 1 xsd:positiveInteger |

### Survey Content: Question

The Question class lies at the core of the ontology and demonstrates the power of the ontological approach to engineering surveys. In particular, it demonstrates how questions can be structured into taxonomic hierarchies, providing a rich set of question classes that can easily be reused and extended. It also demonstrates how distinct concepts, such as survey content and logic, can be combined, by defining a question as having multiple super-classes (next section).

The Question class specifies the text of the question, a sequence number within its enclosing Survey_Part, and a unique identifier. (Note that since a Survey_Part may contain both Questions and sub parts, this sequence numbering may be over both Survey_Parts and Questions.) The upper level of the Question content taxonomy is based on Huq & Karras (2003). The Question class is the root of the taxonomy with three subclasses:

- The Comment_Question does not admit a response. It provides directional text to the respondent.
- The Open_Ended_Question allows the respondent to provide any written text without constraint.
- The Close_Ended_Question provides for a set of predetermined responses that the respondent is to select from.

| Object | Property | Value |
|---|---|---|
| Question | owl:subClassOf | SurveyThing |
| | hasQuestionText | min 1 xsd:string |
| | hasIdentifier | exactly 1 xsd:anyURL |
| | hasSequence | exactly 1 xsd:positiveInteger |
| Comment_Question | owl:subClassOf | Question |
| | disjointWith | Open_Ended_Question |
| | disjointWith | Close_Ended_Question |
| Open_Ended_Question | owl:subClassOf | Question |
| | disjointWith | Comment_Question |
| | disjointWith | Close_Ended_Question |
| Close_Ended_Question | owl:subClassOf | Question |
| | disjointWith | Comment_Question |
| | disjointWith | Open_Ended_Question |
| | hasResponse | min 2 owl:Thing |

Question has the following properties: hasQuestionText is a data property that defines the actual question that is to be presented to the respondent, and hasIdentifier defines the unique identifier that is

associated with the question and also seen by the respondent. Close ended questions have an additional property, hasResponse, that defines the set of responses (which we elaborate below).

There exist a wide variety of close ended questions found in surveys. The following subclasses define a subset of possible responses and can be modified/extended to suit the survey.

| Object | Property | Value |
|---|---|---|
| ConfidenceQT | owl:subClassOf | Close_Ended_Question |
| | hasResponse | only { not_confident, little_confident, somewhat_confident, totally_confident, na } |
| YesNoQT | owl:subClassOf | Close_Ended_Question |
| | hasResponse | only { yes, no, don't_know, refuse } |
| FrequencyQT | owl:subClassOf | Close_Ended_Question |
| | hasResponse | only{ none, some, all, semi-annual, annual, biannual, problem, daily, twice_week, weekly, 2-3times_month, lt_once_month, na } |
| YearQT | owl:subClassOf | Close_Ended_Question |
| | hasResponse | only { ltyear, 1-2year, 2-3year, 3-4year, 4-5year, gt5year, never, na } |
| ExcuseQT | owl:subClassOf | Close_Ended_Question |
| | hasResponse | only { no_time, too_expensive, no_need, afraid, advised, not_near, no_transport, no_helper, language_barrier, long_wait_time, other } |
| SatisfactionQT | owl:subClassOf | Close_Ended_Question |
| | owl:EquivalentTo | only { very_satisfied, satisfied, dissatisfied, very_dissatisfied, dont_know } |
| AgeQT | owl:subClassOf | Close_Ended_Question |
| | hasResponse | only { 1, 2, ..., 120 } |

For example, ConfidenceQT is a close-ended question whose possible responses define the level of confidence elicited by the question. Five possible answers are defined, including "not confident" and "totally confident".

### 3.3. Survey Logic: Contingent Questions, Part Repetition and Survey Termination

In this section we describe how survey logic is integrated with survey content by extending the taxonomy of questions and survey parts to include contingent questions, part repetition and survey termination.

To represent questions whose invocation is contingent upon answers to prior questions, Question is divided into disjoint Contingent_Question and Non_Contingent_Question classes. Contingent_Question specifies one or more Contingent_Survey_Part's whose invocations are contingent upon the answer to a Close_Ended_Question. Contingent_Survey_Part is a subclass of Survey_Part that contains the property contingentOn. This property specifies one or more answers to the contingent question that will invoke this part. A Contingent_Question combined with one or more Contingent_Survey_Parts allows us to capture both the if-then and case structures that appear in some surveys. For example, in a survey about someone's daily activities, the question "What is your

occupation?" may lead to a different set of follow-up questions depending on the answer given. A separate contingent survey part is defined for each possible answer or sets of answers.

| Object | Property | Value |
|---|---|---|
| Contingent_Question | owl:subClassOf | Close_Ended_Question |
| | contingentPart | min 1 Contingent_Survey_Part |
| | terminationResponse | min 0 owl:Thing |
| Non_Contingent_Question | owl:subClassOf | Question |
| | disjointWith | Contingent_Question |
| Contingent_Survey_Part | owl:subClassOf | Survey_Part |
| | contingentOn | min 1 owl:Thing |
| Close_Ended_Question | owl:subClassOf | Question |
| | disjointWith | Comment_Question |
| | disjointWith | Open_Ended_Question |
| | hasResponse | min 2 owl:Thing |
| | exitResponse | min 0 owl:Thing |

In order to represent groups of questions that repeat, such as trips taken by a person, two types of Survey_Part are defined: Repeated_Survey_Part and Non_Repeated_Survey_Part. A Repeated_Survey_Part may repeat an unknown number of times - until some terminating criteria is met. This is captured with the exitResponse property defined in one or more close-ended questions in the repeating survey part. Any Close_Ended_Question may have one or more exitResponse values specified. If the answer matches the specified exit response, then the enclosing Repeated_Survey_Part does not repeat again. In addition, you can specify a maximum on the number of times the part is repeated using the maxRepeat property, or you can specify a Question whose answer defines the number of times the repeated survey part repeats, using the repeatsFrom property.

