# Contents

1  About the BIRD  
  1.1  What is the BIRD  
  1.2  What is out of the BIRD’s scope?  
  1.3  How will the BIRD be a benefit for the European banks?  
  1.4  Which statistical and supervisory regulatory frameworks are (will be) covered by the BIRD?  
  1.5  Cooperation between the authorities and the banks to develop the BIRD  

2  Project Status  

3  BIRD methodology  
  3.1  Introduction  
  3.2  BIRD described data process  
  3.3  Description of the datasets  
  3.4  Transformation rules in the BIRD model  

Annex I: Non-technical introduction to SMCube methodology  
Annex II: New VTL artefacts  
Annex III: New VTL artefacts
1 About the BIRD

1.1 What is the BIRD

Every time a statistical or supervisory regulatory framework is updated or a new one comes into existence, each bank is left to its own devices to interpret it, extract the data from its internal systems and transform them to derive the final figure asked for in a regulation. It is not always straightforward which source data to use and how to process it to produce the number required in Legislation X, Table Y, Cell Z. The greater the misalignment among banks with regard to the meaning of specific sections within a regulatory standard, the more questionable is the quality of the output data and the more difficult are the comparisons between banks. In-depth studying of revised or new legal acts is a costly and time-consuming process for each bank to carry separately, and one for which cooperation among banks can bring significant efficiency gains.

The initiative entitled "BIRD" aims to foster such cooperation in the field of regulatory reporting thus alleviating the reporting burden for the banks and improving the quality of data reported to the authorities. "BIRD" stands for "Banks' Integrated Reporting Dictionary". Its contents, to be published on the BIRD website, are based on a harmonised data model describing precisely the data which should be extracted from the banks' internal IT systems to derive reports demanded by authorities. In addition to this, there are clearly defined transformation rules to be applied to the data extracted from the banks' internal IT systems in order to produce a specific final regulatory figure. The univocal categories of the data to be extracted from the banks' internal IT systems, the so-called "input layer", together with the transformation rules make up the BIRD.

The purpose of the BIRD is to provide a service to the banks. The BIRD is available, as a "public good", to banks and all interested parties (e.g. software houses that develop application packages for financial reporting). The adoption of the BIRD by banks is fully voluntary. It can be used as additional documentation (with respect to regulations and guidelines) or as "active dictionary" for procedures developed by banks. The BIRD represents an "input approach" because it doesn't stop at the regulatory requirements, but goes all the way back to the data in the banks' internal systems. It should be emphasised, however, that banks remain responsible for the organisation of their internal reporting system.
1.2  **What is out of the BIRD’s scope?**

   The BIRD is **not an IT tool** itself nor is the initiative expected to provide such tool. The BIRD does not make any changes to the banks’ internal IT systems. The scope of the BIRD does not cover the mappings of the data from the banks’ internal IT systems to the input layer.

   The BIRD is **not a regulatory act** and it is **not a new rule**. It is a transposition of the legal requirements at a more operational level. In other words, the BIRD provides a formalised representation of how the requirements set in reporting regulations may be satisfied. Its application is fully **voluntary**.

1.3  **How will the BIRD be a benefit for the European banks?**

   The adoption of the BIRD has several **advantages**:

1. Different reports can be produced from a single input layer by applying harmonised algorithms → Lower reporting burden for the banks. Greater consistency and quality of the data. No more need to manage each mandatory data collection in a separate way.

2. Well-defined transformation rules, which include the calculations to obtain certain regulatory figures → A univocal interpretation and clarity of regulations. Enhanced compliance with the regulatory requirements.

3. Decreased time and effort to analyse and satisfy new reporting requirements → Increased efficiency and lower costs.

4. Increased awareness, understanding and interest in what is behind data (what they include, how they are produced) → Improved management and use of data.
1.4 Which statistical and supervisory regulatory frameworks are (will be) covered by the BIRD?

The intention is that several statistical and supervisory reporting requirements will be covered by the BIRD: ECB’s collection of granular credit data and credit risk data (AnaCredit), ECB’s Securities Holdings Statistics (SHS), ECB’s Monetary Financial Institutions’ Balance Sheet Items Statistics (BSI), ECB’s Monetary Financial Institutions’ Interest Rate Statistics (MIR), the needs of other statistics, such as the balance of payments and national accounts, the additional requirements of the Single Supervisory Mechanism and EBA’s Implementing Technical Standards (ITS), which encompasses Common Reporting (COREP) and Financial Reporting (FINREP).

