Processing survey data with VTL

Thomas Dubois, thomas.dubois@insee.fr

Abstract

After renovating its data collection system for business and households surveys based on the concept of active metadata, INSEE has pursued technical investments on the post-collection processing. The Validation and Transformation Language (VTL) proposed by the SDMX initiative is used for the reconciliation of data collected from different survey modes and for first processing tasks. VTL processing rules are used and interpreted thanks to Java and JavaScript implementations provided by the Trevas open source engine.

Introduction

For ten years now, Insee has been developing a survey data collection platform using international standards in a metadata-driven approach. Several components have been developed one after the other: a questionnaire generator (Eno), a platform for business surveys (Coltrane), a questionnaire designer (Pogues), a PAPI data capture tool and a survey data editing environment.

One particular feature sets Coltrane apart from previous systems: the close links that it has had, from the outset, with the RMéS statistical metadata repository and, in particular, the stated objective of using the metadata that describe the surveys questionnaires to directly generate the collection tools, hence the term “active” metadata to describe the system. Insee has adopted the GSBPM to model its statistical processes and the DDI (Data Documentation Initiative) standard to formally describe the life cycle of the data, in particular the “questionnaire” objects. This structural choice made it impossible to use one of the collection software packages on the market, as they were not yet adapted to these standards.

More recently, the Metallica project has set up a new platform serving household surveys, which require better support for multimode, additional functionalities for Pogues and Eno, and the possibility to handle CAPI/CATI in all its dimensions. Metallica pursues the long-term strategy of standardisation and industrialisation of questionnaires, processes, organisation and services that is at the heart of the metadata-driven approach since its inception, and reaps its benefits.

This paper first details how Insee uses DDI to specify survey questionnaires, and how VTL can add more complex features like dynamic behaviour in that context. We then show that VTL can also be used in data integration and post-collection processes, before concluding by listing areas of future work.

Metadata-Driven Surveys

The automated generation of collection instruments (i.e. web or paper or via an interviewer - phone or face-to-face- forms and programs that allow response data to be retrieved) based on the DDI specification of the questionnaire is a powerful driver for standardisation and productivity gains. The Pogues designer and Eno generator make it possible to automate or greatly simplify tasks previously carried out manually: development of the collection instrument by a computer engineer, and then testing and acceptance by the statistician. They also ensure that collection instruments for different survey modes (including paper) correspond to the same questionnaire description, which guarantees maximum data coherence. Also, the survey designers no longer have to write as many specifications as there are collection modes.

Box 1. Representing questionnaires in DDI

DDI, which comes mainly from the world of social science research and libraries, has been developing since 1995 under the main impetus of the University of Michigan. Initially focused on documenting scientific studies and their results, DDI expanded from 2007 to model the entire data lifecycle, from collection to publication and archiving. This version, known as “DDI Lifecycle”, very precisely covers, in particular, the representation of statistical questionnaires.

DDI favours exhaustiveness, precision and the possibilities of reuse and sharing of the objects described, over practicality of use and simplicity of representation. Such precision and semantic richness come at the cost of a certain verbosity, but make DDI a natural choice in a machine-actionable metadata-driven approach.

Other solutions were studied, for example the SDMX metadata model, but using such a high-level and generic model would require most of the business semantics to be injected externally through specialisation and parametrisation, thereby reducing the interoperability and shareability of the approach.

After a prototype created in 2013 generating Open Document questionnaires for the Annual Structural Business Survey, a first release of Eno was put into production in 2015, completed in 2018 by the Pogues designer. Today, all business surveys (using CAWI and PAPI modes) and a growing number of households surveys (using also CAPI and CATI) are served based on this approach, and this represents more than 1.5 million questionnaires annually. The efficiency and flexibility of the solution was strikingly demonstrated at the start of the pandemic crisis, when a new “Covid survey” had to launched in urgency: it took 26 days from the idea to the publication of the results.

Extending the same principles beyond the survey collection phase, it is possible to reuse the questionnaire description in combination with additional control specifications to allow Eno to automatically build a data editing tool. Thus, a survey clerk can review and process the survey data using a view equivalent to that of the questionnaire.

In order to illustrate DDI-based generation with a concrete example, let us consider a single-choice question about Internet usage taken from the annual household survey on information and communication technologies (Household ICT).

