International Journal of Engineering & Technology, 7 (4.31) (2018) 234-241

SPC

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A Library for Articulating the Measurement Streams with Columnar Data

Mario Diván[1]*, María Laura Sánchez Reynoso[2]

Economics and Law School, National University of La Pampa, Santa Rosa, Argentina
**Corresponding author E-mail: mjdivan@eco.unlpam.edu.ar*

Abstract

The CINCAMI/Measurement Interchange Schema (MIS) organizes jointly data and metadata generated from the heterogeneous measurement devices under the same data stream. The Processing Architecture based on Measurement Metadata (PAbMM) is a data stream engine which processes the data streams organized under the CINCAMI/MIS schema. PAbMM is able to replicate in real-time each measurement stream but limited to C-INCAMI/MIS. This constitutes a use limitation of CINCAMI/MIS because the reading, using and writing of the measures was previously performed just by PAbMM. The CINCAMImisConversor library extracts this functionality with the aim of fostering the using along with any measurement project who need it. The functionality extraction was guided by PAbMM´s internal behavior, which is documented through SPEM metamodel. This allowed defining to the CincamimisConversor's object model, its implementation and carry forward a discrete simulation for having an associated time reference. As a contribution, the library allows the data format conversion from the CINCAMI/MIS streams to the columnar organization. The library could translate 4900 measures in approximately 11 ms.

Keywords: *Columnar Data; Data Interchanging; Data Stream Processing; Measurement; Evaluation.*

1. Introduction

Nowadays, the current economy implies a dynamism and periodical adaptation in the customers, the competitors, the suppliers among other different actors, who integrate the market itself [1]. From the general system theory, a market could be viewed as an abstract system, which incorporates different elements, their interactions each other, the environment, the ins, the outs and the feedback between the environment and its associated system out [2]. In this sense, the idea of getting knowledge about a given system implies to measure and evaluate the system as an entity under analysis. Following the idea associated with the general system theory, the internet of thing (or even, the internet of everything) is a conception in which each data source is heterogeneous [3]. For this reason, it is highly likely that the measurement and evaluation process be carry forward on a context with heterogeneous data sources in which each one has its own data type, format, unit, scale, among other associated aspects. Thus, an integration of different kinds of measures would be necessary for adequately interpreting the data from the syntactic and the semantic point of view from the perspective related to the entity under monitoring.

The Processing Architecture based on Measurement Metadata (PAbMM) is a data stream engine specialized in Measurement and Evaluation (M&E) projects [4]. The data processing is guided by the metadata embedded in each data stream coming from the heterogeneous data sources. Each aspect under monitoring is defined in terms of a measurement and evaluation framework called C-INCAMI (Context-Information Need, Concept Model, Attribute, Metric, and Indicator) [5, 6]. The C-INCAMI framework defines the terms, concepts, and relationships necessaries for implementing a M&E project. The M&E project definition is guided by the GOCAME (Goal-Oriented Context-Aware Measurement and Evaluation) strategy [7].

For gaining homogeneity along the interchanging of measurements between the data sources and PAbMM, the CINCAMI/MIS (Measurement Interchange Schema) schema is used. The schema is based on the M&E project definition and it jointly incorporates inside the stream, the data and necessary metadata for matching each concept under monitoring in relation to the data source for a given entity [8]. The CINCAMI/MIS library is an open source library under the terms of the Apache 2.0 license, released in 2018 and freely available from GitHub [9]. The aim of this library is to foster a consistent measurement interchanging among different systems, keeping the track associated with the concepts, the data sources, and the monitored entity. The data interchanging formats related to the library are JSON (*Javascript Object Notation*) and XML (*eXtensible Markup Language*). Additionally, in [9] a discrete simulation associated with the processing times is available.

In this work, the CINCAMI Conversor library is introduced. This library is fully written in Java 8 language and it is released under the terms of the Apache 2.0 license. The library is tiny and freely available from GitHub. As the main contribution, this library allows the data format conversion from the C-INCAMI/MIS streams to the columnar-organization. Thus, the library incorporates the columnar organization as a target of the conversion, because such the data stream processing (i.e. Apache Storm [10]) as the organizational memory storing (i.e. Apache HBase [11]) is oriented to the tuple. Moreover, in PAbMM the statistical computing is based on the R software [12] and the columnar organization is used for the real-time data interchanging between the applications. Because the library is open source, the idea is to foster its extension by the incorporation of other target data interchange formats.

The paper is organized into six sections. Section 2 presents some related works. Section 3 synthesizes the processing architecture based on measurement metadata. Section 4 introduces the CINCAMI/Measurement Interchange Schema and the cincamimis library is synthetically described. Section 5 introduces the CincamimisConversor library. Additionally, a discrete simulation for analyzing its associated times is shown. In Section 6, the conclusions and future works are outlined.

2. Related Works

Esteves, Mousallem et al [13] saw that even when a lot of their machine learning experiments generated outputs, the data interchanging was really a key aspect in comparing them. For this reason, they develop the MEX vocabulary. The MEX's aim is to build a lightweight public vocabulary for interchanging information about the experiments, which allows incorporating a minimum set of meta-information (or metadata) for a better description of the interchanged data. The MEX vocabulary is composed by three sub-vocabularies: a) MEX-Core: It models the essential concepts for describing the steps, phases, and datasets used in each stage along a given experiment, b) MEX-Algorithm: it represents the context in which an algorithm was performed, c) MEX-Performance: it is responsible for representing the concepts associated with the experimental results. However, the vocabulary has no an associated formal strategy which allows guiding step by step in its application by mean of the interpretation of each concept. On the one hand, the CINCAMI Conversor library shares some aspects because the terms, concepts, and relationships are defined based on the C-INCAMI framework, which has an underlying ontology as an analogy of the MEX vocabulary. But on the other hand, there are some differences such as 1) The measurement and evaluation projects are defined using the GOCAME strategy for adequately interpreting each concept in a given order. It is not present in MEX vocabulary; 2) PAbMM uses the M&E project definition for interpreting the measures and making a decision when being necessary (e.g. for detecting miscalibration in a sensor). In MEX the vocabulary is oriented to describe the results or algorithms, but in PAbMM the metadata are used for guiding the real-time data processing; 3) PAbMM incorporates the classifiers for online detecting the typified situations using jointly data and metadata. On the contrary, MEX is oriented to the offline context; and 4) The Cincamimis and cincamimisConversor libraries allow implementing and extending this kind of strategy for the measurement interchanging in other application fields. MEX is oriented specifically to the machine learning.