| Object | Property | Value |
|---|---|---|
| Survey_Part | owl:subClassOf | SurveyThing |
| | hasPart | min 0 Survey_Part |
| | hasQuestion | min 0 Question |
| | hasSequence | exactly 1 xsd:positiveInteger |
| | canRepeat | min 0 xsd:boolean |
| Repeated_Survey_Part | owl:subClassOf | Survey Part |
| | canRepeat | value true |
| | maxRepeat | max 1 xsd:positiveInteger |
| | repeatsFrom | max 1 Question |
| | disjointWith | Non_Repeated_Survey_Part |
| Non_Repeated_Survey_Part | owl:subClassOf | Survey_Part |
| | canRepeat | value false |
| | disjointWith | Repeated_Survey_Part |

A related aspect of the logic of a Survey is the notion of survey termination as a result of some response. We extend the Contingent_Question class with the terminationResponse property to capture this. If the answer matches the value of the terminationResponse, then the survey terminates. If the terminationResponse and a contingent survey part's contingentOn have the same value, then the contingent survey part is invoked first and then the survey terminates.

### Prefilling Answers

In order for answers to questions earlier in the survey to be used as answers by subsequent questions, we extend Question with the property prefillQuestion. This property identifies a Question that has been invoked earlier in the survey, and whose answer prefills the answer to the current question. The answer cannot be changed by the respondent.

It is often the case that the values of some of the questions the first time through a repeating survey part have already been provided by earlier questions. For example, questions about the respondent, such as first/last name, may also be the answer for the first time through the members of a household repeating part. To limit the pre-filling of values of questions contained in a repeating part to the first time through, we create a subclass of Question called RepeatedQuestion that denotes the question is repeated within the enclosing repeating survey part and if a prefillQuestion is specified, it is only to be used the first time through the repeating part.

### Instantiating the Survey

In this section we introduce the classes for instantiating the survey for a specific respondent. Instances of Survey_Response specify a respondent's answers to a survey. The property hasSurvey links it to the specific Survey definition. Survey_Response contains three key roles: the person who is conducting the survey (dataCollector), the person who is providing the answers (hasRespondent) and the person or thing that the questions are being asked about (surveyFocus).

We refer to the entity that the survey is gathering information about as the survey focus. The survey focus can be anything, though we often think of a person, the respondent is not necessarily the survey focus; it could be a product they own[7]. The surveyFocus property links the survey to the survey focus. The dataCollector property identifies the person (or thing) performing the survey. The hasRespondent property identifies the person or people who answered the questions. startTime and endTime specify when the survey started and ended. referenceNumber specifies a unique identifier for the survey. Finally, surveyAnswer documents each question asked and the response provided.

| Object | Property | Value |
|---|---|---|
| Survey_Response | owl:subClassOf | SurveyThing |
| | hasSurvey | exactly 1 Survey |
| | hasDemographic | exactly 1 owl:Thing |
| | surveyFocus | exactly 1 owl:Thing |
| | dataCollector | exactly 1 sc:Person |
| | hasRespondent | exactly 1 sc:Person |
| | startTime | exactly 1 xsd:dateTime |
| | endTime | exactly 1 xsd:dateTime |
| | referenceNumber | exactly 1 xsd:positiveInteger |
| | surveyAnswer | min 0 Question_Answer |

Question_Answer documents each question asked and the response provided by the respondent. hasPart defines the nested sequence of parts within which the question was asked. hasQuestion

[7] If a respondent answers a quality survey about an automobile they own, the focus is the automobile and not the respondent.

identifies the Question. hasSequence specifies where in the global sequence of questions it was asked. Note that hasResponseString is used if the question is open ended, otherwise hasResponse is used.

| Object | Property | Value |
|---|---|---|
| Question_Answer | owl:subClassOf | SurveyThing |
| | hasPart | min 1 Survey_Part |
| | hasQuestion | exactly1 Question |
| | hasResponseString | max 1 xsd:string |
| | hasResponse | min 0 owl:Thing |
| | hasSequence | exactly 1 xsd:positiveInteger |
| | startTime | exactly 1 xsd:dateTime |
| | endTime | exactly 1 xsd:dateTime |
| | hasRepeatPartCount | Exactly 1 xsd:positiveInteger |

In order to better track the repeating of survey parts, Question_Answer has a property hasRepeatPartCount that tracks which repeat a particular answer is for. Trivially, a Question_Answer that is not part of a Repeated_Survey_Part will always have a hasRepeatPartCount value of 1.

## 4. Semantics of Questions and Answers

The goal of the Survey ontology to this point has been to represent the content and logic of the survey, and the answers provided. But this does not mean that the meaning (semantics) of the questions and answers are represented. Instead, we need an ontology that represents the semantics of the survey's domain, which we refer to as the *target ontology*. We then have to specify a mapping of the Question_Answers onto the target ontology[8].

Semantically, each question seeks a value for some property of the survey focus. For example, if John is asked the question "What is your weight" and provides an answer of "75kg", it would be defined by the RDF <Object>, <Property>, <Value> triple[9]:

<URI for the survey focus John>, <URI for weight property>, "75kg".

If we have an ontology that defines the property weight, such as the OM ontology (Rijgersberg et al., 2011)[10], we could substitute the OM URI for weight in the triple:

<URI for the survey focus John>, om:weight, "75kg".

If the survey focus is known to have a URI, for example, John is listed in the University of Toronto faculty RDF triple store: http://uoft/faculty#John, the we can substitute his URI:

http://uoft/faculty#John, om:weight, "75kg".

[8] Note that the mapping can work with relational data models.

[9] The enclosing angle brackets are for the convenience of stating that the object and property of the triple are URIs that have yet to be identified.

[10] The OM ontology can be found at: http://www.wurvoc.org/vocabularies/om-1.8/. We will use the prefix "om:" to identify classes and properties from the ontology. Note that weight is not defined in this ontology.

The object of the above triple is uniquely identified by its URI as John at the University of Toronto, having a property weight defined in the OM ontology, with a value of 75kg.