1.5 Cooperation between the authorities and the banks to develop the BIRD

For reasons of efficiency and effectiveness, the BIRD is carried out and maintained by a number of banks participating in the BIRD group in close cooperation with the European Central Bank and National Central Banks. Both banks and authorities have a very strong interest in an appropriate organisation of the information systems of banks, able to produce timely, consistent and high quality data. The production of a meaningful BIRD requires specific knowledge of reporting agents, which calls for joint work and close collaboration between the banking industry and the authorities.
2 Project Status

In 2013 the Statistics Committee (STC) of the European System of Central Banks has investigated the possibility to promote an integrated approach to supervisory and statistical data.

In 2014 an STC Groupe de Réflexion recommended the development, in close collaboration with the banking industry, of a European “input approach” model which would provide a standardised design of the banks’ internal data warehouses to support the data reporting to the authorities.

In 2015 a workshop with the industry, hosted by the ECB, has indicated as the best approach for launching the implementation of BIRD, to cover the new requirements related to the collection of granular credit and credit risk data (hereafter AnaCredit, see Regulation (EU) 2016/867) and to Statistics on Holdings of Securities (hereafter SHS, see Regulation (EU) No 1011/2012 as amended by Regulation (EU) 2016/1384).

Considering the very little time for implementing AnaCredit and SHS it was decided to initiate a light short term solution with a BIRD pilot covering AnaCredit and SHS. The BIRD pilot started in 2015 with some volunteer National Central Banks (NCBs) and banks contacted by each NCB.

The conclusions of the pilot are expected in the first quarter of 2017. Depending on the result, the BIRD will be further developed with the intention to cover all the European statistical and supervisory requirements.
3 BIRD methodology

3.1 Introduction

The BIRD methodology refers to the technique used to provide a formal documentation of the BIRD proposed data model.

The description of the BIRD proposed data model is composed of:

- Description of datasets
- Transformation process
- Technical guidelines on how to populate the datasets and implement transformations.

The BIRD is documenting a dynamic data process, where some data are provided as input, validated and subsequently transformed into other datasets until the final datasets, which are those that need to be reported to the authorities, are generated. Section 2 of this document describes the BIRD data process.

The BIRD provides a formal description of both the datasets and the validation and transformation rules. Section 3 introduces the methodology for the formal description of the BIRD datasets, while section 4 deals with the validation and transformation language.

The technical guidelines on how to populate datasets, which include the conceptual model of the relation between the cubes in the form of an Entity Relationship Model, constitute a separate document, which is included in the BIRD website.

3.2 BIRD described data process

The data process described by the BIRD follows a typical structure with transactional and analytical layers.

A system following the BIRD process would normally start by feeding the input cubes and running the validations. The system would create enriched cubes, which contain the relevant information for generating the final reports, including input information but also new information derived from that input. From the enriched cubes, all reporting requirements would be generated. The reporting requirements should also be part of the BIRD documentation, in the form of output cubes, although their definition is out of the scope of the BIRD; the BIRD simply includes the translation of the output cubes into its data model.

The process described in the BIRD is divided in three phase, of which two are common to all output frameworks to be generated, and one will be specific for each reporting framework.
The enrichment phase finishes when writing the enriched input layer. The enriched input layer will also be described in the BIRD dictionary as a group of datasets.

The transformation process phases can be further broken down in sub-phases:

The BIRD transformation process is described by means of transformation rules, which are further explained in section 4.
3.3 Description of the datasets

The description of the datasets in the BIRD is provided following the SMCube methodology, which is a methodology developed with the objectives of:

- Enabling the description within one dictionary of all types of datasets (registries, dynamic data for supervisory purposes, for monetary purposes…)
- Facilitating the integration of dictionaries developed with different methodologies (like DPM/XBRL or SDMX) in one single dictionary by keeping a high degree of compatibility with the other methodologies

The version 1.0 of the SMCube methodology is close to final, although some refinements are still pending. A non-technical introduction to the main concepts of the methodology is provided in Annex I of this handbook.

The BIRD pilot is using a simplified version of the methodology, where some artefacts (like those in the rendering and provisioning packages) and features (like historicizing and extensibility) have been removed, since for the time being are not going to be used. The formal description of the BIRD datasets is provided in an Access database, which can be downloaded in the BIRD website.

