Reunión y Ordenamiento de Flujos de Datos Simultáneos y Concurrentes Basados en C-INCAMI

Mario José Diván

Facultad de Ciencias Económicas y Jurídicas – Facultad de Ingeniería
Universidad Nacional de La Pampa

21/03/2011

Trabajo Final presentado para obtener el grado de Especialista en Cómputo de Altas Prestaciones y Tecnología Grid

Facultad de Informática - Universidad Nacional de La Plata

Director	**Dr. Marcelo Naiouf** Instituto de Investigación en Informática - LIDI Facultad de Informática Universidad Nacional de La Plata

Resumen

El presente documento corresponde con el trabajo final de la Especialización en Cómputo de Altas Prestaciones y Tecnología Grid de la Facultad de Informática, de la Universidad Nacional de La Plata.

El trabajo aborda la problemática de procesamiento paralelo de flujos de datos (data streams), con el ingrediente de basarse los mismos en marcos formales de medición y evaluación para luego regir la organización de sus datos en base a ellos.

El trabajo aborda inicialmente el estado del arte en términos de sistemas de gestión de flujos de datos, para luego discutir el marco formal de medición y evaluación C-INCAMI, como referencia para la estructuración del contenido del flujo.

Seguido, se discute globalmente el Enfoque Integrado de Procesamiento de Flujos de Datos Centrado (EIPFD) en Metadatos de Mediciones, el que se asocia a mi tesis para Doctor en Ciencias Informáticas de la misma facultad y que a la fecha, se encuentra en revisión y edición final de escritura. Dicho enfoque, permite estudiar el impacto de paralelizar el procesamiento de la recepción de los flujos y la organización en línea dentro de un buffer centralizado, controlando el acceso concurrente y simultáneo en entornos de arquitecturas con memoria compartida.

Luego, se define un formato de intercambio de mediciones basado en C-INCAMI, junto con el procesador que permite efectuar la serialización/deserialización en línea a los efectos de favorecer el procesamiento paralelo. Hecho ello, se plantea la estructura de organización de las mediciones y cómo guían los metadatos, al proceso de clasificación de mediciones en un buffer central.

Se plantea un caso de aplicación para EIPFD sobre el que se basará la simulación de laboratorio. Esta simulación, persigue validar inicialmente los tiempos de procesamiento y analizar estadísticamente los resultados de la misma, para poder identificar cuellos de botellas y situaciones de mejoras en términos de procesamiento.

Agradecimientos

A Laura, José Ignacio y Santiago Agustín por su inmensa paciencia y amor incondicional.

A mi madre, que aunque no esté más entre nosotros, representa para mí un modelo de rectitud y perseverancia a seguir.

Al Dr. Marcelo Naiouf, por el esfuerzo y dedicación puesto en la dirección del presente trabajo.

Al Dr. Luis Olsina y a la Dra. Silvia Gordillo, por el esfuerzo y dedicación aplicados a la dirección del Doctorado en Ciencias Informáticas (en el que el presente trabajo se encuentra enmarcado).

Al Cr. Roberto Vassia, por la cordialidad y confianza manifiesta en cada desafío planteado.

A la Facultad de Ciencias Económicas y Jurídicas de la Universidad Nacional de La Pampa, por permitir y facilitar la formación de sus docentes con el esfuerzo que ello representa para las instituciones del interior del país.

A la Facultad de Ingeniería de la Universidad Nacional de La Pampa, al Ing. Carlos D'amico y al GIDIS_WEB, por acogerme y constituirse en un ámbito de investigación y desarrollo cordial y ameno, en donde prima la persona sobre toda estructura.

A todos, Muchas Gracias!!!

Mario José Diván

Santa Rosa (La Pampa), Marzo 10 de 2011

Índice

Lista de Figuras

Lista de Tablas

1 Introducción

En la actualidad existen aplicaciones de software que procesan un conjunto de datos a medida, que se generan en forma continua, con el fin de responder a consultas y/o adecuar su comportamiento en función del propio arribo de datos (Namit, Gehrke, & Balakrishan, 2008). Este tipo de aplicaciones tienen por finalidad actuar de un modo pro activo y en donde la persistencia no es una prioridad, sino que el aspecto primordial radica en agilizar el procesamiento de los datos ante su arribo, evitando en lo posible el descarte de los mismos por cuestiones de desborde, minimizando el uso de recursos a los efectos de brindar una respuesta aproximada. En este último punto radica el sacrificio fundamental efectuado por este tipo de aplicaciones, es decir que se prefiere disminuir precisión descartando datos ante potenciales desbordes a los efectos de ganar agilidad en la respuesta en línea.

El presente capítulo expondrá en primer lugar las motivaciones y antecedentes que permitieron arribar al dominio del tipo de aplicaciones indicadas en el párrafo anterior. Presentados los antecedentes y motivaciones, se procede a efectuar el planteamiento del problema en concreto, para continuar con los requerimientos y supuestos que rigen al planteo del mismo. A seguir, se plantean las principales contribuciones y finalmente se describe la estructura de la presente tesis.

Estructura del Capítulo:

1.1 Motivación y Antecedentes

Como se mencionó anteriormente, existen aplicaciones de software que procesan los datos ante su propio arribo primando el principio de agilidad en la respuesta, aunque ello implique un sacrificio en la precisión dado que arrojan resultados aproximados. Este tipo de aplicaciones de procesamiento de flujos de datos continuos, rigen su comportamiento principalmente a partir de los valores actuales de los datos y en donde, la historia presenta una incidencia relativa que si bien aporta, no es esencial para brindar una respuesta. Por ejemplo, las aplicaciones para el monitoreo de signos vitales de pacientes emiten mediciones continuamente, el histórico es importante por cuanto se podría estudiar la evolución de un paciente a partir del mismo, pero ante una situación de urgencia lo importante a los efectos de disparar las alarmas correspondientes, es fundamentalmente la interpretación de los valores actuales arrojados por las mediciones y no su histórico. En forma análoga a las aplicaciones de monitoreo de signos vitales, pueden encontrarse otras aplicaciones tales como las asociadas al comportamiento de los mercados financieros; de centrales nucleares; de tráfico aéreo; entre otras.

Durante el transcurso del 2003 y vinculado a la Maestría en Administración de Negocios que en aquel momento me encontraba desarrollando, se planteó la idea de construir un modelo de centro de distribución celular para acopio de miel en la cooperativa de Doblas (Departamento Atreucó – La Pampa) (Diván M. , 2006). En dicho modelo, uno de los aspectos fundamentales radicaba en el monitoreo continuo de las condiciones ambientales del centro de distribución y de los tambores de miel que contenía, a los efectos de evitar situaciones de deterioro del alimento que pudiesen producir riesgos a la salud derivados de su consumo (Congreso de la República Argentina, 1969 con modificaciones de las resoluciones 84/09 y 709/09 de la Secretaría de Políticas, Regulación e Institutos y SAGPyA respectivamente). De este modo, el requerimiento de medición fue fundamental como así también la formalización de las métricas. Dado que se trataba de una Maestría en Negocios, en aquella tesis se empleó el enfoque de Kaplan y Norton (Kaplan & Norton, Cómo Utilizar el Cuadro de Mando Integral, 2000) para la definición e implementación de las métricas que nutrirían los tableros de comandos involucrados en el monitoreo. Una de las debilidades que se pudo apreciar en el cuadro de mando integral de Kaplan y Norton radicaba en la ausencia de ontologías que sustenten formalmente los conceptos intervinientes en el proceso de medición, como así también la no consideración del contexto de medición.

En 2004 se toma contacto con el marco formal de medición y evaluación INCAMI (Information Need, Concept, Attribute, Metric and Indicator) (Olsina, Molina, & Papa, 2005), a través de la serie de cursos CuPIWeb organizados por la Facultad de Ingeniería de la Universidad Nacional de La Pampa (UNLPAM), el cual permitía formalizar las métricas e indicadores, como sus conceptos asociados careciendo en ese entonces de información contextual relacionada a las mismas. La incorporación de la ontología proveniente de INCAMI, suplía muchas de las dificultades del cuadro de mando integral de Kaplan y Norton que habían surgido durante el desarrollo del modelo de centro de distribución celular para acopio de miel, permitiendo la formalización del proceso de medición y evaluación y por ende su automatización. En 2007, el marco INCAMI incorpora la capacidad de modelar información contextual dentro del proceso de medición y

evaluación pasándose a llamar C-INCAMI (Context - Information Need, Concept Model, Attribute, Metric and Indicador) (Molina & Olsina, 2007), lo que permitió formalizar la situación en la cual el proceso de medición era llevado adelante, facilitando notablemente la posibilidad de automatización del proceso.

La idea de monitorización continua iniciada en el modelo de centro de distribución celular, incorporando la posibilidad de formalizar con información contextual el proceso de medición y evaluación con vistas a su automatización, abrió en su momento otro interrogante surgido del tipo de aplicaciones de procesamiento continuo mencionadas inicialmente ¿Se Necesita detectar o prevenir? La idea era prevenir para minimizar riesgos, motivo por el cual se incorpora un bagaje de herramientas asociadas a la minería de datos (Aleman-Beza, Halaschek, Arpinar, & Sheth, 2003) (Singh, Vajirkar, & Lee, 2003) para actuar pro activamente, de modo predictivo e incorporar automatización en el proceso de toma de decisión, pero fundado ahora en información que proviene en base a un marco de medición y evaluación sustentado en una base ontológica (Olsina & Martín, 2004), a la cual se le incorporaba información relativa a su contexto.

Un aspecto a destacar desde el punto de vista del procesamiento, radica en que cada flujo de datos abastece un conjunto de mediciones y de igual modo, lo pueden realizar cualquier otra cantidad de flujos en paralelo. De este modo, el procesamiento de las mediciones ante un determinado componente que permite reunir de algún modo los flujos, es naturalmente una tarea de procesamiento paralelizable (Andrews, 1999).

1.2 Planteamiento del Problema

A diferencia de las aplicaciones de aprovisionamiento continuo de datos introducidas anteriormente, las bases de datos tradicionales han sido empleadas en situaciones donde se requería principalmente persistencia y respuesta ante consultas complejas. Estas se componen de un conjunto de tuplas relativamente estáticas, sobre las que podrían efectuarse operaciones de inserción, borrado, actualización y reorden con menor frecuencia que lo que se consultan (Golab & Özsu, 2003).

En aplicaciones tales como el monitoreo de signos vitales de pacientes, de centrales nucleares, de mercados financieros y en general, de cualquier aplicación que requiera procesamiento y verificación continua de los datos generados en tiempo real, es menester llevar adelante en forma ininterrumpida la consulta de los atributos críticos que parametrizan la naturaleza del problema u objeto (ente) bajo control, por lo que el enfoque tradicional de bases de datos basado en persistencia no sería el adecuado. De este modo, la caracterización de este nuevo tipo de aplicaciones surge bajo la denominación de *data stream, el cual se define como una secuencia en expansión continua e ilimitada de ítems de datos homogéneos que fluyen en tiempo real* (Getta & Vossough, 2003). La definición de data stream supone la idea de homogeneidad en los datos, tiempo y asume por ende una estructura (metadatos) en los mismos.

A los efectos de posibilitar la repetitividad, consistencia y automatización del proceso de medición y evaluación a partir de data streams, es necesario brindar una estructura a los datos del

flujo, que permita incorporar significado del dato a procesar, como por ejemplo: tipo de dato, definición de métrica, indicadores asociados, precisión requerida, entre otros. Para ello, es necesario basarse en un marco formal de medición y evaluación con sustento ontológico que posibilite no solo la estructuración, sino también la determinación del contenido transmitido a los efectos de su procesamiento automático (Olsina & Martín, 2004). En este sentido, se parte del marco formal de medición y evaluación C-INCAMI, como referencia para la definición de metadatos asociados a las métricas, estructuración de las mediciones respectivas y base ontológica para la incorporación de información contextual referida a la medición (Molina & Olsina, 2007) (Olsina, Papa, & Molina, 2007).

Para aprovechar la gestión de metadatos asociados a las métricas y el resultado provisto a través de las mediciones, es necesario desarrollar un modelo de procesamiento que aborde el proceso de adquisición de flujos de datos basados en C-INCAMI a los efectos de poder reunir y organizar los flujos de datos de cara a facilitar su análisis on line posterior.

De este modo, el núcleo del trabajo final se focalizará en permitir la incorporación de fuentes de datos heterogéneas, cuyos flujos de mediciones están estructurados y enriquecidos con metadatos embebidos basados C-INCAMI, a la vez que se permite el paralelismo de modo natural en la reunión y organización de las mediciones en un buffer centralizador a los efectos de su posterior análisis.

1.3 Consideraciones y Requerimientos

Como se expuso anteriormente, el trabajo final se sustenta en obtener un conjunto de mediciones enriquecidas con metadatos basados en C-INCAMI desde fuentes de datos heterogéneas, para aprovisionar un buffer centralizador con una determinada organización. Dicho buffer permite organizar las mediciones a nivel de grupo de seguimiento como por métrica dentro de cada grupo de seguimiento, lo que permite agilizar lógicamente las tareas de análisis posterior.

El presente trabajo final, es parte componente de la tesis “Enfoque Integrado de Procesamiento de Flujos de Datos centrado en Metadatos de Mediciones” correspondiente a la carrera de Doctorado en Ciencias Informáticas de la Universidad Nacional de La Plata, dirigida por el Dr. Luis Olsina (Facultad de Ingeniería - UNLPAM) y Co-Dirigida por la Dra. Silvia Gordillo (Facultad de Informática – UN LP), actualmente en proceso de revisión y edición. En función de ello, será necesario efectuar una introducción global al enfoque integrado de procesamiento de flujos de datos a los efectos de contextualizar el posterior análisis estadístico sobre la simulación del procesamiento de flujos.

Se asume que los datos provenientes a través de los flujos de datos representan mediciones y se estructuran de algún modo bajo el marco formal C-INCAMI. Dichas mediciones se suponen son numéricas y continuas, pudiéndose receptar una cantidad ilimitada de las mismas a partir de las fuentes y con la evolución del tiempo.

La cantidad de memoria de utilizar para el procesamiento de los flujos, dependerá del proyecto de medición y evaluación y podrá escalar, siempre y cuando no comprometa la estabilidad de la unidad de procesamiento donde se esté ejecutando.

El procesamiento y aplicación de árboles de decisión (Modelo Predictivo dentro del Enfoque Integrado de Procesamiento de Flujos de Datos Centrado en Metadatos de Mediciones) sobre data streams implica una serie de consideraciones que deben tenerse en cuenta (Kirkby, 2007):

- **Procesar un caso a la vez e inspeccionarlo solo una vez en lo posible**: Las mediciones deberán ser analizada en términos de outliers (Johnson, 1998) y/o presencia de ruido sobre el mismo arribo, aplicando instantáneamente el árbol de decisión, permitiendo la obtención de la decisión como así también la actualización incremental del árbol para situaciones futuras.

- **Utilizar una cantidad limitada de memoria**: Las mediciones pueden arribar en forma ilimitada en términos temporales y requerirán ser procesadas como así también clasificadas a los efectos de valorar la decisión. Esto implica un riesgo en términos de memoria y capacidad de procesamiento que debe ser delimitado claramente a los efectos de garantizar la estabilidad del conjunto como sistema.

- **Trabajar en una cantidad limitada de tiempo**: Dado el arribo ilimitado de las mediciones, la cantidad de unidades de tiempo que podrán asignársele a cada grupo de medición con el objeto de procesarla y clasificarla es finita, dado que un exceso o sobrestimación repercutirían en generación de colas de servicio o recursos ociosos respectivamente.