In the following we define extensions to our Survey ontology that specify the mapping for simple questions, then how the mapping changes within a repeating part, followed by more complicated mapping when survey parts are reused.

To support the mapping to a target ontology, Survey has an additional property, targetBaseURI, which is a string that defines the base URI for all objects and properties in the target ontology. Next, each Question is extended with the property targetProperty that specifies the URI for the property that the question is about. If it is not a complete URI, then it is appended to the Survey's target base URI.

The answer to a simple close or open-ended question is mapped into the following triple:

<survey focus>, <question's target property>, <value selected by respondent>

Repeating survey parts are a little more tricky. For example, if a repeating survey part gathers information (e.g., age and name) for each member of a household, we would want to have a separate object created for each member, to which the attributes of the member are attached. Consider the following triples we would like to be created by the semantic mapping to the target ontology for household1, which is the URI (without its base) for the surrounding part:

| Object | Property | Value |
|---|---|---|
| household1 | hasMember | member1 |
| member1 | rdf:type | Person |
| member1 | hasAge | 34 |
| member1 | hasFirstName | Joe |
| member1 | hasLastName | Smith |
| household1 | hasMember | member2 |
| member2 | rdf:type | Person |
| member2 | hasAge | 36 |
| member2 | hasFirstName | Josephine |
| member2 | hasLastName | Smith |

To support this mapping, we interpret a Survey_Part as an indication that intermediation is to be used. In other words, another instance of a class in the target ontology is to be created to be the object of the questions contained within the survey part. We then have to provide two things:

1. The class that the Survey_Part is to instantiate. E.g., the Person class, which member1 and member2 are to be made instances of.
2. The property that links the repeatedly instantiated survey part to the surrounding part? E.g., hasMember as in "household1 hasMember member1", and "household1 hasMember member2".

We add targetClass to Survey_Part to denote the class the Survey_Part will be an instance of (in this example Person). targetProperty specifies how the surrounding Survey_Part (in this example household1) is to be linked to the instance of targetClass (in this example member1 and member2). If targetClass is not specified, intermediation is not invoked.

Figure 1 depicts the mapping of this example onto the target ontology. The mapping specification level depicts how the survey ontology is used to specify the mapping. The survey focus is linked to an instance of the target class specified in the survey part via the target property also specified in the

survey part. This instance is then linked to the answer of each question contained in the survey part via a link defined by the Question target property. Note that in Figure 1, the following classes and properties are from the target ontology: Household, Person, hasMember, hasAge and hasFirstName.

A more complicated situation arises when we have multiple contingent questions reusing the same contingent survey part. For example, in ShineSeniors for each possible ailment, a contingent question asks whether they have that ailment, and if the answer is yes, then the same contingent survey part is invoked asking the same questions (e.g., how long have they had the ailment? do they take medication?). Basically, the contingent question is the same: "Do you have ailment X?" where only X, which is the subject of the contingent question, changes. The problem is that there is no way to specify in the re-used contingent survey part what the subject of the contingent question is. Hence we have no way of discerning for which subject (e.g., ailment) the instantiated contingent survey part is associated with.

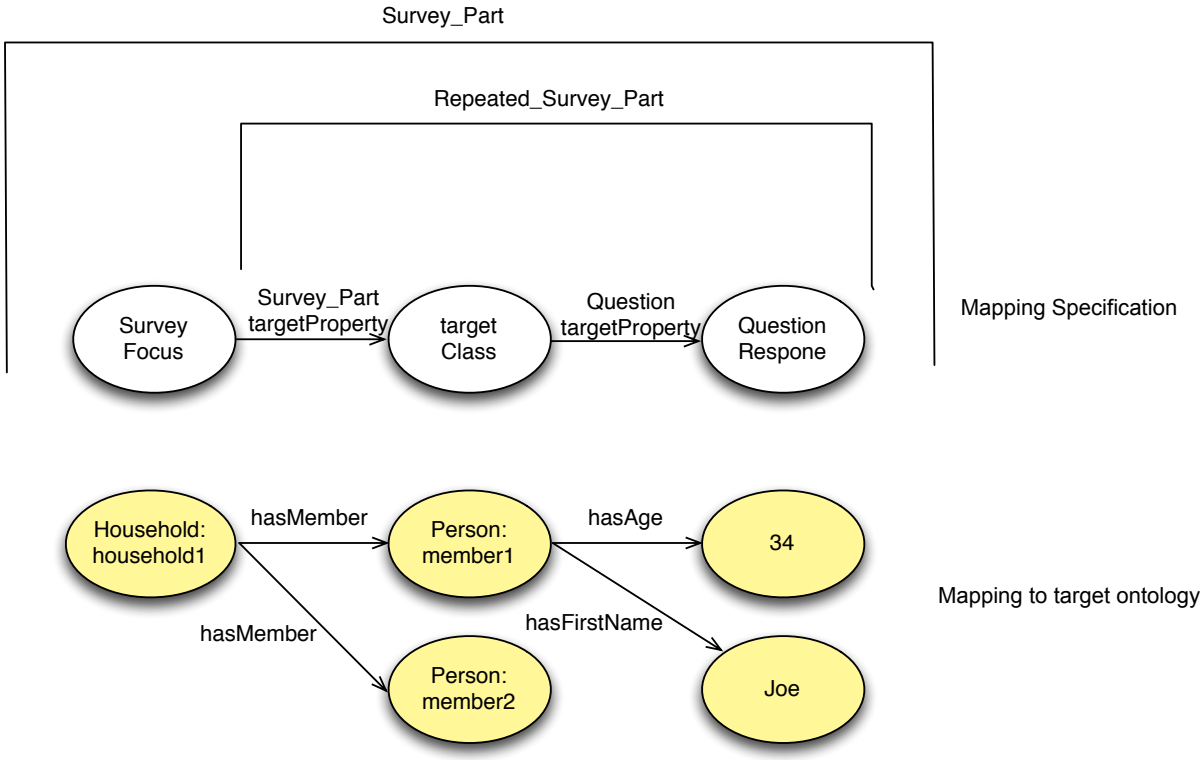



**Figure 1: Repeating Survey Part Mapping**

To distinguish one instance of a contingent survey part from another, we would have to include in the instance a property that identifies the subject of the contingent question, e.g., what the ailment is. We can either add a question to the contingent survey part that asks what ailment the contingent question referred to, which would be annoying to the respondent, or we can automatically add a property to the contingent survey part instance whose value is the subject of the contingent question (e.g., Diabetes). To accomplish the latter, we first have to identify what the subject of the contingent question is. We add the property targetSubject to Question to specify the subject of the question (e.g., Diabetes). The targetSubject value specifies the concept in the target ontology that represents the subject of the question. (Note that all Questions can have a target subject.) Next, we have to add a property, subjectProperty, to the contingent survey part whose value specifies what property to add to the

instantiated contingent survey part and whose value is the target subject of the enclosing contingent question.