The DDI description of the question (simplified version for readability) is the following:

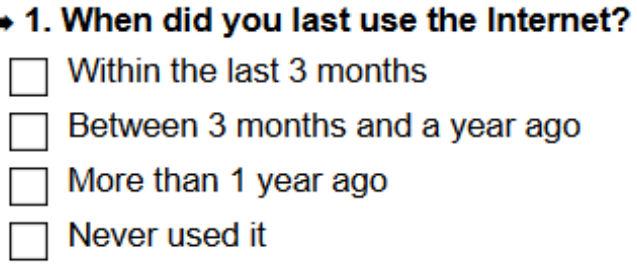
```
<d:QuestionItem>
    <r:Agency>fr.insee</r:Agency>
    <r:ID>kc0h7448</r:ID>
    <r:Version>1</r:Version>
    <d:QuestionItemName>
        <r:String xml:lang="en-IE">NUSEWEB</r:String>
    </d:QuestionItemName>
    <d:QuestionText>
        <d:LiteralText>
            <d:Text xml:lang="en-IE">When did you last use the internet?</d:Text>
        </d:LiteralText>
    </d:QuestionText>
    <d:CodeDomain>
        <r:GenericOutputFormat controlledVocabularyID="INSEE-GOF-CV">
            radio-button</r:GenericOutputFormat>
        <r:CodeListReference>
            <r:Agency>fr.insee</r:Agency>
            <r:ID>kc0hgqph</r:ID>
            <r:Version>1</r:Version>
            <r:TypeOfObject>CodeList</r:TypeOfObject>
        </r:CodeListReference>
        <r:ResponseCardinality minimumResponses="1" maximumResponses="1"/>
    </d:CodeDomain>
</d:QuestionItem>
```

The `CodeListReference` element points to the list of possible answers which is not included here.

Eno generates the following rendering of the web:

Web question

The built-in behaviour of web radio buttons ensures that a unique response will be given. This is of course not the case for the paper version:



Paper question

Here, nothing prevents the respondent to check multiple boxes, so specific post-collection treatments must be put in place to deal with this possibility.

Dynamic behaviour

To each question corresponds one or more variables, called "collected" variables since their values will be provided by the respondent. Calculated or external variables can also be defined. The former are obtained by generally simple formulas based on the collected variables, and are used in particular to specify consistency checks. External variables are not collected but provided in the questionnaire because they are useful for its customisation: they may be values obtained previously and recalled to facilitate completion, check evolutions or set certain elements of the questionnaire (collection wave, geographical zoning, last known population, etc.).

Based on these different types of variables, it is possible to describe logical components for interactive modes (self-administered web questionnaire or face-to-face survey):

- conditional expressions: typically, the way a question is expressed may depend on the values of the responses already obtained;
- checks allowing the data collected to be validated: these checks may cover one or more issues (for example: checking the consistency between the total turnover reported and its breakdown by activity);
- filters: depending on the data already collected, the respondent may be redirected to different questions, or certain sections of the questionnaire may be enabled or disabled.

In a first implementation of the questionnaire designer, these logical expressions were written in a simple *ad hoc* language adherent to the technology used by the collection platform for business surveys (namely XPath/XQuery).

In a second phase of renovation of the collection information system carried out by the Metallica project, the approach was carried out to choose a new language answering the following criteria:

- to favour a specification language that is technically neutral and accessible to survey

designers, and not a technical language (XPath, XQuery, Java, Javascript, R, Python…) in order to avoid the technical migrations inherent to the life of information systems. The specification of a filter or a control must be independent of the execution technology, especially in a complex environment where the same rule can be executed in different execution contexts (offline application for the surveyor, API…)

- to favour a language with a formal grammar so that it is both "machine-actionable" and allows to fit into the logic of active metadata where the documentation/specification of the rules (metadata) is also executable (active) without being rewritten.

Following these criteria, VTL (Validation and Transformation Language) was chosen.

Box 2. VTL

VTL (*Validation and Transformation Language*) is a language for specifying data validation or transformation processing developed within the framework of the SDMX standard well known in the statistical field. Aimed at the statistician user, it provides a neutral view of the data process at the business level. As a specification language, VTL is rich and expressive enough to define relatively complicated operations (the level of complexity is comparable to that of SQL).