Huang, Lange et al [14] proposes the real-time transformation from xml streams to Resource Description Format (RDF) using XPath-based mappings. As a coincidence, the work is oriented to real-time xml data stream processing and transformation. However, the data processing is performed on the data itself without taking into consideration its meaning. In the library, the data are given in XML or JSON data format jointly with their metadata which guides the translating. That is to say, the metadata allow transforming from the json/xml data format to a columnar data organization identifying each metric as attribute quantifier in an entity.

Sun, Franklin et al [15] describe a strategy for the agile reconstruction of tuples (or row) from the columnar data organization. They developed the Generalized Skipping-Oriented Partitioning (GSOP) framework with the aim of implementing a hybrid data skipping in relation to the row-based and column-based tradeoffs. Even when the CincamimisConversor library is able to get the row or column through the using of the hash techniques, it has not implemented a data skipping strategy. This will be considered as future work in the library, with the aim of getting improvements in the data access performance.

3. The Processing Architecture based on Measurement Metadata

The Processing Architecture based on Measurement Metadata is a semi-structured stream engine specialized in M&E projects [4]. The internal behavior was modeled by processes using the SPEM (System and Software Process Engineering Metamodel) [16] specification and fully documented in [17, 18].

Initially, it is necessary to define the M&E project using the C-INCAMI framework through the GOCAME strategy [5, 6, 7]. Once the project is defined, the entity under monitoring is identifiable, its context is defined, and their quantifiable attributes and associated metrics are known. The attributes could be understood as a set of characteristics which allow describing to the entity, while the metrics are the mean for quantifying each attribute or context property. For example, if the entity under monitoring is an outpatient, the attributes could be the cardiac frequency, the systolic pressure, among others. Thus, the metric is defined to get a value from the cardiac frequency from an outpatient using a given device.

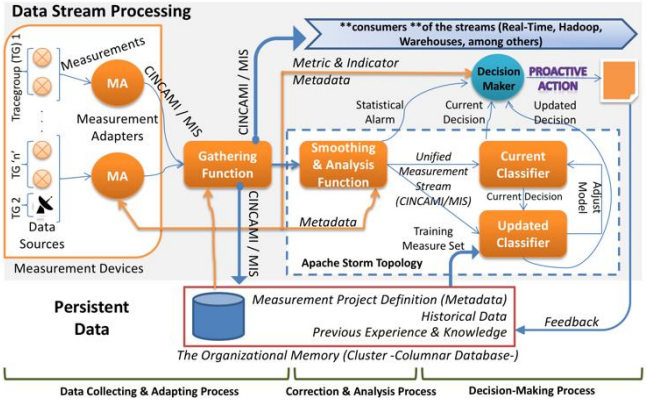

Fig. 1: The Processing Architecture based on Measurement Metadata

At the end of the project definition, each manager establishes which device will provide value for each metric along the measurement process and for a given entity. Thus, the project definition is considered as metadata because help to establish the relationship among each data and its associated concept inside the project. The metadata are always available in memory for each component of a given project along the processing architecture.

Figure 1 shows the main organization related to the processes once the project definition was made. The main processes are related to a) *The data collecting and adapting process*: It is responsible for collecting and integrating the data coming from the heterogeneous data sources and to make available them to the gathering function in PAbMM; b) *The correction and analysis process*: it is responsible for the smoothing and statistical analysis of each measurement stream; c) *The decision-making process*: It is responsible for catching each alarm triggered from the statistical analysis or the online classifiers, analyze them and to solve whether corresponds to a typified situation under supervision or not.

Along the data collecting and adapting process, the measures are collected from the heterogeneous data sources (i.e. the devices) using a component named *Measurement Adapter* (MA in figure 1). Using the project definition, the MA establish the correspondence between the measure and the metric. Once the collecting has finished, the MA send jointly data (i.e. measures) and metadata (tags associated with the identification of each concept along the M&E project) inside a stream under the CINCAMI/MIS schema to the gathering function. The gathering function receives the CINCAMI/MIS streams from different measurement adapters. At the same time in which the measures arrive at the gathering function, this function: 1) It replicates the data to the subscribers using Apache Kafka [19]; 2) It stores the CINCAMI/MIS in the organizational memory on Apache HBase; and 3) It passes the data