| Object | Property | Value |
|---|---|---|
| Survey | owl:subClassOf | SurveyThing |
| | hasPart | min 1 Survey_Part |
| | hasPurpose | exactly 1 xsd:string |
| | targetBaseURI | exactly 1 xsd:string |
| Survey_Part | owl:subClassOf | SurveyThing |
| | hasPart | min 0 Survey_Part |
| | hasQuestion | min 0 Question |
| | hasSequence | exactly 1 xsd:positiveInteger |
| | canRepeat | max 1 xsd:boolean |
| | targetClass | max 1 owl:Thing |
| | targetProperty | max 1 rdf:Property |
| Contingent_Survey_Part | owl:subClassOf | Survey_Part |
| | contingentOn | min 1 owl:Thing |
| | hasSequence | value 1 |
| | subjectProperty | max 1 rdf:Property |
| Question | owl:subClassOf | SurveyThing |
| | hasSequence | exactly 1 xsd:positiveInteger |
| | targetSubject | max 1 owl:Thing |
| | targetProperty | max 1 rdf:Property |

Our last addition for semantic mapping is to extend Question_Answer to identify the triple that will be created. An application can then process the Question_Answer individuals to generate the RDF file. targetObject, targetProperty and targetValue are all string data properties so that they will work equally well for RDF triples and relational data models.

| Object | Property | Value |
|---|---|---|
| Question_Answer | owl:subClassOf | SurveyThing |
| | hasPart | min 1 Survey_Part |
| | hasRepeatPartCount | exactly 1 xsd:int |
| | hasQuestion | exactly1 Question |
| | hasResponseString | max 1 xsd:string |
| | hasResponse | min 0 owl:Thing |
| | hasSequence | exactly 1 xsd:positiveInteger |
| | startTime | exactly 1 xsd:dateTime |
| | endTime | exactly 1 xsd:dateTime |
| | targetObject | exactly 1 xsd:string |
| | targetProperty | exactly 1 xsd:string |
| | targetValue | exactly 1 xsd:string |

## 5. Examples

As examples, we selected questions that are representative of the types found in the SHINESeniors and transportation surveys, We start by instantiating the Survey class as ‘shines_s’, which is composed of two non-repeating Survey_Part ‘shines_a’ and ‘shines_d’ (see Table 1 in the Appendix). It is composed of three primary questions representing questions ‘a1a’, ‘a2a’, and ‘d4’ that are described in the remainder of this section.

### Simple Property Question

The most basic question that can be asked is about a simple property of the survey focus or object of discussion. For example, what is your date of birth, colour of eyes, weight? The choice of using a close or open-ended question depends upon whether you want to constrain the answers for survey purposes (e.g., only want to record certain colours of eyes), or for quality control purposes (i.e., don't want open ended questions that are difficult to understand or interpret). Our preference is to use close-ended questions for both reasons (see Table 2 in the Appendix).

To map the eye colour question and answer onto the target ontology, we would specify the targetProperty to be hasEyeColour (assuming that is the property in the target ontology). So the triple would have the URI for the respondent, the URI for the targetProperty and the URI for the eye colour.

### Simple Contingent Question

| | Ailment | A1 (i) Diagnosis | | | A1 (ii) Age at Diagnosis | A1 (iii) Taking Prescribed Medication | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Yes | No | Not Sure | | Yes | No | Not taking regularly | Not prescribed medication |
| | | | **[Go to next item]** | | | | | | |
| [***Interviewer*** *: Record the first occurence for (a) through (n) below]* | | | | | | | | | |
| a | Heart attack, angina (chest pain originating from the heart), myocardial infarction. | 1 | 2 | 3 | yrs | 1 | 2 | 3 | 4 |

We model this question as a contingent question that accepts one of three answers: Yes, No or Not Sure (See Table 3 in the Appendix). Because there are two questions contingent upon the answer Yes, we group these questions within a contingent survey part that is invoked by the answer Yes. The two questions, age and taking medicine, are both modelled as close-ended. The age question is close-ended as we wish to place a limit on the possible values of ages that can be provided in order to reduce errors.

In this example, we assume that the survey has only one question about an ailment (making this a simple contingent question). Therefore intermediation is not used. Instead, we link the survey focus to each contingent question using the targetProperty defined by each question in the contingent survey part.

### Complex Contingent Question

The SHINESenior survey is more complex than what we saw in the simple contingent question. The section on ailments repeats the same contingent question for several different ailments. If we were to use the simple continent question solution, then for each ailment the answers to the questions contained in their contingent survey part, would be linked directly to the survey focus without any way of distinguishing which answer corresponds to which ailment. The following depicts what the RDF triples would look like, assuming that 'joe' is the survey focus. Note that the first three columns specify the triples. There is no indication what ailment the answers correspond to. The fourth column is provided for convenience to show the ailment that was the subject of each contingent question.

| Object | Property | Value | Subject of Question |
|---|---|---|---|
| joe | hasAilment | Yes | Heart Attack |
| joe | atAge | 64 | |
| joe | takesMedication | Yes | |
| joe | hasAilment | Yes | Diabetes |
| joe | atAge | 60 | |
| joe | takesMedication | Yes | |
| joe | hasAilment | Yes | High Blood Pressure |
| joe | atAge | 55 | |
| joe | takesMedication | Yes | |