VTL has characteristics that make it particularly interesting in a context of industrialization and automation of statistical processes.

First of all, VTL is positioned at the logical level: expressions must be passed to an engine that will run it on a lower level platform such as Java, Python, or C#. This allows a clear separation of concerns between the statistician who focuses on specifications and the computer scientist who is in charge of the implementation. In directly executable languages, the logical formulation of the processing is very often drowned in the details of implementation and difficult to reconstruct.

Another interesting property of VTL is to be based on a data model which derives from international standards (GSIM, SDMX, DDI) and which is well suited to statistics and to different types of data (detailed, aggregated, qualitative, quantitative, etc.). At the heart of the model is the `Data Set`, composed of `Component`s (the columns in a tabular file) playing different roles (identifiers, measures and attributes) and rows (`Data Point`s). This model allows to make assumptions that simplify the expressions.

Finally, VTL is described by a formal grammar, which ensures the logical foundation of the language and allows it to be used in an automated way, in particular by building tools such as editors or execution engines for different platforms. This ensures that the same expression will be executed consistently in different lower level languages.

Going back to our example: since the answer to the question is essential for his study, the survey designer wants to specify a required answer based on his variable called “NUSEWEB”. Using VTL, this translates in the following expression:

```
not isnull(NUSEWEB)
```

The above example represents a simple computation but more complex ones can be performed. For instance the formula for calculating a variable counting the number of checkboxes ticked in a multiple choice question would look something like this:

```
(if (nvl(QCM1, false) = true) then 1 else 0) +
(if (nvl(QCM2, false) = true) then 1 else 0) +
(if (nvl(QCM3, false) = true) then 1 else 0) +
(if (nvl(QCM4, false) = true) then 1 else 0)
```

The following example is a logical expression for filtering questions in the TIC survey about the exposition to screens based on the use of Internet and possession of a smartphone:

```
not ((NUSEWEB = "2" or NUSEWEB = "3" or NUSEWEB = "4") and not
(((nvl(F_SMARTPHONEX3, false) = true or (F_SMARTPHONEX1 = false and
F_SMARTPHONEX2 = false and F_SMARTPHONEX3 = false)) and NUSEWEB = "1")))
```

VTL expressions are transported in the DDI questionnaires by `ComputationItem` elements, as illustrated in the example below:

```
<d:ComputationItem>
    <r:Agency>fr.insee</r:Agency>
    <r:ID>kc0h7448-CI-0</r:ID>
    <r:Version>1</r:Version>
    <d:ConstructName>
        <r:String xml:lang="fr-FR">Required answer</r:String>
    </d:ConstructName>
    ...
    <d:TypeOfComputationItem controlledVocabularyID="INSEE-TOCI-CL-2">
        warning</d:TypeOfComputationItem>
    <r:CommandCode>
        <r:Command>
            <r:ProgramLanguage>vtl</r:ProgramLanguage>
            <r:InParameter isArray="false">
                <r:Agency>fr.insee</r:Agency>
                <r:ID>kc0h7448-CI-0-IP-1</r:ID>
                <r:Version>1</r:Version>
                <r:ParameterName>
                    <r:String xml:lang="fr-FR">NUSEWEB</r:String>
                </r:ParameterName>
            </r:InParameter>
            ...
            <r:CommandContent>isnull(kc0h7448-CI-0-IP-1)</r:CommandContent>
        </r:Command>
    </r:CommandCode>
</d:ComputationItem>
```

Here again, some lines are omitted for brevity: as mentioned already, DDI is quite verbose, but this is completely hidden from the survey designer thanks to the Pogues web interface. The DDI is indeed entirely generated by Eno.

Post-Collection Data Processing

Once collected in different survey modes, response data need to be processed in order to be integrated and provided to the downstream steps of the statistical operation process. The Metallica program has therefore pursued technical investments aiming to reconcile the data from the different modes (GSBPM: “Process/Integrate Data”) and start the first processing tasks (GSBPM: “Process/Classify and Code”).