- **Poder clasificar en cualquier momento**: El modelo de decisión debe poder clasificar en cualquier instante de tiempo, incluso cuando se encuentre procesando nuevas mediciones o bien, cuando no tenga ninguna medición aún con la cual actualizarse.

1.4 Principales Contribuciones

Como se ha mencionado anteriormente, el trabajo final se centra en generar un enfoque (centrado en modelos) que permita integrar fuentes de datos heterogéneas, organizándolas ágilmente mediante empleo de procesamiento paralelo en un buffer centralizado a los efectos de permitir agilizar las etapas de análisis posterior. Con ello se pretende contribuir en los siguientes aspectos:

- **Desde el punto de vista de las fuentes de datos y su procesamiento**:

    - **Permitir un borrado selectivo ante potenciales desbordes en colas de servicio basado en metadatos C-INCAMI**: Ante la situación en que la generación de mediciones provenientes desde diferentes dispositivos de medición, supere la tasa de procesamiento de datos y desborde la cola de servicio de cada procesador, los

metadatos embebidos en las mediciones permiten diferenciar claramente la semántica de cada una de ellas y el procesador podrá efectuar un descarte selectivo de la cola de modo que no perjudique los modelos de decisión a aplicar a posteriori.

- **Desde el punto de vista de implementación y simulación del enfoque integrado**:
  - **Desarrollar una herramienta prototípica que implemente el procesamiento de mediciones:** Se desarrolló un prototipo que permite gestionar y aplicar la minería de datos al flujo de datos en el marco de la administración de metadatos y mediciones, para luego ser visualizada de un modo adecuado al proceso de toma de decisión. El prototipo se construyó partiendo de tecnología de desarrollo JAVA y haciendo su código libre bajo licencia Open Source a los efectos de que se pueda contribuir libremente a los avances del prototipo y del conocimiento en la temática.

  - **La simulación resultante de aplicar el enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones, permite validar el mismo sobre un caso de aplicación y definir los umbrales de aplicación.** La simulación efectuada sobre la definición de un proyecto de medición y evaluación de pacientes trasplantados ambulatorios, ha permitido validar inicialmente el comportamiento detectivo como así también el predictivo, facilitando la recolección de tiempos involucrados en cada etapa de procesamiento, lo cual es fundamental para delinear áreas potenciales de aplicación del enfoque.

1.5 Estructura del Trabajo Final

El trabajo se estructura partiendo del estado del arte de los sistemas de gestión de flujos de datos actuales, para discutir la situación actual de los conceptos y herramientas involucrados, a los efectos de centrar al enfoque integrado de procesamiento de flujos de datos basado en metadatos de mediciones, como una alternativa válida de un nicho específico. A seguir, se describe un panorama del enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones, para luego avanzar sobre la organización del buffer y resultados de la simulación.

- **Capítulo 2**: Se aborda el Estado del Arte donde inicialmente se introduce en una serie de conceptos previos, se analiza el procesamiento de flujos de datos, se hace hincapié en la diferencia paradigmática con respecto al contexto de bases de datos tradicionales, para concluir con la discusión sobre la necesidad del enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones y su esquema de organización.

- **Capítulo 3**: Se plantea un modelo integrado para el procesamiento de flujos de datos (data streams) y soporte al proceso de toma decisión, basado en datos con semántica incorporada en base a C-INCAMI. Aborda cada una de sus componentes y explica el rol de

cada uno de ellos, como así también el efecto de su interacción con los restantes elementos que integran el modelo.

- **Capítulo 4**: Se aborda la problemática de captación de los datos desde distintas fuentes como así también su transmisión, lo que representa la definición formal de los data stream que pueden intervenir en el modelo. Adicionalmente, se expone el mecanismo de incorporación de metadatos en forma conjunta con los datos y las estrategias de abastecimiento/recolección de los mismos.

- **Capítulo 5**: El capítulo plantea un caso de aplicación del enfoque integrado de procesamiento de flujos de datos presentado en el capítulo 3, como así también su implementación prototípica y simulación asociada.

- **Capítulo 6**: Sintetiza las contribuciones efectuadas en la presente tesis y finaliza con los lineamientos fundamentales de los potenciales trabajos futuros que se desprenden de la misma.

Estructura del Capítulo:

2 Estado del Arte

El capítulo 1 ha presentado la motivación y antecedentes de la presente tesis, seguido del planteamiento del problema como así también de las consideraciones y requerimientos asociados al mismo. Ha esbozado las principales contribuciones para culminar con la estructuración de la presente tesis.

En este capítulo se introduce una serie de conceptos previos, necesarios para avanzar en el desarrollo del problema, entre los que se encuentran conceptos asociados a minería de datos (data mining), flujos de datos (data streams), minería sobre flujos de datos (mining data streams), contexto (context) y orientación al contexto (context-aware).

A continuación, se analizan los sistemas de gestión de flujos de datos (data stream management systems -DSMS-) para determinar qué alternativas existen, como así también qué cuestiones permanecen sin resolver, a los efectos del procesamiento de flujos de datos cuyo comportamiento es instruido a partir de metadatos incorporados en el mismo.

Finalmente, el capítulo plantea la necesidad del enfoque integrado de procesamiento de flujos de datos, discutiendo qué aspectos de los conceptos asociados y planteados en el capítulo, permanecen al momento de la escritura de esta tesis sin una respuesta concreta y cuáles podrían ser mejorados a través de dicho enfoque.

2.1 Conceptos Previos

Como se ha expuesto en la sub-sección 1.2, el núcleo del trabajo final se focalizará en incorporar fuentes de datos heterogéneas, cuyos flujos de mediciones estructurados y enriquecidos con metadatos embebidos basados en C-INCAMI, permitan realizar una organización y procesamiento ágil de los flujos para lograr una buffer que facilite las etapas de análisis posterior.

De este modo y para una mejor comprensión, es necesario introducir conceptos fundamentales relacionados al núcleo del trabajo tal como *data mining,* vinculado a la etapa predictiva y consiguiente toma de decisión del enfoque integrado de procesamiento de flujos de datos; *data streams,* concepto en el que

se basa el aprovisionamiento de mediciones para su posterior procesamiento; *mining data streams,* necesario para aspectos del comportamiento predictivo aplicado a los flujos de datos; *context* a los efectos de entender qué se entiende por el mismo y cómo sirve el concepto para delimitar la ocurrencia de un suceso o situación, con respecto al ente de medición; y finalmente, *context-aware* que permite comprender qué implica la orientación al contexto, para luego poder utilizar dicho concepto dentro del flujo de mediciones, ya que la idea del flujo de mediciones como se ha expuesto, es que permita la incorporación de información contextual a los efectos de posibilitar análisis más consistentes y comparables. Finalmente, se reseña C-INCAMI como marco formal de medición y evaluación, el que representa el punto de referencia a partir del cual se estructurarán los flujos de mediciones.

2.1.1 Data Mining

"*Data mining is the process of discovering interesting knowledge from large amounts of data stored Either in databases, data warehouses, or other information repositories*" (Han & Lamber, 2001). Sobre la definición de Han & Lamber debe hacerse una aclaración importante a los efectos de diferenciar el concepto de dato con respecto al de información. Los autores parecieran que emplean ambos términos en forma indistinta, ya que la definición indica que el descubrimiento de conocimiento surge a partir de grandes almacenes de datos y al final de la definición se señala: "u otros repositorios de información". Por ello, es necesario hacer algunas acotaciones:

- Por dato se entiende a la representación de un antecedente captado y registrado en algún medio de almacenamiento necesario para llegar a un conocimiento (Diván M. , 2005)

- Por información se entiende al dato que satisface las características de oportunidad, interés, veracidad y consistencia en forma simultánea (Diván M. , 2005)

- Por conocimiento se entiende la información no trivial y factible de ser aplicada en un contexto determinado a los efectos de lograr la consecución de un objetivo concreto (Han & Lamber, 2001)

En este sentido, se puede reescribir la definición de Han & Lamber como sigue: "*La minería de datos es la aplicación de técnicas en grandes volúmenes de datos para descubrir información interesante, útil, aplicable y no trivial*"

2.1.2 Data Streams

"*A data stream is a possibly unbounded sequence of tuples*" (Koudas & Srivastava, 2005). La definición indica desde un principio, la diferenciación con el esquema de bases de datos tradicional desde el punto de vista que el 'data stream' no tiene limitación y se puede abordar como secuencia de tupla. Esta secuencia dará lugar a la noción de tiempo y por consiguiente, determinará una posibilidad de abordaje mediante esquemas de series temporales (Pérez López, 2005) desde la perspectiva de minería de datos.

"*For unbounded data streams many queries are interested in a subset or part of the complete stream"* (Chaudhry, Shaw, & Abdelguerfi, 2005). Con vistas a efectuar el procesamiento del data stream, es posible interpretar una ventana como subconjunto del flujo, lo que permitirá disponer de un valor finito de tuplas a los efectos del cómputo. Existen diversas técnicas por las que obtener este subconjunto de tuplas (Chaudhry, Shaw, & Abdelguerfi, 2005):

- **Time-Based Window (o físicas)**: El tamaño de la ventana estará expresado por el número de tuplas que arriben en un lapso de tiempo fijo. Por ejemplo: Todas las tuplas que han arribado en los últimos 120 segundos.

- **Tuple-Based Window (o Lógicas)**: El tamaño de la ventana estará expresado por un número fijo de tuplas. Por ejemplo: 200 tuplas por ventana, independientemente del tiempo que insuma el arribo de las mismas.

- **Sliding window**: El ancho de la ventana es fijo y los nuevos valores que arriban van sustituyendo a las viejas tuplas mediante un esquema de colas (FIFO).

- **Landmark window**: Uno de los extremos de la ventana permanece fijo mientras el segundo extremo de la ventana se mueve incorporando nuevas tuplas. Por ejemplo: Transferencias en dólares de los clientes de un banco desde la última vez que el dólar superó los 3 pesos.

2.1.3 Mining Data Streams

"*Mining Data Streams is concerned with extracting knowledge structures represented in models and patterns in non stopping streams of information*" (Medhat Gaber, Zaslavsky, & Krishnaswamy, Mining Data Streams: A Review, 2005). Es menester aclarar que el flujo es de datos no de información. Lo que ocurre es que muchos autores hacen la aclaración al inicio del texto sobre la diferencia de los conceptos y dependiendo del objeto del documento, pueden luego emplearlos como sinónimos a lo largo del mismo.

"*Mining data streams has raised a number of research challenges for the data mining community. These challenges include the limitations of computational resources, especially because mining data streams of data most likely be done on a mobile device with limited resources. Also due to the continuality of data streams, the algorithm should have only one pass or less over the incoming data records*" (Medhat Gaber, Zaslavsky, & Krishnaswamy(b), 2005). Los desafíos señalados por los autores con respecto a la aplicación de métodos y técnicas de minería de datos sobre data streams, reafirman las diferencias paradigmáticas con el contexto de bases de datos tradicional. La minería de datos sobre data streams, por definición, se encuentra limitada temporalmente y computacionalmente con respecto a la lectura del dato como así también por ende a la generación y/o actualización del modelo. Del mismo modo el recolector de datos, el cual también podría emplear mecanismos inferenciales, en general es representado por dispositivos móviles cuyo poder computacional es limitado y su principal rol viene de la mano con el tratamiento previo de los datos en post de garantizar su consistencia.

2.1.4 Context y Context-Aware

"Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves" (Dey, 2001).

Desde el punto de vista de la ejecución de una medición, el concepto revierte una particular importancia, ya que dado una entidad bajo medición, cualquier persona, objeto y/o características del sitio donde reside la misma, que sean plausibles de medición, contribuirán a disminuir la incertidumbre sobre ésta y permitirán analizar las correlaciones entre sus variables representativas.

"*A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on user's task*" (Dey, 2001). A partir de los señalado por Dey, un sistema orientado a la medición es context-aware, en la medida que incorpore algún mecanismo que permita extraer datos útiles referenciales al contexto de la entidad bajo análisis. En este último sentido, las métricas orientadas a parametrizar el contexto que circunda a la entidad bajo medición, representan un mecanismo válido de abstracción que permite incorporar en el modelo información complementaria a la medición efectuada sobre la propia entidad. Por ejemplo, si la temperatura corporal de una persona hipertensa es 36,5°C, en forma aislada nos diría que la temperatura de la misma estaría dentro de los parámetros normales, pero si adicionalmente se tomase la temperatura ambiente inmediata a la posición de la misma y nos arrojase 40°C, este complemento nos estaría indicando un potencial riesgo para la persona (Petersa, y otros, 2009).

2.1.5 C-INCAMI

C-INCAMI (Olsina, Papa, & Molina, 2007) (Molina & Olsina, 2007) es un marco conceptual que define los módulos y conceptos que intervienen en el área de medición y evaluación, para organizaciones de software. Se basa en un enfoque en el cual la especificación de requerimientos, la medición y evaluación de entidades y la posterior interpretación de los resultados están orientadas a satisfacer una necesidad de información particular. Está integrado por los siguientes componentes principales:

- Definición y Especificación de Requerimientos no Funcionales
- Especificación del Contexto del Proyecto
- Diseño y Ejecución de la Medición
- Diseño y Ejecución de la Evaluación

La mayoría de los componentes están soportados por muchos de los términos ontológicos definidos en (Olsina & Martín, 2004) y (Olsina, Papa, & Molina, 2007).

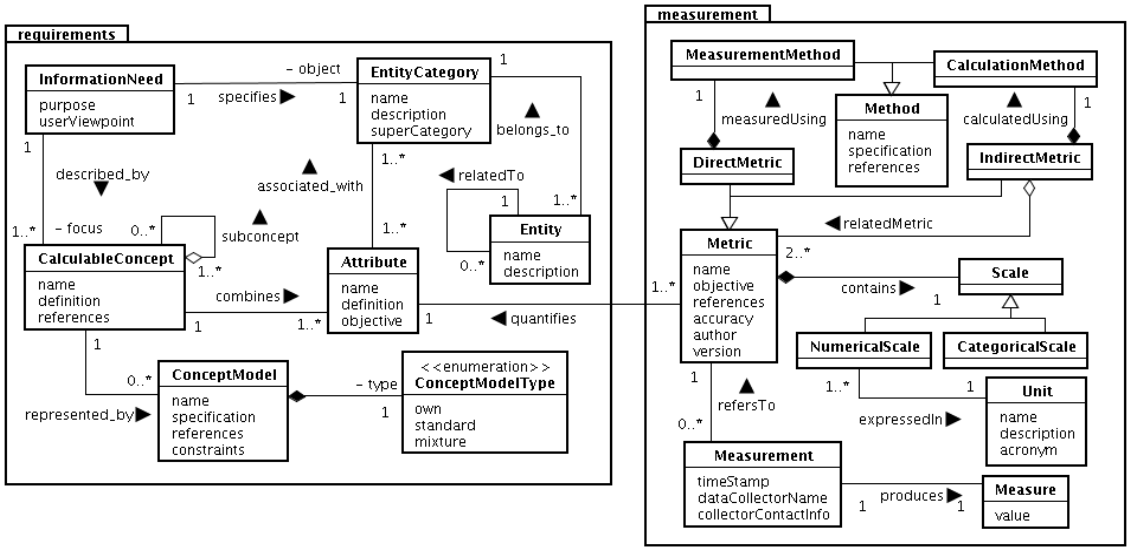

Ilustración 1. Principales Conceptos y Relaciones de los Componentes Definición y Especificación de Requerimientos No Funcionales y Diseño y Ejecución de la Medición

La Ilustración 1 expone un diagrama de clases donde se especifican los componentes de requerimientos y de medición, como así también los principales términos. En lo que sigue se resaltará el texto con itálica cuando se mencionen términos y atributos que forman parte de estos componentes.