There are two approaches to associating an ailment with the answers to the questions that are asked if they have the ailment (i.e., contained in the contingent survey part). Each of these approaches requires intermediation, hence the contingent survey part for each contingent question specifies a targetClass. The first approach requires that the target property of each contingent question be specific to the ailment. For example, rather than having the same target property (i.e., hasAilment) for each ailment (contingent question), each ailment would have a target property, such as hasHeartAttack, hasDiabetes, and each contingent survey part would have a corresponding target property, such as hasHeartAttackInfo, and hasDiabetesInfo. With this approach, the contingent survey part that is instantiated is linked to the survey focus via an ailment specific target property. The following depicts this approach.

| Object | Property | Value | Subject of Question |
|---|---|---|---|
| joe | hasHeartAttack | Yes | Heart Attack |
| joe | hasHeartAttackInfo | ha | |
| ha | atAge | 64 | |
| ha | takesMedication | Yes | |
| joe | hasDiabetes | Yes | Diabetes |
| joe | hasDiabetesInfo | dp | |
| dp | atAge | 60 | |
| dp | takesMedication | Yes | |

Note that the hasHeartAttack and hasDiabetes properties specify whether 'joe' has either ailment. Secondly, that the hasHeartAttackInfo and hasDiabetesInfo link 'joe' to the intermediations (i.e., 'ha' and 'dp') containing the answers to the contingent questions for each ailment.

The problem with the first approach is you end up having a proliferation of properties, at least two for each ailment. A well-designed ontology would not contain such a proliferation and would instead have a single property, e.g., hasAilment, which would link to an instance that contains a property that identifies the ailment. This is the basis of the second approach.

The second approach takes advantage of the targetSubject property of a question, and the subjectProperty property of the contingent survey part. We define the question to be a contingent question and specify the target subject to be the specific aliment (e.g., Diabetes) and the value of subjectProperty (e.g., ailment) to be the property added to the survey part instance whose value will be the target subject (e.g., Diabetes). This will add a property to the survey part instance defining the ailment. The result is depicted as follows:

| Object | Property | Value | Subject of Question |
|---|---|---|---|
| joe | hasAilment | ha | Heart Attack |
| ha | ailment | HeartAttack | |
| ha | atAge | 64 | |
| ha | takesMedication | Yes | |
| joe | hasAilment | dp | Diabetes |
| dp | ailment | Diabetes | |
| dp | atAge | 60 | |
| dp | takesMedication | Yes | |

See Table 4 in the Appendix for more details.

### Repeated Survey Part

The Transportation Tomorrow Survey (TTS) is designed to collect information regarding the travel behaviour of members of selected households. The survey is collected via a single respondent for the entire household, thus there are instances where repetition is required (e.g. to ask about each member of the household).

One part of the survey contains a series of questions to gather personal information about the household members (age, occupation, etcetera). Naturally, this part of the survey must be posed at least once, as there is at least one member of the household -- the survey respondent. However, the number of times this series of questions is repeated is dependent on whether there are any other members of the household, and if so, how many.

**PERSONAL DATA SCREEN**

Message 4: *"Now, I would like to ask you some questions about - yourself…. → go to 1….(for subsequent people) - turning to the next person in the household, that would be your (Label)…. → go to 1.*

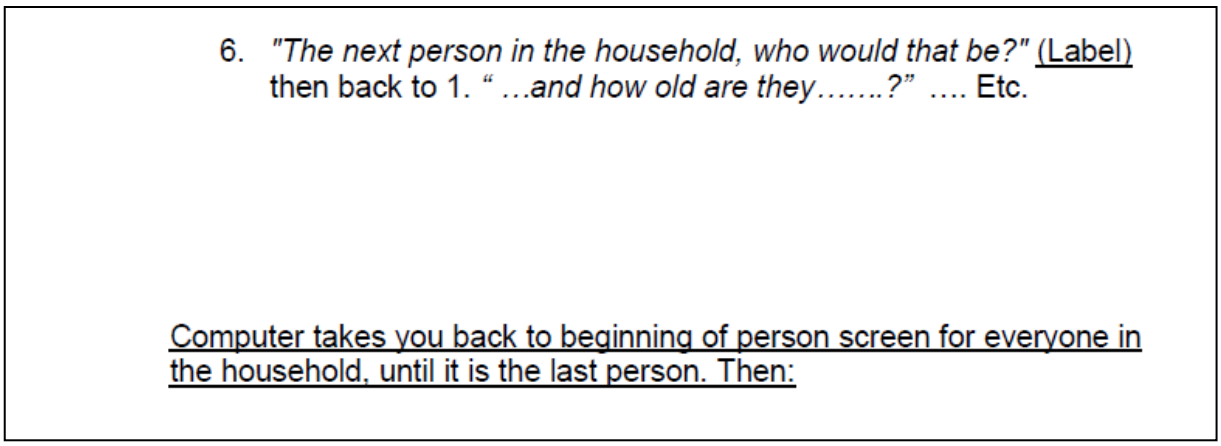
Interviewer: To record identification (Label) for people in the household, you may either type in a family description (e.g., wife, son, roommate, etc.) and/or press **[Enter]** to accept the computer defaults (e.g., RESPONDENT for 1st person, PERSON 2 for 2nd person, PERSON 3 for 3rd person, etc.)

1. *"How old are (is) you (he/she)?"* (**0-98**) and press **[Enter]**.

Interviewer: (a) Code **0** for infant less than 1 year old.
(b) Note gender of person.

2. *"Do you (he/she) have a driver's licence?"* (**Y**/**N**)

…




We capture this structure with the use of a Repeated_Survey_Part. The number of repetitions is dependent on the respondent's answer to the last question of the part: are there any more members of the household? If the answer is no, then the set of questions will not repeat again. This can be defined with the exitResponse property.