stream to the Apache Storm topology for continuing the online data analysis.
Both the Smoothing and analysis process and the decision-making process are embedded in the Apache Storm topology [20]. The smoothing and analysis function uses Redis [21] as a memory cache and R software [12] for the statistical computing. The streams are processed as windows, and it allows counting with partial results related to each entity under analysis. Using R, the architecture drives on each measurement stream the descriptive analysis, the correlation analysis, the principal component analysis, among others. In this situation, the Cincamimis Conversor library would be especially useful, because allows transparently translating each CINCAMI/MIS stream to the columnar-organization/Key-Value expected by R or Redis respectively, without incidence in the Apache Storm topology. If some atypical situation is detected, then an alarm is sent to the decision maker (See figure 1).
Once the smoothing and analysis function has finished, the CINCAMI/MIS stream will be analyzed by the online classifiers based on the Hoeffding Trees from MOA (Massive Online Analysis, See figure 1) [22]. The current classifier will make a decision based on its own training. The updated classifier will make a decision previously incorporating the new CINCAMI/MIS stream. Both decisions are compared using a ROC (Receiver Operating Characteristic) curve [23] and it will be kept the decision associated with the classifier with the bigger area under the curve. In this way, the classifier with bigger area under the curve will be the new current classifier (See figure 1, "Adjust model"). On the contrary, if the updated classifier has a lesser area under the curve than the current classifier, the last will continue as the current classifier and the replacement will not be made.
In each received alarm, the decision-maker analyzes if the alarm corresponds with a typical situation modeled on the base of its training. If so, previous to send an alarm, the decision maker will search in the organizational memory recommended courses of actions associated with previous experience or knowledge. In this way, when the decision maker sends an alarm for the given situation, the recommended courses of actions is attached. For optimizing the searching, PAbMM incorporates the searching based on structural and behavioral coefficients in relation to the M&E project definition, to the effects to keep in memory as cache, the recent information with similarity in relation to the entity under analysis [24].
Mainly, PAbMM is oriented to the real-time data processing and the online measurement replication; However, all the volume of measurements is stored in the organizational memory and it could be queried through the historical data services on demand (See figure 1).

4. The Measurement Interchange Schema

For better understanding, this section is presented in two subsections. First sub-section introduces the object model associated with the measurement interchange schema. The second subsection synthesizes the Cincamimis library.

4.1. The CINCAMI/MIS version 2

The Measurement Interchange Schema named CINCAMI/MIS, it is based on the C-INCAMI measurement and evaluation framework. In this sense, the specific concepts, terms, and relationships among them related to the measurement process are taken from the underlying CINCAMI´s ontology. The aim of CINCAMI/MIS is to foster the measurement interoperability and make easier the processing from heterogeneous data sources keeping the traceability and consistency in relation to the entity under monitoring defined in a M&E project [8]. In this context, the consistency refers to that the data processor (e.g. the gathering function in figure 1) knows the expected unit, scale, value domain, typical values, among other aspects for each attribute related to the entity under monitoring which is defined in the M&E project.

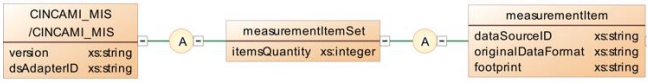

Fig. 2: The Upper level of the CINCAMI/Measurement Interchange Schema version 2. "A" implies a set of subtags without a given order.

The upper level of the measurement interchange schema is identified with the *CINCAMI_MIS* tag (See figure 2). It has two associated properties: *version* and *dsAdapterID*. The version property identifies the version of the measurement interchange schema (e.g. currently 2.0), while the *dsAdapterID* is a representative ID related to the measurement adapter (See MA in figure 1). Under the *CINCAMI_MIS* tag, a set of a *measurementItemSet* tag can be found which represents a set of measurement item. Each *measurementItem* tag has three properties: the *dataSourceID*, *originalDataFormat*, and *footprint*. The *dataSourceID* property represents an ID for the data source related to the measures (e.g. the sensor). The *originalDataFormat* property identifies the original data format in which the data was provided to the measurement adapter. The *footprint* property represents a hash value for integrity control of the described measures under the *measurementItem* tag.

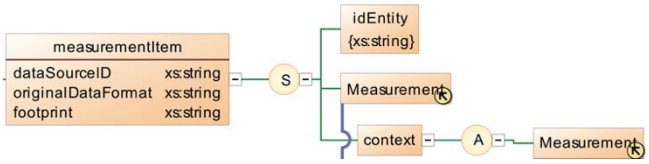

Fig. 3: The *measurementItem* tag in the CINCAMI/Measurement Interchange Schema version 2. "A" implies a set of subtags without a given order. "S" implies a set of tag in the determined order.

Under the *measurementItem* tag (See figure 3), there are three tags: *idEntity*, *Measurement*, and *context*. The *idEntity* identifies the ID related to the entity under monitoring along the M&E projects. The *Measurement* tag contains information about the measurement itself (See figure 4). The *context* tag contains a set of *Measurement* tags related to the quantification associated with the context properties of the entity under analysis. The measurement for a given entity attribute and the context properties are measured at the same time (they are simultaneous) when they are under the same *measurementItem* tag. The *Measurement* tag related to the entity attributes and the context properties have the same structure, which is possible to see in figure 4.

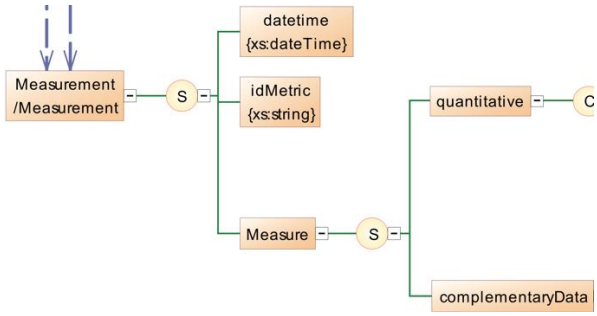

Fig. 4: The Measurement tag in the CINCAMI/Measurement Interchange Schema version 2. "S" implies a set of tag in the determined order. "C" implies that one of a set of tags should be chosen.