El componente *Definición y Especificación de Requerimientos no Funcionales*, permite especificar de forma concisa la necesidad de información (*Information Need*) de un proyecto de medición y evaluación. Generalmente surge de un objetivo específico a nivel de proyecto u organización. La necesidad de información describe el propósito (*purpose*) y el punto de vista de usuario (*userViewpoint*). A su vez, identifica un concepto calculable (*CalculableConcept*) y especifica la categoría de las entidades a evaluar (*EntityCategory*). Un concepto calculable puede ser definido como una relación abstracta entre atributos de un ente y una necesidad de información, el cual puede ser representado por un modelo (*ConceptModel*). En C-INCAMI, la calidad es medida y evaluada mediante la cuantificación de conceptos de menor nivel de abstracción como son los atributos (*Attribute*) de un ente. En la Ilustración 2 se pueden visualizar los términos, atributos y relaciones que forman parte del componente citado (paquete *requirements*).

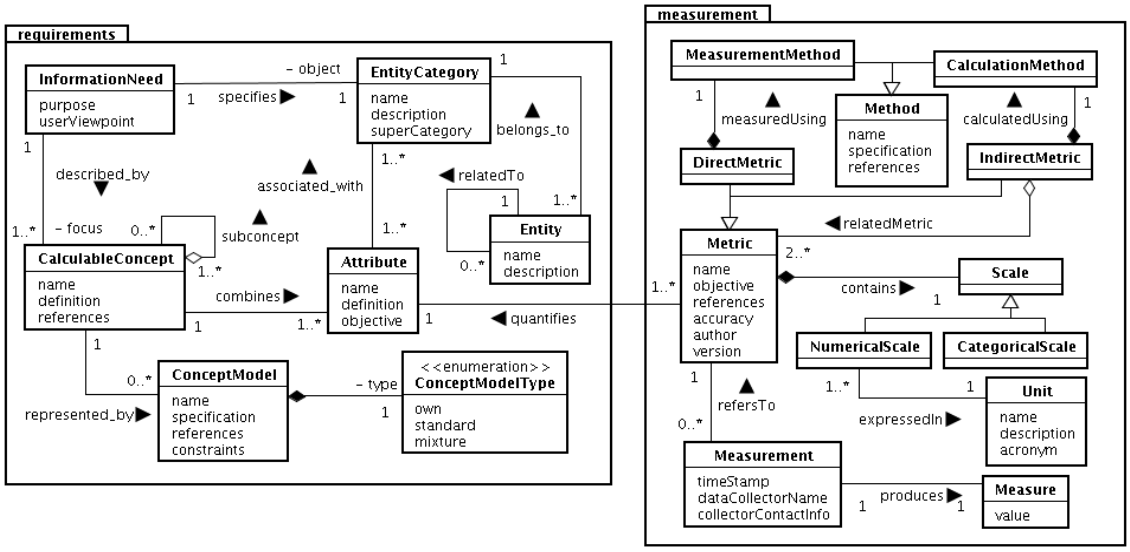

Ilustración 2. Principales Conceptos y Relaciones de los Componentes *Definición y Especificación de Requerimientos no Funcionales* y *Diseño y Ejecución de la Medición*

El componente *Especificación del Contexto del Proyecto*, permite describir de forma estructurada el contexto relevante (*Context*) para una necesidad de información por medio de propiedades (*ContextProperty*). Para mayor información sobre modelado contextual en C-INCAMI remitirse a (Molina & Olsina, 2007).

El componente *Diseño y Ejecución de la Medición* (paquete *measurement* en Ilustración 2), permite especificar las métricas utilizadas en la medición y registrar los valores medidos de los atributos de cada ente. Un atributo puede ser cuantificado por muchas métricas, pero en un proyecto de medición específico sólo se utiliza una métrica. La métrica (*Metric*) define el método de medición o cálculo (*Measurement Method* o *Calculation Method*) para obtener el valor del atributo y la escala (*Scale*) de los valores. Un método de medición se aplica a una métrica directa (*Direct Metric*), mientras que un método de cálculo (en el cual interviene una fórmula) se aplica a una métrica indirecta (*Indirect Metric*). Una vez que las métricas fueron seleccionadas, se utiliza su definición para efectuar la medición (*Measurement*) y así producir una medida (*Measure*) para cada atributo. No obstante, el valor obtenido por una métrica en particular no representa el nivel de satisfacción de un requerimiento elemental (atributo) sino que es necesario realizar una nueva correspondencia utilizando indicadores.

El componente *Diseño y Ejecución de la Evaluación* permite especificar indicadores que son la base para la interpretación de atributos y conceptos calculables. Hay dos tipos de indicadores: elementales y globales. Se define un indicador elemental (*Elementary Indicator*) como aquel que no depende de otros indicadores para evaluar o estimar un concepto de más bajo nivel de abstracción como son los atributos. Por otro lado, un indicador parcial o global (*Global Indicator*) es derivado de otros indicadores para evaluar o estimar un concepto de alto nivel de abstracción (como son los subconceptos y conceptos calculables, por ej. de un modelo de calidad).

El valor del indicador global finalmente representa el grado de satisfacción global de los requerimientos para dicha necesidad de información.

C-INCAMI tiene como virtud el hecho de sustentar su definición sobre una ontología diferenciando claramente los conceptos involucrados en la medición, facilita la formalización de dichos conceptos a los efectos de automatizar los procesos de medición, es un marco específico a procesos de medición, incorpora manejo del contexto y al formalizar una base común en la definición, permite la repetitividad y comparación. No obstante y sin perjuicio de lo mencionado, no plantea la idea de grupo de seguimiento, es decir que una serie de métricas aplicadas a una misma entidad en un mismo contexto puedan ser agrupados a los efectos de su análisis; por ejemplo: en un paciente trasplantado al cual se le mide presión arterial, temperatura axilar y frecuencia cardíaca, mientras que en su contexto inmediato se mide humedad ambiente, temperatura y presión ambiental, sería altamente deseable que las mediciones sean agrupadas y analizadas conjuntamente a los efectos de verificar si existen efectos de arrastres entre las mediciones. Este último aspecto, si bien necesario para el procesamiento de flujos de mediciones, es una cuestión menor a la que se pueden proponer extensiones del marco. De los marcos de medición analizados, el que mayor grado de formalización y claridad conceptual propone, es C-INCAMI. Esto último, es fundamental para poder lograr la automatización de procesos de medición que permitan la repetitividad, la comparación de resultados y establecer una base homogénea que facilite un análisis posterior, sea este estadístico o predictivo.

2.2 Sistemas de Gestión de Flujos de Datos (Data Stream Management Systems)

Dado que uno de los puntos fundamentales del trabajo final pasa por procesar los flujos ante su propio arribo, la gestión involucrada en el procesamiento de los flujos de datos, pasa a ocupar un aspecto necesario de considerar. En este sentido, debe considerarse desde la gestión de memoria, a los efectos de garantizar el funcionamiento continuo y por ende el comportamiento detectivo y/o predictivo mencionado; hasta aspectos asociados a cómo debiera reaccionar un procesador de flujos, si la tasa de arribo de los datos supera la tasa de procesamiento a los efectos de garantizar la estabilidad.

Desde el punto de vista clásico del procesamiento de data stream, existen trabajos relacionados tales como los llevados adelante en el proyecto STREAM de la Universidad de Stanford (Babcock, Datar, & Motwani, 2004) (Babcock, Babu, Datar, Motwani, & Thomas, 2004) (Srivastava & Widom, 2004) y el proyecto Aurora, desarrollado en colaboración entre el MIT y las Universidades de Brown y Brandeis (Hwang, Balazinska, Rasin, Çetintemel, Stonebraker, & Zdonik, 2005) (Ryvkina, Maskey, Cherniack, & Zdonik, 2006) (Tatbul & Zdonik, 2006). Básicamente, estos proyectos han abordado desde diferentes puntos de vista los tópicos y problemáticas de las operaciones tradicionales de bases de datos dentro del contexto de procesamiento de data streams.

Desde un punto de vista aplicado de los Data Stream Management System y analizando un campo más afín a la presente tesis, como la gestión de datos en redes de sensores, se presentan distintos enfoques de trabajos vinculados con sus respectivas particularidades:

- **Presto** (Li, Ganesan, & Shenoy, 2006): Presenta una arquitectura en dos capas que comprende un Proxy y sensores que cooperan unos con otros para adquirir datos y procesar consultas. PRESTO construye un modelo de serie temporal basado en SARIMA (Box & Jenkins, 1991) de acuerdo a las tendencias observadas y transmite los parámetros del modelo a los sensores. Los sensores contrastan la medición con su valor predicho de acuerdo a los parámetros informados por el Proxy, en caso de que el valor sensado u obtenido se corresponda con el predicho, más o menos un umbral definido como parámetro en el sensor, el dato no es informado al Proxy, mientras que en caso de desvío éste si es informado. De este modo la arquitectura pretende disminuir el overhead en la red por la sola transmisión de los desvíos a los valores predichos por el modelo de serie temporal y adicionalmente, ajustar el modelo en función de los nuevos desvíos. Un aspecto adicional e interesante en la resolución de consultas en PRESTO, es que el Proxy en caso de que la precisión exigida en la consulta sea inferior o igual a la administrada por el mismo, puede responderla en base a sus datos locales e interpolar los faltantes mediante el modelo de series temporales, garantizando siempre la posibilidad de respuesta y evitando la consulta a los sensores. En caso de que la consulta que se le exige al Proxy sea de una precisión superior a la administrada, entonces deberá solicitar los datos a los sensores mediante un mecanismo “Pull” para resolver la misma.

- **Procesamiento de datos sobre data streams basados en RFID** (Diao, Liu, Peng, & Sutton, 2009) (Tranh, Sutton, Cocci, Nie, Diao, & Shenoy, 2009)**:** Se plantea un sistema capaz de recibir lecturas de datos crudas o sin tratamiento previo, provenientes de diferentes lectores Radio Frecuency Identification (RFID), los cuales son objeto de inconvenientes tales como ruido, incompletitud y/o imprecisión. Éstos son abordados mediante modelos probabilísticos e inferencia a los efectos de generar una descripción de dicha incertidumbre que envuelven los datos y se emplean diferentes técnicas estadísticas para capturar los cambios que va sufriendo dicha ‘incertidumbre’. El sistema modela las variables como aleatorias y continuas, diferenciando entre dos tipos: observadas y ocultas. Las variables observadas son los datos de entrada que provienen desde la realidad captada mediante los lectores RFID. Como las variables observadas posiblemente incorporen imprecisiones, incompletitud y/o ruido, se modela la incertidumbre mediante funciones de probabilidad conjuntas entre las variables observadas y ocultas. Las variables ocultas representan dentro del modelo la verdadera ubicación de un objeto, con lo que en definitiva dicha función de probabilidad tratará de modelar la incertidumbre que proviene desde las variables observadas, estableciendo intervalos de confianza a los efectos de satisfacer requerimientos de precisión en la ubicación de un objeto.

- **Storage Manager of Streams (SMS)** (Botan, Alonso, Fischer, Kossmann, & Tatbul, 2009)**:** Se expone la necesidad de separar el procesamiento de consultas de la gestión de almacenamiento en términos de data streams a los efectos de desacoplar las componentes del DSMS. Cabe destacar que no pretende almacenar el data stream como tal, sino que por el contrario pretende brindar una alternativa independiente para lectura, actualización y control de accesos sobre cada data stream centralizando la gestión de dichas operaciones en el componente SMS y facilitando al motor de consultas, mediante una API predefinida, el acceso a cada ítem de dato en el data stream. El SMS plantea distintos parámetros arquitecturales, funcionales y relacionados con el rendimiento que le permiten adicionalmente al motor de consultas particularizarlos de acuerdo a la necesidad de lectura del data stream que desee definir.

- **DejaVu** (Lindar, Güc, Lau, Özal, Soner, & Tatbul, 2009)**:** Se trata de un sistema abocado al procesamiento de eventos, el cual integra emparejamiento de patrones sobre flujos históricos y 'vivos'. Se entiende por flujos históricos a aquellos eventos que han sido almacenados de un modo total o selectivo en un archivo físico; mientras que los flujos vivos, representan el arribo continuo de los datos mediante un mecanismo de empuje, los cuales son almacenados en memoria, actuando esencialmente como una cola. Lo interesante de este sistema en particular, es que emplea SMS (Botan, Alonso, Fischer, Kossmann, & Tatbul, 2009) para la gestión de almacenamiento en memoria de los flujos vivos y adicionalmente, ha desarrollado una extensión al procesador de consultas MySQL agregando la implementación de una máquina de estados finitos para guiar las consultas de emparejamiento de patrones, ejecutándose ésta última como parte integral del plan de consultas MySQL.

  El sistema ha sido utilizado a nivel de prototipo en seguimientos de eventos RFID en una librería. En la misma, tanto libros como personas disponían de etiquetas RFID las cuales se leían continuamente a los efectos de detectar eventos de extracción y/o depósito de libros como así también hurtos.

- **MaxStream** (Botan, y otros, 2010) (Tatbul, 2010)**:** Es un sistema de procesamiento de flujos federados que transparentemente integra motores de procesamiento de flujos (Stream Processing Engines - SPE) y bases de datos tradicionales. La idea central de su arquitectura es actuar como una capa intermedia de acceso homogéneo entre las aplicaciones clientes para con los SPEs y los Sistemas de Gestión de Bases de Datos tradicionales (SGBD) mediante una interfase única, permitiendo la reutilización del soporte prexistente al Structured Query Language (SQL) y extendiéndolo para facilitar el empleo de consultas continuas y tratamiento de ventanas.

Las alternativas de sistemas de gestión de flujos de datos presentadas presentan enfoques interesantes desde el punto de vista del procesamiento, pero ninguna de ellas plantea la incorporación de metadatos que guíen el procesamiento de los datos en sí, sino que interpretan al dato de un modo sintáctico sin considerar el significado que pudiese tener asociado.

Adicionalmente, no modelan el contexto en que los datos son obtenidos, lo cual por ejemplo en RFID podría posiblemente ayudar a disminuir la incertidumbre con respecto a posibles interferencias del ambiente ante la transmisión de datos. En general, los mecanismos de integración de fuentes de datos son “propietarias”, es decir que no permitirían la integración de otra fuente de datos de modo dinámico sin una etapa de acoplamiento previa. El análisis detectivo no está presente debido a que en principio no se trata de sistemas cuya finalidad sea el monitoreo o control de algún concepto.

Sin perjuicio de lo dicho, debe recordarse que el objeto de este tipo de aplicaciones es el procesamiento de flujos de datos específicamente y si bien, no es exactamente el planteo esta tesis, deben puntualizarse algunas cuestiones destacadas de los modelos mencionados, que deben tenerse en cuenta a los efectos del abordaje de la problemática.

PRESTO incorpora al procesamiento de flujos de datos un mecanismo predictivo basado en series temporales para disminuir el overhead de la red, mientras que la predicción es realizada por el sensor, quien decide en última instancia si transmite o no el dato. Este punto es de suma utilidad por lo que (a) delega responsabilidad funcional en el sensor y adicionalmente, (b) el sensor puede dentro de sus restricciones computacionales, puede predecir efectivamente.

El procesamiento basado en RFID plantea un abordaje muy interesante del tratamiento del ruido en la transmisión de los datos, máxime desde el punto de vista estadístico, diferenciando claramente entre variables observadas (las cuales se asocian a la realidad medida y por consiguiente, adolece de ruido, incompletitud e imprecisiones) y ocultas (las que representan la verdadera ubicación del objeto y que se desean determinar).