Each repetition gathers the same personal information about a different member of the household. This is a case where an intermediation must be used, as each repetition of the survey part instantiates a new person, who is a member of the household (the focus of the survey). We employ a target ontology, let's call it iCity, to define the semantics of the survey -- the concepts of a Household, Person, and the membership relation between a Household and a Person, among other things. To capture this intermediation, we can then specify the value of targetClass to be the Person class in the iCity ontology, and targetProperty to be the hasMember property in the iCity ontology. For each set of answers to this survey part in a particular response, a new instance of a Person is created (who is a member of the Household that is the focus of the survey). The semantics of each Question Answer is then captured with a tuple where the new instance is the targetObject and the targetProperty is as specified by the Question, with some targetValue as captured in the response.

The first time through the repeated survey part, we want to prefill the question that asks the member's name with the name of the survey focus. This is accomplished by specifying the first question as a repeated question with the property prefillQuestion having the value of the question found earlier in the survey that gets the survey focus' name. Because it has been defined as a repeated question, it will be prefilled with the survey focus' name only on the first repetition.

See Table 5 in the Appendix for more details on this example.

## 6. Conclusion

This paper addresses two issues. The first is to provide an ontology, based on Semantic Web/Linked Data standards, for representing the combined content and logic of surveys. Instead of simply translating standards such as DDI or Triple-S into an ontology language such as OWL, we take advantage of OWL's concept modelling capabilities to combine content and logic into simple to understand and reuse classes.

The second issue it addresses is how to map survey answers into a pre-defined ontology. Given the breadth of topics that survey's span, it is not possible to integrate the semantics of every survey domain directly into the survey ontology. Instead, there needs to be a way of mapping the questions and answers onto a domain specific (target) ontology. In this paper we extend the survey ontology to

include such mapping information. This affords two benefits. First, it enables the automation of the mapping into the target ontology. Secondly, and more importantly, it captures what the survey designer believes is the relationship between survey content and target ontology, thereby reducing possible ambiguities of interpretation.

The combination of these two solutions makes it possible to support a variety of reasoning about the logic, content and answers to the questions. It also enables interoperability, that is, being able to communicate survey content, results and questions across the Internet, using linked data standards, and merge and analyse them.

Finally, with respect to DDI, this ontology can be integrated within DDI's broader scope with linkages to the RDF specification of DDI's Geographic information, Concepts, Universes, etc.

## Appendix

### Survey Structure

The following table defines a survey as composed of four parts, as denoted by the hasSequence numbers, but we only show two of them. The first survey part has two questions, and the fourth has 1.

| Object | Property | Value |
|---|---|---|
| shines_s | rdf:type | Survey |
| | hasPart | shine_a |
| | hasPart | shine_d |
| | hasPurpose | "Survey of SHINESeniors clients." |
| shines_a | rdf:type | Non_Repeated_Survey_Part |
| | hasQuestion | a1ai |
| | hasQuestion | a2a |
| | hasSequence | 1 |
| shines_d | rdf:type | Non_Repeated_Survey_Part |
| | hasQuestion | d4 |
| | hasSequence | 4 |

**Table 1: Survey with two parts**

### Simple Property Question

As discussed on page 13 the following defines a simple question about eye colour. It assumes a close-ended question eyeColourQT defines the possible set of colours that the respondent is allowed to select from. Secondly, it defines the property in the target ontology to be hasEyeColour. The object of RDF triple will be defined in the surrounding part (or the survey) and the value will be the answer provided linked via hasEyeColour.

| Object | Property | Value |
|---|---|---|
| q1 | rdf:type | eyeColourQT |
| | hasQuestionText | "What colour are your eyes?" |
| | hasSequence | 1 |
| | targetProperty | hasEyeColour |

**Table 2: Simple Property Question**

### Simple Contingent Question

The example on page 13 is composed of three questions where A1a(ii) and A1a(iii) are contingent upon A1a(i). A1a(i) is a YesNoQTClose_Ended_Question. A1a(ii) is also a AgeQTClose_Ended_Question. A1a(iii) is an extension of the YesNoQTClose_Ended_Question. Questions A1a(ii) and A1(iii) are contingently linked to the A1a(i). To represent this within the Survey, a YesNOQT Question instance is created and also made a type of Contingent_Question. It specifies A1(i) as the question to be asked, a Response (in this case "yes" that if matched invokes a second Survey_Part that specifies A1a(ii) and A1a(iii) as Questions to be asked.

Intermediation is not invoked for the survey part because the targetClass property is not specified. Consequently, the answers to the two contingent questions are linked directly to the survey focus via the targetProperty specified for each question.

| Object | Property | Value |
|---|---|---|
| a1ai | rdf:type | YesNoQT |
| | rdf:type | Contingent_Question |
| | hasQuestionText | “Heart attack, angina (chest …” |
| | contingentPart | a1a_p |
| | targetProperty | hasHeartAttack |
| | hasSequence | 1 |
| a1a_p | rdf:type | Non_Repeated_Survey_Part |
| | contingentOn | yes |
| | hasQuestion | a1aii |
| | hasQuestion | a1aiii |
| | hasSequence | 1 |
| a1aii | rdf:type | Number_OEQ |
| | hasQuestionText | “Age at Diagnosis” |
| | hasMax | 150 |
| | hasSequence | 1 |
| | targetProperty | atAge |
| a1aiii | rdf:type | Medication_YesNo_RT |
| | hasQuestionText | “Taking prescribed medication” |
| | hasSequence | 2 |
| | targetProperty | takesMedication |

**Table 3: Contingent question with a contingent survey part containing two questions**

### Complex Contingent Question

As discussed in Section 5, the same contingent question is asked about multiple diseases. Starting with the simple contingent question example above, we modify it to specify that a new instance is to be created for each survey part invoked by a contingent question. This is accomplished by specifying a targetClass that is to be instantiated, and a targetProperty that links the instance to the surrounding part or survey. Each question contained in the invoked part has its answer linked to the part's instance via the targetProperty specified by the question.