3.4 Transformation rules in the BIRD model

Transformation rules represent a description of logical operations applied to objects of the BIRD database to create new / additional information.

Transformations rules are organised, following the VTL model, in two levels: Transformation scheme and transformation.

Transformations constitute the basic element of the calculations, and consist of a statement assigning the outcome of an expression to a VTL element. Transformations have thus always the following structure:

```
transformationResult := expression
```

Transformations are grouped into transformation schemes, which are sets of transformations aimed at obtaining some meaningful results for the user. Transformation schemes group transformations that share some functional or business purpose. An example of transformation scheme is the derivation of enterprise size. This transformation scheme is formed by several different transformations aimed to give as a result the final enterprise size.

Transformations are defined in verbal and formal form. The verbal description ("Natural language") represents the business perspective trying to avoid specifications regarding the technical implementation. The formal description is based on the "Validation and Transformation Language (VTL)" with minor adaptions to improve usability / readability by the end user and aims at a technical description of the transformation rules independent of the technical implementation.
Regarding the formal description, please note that due to the ongoing developments in the Task force for the Validation and Transformation Language (VTL) and the publication of VTL version 1.1 at the beginning of 2017, changes to the syntax are possible.

Transformation schemes are classified in the following four types:

i. **Validation rules**: Transformations that evaluate a logical condition that data should comply with.

ii. **Derivation rules**: Transformations that create new variables from existing data.

iii. **Generation rules**: Transformations focused on the data preparation based on the formalities described for each output framework.

iv. **Technical rules**: Transformations required for technical purposes, in order to have a complete description of the transformation process, but are not relevant for business users.

Annex II describes additional VTL features used to describe the BIRD transformation process: Procedures, functions and rulesets.

Annex III provides a technical description of the model used in the database to represent VTL.
SMCube is a methodology for defining metadata to describe datasets. Its pivotal role is in defining cubes, which define the structure of a dataset, intended as a set of data organised as a table with fields (columns) and records (rows).

Table 1 provides an example of a dataset with information related to securities.

**Table 1: Securities dataset.**

<table>
<thead>
<tr>
<th>ISIN</th>
<th>Type of instrument</th>
<th>Inception date</th>
<th>Legal final maturity date</th>
<th>Subordinated debt</th>
<th>Nominal value</th>
<th>Fair value</th>
<th>Accrued interest</th>
<th>Currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>YY7365</td>
<td>F_32</td>
<td>16/06/2015</td>
<td>16/06/2035</td>
<td>1</td>
<td>1.000</td>
<td>1.020</td>
<td>10</td>
<td>EUR</td>
</tr>
<tr>
<td>923618</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZZ3941</td>
<td>F_511</td>
<td>05/10/2015</td>
<td></td>
<td></td>
<td>6.000</td>
<td>6.050</td>
<td></td>
<td>USD</td>
</tr>
<tr>
<td>829354</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The minimum information required to describe the structure of the dataset can be summarised with the following four questions:

1. **What are the fields (columns) of the dataset?**

   In the SMCube methodology, the fields of a cube are called variables. The variables are defined independently of the cube, so that a cube will refer to as many variables as it needs; but the same variable may be referred by many cubes, thus allowing for reusability of concepts.

   Referring to Table 1, the variables are: “ISIN”, “Type of instrument”, “Inception date”, “Legal final maturity date”, “Subordinated debt”, “Nominal value”, “Fair value”, “Accrued interest” and “Currency”.

2. **What are the allowed values for each field?**

   In the SMCube methodology, each variable is defined on a domain, which states the allowed values for the variables. Domains can be enumerated, if they provide a list with the allowed values, or non-enumerated, if they provide a data type.

   In the example dataset, some variables are defined on non-enumerated domains, like the “ISIN” (string domain), the “Inception date” (date domain) or the “Fair value” (monetary domain). Three variables are defined on enumerated domains: “Type of security”, “Subordinated debt” and “Currency”; for each of these variables, it is necessary to provide the list of allowed values. The possible values for an enumerated domain are called members in the SMCube methodology. For each member, the dictionary shall provide additional information, like the description of the concept (e.g. what does F_32 mean?)