La iniciativa de Storage Manager of Streams (SMS) de abstraer el procesamiento de consultas de la gestión de almacenamiento ha sido relevante, por cuanto permite evitar situaciones en donde la tasa de arribo supera a la de procesamiento (cuello de botella) e independizar de hecho, la operatoria de administración de recursos asociadas al flujo con respecto a la resolución de consultas que de por sí son disímiles.

El enfoque de DejaVu se destaca en cuanto a la utilización que efectúa de SMS y en lo referido a la aptitud de combinar flujos de datos en forma conjunta con datos persistentes, incorporando dicha situación en un motor de bases de datos tradicional.

Finalmente y no por ello menos importante, MaxStream se plantea como una capa intermedia la que permite la interlocución entre entornos legados y flujos de datos, a la vez que permite el empleo de una extensión del bien conocido SQL, para realizar las consultas.

2.3 Necesidad de un Enfoque Integrado de Procesamiento de Flujos de Datos Centrado en Metadatos de Mediciones

Cuando de tomar decisiones se trata, medir no es una posibilidad sino una necesidad; representa uno de los medios lógicos por el cual se puede cuantificar el estado de un objeto y/o situación. Si hay un aspecto que se tiene en claro en la medición, es que para comparar dos mediciones diferentes las mismas deben ser consistentes entre sí, esto es poseer la misma escala y obtenerse bajo métodos o reglas de cálculos equivalentes. Los marcos formales de medición y evaluación (Ver sub-sección 2.1.5) representan un intento por formalizar el modo de definir las métricas, sus objetivos y sus restantes aspectos asociados, a los efectos de garantizar la repetitividad del proceso de medición.

C-INCAMI es un marco de medición y evaluación el cual permite formalizar métricas, indicadores, contexto de medición, entidades bajo análisis, atributos asociados a las entidades sujetos de medición, entre otros aspectos que nos permiten garantizar la repetitividad y consistencia del proceso de medición. Desde el punto de vista del proceso de medición, C-INCAMI sufre de algunos inconvenientes que posiblemente requieran adaptación, en este último sentido, la situación en que una medición es no determinista y su resultado es en realidad una distribución de probabilidad (Abajo Martínez, ANN quality diagnostic models for packaging manufacturing: an industrial data mining case study, 2004) no está contemplada y afecta tanto a métricas como a elementos contextuales, adicionalmente será necesario extender la clase *dataset* para que administre valores deterministas y probabilistas, como así también la posibilidad de generar un concepto lógico que agrupe *dataset* a modo de grupo de seguimiento. Un ejemplo de esta situación indicada puede darse con un paciente trasplantado, el cual posiblemente tenga asociado 'n' sensores que monitorearán desde frecuencia cardiaca hasta presión sanguínea, cada sensor generará su flujo de datos y en el cual es fundamental comprender al conjunto de datos generado por los diferentes sensores como una única unidad lógica de seguimiento, esa correspondencia e interpretación posterior es la que se desea incorporar extensivamente en C-INCAMI.

En cuanto al procesamiento de flujos de datos, existen tópicos que a la fecha permanecen bajo estudio en la disciplina. Alguno es estos temas versan sobre esquemas de JOINING entre flujos de datos (Arasu, Ganti, & R., 2006) (Arasu, Chaudhuri, & R., 2008) , optimización de consultas sobre flujos de datos (Babcock & Chaudhuri, 2005) (Babu, Bond, Chandramouli, & Yang, 2007) (Babu & Duan, 2007), lenguaje de consulta sobre flujos de datos (Arasu, Babu, & Widom, 2006) (Toman, 2009), entre otros. Dentro de los sistemas de gestión de flujos de datos, analizados en 2.2, ninguno se basa en marcos formales a los efectos de brindar sustento a los metadatos basados en variables numéricas, sino que cada uno plantea ad-hoc su propia semántica, lo cual afectará inevitablemente la interoperabilidad de los datos y la aplicación de los prototipos en diferentes industrias. No obstante, existen planteos muy interesantes como el de independizar el componente de gestión de almacenamiento (Storage Manager of Streams – SMS-) del motor de consultas, brindando una interfase de acceso uniforme, actualmente empleada por DejaVu y Maxstream; o bien, como en el caso de PRESTO, incorporar la posibilidad de que el sensor haga empleo de un modelo predictivo SARIMA mediante una tolerancia de error

predefinida, el cual es empleado para analizar en el mismo si el dato captado difiere del pronosticado más o menos la tolerancia y solo es transmitido, en caso de que el error de pronóstico supere la misma.

Concluyendo el capítulo, si bien existen intentos específicos en cada área en particular, *no existe un planteo integrado y global que permita simultáneamente*:

1. *Integrar flujos de mediciones a través de fuentes de datos heterogéneas*
2. *Permitir que el proyecto de medición se sustente completamente en un marco de medición y evaluación*
3. *Que el flujo de mediciones informado desde las fuentes de datos, incorpore metadatos basado en un marco formal de medición y evaluación, que permita dilucidar el significado de la medición*
4. *Que la organización y reunión de los flujos pueda procesarse de modo paralelo basado en los metadatos de medición.*

El *Enfoque Integrado de Procesamiento de Flujos de Datos* Centrado en Metadatos de Mediciones (EIPFDcMM), a introducir en el siguiente capítulo, tiene por finalidad satisfacer las necesidades indicadas de un modo integral y simultáneo, a los efectos de poder garantizar la repetitibilidad de los procesos de medición, actuar detectivamente en base a la formalización de un proyecto de medición y evaluación, como así también anticipar riesgos ante el propio arribo de las mediciones enriquecido con información contextual.

Estructura del Capítulo:

3 Panorama del Enfoque Integrado de Procesamiento de Flujos de Datos Centrado en Metadatos de Mediciones

El capítulo 2 ha permitido introducir conceptos claves para el procesamiento de flujos de datos centrado en metadatos de mediciones tales como data mining, data streams, mining data streams, context y context-aware. Luego, se ha discutido C-INCAMI como marco de medición y evaluación, a los efectos de garantizar la consistencia, repetitividad y automatización de los procesos de medición. Luego, se analizó el estado de los sistemas de gestión de flujos de datos y el asociado a árboles de clasificación sobre data streams, para concluir finalmente con la necesidad de un enfoque integrado de procesamiento de flujos de datos centrado en mediciones.

El presente capítulo presenta, en primer lugar, una visión global del enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones, para luego avanzar sobre sus principales componentes y procesos. El orden de abordaje de éstos, es determinado por la lógica del proceso de medición, análisis y toma de decisión, por lo que a seguir se describirán los procesos de recolección y adaptación, donde podrá observarse de qué modo se puede integrar diferentes fuentes heterogéneas a los efectos de transmitir mediciones.

Luego, se continúa con el análisis de los procesos de corrección y análisis, los cuales implementan la fase detectiva del enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones.

A seguir, se examinan los procesos de toma de decisión, los cuales implementan la etapa predictiva del modelo, concluyendo el capítulo con la presentación de un caso de aplicación (que se empleará a lo largo de la tesis) para el enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones.

3.1 Modelo Conceptual

Conceptualmente, la idea en términos de procesamiento de flujos (ver Ilustración 3) es la siguiente: Los datos son captados por uno o más dispositivos de medición, éstos remiten los datos sin procesar a una interface capaz de comprender el lenguaje del dispositivo e incorporar metadatos en las mediciones. Dicha interface, denominada adaptador de mediciones (Measurement Adapter -MA), comunicará el ingreso

de los datos y metadatos a una función $F^{rt}(d_{si})$ (Gathering Function -GF-), que tendrá por objeto la reunión de los flujos en función de las métricas asociadas. Los flujos reunidos, se transmiten a una función $F^{ut}(d_{si})$ (Analysis & Smoothing Function -ASF-) cuyo objetivo será suavizar las mediciones vinculadas a cada métrica. Una vez que las mediciones han sido suavizadas, se aplicará el modelo actual de clasificación $G(d^{t}_{si})$ (Current Classifier) en base al marco de métricas e indicadores definidos junto con el conocimiento previo almacenado, produciendo una decisión en tiempo t, a la cual se denomina D^{t}. Luego, el modelo actual de clasificación (Updated Classifier) se ajusta y/o sustituye incrementalmente en base a los nuevos datos y situación contextual para producir la decisión en tiempo t+1 con el nuevo modelo, permitiendo la comparación de decisiones entre D^{t} y D^{t+1} la cual estará a cargo del Tomador de Decisiones (Decision Maker –DM-), como así también la interpretación de los indicadores elementales definidos en función de los datos arribados y el análisis del conocimiento pre existente en función del contexto actual de la entidad bajo estudio.

Ambas decisiones permitirán proactivamente al tomador de decisiones, disparar alarmas, notificaciones, ajustes y/o lo que el usuario de C-INCAMI haya definido en términos de indicadores, a los efectos de detectar las desviaciones on-line.

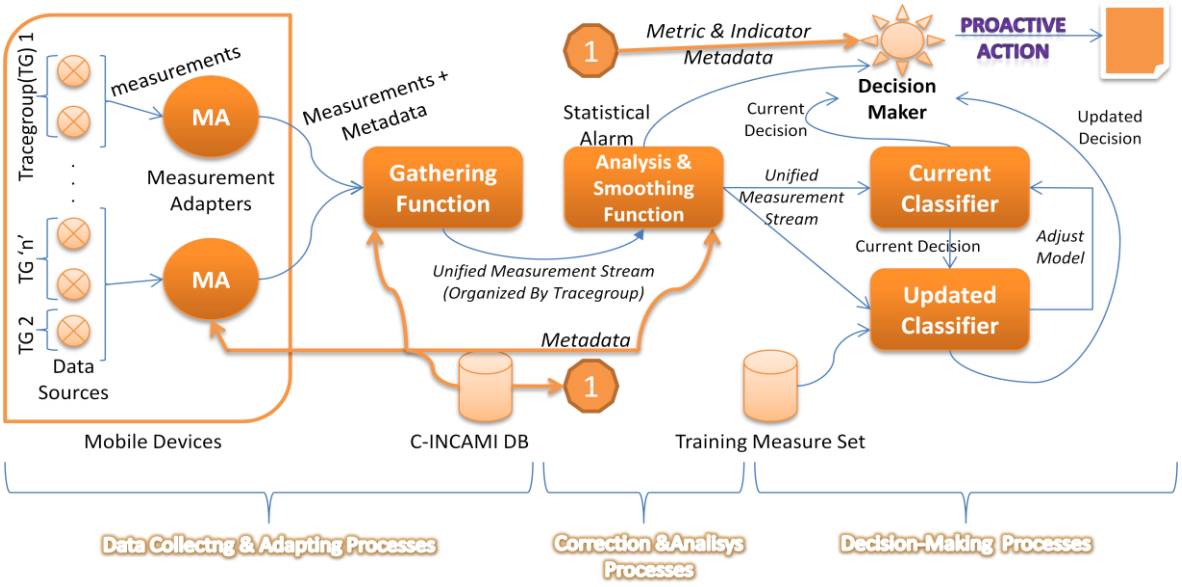

Ilustración 3. Esquema Conceptual del Modelo Integrado de Procesamiento de Flujos de Datos Centrado en Metadatos de Mediciones

El modelo se agrupa lógicamente en procesos, los cuales aquí se presentan y se detallan en las siguientes sub-secciones del capítulo:

- **Procesos de Recolección y Adaptación**: Tiene por principal responsabilidad la recolección de mediciones. Sus funciones inician a partir de la intercomunicación con los dispositivos de medición hasta la comunicación a los procesos de corrección y análisis de un flujo de mediciones unificado.

- **Procesos de Corrección y Análisis**: Su responsabilidad central radica en dar coherencia al orden de procesamiento de las diferentes mediciones en función de sus metadatos, como así también identificar y resolver automáticamente problemas típicos de datos

tales como ruido, outliers y ausencia de valor. Sus funciones comienzan con la recepción del flujo unificado desde los procesos de recolección y adaptación, culminando con la transferencia del flujo de datos con un orden de procesamiento dado y consistente numéricamente a los procesos de toma de decisión.

- **Procesos de Toma de Decisión**: Su objetivo principal es tomar una decisión en función de los datos provistos y disparar el evento correspondiente. Sus funciones comienzan con la recepción de un flujo de datos bajo los supuestos de que el orden de procesamiento seriado en el que arriban es el adecuado y que los mismos presentan la menor cantidad posible de problemas típicos de datos. Sus funciones finalizan al momento en que el componente que toma la decisión dispara el evento que correspondiese en caso de ser necesario.

Como puede apreciarse en el modelo conceptual del enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones (ver Ilustración 3), todos los procesos que abordan el procesamiento de datos tienen acceso a lo que se denomina la base de datos C-INCAMI. Esto es así, debido a que el flujo de mediciones proviene con el dato de la medición más una serie de metadatos que permiten identificar el significado del mismo como así también a la instancia de medición. El subconjunto de metadatos transmitidos con el flujo de mediciones es mínimo, por cuestiones de rendimiento de la comunicación y requieren ser extendidos con la información complementaria en la BD C-INCAMI ante su arribo, para lograr un procesamiento consistente.

3.2 Procesos de Recolección y Adaptación

Los procesos de recolección y adaptación se centran en como adecuarse a los diferentes dispositivos de medición, a los efectos de tomar las mediciones y comunicarlas luego a los procesos de corrección y análisis. Los principales componentes de la recolección y adaptación son los flujos de datos propiamente dichos, el adaptador de mediciones y la denominada función de reunión.

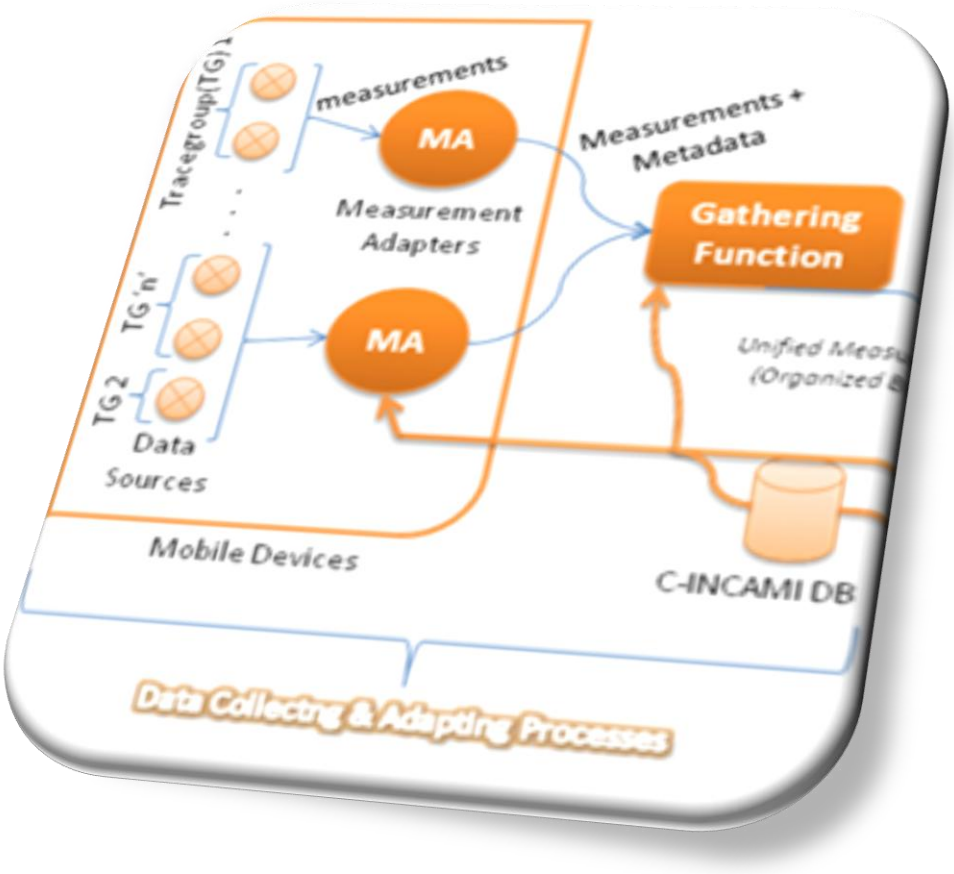

Ilustración 4. Procesos de Recolección y Adaptación

Sintéticamente, las mediciones se generan en las fuentes de datos heterogéneas, las cuales abastecen a un módulo denominado *adaptador de mediciones* (*MA en* Ilustración 4) generalmente embebido en dispositivos móviles por una cuestión de portabilidad y practicidad, aunque podría embeberse en cualquier dispositivo de cómputo con asociación a fuentes de datos. MA incorpora junto a los valores medidos, los metadatos del proyecto de medición y los informa a una *función de reunión central* (*Gathering Function –GF*). GF incorpora los flujos de mediciones en un buffer organizado por grupos de seguimiento –modo dinámico de agrupar a las fuentes de datos definido por el director del proyecto de M&E- con el objeto de permitir análisis estadísticos consistentes a nivel de grupo de seguimiento o bien por región geográfica donde se localicen las fuentes de datos, sin que ello implique una carga adicional de procesamiento. Adicionalmente, GF incorpora técnicas de *load shedding* (Rundensteiner, Mani, & Wei, 2008) que permiten gestionar la cola de servicios asociada a las mediciones, mitigando los riesgos de desborde independientemente el modo en que se agrupen.