Next we have to reflect that each instance of the target class refers to a different ailment. In the contingent question a1ai, we specify the target subject to be HeartAttack, we then specify in the contingent survey part the value of subjectProperty to be ailment. This will result in the property ailment with a value of HeartAttack being added to the instance of the contingent survey part (that is of type Ailment).

| Object | Property | Value |
|---|---|---|
| a1ai | rdf:type | YesNoQT |
| | rdf:type | Contingent_Question |
| | targetSubject | HeartAttack |
| | targetProperty | hasHeartAttack |
| | hasQuestionText | “Heart attack, angina (chest …” |
| | contingentPart | a1a_p |
| | hasSequence | 1 |
| a1a_p | rdf:type | Contingent_Survey_Part |
| | contingentOn | yes |
| | hasQuestion | a1aii |
| | hasQuestion | a1aiii |
| | hasSequence | 1 |
| | subjectProperty | ailment |
| | targetClass | Ailment |
| | targetProperty | hasAilment |
| a1aii | rdf:type | Number_OEQ |
| | hasQuestionText | “Age at Diagnosis” |
| | hasMax | 150 |
| | hasSequence | 1 |
| | targetProperty | atAge |
| a1aiii | rdf:type | Medication_YesNo_RT |
| | hasQuestionText | “Taking prescribed medication” |
| | hasSequence | 2 |
| | targetProperty | takesMedication |

Table 4: Complex Contingent Question

### Repeated Survey Part

As discussed in Section 5, this table defines the repeating part of the TTS that captures personal information for each member of a given household.

| Object | Property | Value |
|---|---|---|
| tts | rdf:type | Survey |
| | hasPart | personal_part |
| | hasPurpose | " The survey is designed to collect information on the travel patterns of allmembers of the selected households eleven years of age and older" |
| | hasBaseURI | "http://ontology.eil.utoronto.ca/Survey/TTS.owl" |
| personal_part | rdf:type | Repeated_Survey_Part |
| | hasSequence | 3 |
| | canRepeat | True |
| | targetClass | icity:Person |
| | targetProperty | icity:hasMember |
| | hasQuestion | personal_Q1 |
| | hasQuestion | personal_Q2 |
| | .... | |
| | hasQuestion | personal_Q6 |
| personal_Q1 | rdf:type | Open_Ended_Question |
| | rdf:type | Repeated_Question |
| | prefillQuestion | q1a |
| | hasSequence | 1 |
| | hasQuestionText | "What is the name of this member?" |
| | targetProperty | icity:hasName |
| personal_Q2 | rdf:type | Open_Ended_Question |
| | hasSequence | 2 |
| | hasQuestionText | "How old are (is) you (he/she)?" |
| | targetProperty | icity:hasAge |
| ... | | |
| personal_Q6 | rdf:type | YesNoQT |
| | rdf:type | Close_Ended_Question |
| | hasSequence | 6 |
| | hasQuestionText | "Is there another member of the household?" |
| | exitResponse | "no" |

**Table 5: Repeated Survey Part**

The following table provides an abbreviated example of a how a particular survey response might be stored for the repeated survey part.

| Object | Property | Value |
|---|---|---|
| tts_response_01234 | rdf:type | SurveyResponse |
| | hasSurvey | tts |
| | surveyFocus | http://www.example.com/someuri#household1234 |
| | hasRespondent | http://www.example.com/someuri#johndoe |
| | surveyAnswer | a1-1 |
| | surveyAnswer | ... |
| | surveyAnswer | a6-1 |
| | surveyAnswer | a6a-1 |
| | surveyAnswer | a1-2 |
| | surveyAnswer | ... |
| | surveyAnswer | a6-2 |
| a1-1 | rdf:type | Question_Answer |
| | hasPart | personal_part |
| | hasQuestion | personal_Q1 |
| | hasResponseString | "Joe Smith" |
| | hasResponse | http://ontology.eil.utoronto.ca/Survey/TTS.owl#JoeSmith |
| | hasSequence | 1 |
| | hasRepeatPartCount | 1 |
| | targetObject | http://www.example.com/someuri#person1 |
| | targetProperty | icity:hasName |
| | targetValue | http://ontology.eil.utoronto.ca/Survey/TTS.owl#JosSmith |
| a2-1 | rdf:type | Question_Answer |
| | hasPart | personal_part |
| | hasQuestion | personal_Q2 |
| | hasResponseString | "42" |
| | hasResponse | http://ontology.eil.utoronto.ca/Survey/TTS.owl#42 |
| | hasSequence | 1 |
| | hasRepeatPartCount | 1 |
| | targetObject | http://www.example.com/someuri#person1 |
| | targetProperty | icity:hasAge |
| | targetValue | http://ontology.eil.utoronto.ca/Survey/TTS.owl#42 |
| a6-1 | rdf:type | Question_Answer |
| | hasPart | contingentpart_Q6 |
| | hasQuestion | personal_Q6 |
| | hasResponseString | "yes" |
| | hasResponse | http://ontology.eil.utoronto.ca/Survey/TTS.owl#yes |
| | hasSequence | 6 |

## Acknowledgements

This research was supported, in part, by the iCity Lab, a partnership between Tata Consultancy Services and Singapore Management University, and the Ontario Research Fund Research Excellence iCity project. We like to thank the reviewers for their comments.

SHINESeniors (Nov 14 – Oct 17) is a Singapore Management University led research project supported by the Singapore Ministry of National Development and National Research Foundation under the Land and Liveability National Innovation Challenge (L2NIC) Award No. L2NICCFP1-2013-5.

## References

Bai, L., Gavino, A.I., Lee, P., Kim, J., Liu, N., Tan, H.P., Tan, H.X., Tan, L.B., Toh, X., Valera, A., Yu, E.J., Wu, A., and Fox, M.S., (2015), "SHINESeniors: Personalized Services for Active Ageing-in-Place, *Proceedings of the IEEE Smart Cities Conference*, Guadalajara, Mexico.

Barclay MW, Huq S, Karras BT, and Lober WB. (2002) SuML: Implementation of a Multi-domain Survey Markup Language. *Proceedings of the Association for Survey Computing*; Sept 2002 London: Publisher.