Each measurement has a) a given time in that was taken from the data source (the *datetime* tag); b) the entity´s attribute from the M&E project which is quantified (the *idMetric* tag); and c) the *Measure* tag which jointly contains the quantitative measure and their complementary data. Synthetically, each measure could have a set of complementary data under the form of image, audio, video, plain text or geographic information under the Geographic

Markup Language (GML) [25]. The complementary data are optional, and for that reason not always they are present in each measurement stream. The design reason is that each complementary datum implies an overhead in the real-time processing, and this is valid just when they are necessaries.

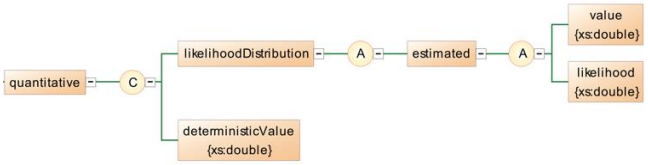

Fig. 5: The Quantitative tag in the CINCAMI/Measurement Interchange Schema version 2. "C" implies that one of a set of tags should be chosen. "A" implies a set of subtags without a given order.

For the measure interchange schema, all the measures are quantitative. However, a measure could be deterministic or estimated and the choice is mutually exclusive. On the one hand, when a measure is deterministic, just one value is given. On the other hand, when the measure is non-deterministic, a likelihood distribution is informed as a set of estimated tags. Each estimated tag contains a <value, likelihood> pair.
Finally, as it is possible to see along the figures 1 to 5, the measurement interchange schema constitutes an essential asset in the data interchanging from the heterogeneous data sources to the data processor in PAbMM. The schema allows: a) verifying the integrity of the content for detecting errors in the transmission (See *footprint* property in figure 3), b) keeping the data traceability (See the *dsAdapterID* property in figure 2 and the *dataSourceID* property in figure 3), c) incorporating complementary data when it is necessary (See the *complementaryData* tag in figure 4), d) managing deterministic and non-deterministic measures, among other additional aspects explained in [8].

4.2. The CINCAMI/MIS Library

The aim of the library is to foster the consistency, traceability and the integrity in the measurement interchange based on the C-INCAMI framework. The library is able to translate from the C-INCAMI/MIS object model from/to XML and JSON, incorporating GZIP compression [8, 9].
The idea is that independently of the data source, the measurements could be interchanged among the systems which need to process them, keeping the data traceability and being able to detect any error in the transmission as was shown in the previous subsection. In this sense, it is worthy to mention that the library could be used independently of PAbMM, and for that reason was modularized as an independent component.
The CINCAMIMIS library was developed in Java 8 and released in March of 2018 under the terms of the Apache 2.0 license. The library is small, open source and it is freely available on GitHub (https://github.com/mjdivan/cincamimis). It uses the Java Architecture for XML Binding (JAXB) for translating the C-INCAMI object model from/to XML data format (See figure 6). The only dependencies are associated with: a) GSON [26] for translating to/from the extended C-INCAMI object model and JSON data format, and b) The gson-javatime-serialisers package for serializing the java.time objects from/to JSON data format [27].

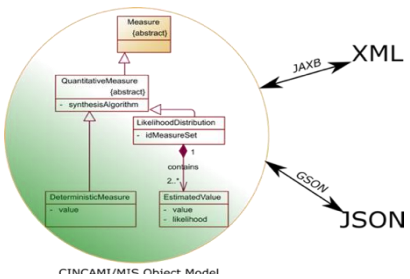

Fig. 6: The translation alternatives included in the CINCAMI/MIS library.

Each embedded concept in the measurement interchange schema has its correspondence in the object model based on the C-INCAMI framework. In this way, a CINCAMI/MIS stream could be translated from/to XML and from/to JSON again, even when the original data format was XML or JSON and not the object model. That is to say, the translation is bidirectional between the object model and XML and the same in relation to JSON (See figure 6). For more details about the object model, it is possible to refer to [9].
In [9] a discrete simulation was performed through the variation of the measures, starting from 100 to 1000 measures inside of a given CINCAMI/MIS stream. Table 1 synthesizes the simulation information in relation to the times and sizes associated with the JSON data format.
Table 1 synthesizes the associated times for converting one stream arriving under C-INCAMI/MIS but in JSON data format (with the number of the indicated measures), to the C-INCAMI/MIS object model representation. The initial times are higher than times associated with the streams with an upper number of measures because the required start-time for the communication has high impact at the beginning of the simulation and it has not been discounted. In this sense, the processing times upper than 200 measures seems to be more stable than before.

Table 1: Times and sizes related to the conversion from the JSON data format to the CINCAMI/MIS Object Model (OM). Adapted table from [9].

#Measures	JSON→OM (ms)	Original Stream (KB)	Compressed Stream (KB)
100	71.95	54.52	4.79
200	77.72	108.32	9.18
300	40.11	162.11	13.53
400	34.58	215.90	17.89
500	22.11	269.29	22.27
600	24.75	323.47	26.61
700	26.88	377.25	30.95
800	50.27	431.01	35.28
900	43.92	484.76	39.64
1000	40.86	538.63	44.06

The difference between the compressed and uncompressed size in Table 1 is around of the order of 12 times in mean. It is worthy to mention that just the decompression average time is 0.85 ms, being 1.25 ms in a stream with 1000 measures, and 0.29 ms in a stream with 200 measures [9]. Figure 7, and on the one hand, it shows the evolution related to the compressed size of each stream (dotted line, right vertical axis) when the number of measures grow-up. On the other hand, the figure exposes the processing time in ms associated with the conversion from JSON to the object model (continuous line, left vertical axis), in each stream and when the number of measures grow-up. It is interesting to mention that even when the compressed size seems to be grow-upping in a linear way, the processing time seems to be varying in a range of 20 to 80 ms.