Las fuentes de datos solo transmiten datos, no tienen ningún tipo de gestión de metadatos (ver Ilustración 4), de modo que ésta representa a cualquier dispositivo capaz de exteriorizar su medición e informarla al componente denominado Adaptador de Mediciones (Measurement Adapter -MA). Dicha iniciativa tiene por objeto independizarse de la tecnología

de implementación de los dispositivos de medición, permitiendo a cualquier dispositivo transmitir sus mediciones a través de una interface común definida en MA. De este modo, la fuente de datos más allá de cómo llevará adelante el proceso de medición para el cual fue diseñado, no tiene más que implementar la interface común de comunicación para con el MA si desease formar parte del enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones.

El adaptador de mediciones tiene una responsabilidad más compleja en términos funcionales con respecto a la fuente de datos. Inicialmente debe mantener registro de qué fuentes de datos han solicitado transmitir mediciones a través de éste y a qué entidad se asocian (bajo algún proyecto de medición dado). Esto último impone una etapa de configuración del adaptador de mediciones para con la fuente de datos, en donde el MA buscará asociar la fuente de datos con la métrica o métricas asociadas a algún atributo en particular de una entidad dentro de un proyecto de medición. Tal configuración, supone una consulta a un repositorio donde esté claramente definido las diferentes componentes del proyecto de medición y es allí donde, desde el MA, se consulta la base de datos C-INCAMI. Adicionalmente, el MA puede actualizar el repositorio de metadatos C-INCAMI con información tal como los tipo de dispositivos que implementan la métrica, último dispositivo asociado, si la métrica está siendo efectivamente implementada en un momento de tiempo dado, entre otros parámetros que permiten enriquecer el monitoreo del proyecto de medición.

A partir de la asociación entre flujos de datos y métricas que implementa, el MA *conoce* cuáles son sus metadatos a los efectos de incorporarlos en el flujo de mediciones. El MA debiera ser lo más ágil posible en términos de procesamiento y empleo de memoria, ya que debe residir en las proximidades de la localización de los dispositivos de medición y esto en general, impone límites de recursos propio de los dispositivos móviles. Es por ello, que el MA debe incorporar solo los metadatos básicos y necesarios en los datos a comunicar y transmitir finalmente cuando sea necesario. Así, la idea planteada en los sensores de PRESTO (Li, Ganesan, & Shenoy, 2006), donde un modelo de serie temporal basado en SARIMA (Box & Jenkins, 1991) es construido de acuerdo a las tendencias observadas, transmitiéndose a los sensores los parámetros del modelo a los efectos de que determinen si la medición es acorde a la predicción con el objeto de informar solo las desviaciones, toma particular importancia en este punto. Siguiendo con esta última línea de razonamiento, se sabe que tiene un costo computacional estimar los parámetros de un modelo SARIMA, pero dada la información observada para una serie temporal, limitada y de historia reciente de datos, almacenada dentro de la base de datos C-INCAMI, es posible disponer de parámetros iniciales e informarlos al MA para que efectúe la predicción. De este modo y al igual que PRESTO, el MA informaría desviaciones a la predicción ante la presencia de parámetros del modelo SARIMA, lo que permitiría optimizar el empleo de recursos desde anchos de banda, disminuir overhead de la comunicación, adecuar tiempos de procesamiento, entre otros. En caso de no disponer el MA de los parámetros del modelo SARIMA, transmitiría la totalidad de las mediciones bajo la modalidad de ráfagas, en función de la configuración y limitación de la memoria local del dispositivo que ejecuta el MA.

De lo mencionado, se deduce por un lado que la estimación de los parámetros SARIMA no es responsabilidad del MA, sino que en realidad caen bajo las obligaciones de los procesos de toma de decisión, los cuales deberán informar los mismos mediante el proceso de inicialización de un MA. Por otro lado, una fuente de datos solo puede asociarse con un MA, esto es así para evitar la duplicación de datos y una doble predicción sobre las mediciones en MA diferentes. Finalmente, un MA puede receptar tantos parámetros SARIMA como métricas diferentes sean implementadas por las diferentes fuentes de datos, ya que cada serie de mediciones asociada a una métrica puede entenderse como una serie temporal en sí misma.

Una vez que el MA ha calculado sus predicciones y determina que información deberá transmitir a la función de reunión (Gathering Function -GF-), viene el aspecto de con qué estrategia transmitir. Esto último implica decidir sobre si gestionará un buffer local o bien, transmitirá sobre cada desviación detectada. Ambas posibilidades pueden ser implementadas por el componente, bajo la premisa de mantener estabilidad del sistema, uso limitado de memoria y capacidad de procesamiento, teniendo en cuenta que posiblemente se encontrará sobre una plataforma de dispositivos móviles aunque no es restrictivo a ellos.

La función de reunión receptará los datos y metadatos desde el adaptador de mediciones bajo un esquema CINCAMI/MIS (Diván, Molina, & Olsina, 2008) y podrá agrupar las mismas si el objeto de medición fuese equivalente. Esto último implica que si una métrica aplicada al mismo atributo de entidad bajo un proyecto dado, es implementada por dos fuentes de datos diferentes, al arribar los datos, la función de reunión deberá agruparlas bajo un mismo flujo a los efectos del análisis estadístico posterior, dado que se trata en definitiva de dos dispositivos físicos que implementan igual objetivo de medición. Es importante mencionar que la función de reunión puede efectuar tal agrupamiento de flujos, gracias a que es capaz de consultar la base de datos C-INCAMI donde residen los metadatos del proyecto de medición. Esta situación particular de la función de reunión, abre la posibilidad a que se aborde un entorno de procesamiento paralelo, dado la independencia de los flujos de datos y la agilidad de fusionamiento que se dispondría en caso de flujos vinculados, como la situación mencionada en que se implementen igual métrica en diferentes dispositivos de medición.

3.3 Procesos de Corrección y Análisis

Sintéticamente, una vez que las mediciones se encuentran organizadas en el buffer de la función de reunión (Gathering Function), se aplica mediante arrastre de los datos desde el buffer *análisis descriptivo, de correlación y componentes principales* (*Analysis & Smoothing Function –ASF-*) guiados por sus propios metadatos, a los efectos de detectar situaciones inconsistentes con respecto a su definición formal, tendencias, correlaciones y/o identificar las componentes del sistema que más aportan en términos de variabilidad. De detectarse alguna situación en ASF, se dispara una alarma estadística al *tomador de decisiones* (*Decision Maker - DM*) para que evalúe si corresponde o no disparar la alarma externa (vía, e-mail, SMS, etc) que informe al personal responsable de monitoreo sobre la situación.

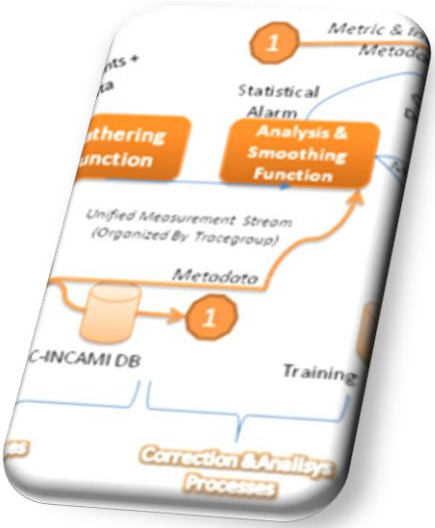

Ilustración 5. Procesos de Corrección y Análisis

El proceso de corrección se basa en un abordaje estadístico de los datos que se sustenta en la presencia de metadatos, esto permite enriquecer el análisis mediante el conocimiento previo de la semántica del mismo.

Adicionalmente, el proceso abordará y determinará el orden coherente de procesamiento, el cual es consensuado mediante la prioridad que se haya fijado para cada una de las métricas dentro del proyecto de medición. Esto último incorpora la idea de que no todas las métricas dentro del proyecto poseen igual importancia, en efecto, existe en cualquier circunstancia de medición aspectos más destacados y prioritarios y otros no tanto, de este modo este proceso permite abordar selectivamente, previo a aplicar técnicas estadísticas, el tratamiento de los datos prioritarios.

Una vez que se ha producido el reorden de los flujos de mediciones dado por la prioridad de las métricas, la cual es obtenida mediante consulta a la base de datos C-INCAMI (ver Ilustración 5), se aplican distintas técnicas estadísticas (Análisis Descriptivo, Análisis de Componentes Principales y Análisis de Correlación) para abordar problemáticas de ruidos, outliers y valores faltantes. Claro está, que dichas técnicas requerirán que se configuren las acciones a tomar o parámetros previos tales como confianza requerida dentro del proyecto de medición, para que el análisis sea en verdad automatizado y coherente.

Culminado el tratamiento de los datos de acuerdo a la parametría del proyecto de medición, se informa el flujo al proceso de toma de decisión y en particular a dos componentes: Al modelo de clasificación y al tomador de decisión. El modelo de clasificación requerirá los datos para tomar una decisión (Current Classifier en Ilustración 3) y posteriormente actualizar su modelo, mientras que el tomador de decisión (Decision Maker en Ilustración 3) se nutrirá de los datos bases más las decisiones de los modelos de clasificación para sustentar su accionar.

3.4 Procesos de Toma de Decisión

Una vez que los nuevos flujos de mediciones son comunicados al *clasificador vigente* (*Current Classifier –CC- en* Ilustración 3) desde la función de reunión (Gathering Function -GF-), éste deberá clasificar las nuevas mediciones si corresponden o no a una situación de riesgo e informar dicha decisión al DM. Simultáneamente, se reconstruye el CC incorporando las nuevas mediciones al conjunto de entrenamiento y produciendo con ellas un *nuevo modelo* (*Updated Classifier –UC- en* Ilustración 3). El UC clasificará las nuevas mediciones y producirá una decisión actualizada que también será comunicada al DM. El DM determinará si las decisiones indicadas por los clasificadores (CC y UC) corresponden a una situación de riesgo y en cuyo caso con qué probabilidad de ocurrencia, actuando en consecuencia según lo definido en el umbral mínimo de probabilidad de ocurrencia definido por el director del proyecto. Finalmente, independientemente de las decisiones adoptadas, el UC se torna en CC sustituyendo al anterior, en la medida que exista una mejora en su capacidad de clasificación según el modelo de ajuste basado en curvas ROC (*Receiver Operating Characteristic*) (Duin, Tortorella, & Marrocco, 2008).

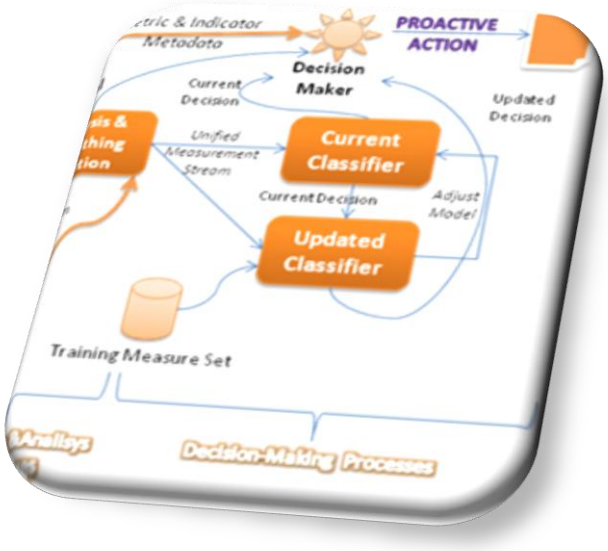

Ilustración 6. Procesos de Toma de Decisión

El proceso de toma de decisión deberá tomar los flujos de datos, bajo el supuesto de que los mismos han sido previamente tratados en términos estadísticos y presentan la menor cantidad posible de inconvenientes en términos de ruidos, outliers y valores faltantes. No obstante, aunque minimizada la problemática debe considerarse el margen de error propio de la parametría del proyecto de medición y por ende, la posibilidad de que persista rastro de incongruencias de datos no detectados y a los cuales deberán hacer frente los clasificadores al momento de procesar los datos.

El proceso de toma de decisión, como puede apreciarse en la Ilustración 6, se compone de una BD de entrenamiento, un clasificador actual, un clasificador actualizado y el tomador de decisión. La BD de entrenamiento almacena en forma estructurada la parametría y datos con la historia reciente, que son empleados para la estimación de los parámetros de

SARIMA a los efectos de inicializar los MA, como así también el subconjunto de mediciones de entrenamiento basado en experto, de los casos de riesgo y normalidad para la entidad bajo medición, con los cuales se entrena los clasificadores durante su inicialización.

Se dispone de dos clasificadores incrementales basado en árbol, uno clasifica sin actualizar el modelo y en base al dato recientemente arribado, denominado $G(d^t_{si})$ (Current Classifier en Ilustración 6), mientras que el segundo actualiza el modelo con el dato recientemente arribado y con el nuevo modelo, denominado $G'(d^t_{si})$(Updated Classifier en Ilustración 6), clasifica el dato recientemente arribado. Ambas clasificaciones, se informan en forma conjunta con el flujo de datos proveniente del proceso de corrección y análisis al tomador de decisiones quien, en base a la información de los indicadores definida en el proyecto de medición, tomará las acciones pertinentes. El clasificador $G(d^t_{si})$ es reemplazado por $G'(d^t_{si})$, de modo que ante un nuevo arribo de datos, el clasificador que en un tiempo t+1 se consideró actualizado, en el tiempo t+2 será el actual, $G(d^t_{si})$.

3.5 Paralelismo y Concurrencia

El enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones, presenta una oportunidad clave para aplicar paralelismo y éste se asocia con el momento en que los flujos de mediciones son receptados en la función de reunión, tal y como puede observarse en la Ilustración 3 y Ilustración 4.