Returning to our running example of single-choice question from the Household ICT survey, the reconciliation procedure of data from several modes will consist in specifying, for the paper response, what to do in cases where several boxes are ticked: retain none, the first, the one consistent with other responses, etc. In order to implement this type of calculations, Insee also uses VTL. In our specific example, the expression is as follows:

```
ICT_PAPI :=  ICT_PAPI [calc NUSEWEB :=
if NUSEWEB_1 = '1' then '1' else (
    if NUSEWEB_2 = '1' then '2' else (
        if NUSEWEB_3 = '1' then '3' else (
            if NUSEWEB_4 = '1' then '4' else '')))];
```

The choice made by the designer can be different in other surveys. In the French VQS survey, the SEX variable is dropped if several boxes are ticked:

```
VQS_PAP := VQS_PAP [calc SEXE :=
if SEXE_1 = '1' and SEXE_2 = '0' then '1'
else if SEXE_2 = '1' and SEXE_1 = '0' then '2'
else ''];
```

A more sophisticated example is given by the VQS survey: a score is computed from the responses on age and visual/audio capacities in order to identify sub-sample for a follow-up survey on autonomy. A simplified version of the specification is given below:

```
VQS := VQS[calc SCORE :=

if 0 <= AGE < 5 then (
    (if VUE = '3' or VUE = '4' then 3
    else if VUE = '2' then 1
    else 0)

    + (if AUDITIF = '3' or AUDITIF = '4' then 3
    else if AUDITIF = '2' then 1
    else 0)
)
else if 5 <= AGE <= 59 then (
    (if VUE = '3' or VUE = '4' then 6
    else if VUE = '2' then 3
    else 0)

    + (if AUDITIF = '3' or AUDITIF = '4' then 6
    else if AUDITIF = '2' then 3
    else 0)
)
else -1];
```

```
VQS := VQS[calc GROUPE :=
if 0 <= SCORE < 5 then '1'
else if 5 <= SCORE < 10 then '2'
else if 10 <= SCORE < 25 then '3'
else if 25 <= SCORE <= 100 then '4'
else ''
];
```

We see in this example how VTL can be used to specify relatively complex calculations. Insee has developed a set of VTL tools that allow the survey designer to specify and test this type of processing independently without calling on a computer scientist.

Box 3. Insee's VTL tools

For actual execution, VTL expressions need to be translated to the target runtime environment. Trevas provides this step for the Java platform, by using the VTL formal grammar and the Antlr parser generator. For a given execution, Trevas receives the VTL expression and the data bindings that associate variable names in the expression to actual data sets. The execution results can then be retrieved from the bindings for further treatments.

Trevas provides an abstract definition of a Java VTL engine, as well as two concrete implementations: - an in-memory engine for relatively small data, for example at design time when developing and testing VTL expressions on data samples - an Apache Spark engine for Big Data production environments

Other implementations can be easily developed for different contexts.

Companion software for Trevas are available, in particular a web environment and a Jupyter notebook integration. Trevas JS, a JavaScript VTL engine can be used for simple client-side computations. All these tools are open source.

Conclusion

Assessment of the Solution

While the redesign of the collection information system has led to a great deal of work on standardizing processing, there are still a number of specificities to be taken into account for each survey. The use of the VTL processing language, dedicated to the designer and interoperable with the rest of the highly standardized system, has already made it possible to optimize the implementation and renovation of certain household surveys (all Insee household surveys will be migrated to this system within the next 3-4 years), while guaranteeing the specificities of each one. The VTL grammar makes it possible to cover the vast majority of needs in terms of post-collection processing specific to each survey, even in the case of complex

protocols.

Next Features for Surveys

In order to further develop the use of VTL in the statistical surveys, the following activities are currently under study:

- to further develop the concept within complex panel and mutlimode processes (e.g. the use of VTL rules for the post-collection processing necessary for re-collection or change of mode, including through the use of paradata)
- to extend the use of VTL for first post-collection data processing: validating data, codifying variables, computing new variables, bringing together different data and paradata sources
- to develop a tool dedicated to the designer's work: the simplified specification of VTL rules for post-collection processing in a working environment, integrated with the one that already exists for the specification of questionnaires (Pogues).

Beyond Surveys

Other possible uses of VTL are being investigated, in particular:

- the specification of validation checks and data transformations for the acquisition of administrative data or more generally third-party data
- the possibility to automatically generate VTL rules from structural metadata expressed using standards (e.g. DDI or SDMX) in order to assess the compliance between data and associated structural metadata.