   But one variable can have different allowed values in different datasets. For instance; the example dataset refers to securities, so the allowed values for “Type of instrument” should only refer to securities, but in a dataset covering more instruments the list of allowed values could be enlarged. In order to allow reusing the variables, the domain will contain all the possible members for variables.
The same holds true for non-enumerated domains, where the subdomain will serve to constrain the allowed values, even if they are not enumerated. For instance, the variable “Fair value” is a monetary variable, but in the context of the securities dataset, its allowed values may be limited to only positive amounts.

In the context of a cube, one variable has to be associated to a concrete subdomain, i.e. the subset of the domain that is allowed in the context of the cube.

3. What is the role of one field within one dataset?

One of the most relevant aspects of the structure of the dataset is what the identifier of the record is or, in other words, what combination of fields makes a record unique. In the example dataset, if nothing is said regarding the structure, applying some business knowledge one may conclude that each record is uniquely identified by the ISIN. But it is also true that one security with one ISIN may be denominated in more than one currency, so if the variable currency is referred to the denomination of the security, it could be the case that more than one record per ISIN is allowed, as long as the currency in each record is different. Thus in order to get a thorough understanding of the described dataset, the role of the variables has to be explicit. In the SMcube methodology, variables that serve as identifiers of the records take the dimension role.

Let’s suppose that in the example the only dimension is the ISIN. The rest of variables may have also different roles, depending on the variables to which they add information. Let’s take again the currency variable as an example, once it is known that it is not a dimension. Without additional information, a user could have two different interpretations of it: It may be adding information to the ISIN (i.e. to the combination of dimensions, in this case only one) specifying the currency in which the currency is nominated, or it may be referred to the monetary amounts, specifying the currency in which the amounts are nominated. In the SMcube methodology, the variables that add information to the combination of dimensions take the observation value role, while the variables that add information only to one variable of the dataset take the attribute role.

It is worth highlighting that one variable can take different roles in different datasets. For instance, the example dataset is disaggregated to the level of security. Another dataset may contain, for one reporting institution, the total securities broken down by instrument.

Table 2: Aggregated dataset.

<table>
<thead>
<tr>
<th>Reporting institution</th>
<th>Type of instrument</th>
<th>Nominal value</th>
<th>Fair value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC F_32</td>
<td>1.000</td>
<td>1.020</td>
<td></td>
</tr>
<tr>
<td>ABC F_511</td>
<td>6.000</td>
<td>6.050</td>
<td></td>
</tr>
</tbody>
</table>

In the context of this dataset, the type of instrument is a dimension, since it is part of the identifier (one record is uniquely identified by the combination of “Reporting institution” and “Type of instrument”).

Summary

With the SMCube methodology, one cube serves to define the structure of a dataset. One cube is a set of variables, for which the allowed values are specified by a subdomain, and that have a role in the context of the cube. The SMcube definition of the dataset 1 could be summarised in the following table:
Table 3: Cube describing the securites dataset.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Subdomain</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISIN</td>
<td>12-character alpha-numerical code</td>
<td>Dimension</td>
</tr>
<tr>
<td>Type of instrument</td>
<td>Types of instruments for dataset 1</td>
<td>Observation value</td>
</tr>
<tr>
<td>Inception date</td>
<td>All dates</td>
<td>Observation value</td>
</tr>
<tr>
<td>Legal final maturity date</td>
<td>All dates</td>
<td>Observation value</td>
</tr>
<tr>
<td>Subordinated debt</td>
<td>Boolean including not applicable</td>
<td>Observation value</td>
</tr>
<tr>
<td>Nominal value</td>
<td>Positive monetary amounts</td>
<td>Observation value</td>
</tr>
<tr>
<td>Fair value</td>
<td>Non-negative monetary amounts</td>
<td>Observation value</td>
</tr>
<tr>
<td>Accrued interest</td>
<td>Non-negative monetary amounts</td>
<td>Observation value</td>
</tr>
<tr>
<td>Currency</td>
<td>ISO 4217 codes</td>
<td>Observation value</td>
</tr>
</tbody>
</table>
Annex II: New VTL artefacts

Procedures

Procedures are aimed at automating common processing tasks, and can be used as a means for shortening the code by replacing common processing tasks with a procedure call.

Procedures take input and output arguments and describe the set of transformations performed with those arguments.