Naturalmente, cada adaptador de mediciones (Ver Ilustración 4) agrupa una serie de sensores los cuales, a través del mismo, permiten conformar uno o más grupo de seguimientos. De este modo, el adaptador de mediciones puede transmitir un flujo de datos con datos y metadatos de varios sensores pero en donde, todos ellos, corresponden a un mismo flujo de datos. De este modo, la función de reunión, quien es responsable de la recepción y reunión de los flujos, debe abordar la posibilidad cierta de recibir flujos de mediciones paralelo y simultáneamente desde ‘n’ adaptadores de mediciones. Tal y como se presentará en la sub-sección siguiente mediante el caso de aplicación, un paciente trasplantado ambulatorio puede tener sobre sí un conjunto de sensores determinados para medir presión arterial, ritmo cardíaco, humedad ambiente, entre otros. Dichos sensores se comunican mediante tecnología bluetooth con un dispositivo móvil asociado al paciente, por ejemplo una Palm Treo 750 u otro similar, en donde se ejecuta el programa móvil adaptador de mediciones quien reúne las medidas y las transmite a la función de reunión como flujo único para el grupo de seguimiento del paciente “X”. De este modo, se puede tener una cantidad variable de pacientes bajo monitoreo, lo que abre la posibilidad del procesamiento simultáneo y paralelo en cuanto a la recepción y reunión de los mismos. Éste aspecto en particular, es el abordado por el enfoque integrado de procesamiento de flujos de datos en términos de paralelismo por su criticidad, dado que en este contexto no sería aceptable bajo ningún punto de vista una serialización, dado que implicaría que las mediciones provenientes del paciente B serían procesadas recién luego de procesar al paciente A, por lo que si el paciente B se encontrase en riesgo y el paciente A estuviese sano, se estaría incrementando el riesgo del paciente B por la misma demora en la detección de su caso.

Dado que la recepción de flujos de mediciones (los cuales por definición pueden asumir un comportamiento bajo la modalidad de ráfagas) se efectúa mediante servicios web,

la recepción de los mismos es paralelizable tanto como adaptador de mediciones desee transmitir. Ahora bien, el enfoque integrado de procesamiento de flujos de datos ha planteado la necesidad de organizar y centralizar el buffer de mediciones a los efectos de ganar consistencia mediante los metadatos incorporados en los propios flujos de mediciones, sustentados éstos por una base conceptual consistente y robusta como la asociada a C-INCAMI. Este último aspecto y considerando la recepción de flujos de mediciones en paralelo, plantea la situación cierta de que dos o más flujos de mediciones, intenten leer y/o escribir el buffer central en forma simultánea. Más aún, no solo los flujos de mediciones pueden intentar escribir el buffer central en forma simultánea, sino desear acceder a la misma región del buffer con operaciones posiblemente diferentes o no, desencadenando la posibilidad concreta de un acceso concurrente que debe ser gestionado desde el enfoque integrado de procesamiento de flujos de datos (EIPFD) centrado en metadatos de mediciones.

Finalmente y sintetizando lo expuesto, el EIPFD deberá gestionar el paralelismo incorporado naturalmente en la recepción y reunión de los flujos de datos provenientes de los adaptadores de mediciones, como así también a la gestión de transacciones con vistas a regular los accesos concurrentes para evitar corrupción de datos, sin que ello implique serializar el acceso al buffer global.

3.6 Caso de Aplicación

El caso de aplicación a presentar en esta sub sección tiene por objeto ilustrar el Enfoque Integrado de Procesamiento de Flujos de Datos Centrado en Metadatos de Mediciones (Diván & Olsina, 2009). La idea subyacente es que los médicos del centro de salud puedan evitar reacciones adversas y daños mayores en la salud del paciente en la medida en que puedan disponer de un seguimiento continuo del mismo. Es decir, que puedan disponer de un mecanismo por el cual les informe ante variaciones no previstas y/o inconsistencias en indicadores de salud definidos por ellos para un tipo de trasplante realizado en particular. En definitiva, la idea central es que exista proactivamente algún mecanismo que, basado en las métricas e indicadores de salud definidas por los especialistas para un tipo de trasplante (y potencialmente para segmentos de edades), informe sobre situaciones que pudieran afectar la salud del paciente bajo monitoreo.

A los efectos de ejemplificar los factores contextuales a evaluar como así también los atributos de los cuales se desea mantener información permanente del paciente, los especialistas definen el siguiente proyecto de medición en base al marco C-INCAMI, presentado en la sub sección 2.1.5.

La necesidad de información es "*monitorear los principales signos vitales en un paciente trasplantado al momento en que se le da el alta desde el centro médico*". La entidad bajo análisis es el *paciente trasplantado ambulatorio* (notar que podría definirse entes más específicos como *paciente trasplantado anciano*, entre otros, con el fin de ser más precisos en la definición e interpretación de indicadores). Según los expertos, la *temperatura corporal*, la *presión arterial sistólica* (máxima), la *presión arterial diastólica* (mínima) y la *frecuencia cardiaca* representan los atributos de los signos vitales relevantes a monitorear en este tipo de paciente. Además, los expertos señalan que es necesario monitorear la *temperatura ambiental*, la *presión ambiental*, la *humedad* y la *posición del paciente* (latitud y longitud) como parte de las propiedades de contexto. La necesidad de información junto con la definición de la entidad, sus atributos y contexto, forman parte de la "*Definición y Especificación de Requerimientos no Funcionales*" y de la "*Definición del Contexto del Proyecto*" (ver sub sección 2.1.5).

La cuantificación de los atributos se realiza por medio de las métricas conforme al componente *Diseño y Ejecución de la Medición*. Para el monitoreo, se desea disponer de las métricas que cuantifiquen a los atributos citados, a saber: la presión arterial sistólica, presión arterial diastólica, temperatura corporal y frecuencia cardiaca.

Con el objeto de ejemplificar cómo se definen los metadatos vinculados a las métricas según C-INCAMI, se especifica en la Tabla 1, la definición de métrica asociada al atributo temperatura corporal.

Atributo	Temperatura Corporal
Métrica	Valor de la temperatura axilar
Código Id.	VTA
Tipo de Métrica	Directa
Escala	**Tipo de Escala** / **Intervalo** **Dominio de Valores** / **Numérica, Continua, Real^{+}** **Unidad** / **Grados Centígrados (^{o}C)**
Método	**Nombre** / **Axilar** **Método de Medición** / **Objetivo** **Especificación** / **Embebida en el instrumento** **Instrumento** Nombre / Omron ECO-TEMP Versión / ECO-TEMP Proveedor / Omron Descripción / Termómetro electrónico digital para uso oral, rectal o axilar

Tabla 1. Definición de la Métrica Valor de la Temperatura Axilar

En cuanto a las propiedades de contexto, se desea disponer de un monitoreo sobre la temperatura ambiental, la presión ambiental, la humedad y la posición del paciente (latitud y longitud). La Tabla 2 especifica la definición para una de ellas: La Temperatura Ambiente.

Context ID	CtxPacienteTrasplantadoAmbulatorio
Propiedad de Contexto	**Temperatura Ambiente**
Métrica	Valor de la temperatura ambiental del paciente transplantado
Código Id.	VTAPT
Tipo Métrica	Directa
Escala	**Tipo de Escala** / **Intervalo** **Dominio de Valores** / **Numérica, Continua, Real^{+}** **Unidad** / **Grados Centígrados (^{o}C)**
Método	**Nombre** / **Sensado de temperatura** **Método de Medición** / **Objetivo** **Especificación** / **Embebida en el instrumento** Instrumento **Nombre** / **Termómetro Digital** **Versión** / **TCH305003** **Proveedor** / **Technidea S.A.** **Descripción** / **Dispositivo que permite medir la temperatura interna y humedad relativa al ambiente con un sensor de 3 metro.** **Rangos:** • **Temperatura 10^{o} a +60^{o}** • **Humedad 10% a 99%**

Tabla 2. Definición de la Propiedad de Contexto Temperatura Ambiente

Para el escenario actual, los expertos ya han consensuado el conjunto de métricas y propiedades contextuales a monitorear. Ahora, deben establecer los indicadores elementales,

a los efectos de sentar la base para la interpretación de los atributos y conceptos calculables. De este modo, han definido los siguientes indicadores elementales: el nivel de temperatura corporal, el nivel de presión, el nivel de la frecuencia cardiaca y el nivel de diferencia de la temperatura corporal versus la temperatura ambiental. Estos indicadores integran el componente *Diseño y Ejecución de la Evaluación* (ver sub-sección 2.1.5).

Una vez establecida la necesidad de información, el ente a monitorear, los atributos a medir, las propiedades contextuales y los indicadores elementales conforme a los criterios dado por los expertos, se estará en condiciones de instalar y configurar el MA en un dispositivo móvil –el del paciente-, el cual trabajará en forma conjunta con los sensores tal y como se expone en la

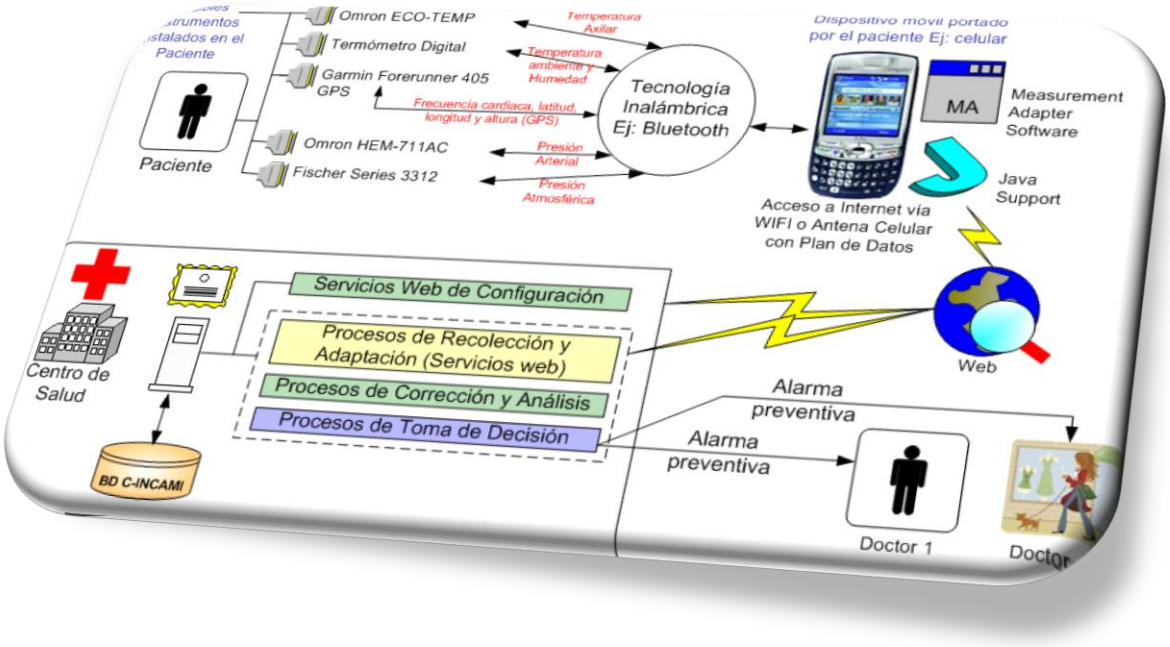

Ilustración 7. Esquema de Aplicación del Enfoque Integrado de Procesamiento de Flujos de Datos Centrado en Metadatos de Mediciones a Pacientes Ambulatorios Trasplantados

El MA tomará las mediciones desde los sensores (fuentes de datos) e incorporará los metadatos vinculados a las métricas y propiedades de contexto que se relacionaron con éstos. Siguiendo con el ejemplo, incorpora el identificador de la propiedad de contexto temperatura ambiente (VTAPT, ver Tabla 2) en forma conjunta con el valor del dato a transmitir, supongamos 37°C para la temperatura ambiente; y así para cada atributo y propiedad de contexto. Datos y metadatos se transmiten mediante el esquema C-INCAMI/MIS (Diván, Molina, & Olsina, 2008), el cual será profundizado en el siguiente capítulo, a la función de reunión (Gathering Function -GF-, ver Ilustración 4. Procesos de Recolección y Adaptación). La función de reunión es el único componente de los procesos de recolección y adaptación que se sitúa en el centro de salud en forma conjunta con los procesos de corrección y análisis y toma de decisión.

Cuando la función de reunión recibe las mediciones desde los diferentes pacientes bajo monitoreo, ordena las mismas por paciente (Grupo de Seguimiento) y las transmite a los

procesos de corrección y análisis los que, principalmente, tratarán de resolver problemáticas propias de los datos tales como los valores faltantes, ruidos, etc. En este sentido y gracias a los metadatos, si se recepta por ejemplo para la métrica "Valor de la Temperatura Axilar" un valor 0, por la misma definición de la métrica el modelo de procesamiento identifica automáticamente un error en el dato, dado que el tipo de escala es intervalo y su dominio de valores es Numérico, Continuo y Real+.

Si bien desde el paciente arriban y se analizan en simultáneo todas las métricas y propiedades de contexto configuradas por los expertos, por un momento considere que sólo se están recibiendo datos de la temperatura axilar y la temperatura ambiental desde el paciente y que es factible visualizarlo. De este modo, si observa la Ilustración 8, podrá apreciar el límite inferior y superior definido para el indicador "*Nivel de la Temperatura Corporal*" junto con la evolución de la *temperatura ambiental* (serie indicada con línea continua) y la *temperatura axilar* (serie indicada con línea punteada).

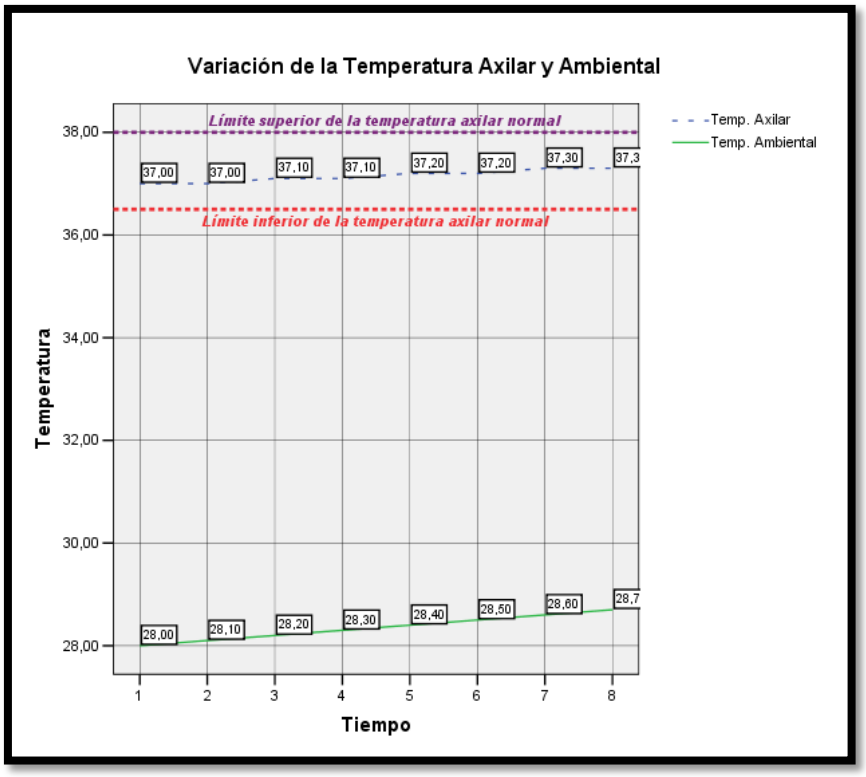

Ilustración 8. Visualización de las Mediciones para la Temperatura Axilar y la Temperatura Ambiental

Los datos de las medidas, e incluso el nivel de aceptabilidad del indicador elemental expuestos en la Ilustración 8, arrojarían que el paciente se encuentra en una situación normal. No obstante, los procesos de toma de decisión (ver Ilustración 6. Procesos de Toma de Decisión en la sub-sección 3.4) además de analizar los indicadores en sí mismos, analizan en forma conjunta la interacción entre indicadores, métricas y factores contextuales lo que permite detectar una situación como la expuesta por la Ilustración 9.