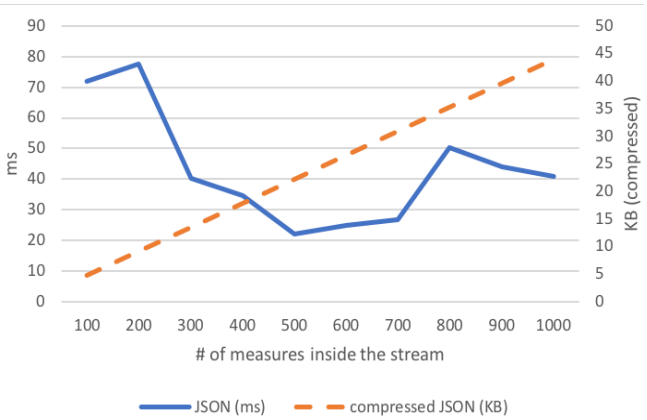

Fig. 7: CINCAMI/MIS Library: The Processing Time and Size Evolution in relation to the variation in the number of measures inside the stream.

Thus, any user which use this library, it is under knowledge about the processing times and estimated sizes for each stream depending on the volume of measures, which allows the posterior analysis related to the application fields. Clearly, any user could be downloading the library from GitHub and it carries forward its own tests for evaluating the pertinence or not of the library in a given area

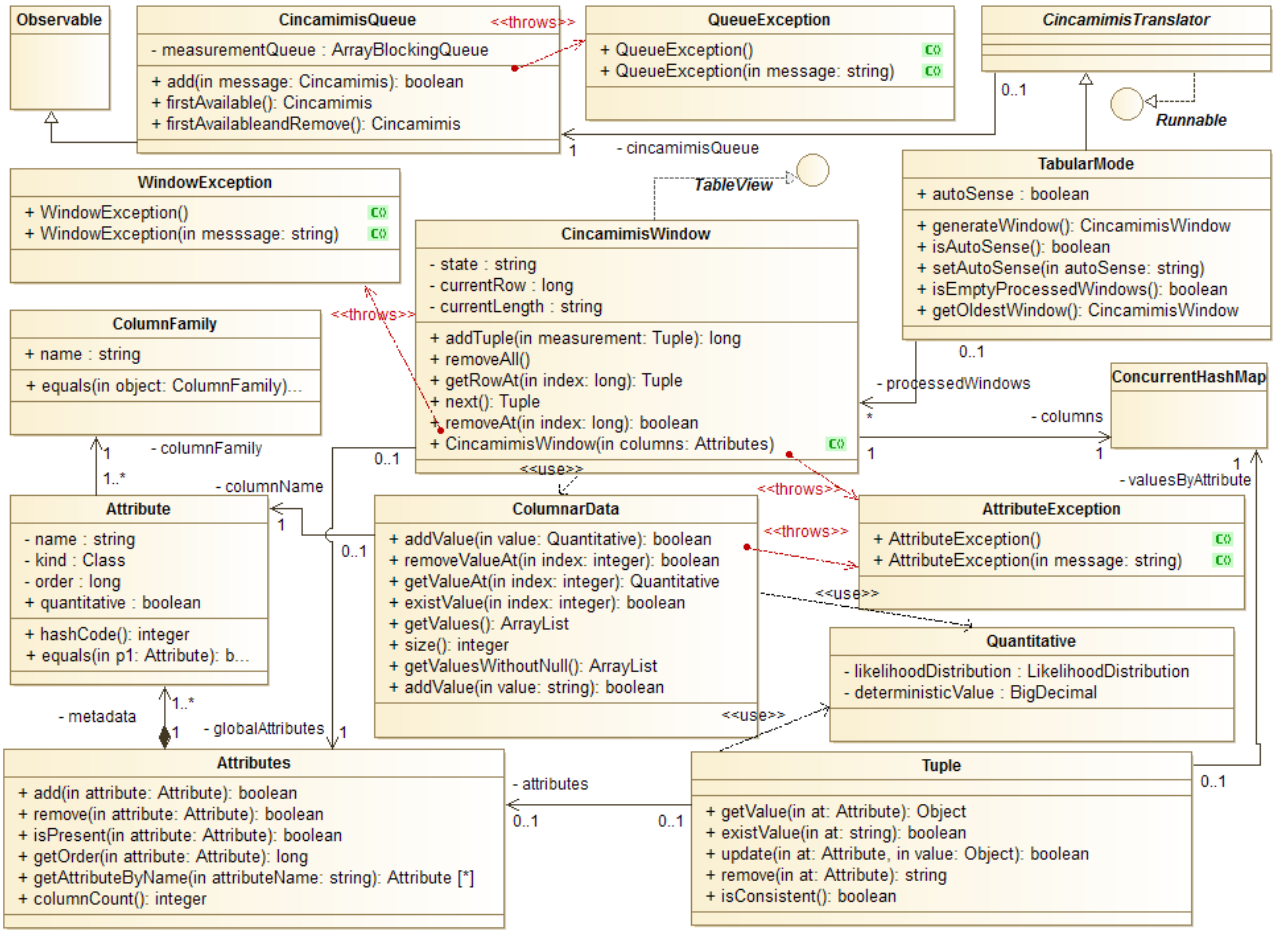

Fig. 8: Main concepts associated with the object model of the CincamimisConversor library

5. The CINCAMI/MIS Conversion

The CincamimisConversion is a library fully written in Java 8 language and released under the terms of the Apache 2.0 license. It is freely available from GitHub (https://github.com/mjdivan/cincamimisConversor). The library aim is fostering the use of CINCAMI/MIS as measurement interchange schema among different systems which require the online monitoring (without restrictions and under the open source principles).