The following example shows a procedure for checking the identifiers that are used in a cube (FRGN_CB) and not in another cube (PRMRY_CB).

```vql
define procedure check_ref_int (in FRGN_CB as dataset, in FRGN_VRBL as string, in PRMRY_CB as dataset, in PRMRY_VBL as string, out valResult as dataset) as {
  FRGN_IDS := FRGN_CB[keep FRGN_VRBL] [rename FGN_VRBL as "ID" ]
  PRMRY_IDS := PRMRY_CB[keep PRMRY_VBL ] [rename PRMRY_VRBL as "ID" ]
  calResult := setdiff (FRGN_IDS, PRMRY_IDS)
}
```

The procedure can be afterwards called with concrete arguments:

```vql
call check_ref_int(TRNSCTNS_CNTRPRTS,CNTRPRTY_ID,CNTRPRTS,CNTRPRTY_ID, V_CNTRPRTY_ID1)
```

Functions

VTL allows extending the available operators by defining functions. Functions take as input some variables, and give as a result a predefined calculation.

The following example shows a function to calculate the carrying amount from the required input variables:

```vql
define function fnctnCrryngAmnt(ACCNTNG_CLSFCTN, FV, GRSS_CRRYNG_AMNT_E_INTRST, ACCRD_INTRST, FV_CHNGS_HDG_ACCNTNG, ACCMLTD_IMPRMNT) as
  if (ACCNTNG_CLSFCTN in (2, 4, 8, 41)) then
    FV
  elseif (ACCNTNG_CLSFCTN in (6, 14)) then
    (GRSS_CRRYNG_AMNT_E_INTRST + ACCRD_INTRST - ACCMLTD_IMPRMNT + FV_CHNGS_HDG_ACCNTNG);
```
This function can be then used to derive new data:

\[
\text{RESULT} := \text{CUBE} \left[ \text{calc fncnCrryngAmnt(ACCNTNG\_CLSFCTN, FV, GRSS\_CRRYNG\_AMNT\_E\_INTRST, FV\_CHNGS\_HDG\_ACCNTNG, ACCRD\_INTRST)} \right. \\
\left. \text{as "CRRYNG\_AMNT" role MEASURE} \right]
\]

**Rulesets**

Rulesets define validation rules between variables that have to be applied to each individual record of a given dataset. Rulesets take as input the variables to be validated, and contain at least one consistency rule that the validations need to comply with. Each rule has two conditions (introduced by the clauses when and then), and the validation will be satisfied if both conditions are satisfied.

As an example, the following ruleset includes two consistency rules between the variables accounting classification and accumulated changes in the fair value due to credit risk

**Define datapoint ruleset** VRC1 (ACCNTNG\_CLSSFCTN, ACCMLTD\_CHNGS\_FV\_CR) {

rule1: when ACCNTNG\_CLSSFCTN in (2, 4, 41) then not isnull(ACCMLTD\_CHNGS\_FV\_CR)

rule2: when ACCNTNG\_CLSSFCTN in (6, 8, 14) then isnull(ACCMLTD\_CHNGS\_FV\_CR)

The ruleset can then be used with the check operator:

\[
\text{VLDTN\_RSLT} := \text{check (Dataset, VRC1)}
\]
Annex III: New VTL artefacts

The representation of transformations in the BIRD database is based on the SDMX information model (see section II, 13.2 Model – Inheritance View, 13.2.1 Class Diagram).

According to the VTL mode, a Transformation scheme is an ordered list of transformations. Therefore such a transformation scheme contains one or many transformations (i.e. one line of valid VTL code). The SDMX model specifies that transformations can contain one or many transformation elements (i.e. the components of this line of valid VTL code). Therefore a transformation element is a constant, an operation or a BIRD model object (i.e. a variable, cube, etc.). In case the transformation element represents an operation such a transformation element itself can have a relation to one or many transformation elements.

The BIRD database contains the complete information about transformation schemes in the sense that not only the decomposition of a transformation scheme into its transformations is stored in the database but also the decomposition of transformations into its transformation nodes according to the SDMX information model.

Example

The following example will try to clarify the current status of the representation of transformations in the BIRD database:

Let’s assume we have a table containing coordinates having the variable x and y which (clearly) relate to some coordinate system. Our transformation scheme’s goal is to derive a new variable distance for all records where x and y are greater than or equal to 0 defined in the following way:

\[ \text{distance} = \sqrt{x \times x + y \times y}. \]

Using VTL syntax we would write the following lines:

\begin{verbatim}
dataset := get("coordinates", filter(x>=0 and y>=0), keep(x, y));
result := dataset [calc sqrt(x*x+y*y) as "distance"];  
\end{verbatim}

The tree structure with respect to the first line is as follows:

\begin{verbatim}
origin ::= [:=
  dataset [DataSet]
  keep [keep]
  filter [filter]
\end{verbatim}
Where a tabulator means that the node is a member of the next level of the tree, for example the nodes “dataset [DataSet]” and “keep [keep]” are children of the root of this tree (i.e. the node “:= [=]”). The term written in brackets is the type of transformation element which can be used to identify constants and BIRD model objects (i.e. variables, cubes, etc.).