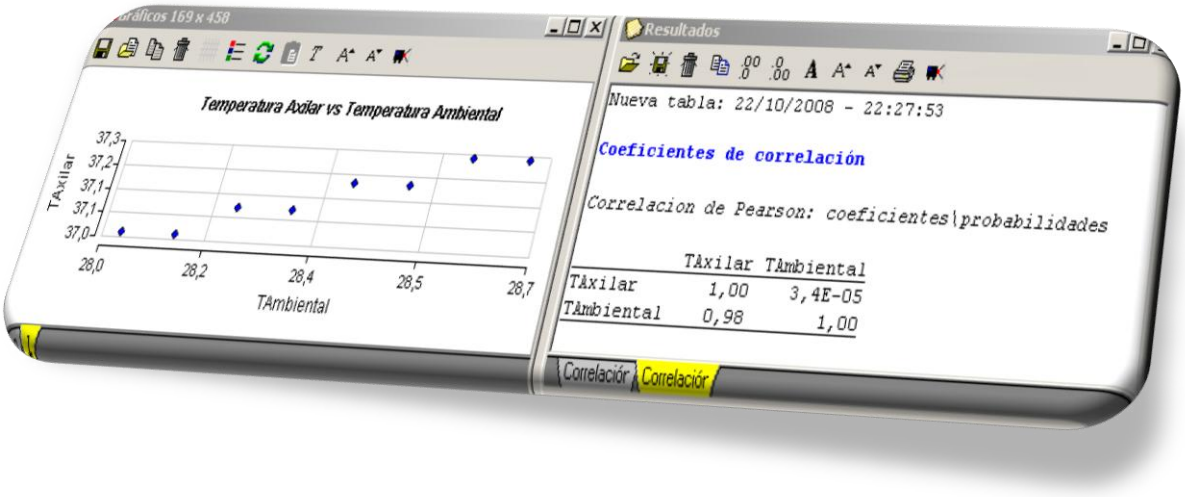

Ilustración 9. Análisis de Correlación de la Temperatura Axilar versus la Temperatura Ambiental

Lo que parecía normal y evidente, posiblemente no lo era tanto, dado que el modelo detectó una correlación, como puede apreciarse en la Ilustración 9, entre la temperatura axilar y la temperatura ambiental. Esto determinará posiblemente una emisión de alarma preventiva desde el centro de salud a los médicos, ya que el incremento de la temperatura ambiental arrastra a la corporal y esta situación puede implicar un incremento gradual del riesgo para el paciente.

4 Transmisión de Mediciones

El capítulo 3 ha permitido obtener una visión global del enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones. Ha expuesto claramente los objetivos de los procesos de recolección y adaptación, el rol de los procesos de suavización y análisis para con la fase detectiva, como así también la relación entre los procesos de toma de decisión y la fase predictiva.

Este capítulo, presenta el esquema C-INCAMI/MIS mediante el cual se estructura y transmiten los flujos de mediciones. Luego, plantea las interfaces de transmisión que emplea el enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones. A seguir, discute el rol del adaptador de mediciones a los efectos de la transmisión de las mediciones a través del flujo. Se analiza la gestión de mediciones mediante buffer multinivel, quedando expresamente indicado el modo de organización central de las mediciones a partir de múltiples flujos de datos, los cuales arriban en paralelo para su procesamiento. Finalmente se discuten aspectos de load shedding para regular la tasa de procesamiento y el proceso de recepción de mediciones.

4.1 Esquema C-INCAMI/MIS

El esquema utilizado para el intercambio de mediciones entre las fuentes de datos y el proveedor de servicios de transmisión, se denomina CINCAMI / Measurement Interchange Schema (CINCAMI/MIS). El mismo se basa en el marco de medición y evaluación C-INCAMI, con el objeto de incorporar en forma conjunta a as mediciones, metadatos que permitan enriquecer a posteriori el análisis de consistencia.

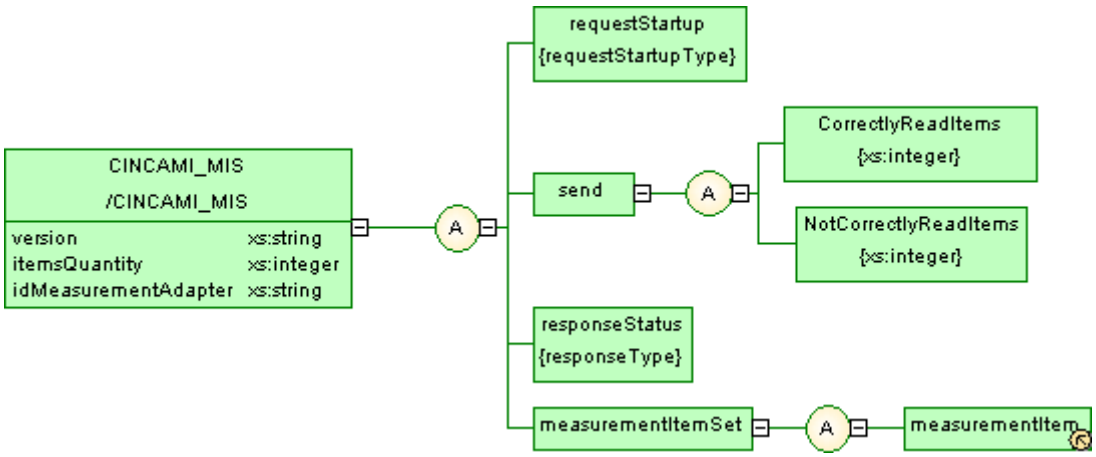

Ilustración 10. Nivel Superior de C-INCAMI/MIS

La etiqueta principal del esquema es CINCAMI/MIS, el mismo tiene asociado los atributos *version,* el cual señala la versión del esquema, *itemsQuantity* el cual indica la cantidad de unidades lógicas de medición a transmitir y *idMeasurementAdapter* que es el identificador único del adaptador de mediciones del cual proviene el mensaje.

Las cuatro subetiquetas que componen CINCAMI/MIS, expuestas en la Ilustración 10, son opcionales. Esto implica que el mismo esquema puede ser empleado para mensajes de control o bien para transmisión de mediciones propiamente dichas. Dentro de las subetiquetas, requestStartup es una etiqueta opcional que representa un mensaje de control, la cual puede asumir solo los valores *startup* o *reverse. Startup* puede enviarse previo al envío de mediciones, para conocer el estado del proveedor del servicio de transmisión. *Reverse* es un mensaje de control que se envía porque ha ocurrido un timeout durante un mensaje anterior y no se posee confirmación de estado del receptor. La subetiqueta *responseStatus* se vincula solo con respuestas del proveedor e indica el estado del mismo, pudiendo solo asumir los valores *online* u *offline*. La subetiqueta *send* se asocia con mensajes de respuesta e indica cuantas unidades lógicas de medición han podido ser leídas correctamente y cuantas han sido descartadas. Finalmente, la etiqueta *measurementItemSet* es la etiqueta que engloba las unidades lógicas de medición, agrupando un conjunto de etiquetas *measurementItem,* la cual es ampliada en la Ilustración 11.

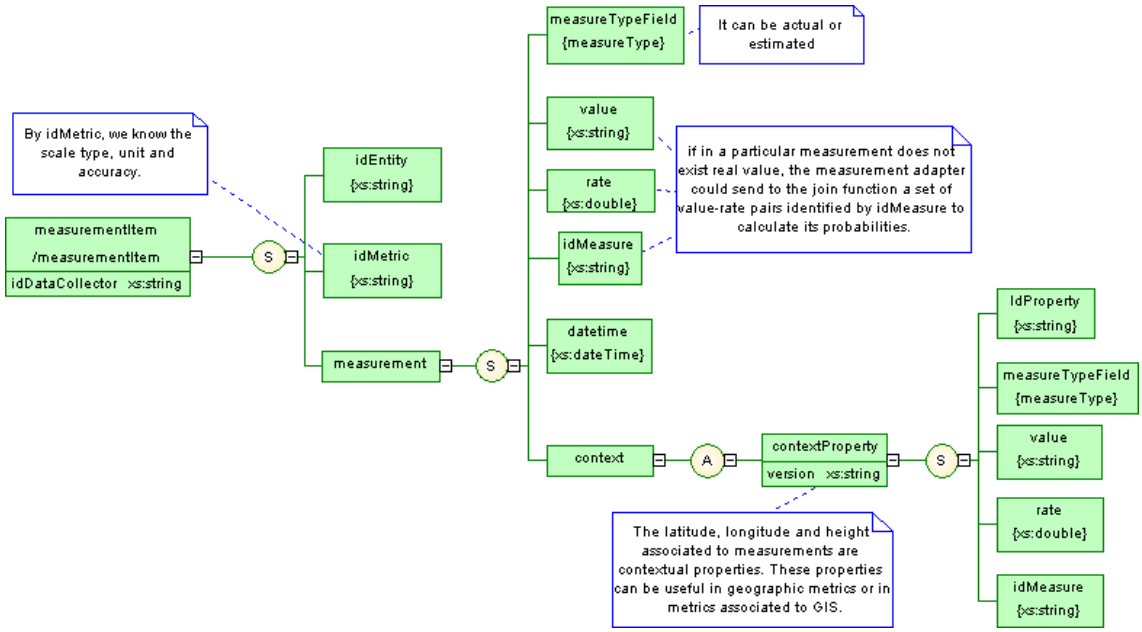

Ilustración 11. Nivel C-INCAMI/MIS para Unidades Lógicas de Medición

La etiqueta *measurementItemSet* posee un único atributo, *idDataCollector*. Este atributo es de singular importancia dado que informa el identificador de la fuente de datos que está informando la unidad lógica de medición. De este modo, el esquema es apto para informar a través de un único mensaje, mediciones provenientes de diferentes fuentes de datos. La etiqueta se compone de tres subetiquetas obligatorias, *idEntity* la cual representa el identificador de la entidad con la que se asocia *idMetric,* que a su vez indica el identificador de la métrica sobre la que se está informando una medición. La medición propiamente dicha se informa agrupada mediante la etiqueta *measurement.*

La etiqueta *measurement* brinda la posibilidad de informar en forma conjunta la medición con respecto a las propiedades de contexto. Las subetiquetas obligatorias son *measureTypeField, value y datetime.* La subetiqueta *measureTypeField* puede asumir solo los valores: *actual* y *estimated,* los cuales indican si la medición fue determinista o probabilística respectivamente. La subetiqueta *value* representa un valor concreto, que se asume determinista o no en función del valor que asuma *measureTypeField*. La etiqueta *datetime* informa fecha y hora en que se tomó la medición en la fuente de datos.

En caso de que el valor asumido por *measureTypeField* fuese *estimated*, se informarán las subetiquetas *rate* y *idMeasure.* La subetiqueta *rate* indica la probabilidad asociada al valor indicado en *value,* mientras que *idMeasure* permite agrupar la distribución de probabilidad entre diferentes etiquetas *measurement,* considerándolas fruto de una misma medición y no como mediciones separadas.

La etiqueta *context* es opcional ya que no existe obligación que todo proyecto deba tener asociado un contexto, con lo cual puede ocurrir que deban informarse mediciones y las mismas no tener contexto asociado. En caso de poseer contexto asociado, *context* se compone de un conjunto de etiquetas *contextProperty* la cual contiene un único atributo, *version,* el cual representa la versión de propiedad de contexto empleada. Cada etiqueta *contextProperty* representa una medición efectuada sobre una propiedad de contexto y tendrá como

subetiquetas obligatorias a *idProperty*, *measureTypeField* y *value*. *idProperty* indica la propiedad de contexto con la que se asocia, *measureTypeField* indica el tipo de valor informado con la propiedad de contexto *y value* especifica un valor dado. La subetiqueta *measureTypeField* puede solo asumir *actual* o *estimated* como valor. Si asumiese el valor *actual* entonces el valor de la subetiqueta *value* es determinista, pero si por el contrario fuese *estimated*, deberán informarse las subetiquetas *rate* y *idMeasure.* La subetiqueta *rate* indica la probabilidad asociada al valor indicado en *value,* mientras que *idMeasure* permite agrupar la distribución de probabilidad entre diferentes etiquetas *contextProperty,* considerándolas fruto de una misma medición y no como mediciones separadas.

4.2 Interfaces de Transmisión

Una vez que las diferentes fuentes de datos han sido configuradas dentro del modelo integrado de procesamiento de flujos, es necesario definir la funcionalidad a implementar, a los efectos de materializar la transmisión de las mediciones. En este sentido se plantea una interfase base denominada *transmitionService* y a partir de ella, se especializa en las interfaces *transmitionServiceConsumer* y *transmitionServiceProvider* según si el rol a desempeñar sea consumidor o proveedor del servicio respectivamente, como puede apreciarse en la Ilustración 12.

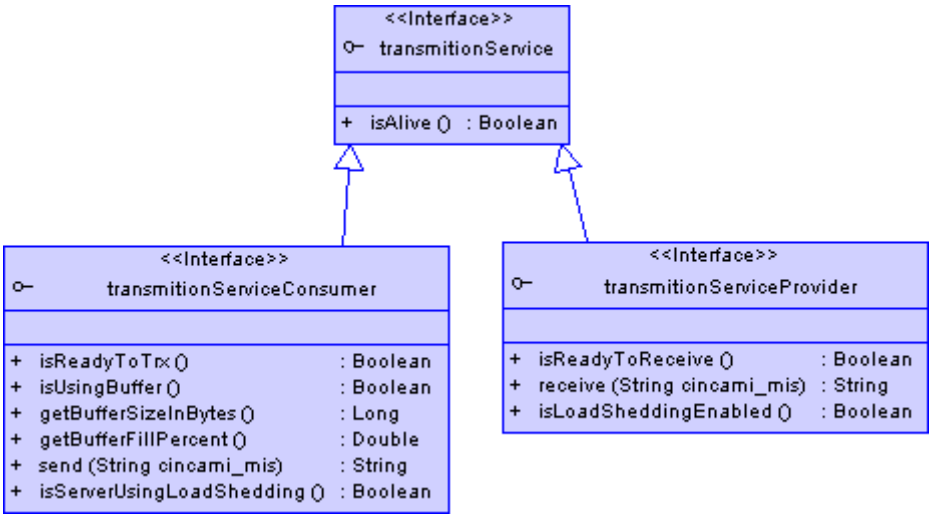

Ilustración 12. Interfaces asociadas a la transmisión

La interface *transmitionService* posee un único método asociado el cual se explicita en la tabla a continuación.

Operación	isAlive()
Objetivo	Verifica si consumidor o receptor se encuentra disponibles al momento de invocar el método.
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE = Si se encuentra disponible

	FALSE = En caso de no estar en condiciones de transmitir ó recibir mediciones

Tabla 3. Interfaces transmitionService. Método isAlive.

La interface *transmitionServiceConsumer* representa la interface que *debe* ser implementada por quien desee transmitir las mediciones recolectadas desde diferentes fuentes de datos. A continuación, se exponen cada una de los métodos específicos asociados a dicha interface, organizados en forma de tablas a los efectos de explicitar el alcance total de su funcionalidad.

Operación	**isReadyToTrx()**
Objetivo	Indica si se está en condiciones para enviar el flujo de mediciones a la unidad de procesamiento central y si ésta última está en condiciones de recibirlo de acuerdo a su situación actual.
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE = Si se encuentra listo FALSE = En caso de no estar en condiciones de transmitir mediciones

Tabla 4. Interface transmitionServiceConsumer. Método isReadyToTrx.

Operación	**isUsingBuffer()**
Objetivo	Indica si a los efectos de la transmisión de mediciones se está empleando Buffer local
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE = Si emplea un buffer local para almacenar las mediciones previo envío, FALSE = En caso contrario

Tabla 5. Interface transmitionServiceConsumer. Método isUsingBuffer.