This first version allows the data format conversion from the C-INCAMI/MIS streams to the columnar-organization. Thus, the library incorporates the columnar organization as a target of the conversion, which is particularly useful in the gathering function (See figure 1) when the real-time communication with R software should be made for the statistical computing.

This section is organized into two subsections. The first introduces the object model representation related to the concepts, terms, and relationships embedded in the library. The second shows the obtained results from the discrete simulation.

5.1. CincamimisConversor: the Object Model

The *CincamimisConversor* library is based on the C-INCAMI framework and its underlying ontology. In this sense, there is a common understanding about the terms, concepts, and relationships related to a measurement project. That is to say, concepts such as Metric, Scale, Unit, Calculation Method, among others jointly with their relationships are identified from the structural and semantic point of view. Additionally, the *CincamimisConversor* library is based on the object model related to the *Cincamimis* library for understanding the translating. Because the *Cincamimis* library is based on the C-INCAMI framework too, by mean of the use of the *CincamimisConversor* library, it is possible to read the CINCAMI/MIS streams for interpreting each concept-value pair.

The core concept in the *CincamimisConversor* library is associated with the *CincamimisTranslator* class. This abstract class is responsible for grouping the common behavior for any kind of derived translator (current or in the future). As shown in figure 8, each instance of the *CincamimisTranslator* class has just one associated instance of *CincamimisQueue* class. The *CincamimisQueue* instance receives and queues the *Cincamimis* instance in arrival order from the translation between measurement streams and the *Cincamimis's* object model. In PAbMM the translating happens in the gathering function (See figure 1).

Each *CincamimisQueue* instance is periodically read by a derived class of *CincamimisTranslator*. In each reading, the aim is to convert the *Cincamimis* instance to the wished data format. For this reason, each *CincamimisQueue* instance inherits the *Observable* class from Java. Eventually, the *CincamimisQueue* instance is able to raise a *QueueException* when an unwished situation happened (e.g. when the queue become full).

The *TabularMode* class is a specialization of the *CincamimisTranslator* class. Even when it is oriented to manage the relational data, it has a columnar data organization (See figure 8) as background. Its aim is to read each *Cincamimis* instance from the corresponding queue and translate it to a *CincamimisWindow* instance. The *CincamimisWindow* class is a representation of a logical window in terms of the data stream engine [28]. Thus, if the consuming rate would be exceeded, then the oldest tuple is discarded, and the new tuple is added at the end of the window. The tuple is interpreted as a group of attributes related to a given record. Because the *CincamimisWindow* class implements the *TableView* interface, each *CincamimisWindow* instance is able to show the data in terms of the tabular or columnar data organization.
Because the data management in the background is really columnar, the responsibilities are managed through the *ColumnarData* class. The concept of a tuple is associated with the *CincamimisWindow* class for implementing the *TableView* interface. Thus, thanks to the columnar organization, the data interchanging among PAbMM with the R or Redis software is easier than before. This allows modularizing the data format translation in architectures such as PAbMM, minimizing the associated operative risks and fostering the reusability of this kind of libraries.

5.2. CincamimisConversor: the Simulation Results

The library was tested on a virtual machine with the next configuration: Windows 7 Professional 64-bit, 2 CPUs, 8GB of RAM, and 48GB SSD disk. The virtual PC ran on Virtual Box 5.2.0 on MacOS High Sierra 10.13.3. The host is a MacBook Pro 16GB of RAM, Core i7 2.9Ghz, and 500GB SSD disk. The associated times were collected taking the time before and after of a given operation by mean of the *nanoTime* method of the System class of the Java Virtual Machine.
The test was planned as follows:

1. The number of measures was varied from 200 to 4900 inside each stream.
2. Each IDs was artificial (e.g. entities, metrics, etc.) and the measures were obtained using a pseudo-aleatory way through the class *java.util.Random*.
3. The measures multiple of 2 had not an associated context information. While the measures whose order was not multiple of 2, incorporated a likelihood distribution as an estimated measure inside the measurement context. The likelihood distribution obtained the values from the *java.util.Random* class and the same likelihood distribution was established for all the measures.
4. A *CincamimisQueue* instance was initialized.
5. Five *TabularMode* instances were initialized and linked with the queue. The number five was arbitrary and it did not have a particular reason.
6. Once the measurement streams have been generated and stored under the C-INCAMI/MIS object model, they were fed to the queue in an irregular time period (i.e. in a random way). Each time a *Cincamimis* instance was detected in the queue:
   a. The translation time from the object model to columnar-organization was measured,
   b. The total length of records incorporated in the table is shown in console for a given quantity of measures

Thus, the simulation varied from 100 to 4900 measures collecting the measured processing times. Figure 9 synthesizes the behavior associated with the time processing throughout the variation of measures in each object model.
The outlier associated with the first time for 100 measures in the "Processing Times" (continuous line, left vertical axis) in Figure 9 is highly likely that be related to the communication start-up time because the same time is not reproducible along the simulation. However, there are at least five outliers associated with the next number of measures: 600, 1100, 2100, 2700 and 4200. Those five peaks could be related to the garbage collector of the JVM.
The quantity of processed measures by a time unit (dotted line, right vertical axis) has a positive tendency, which would imply that an upper quantity of measures is processed in the same time unit when the volume of measures grow-up in each stream. However, it presents the inverse peaks in relation to the outliers associated with the processing times, which indicates that in each peak the processing capacity is possibly affected by an external factor. This is because at each peak, the processing times grow-up in a significative way in relation to the tendency (but the processed records by unit time goes down).