Berners-Lee, T., Hendler, J., and Lassila, O., (2001), "The Semantic Web", *Scientific American*, May.

Bosch, T., Cyganiak, R., Wackerow, J., and Zapilko, B.,(2015), "DDI-RDF Discovery Vocabulary", rdf-vocabulary.ddialliance.org/discovery.html, Accessed 2 July 2016.

Campbell, A.E., and Schapiro, S.C., (1995), "Ontologic Mediation: An Overview", *Proceedings of the IJCAI Workshop on Basic Ontological Issues in Knoweldge Sharing*, Menlo Park CA, USA: AAAI Press.

Fox, M.S., (2013), "A Foundation Ontology for Global City Indicators", Working Paper No. 3, Global Cities Institute, University of Toronto, Revised 12 April 2015. http://eil.utoronto.ca/wp-content/uploads/smartcities/papers/GCI-Foundation-Ontology.pdf

Fox, M.S., (2015), "The role of ontologies in the publishing and analysis of city indicators", *Computers, Environment and Urban Systems*, Vol. 54, pp. 266-279.

Gruber, T. R., (1993), "Towards Principles for the Design of Ontologies used for Knowledge Sharing." *Proceedings of the International Workshop on Formal Ontology*, Padova, Italy.

Harris, S., and Seaborne, A., (2012), "SPARQL 1.1 Query Language", W3C Recommendation, https://www.w3.org/TR/sparql11-query/

Heath, T., and Bizer, C., (2011), *Linked Data: Evolving the Web into a Global Data Space*, Morgan & Claypool Pub.

Hitzler, P., et al., (2012), "OWL 2 Web Ontology Language Primer (2nd Edition)", http://www.w3.org/TR/owl-primer.

Horridge, M., and Patel-Schneider, P.F., (2012), "OWL 2 Web Ontology Language Manchester Syntax (2nd Edition)", https://www.w3.org/TR/owl2-manchester-syntax/

Huq, S. Z., and Karras, B. T., (2003). A Proposed Ontology For Online Healthcare Surveys. *AMIA Annual Symposium Proceedings*, 2003, 304–308.

MacKay, I., (2014), "We have big data, but need big knowledge – weaving surveys into the semantic web", *Proceedings of the Association for Survey Computing*, September.

McDowell I, and Newell C., (2006), *Measuring Health: A Guide to Rating Scales and Questionnaires*. 3rd ed. New York: Oxford University Press.

McNeil, V., (2015), "How To Convert A Survey From Word To queXML Using ALTOVA XML SPY 2005", https://quexml.acspri.org.au/sites/quexml.acspri.org.au/files/images/introducing_xml.pdf

Nardi, D., and Brachman, R.J., (2002), "Introduction to Description Logics", In *Description Logic Handbook*, edited by F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, P.F. Patel-Schneider, Cambridge University Press, pp. 5-44.

Rijgersberg, H., Wigham, M., and Top, J.L., (2011), "How Semantics can Improve Engineering Processes: A Case of Units of Measure and Quantities", *Advanced Engineering Informatics*, Vol. 25, pp. 276-287.

Thomas, W., Gregory, A., Gager, J., Johnson, J., and Wackerow, J., (2014), "Technical Document for DDI 3.2", Data Documentation Initiative, http://www.ddialliance.org/Specification/DDI-Lifecycle/3.2/XMLSchema/HighLevelDocumentation/DDI_Part_I_TechnicalDocument.pdf, Downloaded 2 July 2016.

## About the Authors

**Mark S. Fox** is a Distinguished Professor of Urban Systems Engineering, and Professor of Industrial Engineering and Computer Science at the University of Toronto. He received his BSc in Computer Science from the University of Toronto in 1975 and his PhD in Computer Science from Carnegie Mellon University in 1983. In 1979 he was a founding member of the CMU Robotics Institute as well as the founding Director of the Institute's Intelligent Systems Laboratory. He co-founded Carnegie Group Inc. in 1984, a software company that specialized in Artificial Intelligence-based systems. He was Associate Professor of Computer Science and Robotics at CMU from 1987 to 1991, In 1988 he was the founding Director of the Center for Integrated Manufacturing Decision Systems in CMU's Robotics Institute. In 1991, Dr. Fox returned to the University of Toronto where he was appointed the NSERC Research Chairholder in Enterprise Integration and Professor of Industrial Engineering and Computer Science. In 1993, Dr. Fox co-founded and was CEO of Novator Systems Ltd., a pioneer in E-Retail software and services. In 2014 he became the founding director of the Center for Social Services Engineering.

Dr. Fox pioneered the field of Constraint-Directed Scheduling within Artificial Intelligence and played a significant role in the development of Ontologies for modelling Enterprises. He was the designer of one of the first commercial industrial applications of expert systems: PDS/GENAID, a steam turbine and generator diagnostic system for Westinghouse, which was a recipient of the IR100 in 1985 and is still in commercial use at Siemens today. He was the co-creator of the Knowledge Representation SRL from which Knowledge Craft™ and ROCK™, commercial knowledge engineering tools, were derived, and KBS from which several commercial knowledge based simulation tools were derived. His current research focuses on the ontologies and common sense reasoning and their application to Smart Cities.

Dr. Fox was elected a Fellow of Association for the Advancement of Artificial Intelligence in 1991, a Joint Fellow of the Canadian Institute for Advanced Research and PRECARN in 1992, and a Fellow of the Engineering Institute of Canada in 2009. Dr. Fox has published over 200 papers.

**Megan Katsumi** is a post-doctoral fellow with the University of Toronto Enterprise Integration Lab, and the Transportation Research Institute (UTTRI) Group. She has a B.A.Sc. in Industrial Engineering (Toronto), and completed her M.A.Sc. and Ph.D. with the Semantic Technologies Group at the University of Toronto where her research focused on foundations for ontology development -- most notably the verification of ontologies, and the reuse of ontologies. Currently, Megan's research is concerned with the development of a set of ontologies to capture the urban system -- both as it exists in reality and as it is conceptualized in the sort of simulations and analyses that are performed by the UTTRI Group.