Please note that the Boolean condition applied to the filter operator (i.e. “x>=0 and y>=0”) is completely decomposed into its components (i.e. transformation elements) in a structured way in the sense that the Boolean condition can be reengineered from this tree structure.

The decomposition of transformation into its transformation elements supports specific implementations of these transformations. For example in case of a SQL implementation we could apply the following mappings:

- := → CREATE VIEW _______ AS
- Get → FROM
- Filter → WHERE
- Keep → SELECT

Walk the tree and create the corresponding line of SQL code:

```
CREATE VIEW dataset AS SELECT x, y FROM coordinates WHERE x>=0 AND y>=0;
```

Please note that – in order to generate such an SQL statement – one must additionally rearrange the nodes of the tree according to the SQL syntax. Please also note that the elements after each keyword (i.e. CREATE VIEW, SELECT, FROM, WHERE) are similar to the elements represented in the tree structure.
The second line’s tree structure is as follows and could be transformed in a similar way as described in the above paragraph:

\[
:= [:=] \\
\text{result [DataSet]} \\
\text{calc [calc]} \\
\text{dataset [DataSet]} \\
\text{as [as]} \\
\text{sqrt [sqrt]} \\
\text{+ [+] \\
\text{* ["] \\
\text{x [Variable]} \\
\text{x [Variable]} \\
\text{* ["] \\
\text{y [Variable]} \\
\text{y [Variable]} \\
\text{"distance" [StringValue]} \\
\text{y [Variable]} \\
\text{y [Variable]}}
\]

### Representation in the database

The transformation scheme is stored in the table TRANSFORMATION_SCHEME:

<table>
<thead>
<tr>
<th>SCHEME_ID</th>
<th>DESCRIPTION</th>
<th>NATURAL_LANGUAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEST_SCHEME</td>
<td>transformation scheme example</td>
<td>transformation scheme deriving the distance between x and y</td>
</tr>
</tbody>
</table>

Please note that this is a reduced version of the original table, presented for illustrative purposes.

Each individual transformation is stored in the TRANSFORMATION table:
TRANSFORMATION_ID | EXPRESSION | DESCRIPTION | SCHEME_ID
---|---|---|---
3772 | dataset := get("coordinates", filter(x>=0 and y>=0), keep(x, y)) | | TEST_SCHHEME
3788 | result := dataset [calc sqrt(x*x+y*y) as "distance"]; | | TEST_SCHHEME

Using the SCHEME_ID we can connect these transformations with the related transformation scheme (which is similar to stating that "these transformations are children of the transformation scheme with SCHEME_ID 3771").

All Transformation elements are stored in the table TRANSFORMATION_ELEMENT:

<table>
<thead>
<tr>
<th>TRANSFORMATION_ID</th>
<th>ELEMENT_ID</th>
<th>EXPRESSION</th>
<th>TYPE_OF_ELEMENT</th>
<th>LEVEL</th>
<th>PARENT</th>
<th>ORDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>3772</td>
<td>3773</td>
<td>:=</td>
<td>:=</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3772</td>
<td>3774</td>
<td>dataset</td>
<td>DataSet</td>
<td>2</td>
<td>3773</td>
<td>0</td>
</tr>
<tr>
<td>3772</td>
<td>3775</td>
<td>keep</td>
<td>keep</td>
<td>2</td>
<td>3773</td>
<td>1</td>
</tr>
<tr>
<td>3772</td>
<td>3776</td>
<td>filter</td>
<td>filter</td>
<td>3</td>
<td>3775</td>
<td>0</td>
</tr>
<tr>
<td>3772</td>
<td>3777</td>
<td>get</td>
<td>get</td>
<td>4</td>
<td>3776</td>
<td>0</td>
</tr>
<tr>
<td>3772</td>
<td>3778</td>
<td>&quot;coordinates&quot;</td>
<td>DataSet</td>
<td>5</td>
<td>3777</td>
<td>0</td>
</tr>
<tr>
<td>3772</td>
<td>3779</td>
<td>and</td>
<td>and</td>
<td>4</td>
<td>3776</td>
<td>0</td>
</tr>
<tr>
<td>3772</td>
<td>3780</td>
<td>&gt;=</td>
<td>&gt;=</td>
<td>5</td>
<td>3779</td>
<td>0</td>
</tr>
<tr>
<td>3772</td>
<td>3781</td>
<td>x</td>
<td>Variable</td>
<td>6</td>
<td>3780</td>
<td>0</td>
</tr>
<tr>
<td>3772</td>
<td>3782</td>
<td>0</td>
<td>NumericValue</td>
<td>6</td>
<td>3780</td>
<td>1</td>
</tr>
<tr>
<td>3772</td>
<td>3783</td>
<td>&gt;=</td>
<td>&gt;=</td>
<td>5</td>
<td>3779</td>
<td>1</td>
</tr>
<tr>
<td>3772</td>
<td>3784</td>
<td>y</td>
<td>Variable</td>
<td>6</td>
<td>3783</td>
<td>0</td>
</tr>
<tr>
<td>3772</td>
<td>3785</td>
<td>0</td>
<td>NumericValue</td>
<td>6</td>
<td>3783</td>
<td>1</td>
</tr>
<tr>
<td>3772</td>
<td>3786</td>
<td>x</td>
<td>Variable</td>
<td>3</td>
<td>3775</td>
<td>1</td>
</tr>
<tr>
<td>3772</td>
<td>3787</td>
<td>y</td>
<td>Variable</td>
<td>3</td>
<td>3775</td>
<td>2</td>
</tr>
<tr>
<td>3788</td>
<td>3789</td>
<td>:=</td>
<td>:=</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3788</td>
<td>3790</td>
<td>result</td>
<td>DataSet</td>
<td>2</td>
<td>3789</td>
<td>0</td>
</tr>
<tr>
<td>3788</td>
<td>3791</td>
<td>calc</td>
<td>calc</td>
<td>2</td>
<td>3789</td>
<td>1</td>
</tr>
<tr>
<td>3788</td>
<td>3792</td>
<td>dataset</td>
<td>Variable</td>
<td>3</td>
<td>3791</td>
<td>0</td>
</tr>
<tr>
<td>3788</td>
<td>3793</td>
<td>as</td>
<td>as</td>
<td>3</td>
<td>3791</td>
<td>1</td>
</tr>
<tr>
<td>3788</td>
<td>3794</td>
<td>sqrt</td>
<td>sqrt</td>
<td>4</td>
<td>3793</td>
<td>0</td>
</tr>
<tr>
<td>3788</td>
<td>3795</td>
<td>+</td>
<td>+</td>
<td>5</td>
<td>3794</td>
<td>0</td>
</tr>
<tr>
<td>3788</td>
<td>3796</td>
<td>*</td>
<td>*</td>
<td>6</td>
<td>3795</td>
<td>0</td>
</tr>
<tr>
<td>3788</td>
<td>3797</td>
<td>x</td>
<td>Variable</td>
<td>7</td>
<td>3796</td>
<td>0</td>
</tr>
<tr>
<td>3788</td>
<td>3798</td>
<td>x</td>
<td>Variable</td>
<td>7</td>
<td>3796</td>
<td>1</td>
</tr>
<tr>
<td>3788</td>
<td>3799</td>
<td>*</td>
<td>*</td>
<td>6</td>
<td>3795</td>
<td>1</td>
</tr>
<tr>
<td>3788</td>
<td>3800</td>
<td>y</td>
<td>Variable</td>
<td>7</td>
<td>3799</td>
<td>0</td>
</tr>
<tr>
<td>3788</td>
<td>3801</td>
<td>y</td>
<td>Variable</td>
<td>7</td>
<td>3799</td>
<td>1</td>
</tr>
<tr>
<td>3788</td>
<td>3802</td>
<td>&quot;distance&quot;</td>
<td>StringValue</td>
<td>4</td>
<td>3793</td>
<td>1</td>
</tr>
</tbody>
</table>
Transformations parser

We have developed a VTL transformations parser in order to create and visualize tree structures, and produced the output data model required for the BIRD. The parser is done with Java, and available in GitHub.