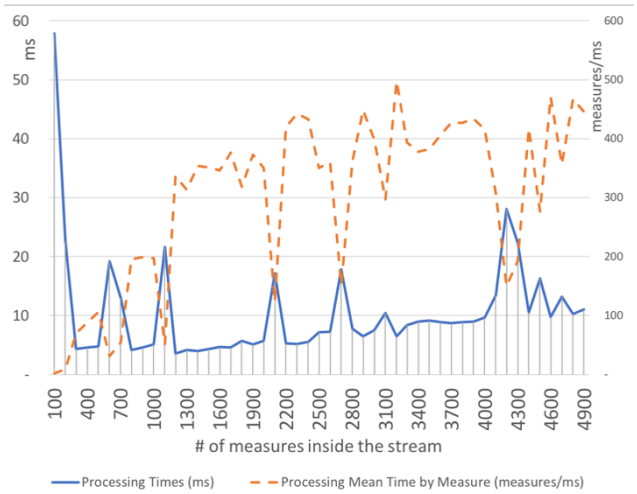

Fig. 9: CincamimisConversor Library: Evolution of the processing time in function of the number of measures.

The completeness was checked as follows: a) From each *Cincamimis* instance the number of measures for each metric was obtained, b) Once the translation was made, the number of records for each metric (column or variable) from the table view was obtained, c) From the comparison between the quantities, it was possible to detect the eventual absence and/or inconsistency. All the cases were coincident in term of quantity of records. The accuracy was verified comparing the arithmetic mean, trimmed mean, standard deviation, kurtosis, skewness, minimum and maximum among each metric (column or variable) in its translated way versus the original way. Any difference along the mentioned measures would implicate some kind of alteration in terms of accuracy. In this case, no difference was found related to the translating. However, in case of differences, the statistical measures would allow detecting the kind of problem in relation to the data distribution.
From the results shown in figure 9, the library is able to convert 4900 measures in 11ms, while the rate for 300 measures is 4.31ms. It implies that the library gains scale economy, optimizing the available resources, which is appreciable along the associated processing times. That is to say, if the scale economy would be not present, then the processing rate should be constant. Taking what was said as hypothesis: a) The associated processing time for 300 measures is 4.31ms, which gives an approximated rate of 69 (measures/ms) or 0.01437 (ms/measure); b) Based on the linearity, the expected time for processing 4900 measures should be similar to 4900 measures * 0.01437 (ms/measure), which would give 70.4 ms; c) Comparing the measured time from figure 9 with the previous calculus, it is possible exposes the scale economy. That is to say, the library consumed 11 ms for processing 4900 measures (i.e. 11 ms /4900 measures = 0.0022 ms/measure) but not 70.4 ms as it was expected in terms of linearity.
The Cincamimis and CincamimisConversor libraries could be jointly used. While the Cincamimis library allows converting from JSON or XML from/to the C-INCAMI/MIS object model; the CincamimisConversor library allows the transformation between

the C-INCAMI/MIS streams and the columnar-data organization. It is important for PAbMM, particularly in relation to the gathering function (GF) and the analysis and smoothing function (ASF) because allows isolating in a library part of its functionality (See Section 3).
It is important to remark that the results have emerged from the discrete simulation, and for that reason, they are not statistically valid. However, the results are used as a concept proof. They allow giving an idea related to the times and possible application fields.

6. Conclusions and Future Works

In this work, the Processing Architecture based on Measurement Metadata was synthesized. The processing architecture is a data stream engine specialized in measurement and evaluation projects. All measurement and evaluation projects are defined in terms of the C-INCAMI framework through the GOCAME strategy. The collected measurements from sensors are organized under a CINCAMI/MIS stream by the measurement adapter, which then is informed to the gathering function in PAbMM.
The Cincamimis library allows the translation among the CINCAMI/MIS object model and XML or JSON incorporating compression by GZIP. The library aim is to foster the interoperability, traceability, and consistency along the measurement interchange, but independently of the kinds of data sources. Thus, CINCAMI/MIS is an interesting schema for incorporating the data traceability and integrity verification independently of the measure providers. Nowell, for fostering its integration with other systems, it is necessary a way in which the CINCAMI/MIS object model can be translated from/to other data formats, such as the columnar-data organization. In this point, the CincamimisConversor library allows translating the CINCAMI/MIS stream in a columnar-data organization to be used in other kinds of software, such as R software (e.g. as a data frame).
The CincamimisConversion library was presented in section 5. The associated object model was introduced, and the simulation results are synthesized. This library allows isolating and encapsulating the translation functionality related to the CINCAMI/MIS streams and the columnar-data organization. In this way, the functionality is useful for the GF and ASF in PAbMM but also for others software who want integrates it. Nowadays, PAbMM is just another user in relation to the Cincamimis and CincamimisConversion library. For example, some cases in where the Cincamimis and CincamimisConversor libraries are used inside of PAbMM: a) When the R data frame should be created for computing the statistical measures on a given measurement window; b) When the synopses associated with a given entity under monitoring should be updated in Redis (memory cache in PAbMM); c) When data from the historical data repository should be served as table to some user; d) When it is necessary to replicate the measurement stream as set of tuple (e.g. Apache Kafka); among other cases.
The libraries allow gradually going down the complexity in PAbMM, encapsulating a defined functionality which is able to be useful in other software. At the same time, it fosters the gradual opening of PAbMM with its interoperability and extensibility.
The results obtained from the discrete simulation are useful for giving an idea associated with the application cases (e.g. an outpatient monitoring). In this sense, the CincamimisConversor library is able to translate a stream with 4900 measures in just 11 ms. The library is able to get a scale economy when the volume of measures grow-up in each stream. That is to say, better unitary processing rates are obtained when the size of the stream grow-up. For example, in a stream with 300 measures, the associated rate is 0.01437 (ms/measure), but when the stream has 4900 measures, the processing rate is 0.0022 (ms/measure). The results were obtained from a discrete simulation, and for that reason, they are used as a concept proof. However, they should not be considered statistically validated results.
As future work, the CincamimisConversor library will be gradually extended through new data formats. Additionally, the analysis of the data skipping strategies for the tuples reconstruction will be considered.

Acknowledgement

This research is partially supported by the project 278/16 of the Economics and Law School of the National University of La Pampa (Argentina).

References

[1] Cohn T, Global Political Economy: Theory and Practice, 7th ed., New York: Routledge, 2016.
[2] von Bertalanffy L & Hofkirchner W, General System Theory: Foundations, Development, Applications, 1st ed., New York: George Braziller Inc., 2015.
[3] Kang J, Lee S, Kim H, Kim S, Culler D, Jung P, Choi T, Jo K & Shim J, "High-fidelity Environmental Monitoring Using Wireless Sensor Networks," in Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems, Roma, 2013.
[4] Diván M, "Processing Architecture based on Measurement Metadata," in International Conference on Reliability, Infocom Technologies and Optimization (ICRITO), Noida, 2016.
[5] Olsina L, Papa F & Molina H, "How to Measure and Evaluate Web Applications in a Consistent Way," in Web Engineering: Modelling and Implementing Web Applications, Rossi G, Pastor O, Schwabe D & Olsina L, Eds., London, Springer-Verlag, 2007, pp. 385-420.
[6] Molina H & Olsina L, "Towards the Support of Contextual Information to a Measurement and Evaluation Framework," in Quality of Information and Communications Technology (QUATIC), Lisbon, 2007.
[7] Becker P, Lew P & Olsina L, "Strategy to Improve Quality for Software Applications: A Process View," in Conference on Software and Systems Process, Waikiki, Honolulu, 2011.
[8] Diván M & Martín M, "Towards a Consistent Measurement Stream Processing from Heterogeneous Data Sources," International Journal of Electrical and Computer Engineering (IJECE), vol. 7, no. 6, pp. 3164-3175, December 2017.
[9] Diván M & Sánchez Reynoso ML, "A Library for Measurement Interchanging from Heterogeneous Data Sources," in Under reviewing in The XLIV Latin-American Computer Conference (CLEI), Sao Paulo, Brazil, 2018.
[10] Apache Storm Core Team, Apache Storm, Wakefield, MA: Apache Software Foundation, 2018.
[11] Apache HBase Core Team, Apache HBase, Wakefield, MA: Apache Software Foundation, 2018.
[12] R Core Team, R: A Language and Environment for Statistical Computing, Vienna: R Foundation for Statistical Computing, 2017.
[13] Esteves D, Moussallem D, Neto C, Soru T, Usbeck R, Ackermann M & Lehmann J, "MEX Vocabulary: A Lightweight Interchange Format for Machine Learning Experiments," in 11th International Conference on Semantic Systems, Vienna, Austria, 2015.
[14] J. Huang, C. Lange and S. Auer, "Streaming Transformation of XML to RDF Using XPath-based Mappings," in 11th International Conference on Semantic Systems, Vienna, Austria, 2015.
[15] Sun L, Franklin M, Wang J & Wu E, «Skipping-oriented Partitioning for Columnar Layouts, » VLDB Endowment, vol. 10, nº 4, pp. 421-432, 2016.
[16] Object Management Group (OMG), "Software and Systems Process Engineering Meta-Model Specification (SPEM)," Object Management Group (OMG), 2008.
[17] Diván M & Olsina L, "Process View for a Data Stream Processing Strategy based on Measurement Metadata (Extended Version)," Electronic Journal of Informatics and Operations Research, vol. 13, no. 1, pp. 16-34, June 2014.
[18] Diván M & Martín M, "Towards the feedback in the data stream processing based on an organizational memory," in National Conference on Informatic Engineering / Information System, Córdoba, 2013.
[19] Apache Kafka Core Team, Apache Kafka. A distributed streaming platform, Wakefield, MA: Apache Software Foundation, 2018.

[20] Diván M & Martín M, "A New Storm Topology for Synopsis Management in the Processing Architecture," in XLIII Latin American Computer Conference (CLEI), Córdoba, 2017.
[21] Redis Labs Core Team, Redis, Mountain View, CA: Redis Labs, 2018.
[22] Bifet A, Holmes G, Kirkby R & Pfahringer B, "MOA: Massive Online Analysis," The Journal of Machine Learning Research, vol. 11, pp. 1601-1604, August 2010.
[23] Dalton L, "Optimal ROC-Based Classification and Performance Analysis Under Bayesian Uncertainty Models," IEEE/ACM Trans. Comput. Biol. Bioinformatics, vol. 13, no. 4, pp. 719--729, 2016.
[24] Diván M & Sánchez Reynoso ML, "Behavioural Similarity Analysis for Supporting the Recommendation in PAbMM," in 1st International Conference on Infocom Technologies and Unmanned Systems (ICTUS), Dubai, 2017.
[25] Open Geospatial Consortium and International Standard Organization, "ISO 19136:2007 Geographic Information - Geography Markup Language," International Standard Organization, Geneva, 2007.
[26] Google, "google-gson: A Java serialization/deserialization library to convert Java Objects into JSON and back," 26 March 2018. [Online]. Available: https://github.com/google/gson. Last visit: 26.03.2018.
[27] Kopff G, "gkopff/gson-javatime-serialisers: A set of GSON serialiser/deserialisers for dealing with Java 8 java.time entities.," 26 March 2018. [Online]. Available: https://github.com/gkopff/gson-javatime-serialisers. Last visit: 26.03.2018.
[28] Garofalakis M, Gehrke J & Rastogi R, Eds., Data Stream Management: Processing High-Speed Data Streams, Berlin: Springer-Verlag Berlin Heidelberg, 2016.