Operación	**getBufferSizeInBytes ()**
Objetivo	Retorna el tamaño aproximado en bytes que actualmente ocupa el buffer local
Parámetros	-
Tipo Retorno	Long
Retorno	Tamaño aproximado en bytes que actualmente ocupa el buffer local en caso de estar utilizándose, null en caso contrario.

Tabla 6. Interface transmitionServiceConsumer. Método getBufferSizeInBytes().

Operación	getBufferFillPercent ()
Objetivo	Retorna el porcentaje de llenado del buffer local.
Parámetros	-
Tipo Retorno	Double
Retorno	Retorna el porcentaje de llenado del buffer en caso de estar utilizando uno, null en caso contrario.

Tabla 7. Interface transmitionServiceConsumer. Método getBufferFillPercent.

Operación	send(String cincamimis)		
Objetivo	Envía el flujo de mediciones según el esquema C-INCAMI / MIS.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	cincamimis	Mediciones estructuradas según el esquema C-INCAMI /MIS.
Tipo Retorno	String		
Retorno	Datos y metadatos según CINCAMI/MIS con un resumen retornado por la unidad de procesamiento sobre las mediciones recibidas junto con información de estado del mismo.		

Tabla 8. Interface transmitionServiceConsumer. Método send.

Operación	isServerUsingLoadShedding ()
Objetivo	Verifica si el servidor está empleando algún método de load shedding.
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE si el servidor emplea algún método de load shedding, FALSE en caso contrario.

Tabla 9. Interface transmitionServiceConsumer. Método isServerUsingLoadShedding.

La interface *transmitionServiceProvider* representa la interface que *debe* ser implementada por quien desee receptar las mediciones recolectadas desde diferentes fuentes de datos. A continuación, se exponen cada una de los métodos específicos asociados a dicha interface, organizados en forma de tablas a los efectos de explicitar el alcance total de su funcionalidad.

Operación	isReadyToReceive()
Objetivo	Indica si la unidad de procesamiento se encuentra en condiciones de recibir nuevas mediciones.
Parámetros	-

Tipo Retorno	Boolean
Retorno	TRUE si la unidad de procesamiento está lista para recibir mediciones, FALSE en caso contrario.

Tabla 10. Interface transmitionServiceProvider. Método isReadyToReceive.

Operación	**receive(String cincami_mis)**		
Objetivo	Recibe un flujo C-INCAMI / MIS con datos y metadatos de las mediciones para su procesamiento.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	cincamimis	Mediciones estructuradas según el esquema C-INCAMI /MIS.
Tipo Retorno	String		
Retorno	Datos y metadatos según CINCAMI/MIS con un resumen de lo procesado a partir de la ventana de mediciones remitida, junto con información de estado de la unidad de procesamiento.		

Tabla 11. Interface transmitionServicesProvider. Método receive.

Operación	**isLoadSheddingEnabled ()**
Objetivo	Indica si la unidad de procesamiento está empleando mecanismos de descarte selectivo de mediciones cuando se producen colas de servicio.
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE si la unidad de procesamiento emplea descarte selectivo, FALSE en caso contrario.

Tabla 12. Interface transmitionServiceProvider. Operación isLoadSheddingEnabled.

4.3 Procesador C-INCAMI/MIS

La clase *CINCAMIMISProcessor* implementa la funcionalidad de traducir flujos XML bajo el esquema C - INCAMI / MIS a objetos y viceversa. Previo a ingresar en la explicación de la funcionalidad básica de dicha clase, es prioritario presentar las clases y relaciones que se emplean para efectuar el mapeo. Dichos objetos se basan indefectiblemente en el marco C-INCAMI, ya que responden al esquema C-INCAMI / MIS para efectuar la traducción. El proceso de traducción desde esquema a objetos se ha implementado mediante JAXB (Java Architecture for XML Binding) y su idea básica es esquematizada en la Ilustración 13.

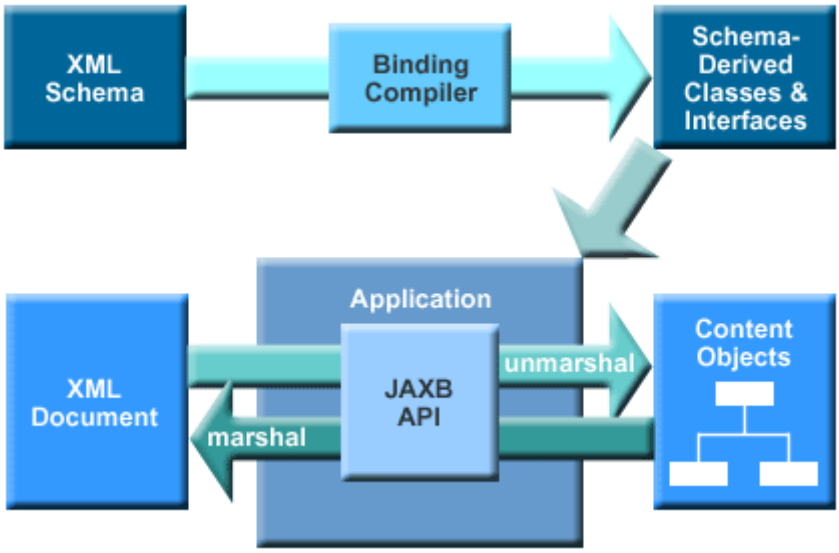

Ilustración 13. Conceptualización del Funcionamiento de JAXB (Ort & Mehta, 2003)

La idea básica de JAXB es generar clases e interfaces derivadas a partir de un esquema XML validado. Luego, a partir de las clases e interfaces derivadas, es posible enlazar a objetos aquellos documentos XML que respeten el esquema indicado, como así también, a partir de documentos XML basados en el esquema, obtener el conjunto de objetos y sus relaciones, efectuando la traducción en ambos sentidos de un modo transparente.

A partir del esquema C-INCAMI / MIS la estructura de clases obtenida se esquematiza en la Ilustración 14. Dentro del esquema asociado a las clases derivadas de C-INCAMI/MIS, se ha optado por no exponer los métodos para facilitar la lectura del diagrama de clases. Puede observarse, comparado con C-INCAMI, que las clases expuestas coinciden y respetan las clases del marco formal C-INCAMI. Debe mencionarse a los efectos de evitar incurrir en inconsistencia, que dichas clases son solo empleadas para el mapeo entre XML y sus correspondientes objetos y relaciones, pero no son bajo ningún punto de vista persistentes.

Hay cuestiones interesantes de destacar a los efectos de exponer cómo los metadatos permiten incrementar la consistencia de las mediciones, a saber:

- La clase CINCAMIMIS representa el encabezado del mensaje y ella contiene toda información interna asociada a dicho mensaje, de este modo cada objeto CINCAMIMIS representa una ventana de procesamiento en términos de data stream.

- Si se analiza cada una de las relaciones de la clase CINCAMIMIS, podrá observarse que en ningún caso presenta obligatoriedad de existencia, por lo que se traduce esto en la aptitud de administrar mensajes de control y mediciones en forma totalmente transparente.

- La clase MeasurementItem, en lugar de tener un valor directamente, posee un conjunto de objetos del tipo Measurement, lo que expone la capacidad de para *una*

medición dada informar un valor o un conjunto de ellos si se tratase de un resultado no determinístico.

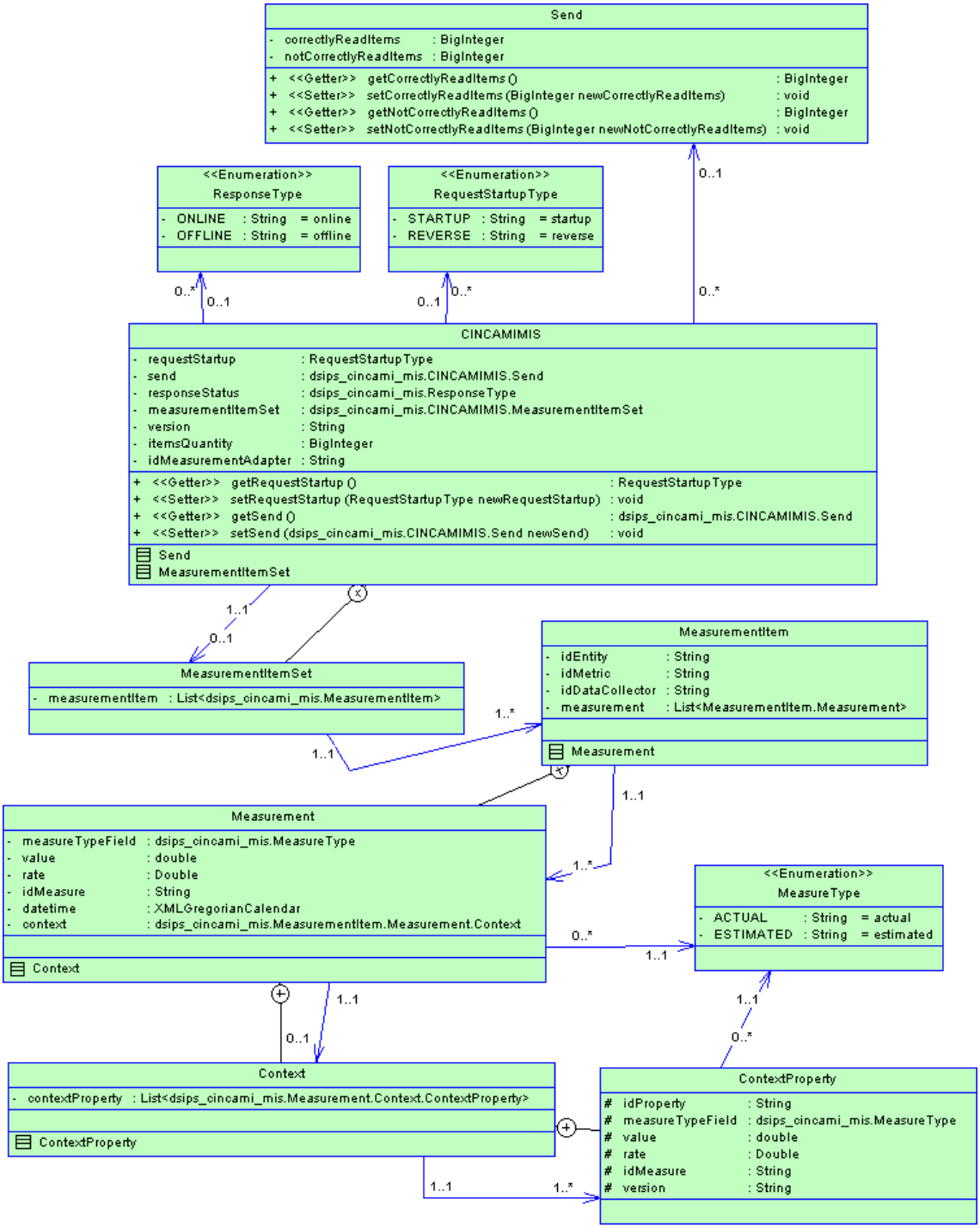

Ilustración 14. Clases Derivadas del Esquema C-INCAMI/MIS

- Cada objeto del tipo Measurement, es capaz de informar o no su situación contextual, con lo que no se obliga a presentar información de contexto si una medición no la posee. Pero en caso de estar disponible, la propia medición es la que informa su situación de contexto, lo que en términos de procesamiento estadístico permite

efectuar un análisis conjunto de las mediciones y su contexto global o bien, permitir analizar las mediciones de una métrica dada en su contexto particular.

Previo a avanzar sobre el procesamiento de C-INCAMI / MIS es menester aclarar aunque parezca evidente, que cada atributo y clase expuesta en la Ilustración 14, responden *estrictamente* al acebo conceptual abordado en la sub-sección 4.2, cuando se presentase la estructura y significado del esquema para intercambio de mediciones.

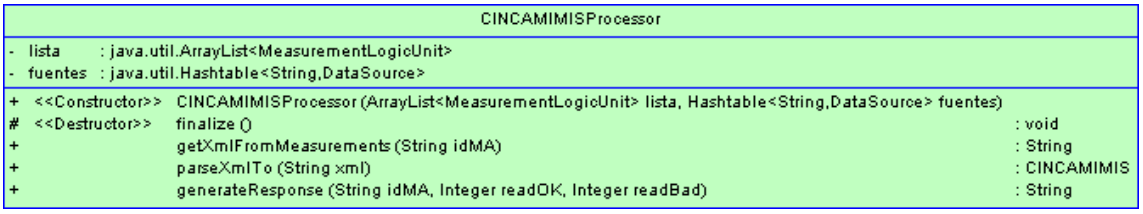

Ilustración 15. Clase CINCAMIMISProcessor

EL procesador C-INCAMI / MIS es instanciado solo cuando se desea convertir las unidades lógicas de medición a XML, en dicho caso debe inicializar la instancia informando la lista de mediciones a transmitir como así también el conjunto de fuentes de datos que han intervenido en su generación. La obtención de los objetos y relaciones a partir de un XML bajo el esquema de intercambio de mediciones está asociados con métodos estáticos, cono lo que no requiere instanciación del procesador.

A continuación, se expone cada una de los métodos específicos asociados a dicha *CINCAMIMISProcessor*, organizados en forma de tablas a los efectos de explicitar el alcance total de su funcionalidad.

Operación	**CINCAMIMISprocessor(java.util.ArrayList<MeasurementLogicUnit> lista, java.util.Hashtable<String,dsips.datastream.DataSource> mySources)**		
Objetivo	Constructor que incorpora la lista de mediciones a transmitir como así también las interfaces asociadas a las fuentes de datos que han intervenido en su obtención.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	ArrayList<MeasurementLogicUnit>	lista	Lista de mediciones a transmitir.
	Hashtable<String,DataSource>	mySources	Fuentes de datos que han intervenido en la obtención de mediciones
Tipo Retorno	-		
Retorno	-		

Tabla 13. Clase CINCAMIMISProcessor. Constructor para transmisión de mediciones.

Operación	**finalize()**
Objetivo	Limpia las mediciones transmitida, dejando liberada la memoria
Parámetros	-

Tipo Retorno	-
Retorno	-

Tabla 14. Clase CINCAMIMISProcessor. Destructor para transmisión de mediciones.

Operación	**getXmlFromMeasurements(String idMA)**		
Objetivo	Obtiene una cadena de texto con formato XML en base al esquema C-INCAMI / MIS generado a partir de la lista de mediciones informadas.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	idMA	Identificador único del adaptador de mediciones que solicita la generación de mensaje para transmisión.
Tipo Retorno	String		
Retorno	Normalmente, retorna una cadena de texto con formato XML en base al esquema C-INCAMI / MIS. Puede ocurrir alguna de las siguiente excepciones: • *CincamiMisException*: Si no existen mediciones que transmitir (Lista nula o con cero mediciones), No se informa el ID del Adaptador de Mediciones, No se informan las fuentes de datos que han intervenido en la recolección de mediciones. • *JAXBException*: Si ocurriese algún inconveniente en el proceso de enlace. • *XMLStreamException*: Si ocurriese alguna excepción durante el procesamiento del flujo que se está generando. • *UnsupportedEncodingException*: Si alguno de los caracteres informados dentro de las mediciones no es convertible a UTF-8.		

Tabla 15. Clase CINCAMIMISProcessor. Método getXmlFromMeasurements.

Operación	***static* parseXmlTo(String xml)**		
Objetivo	Obtiene el conjunto de objetos y sus relaciones, asociados al flujo de mediciones informados.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	xml	Documento xml estructurado de acuerdo al esquema C-INCAMI/MIS que contiene un flujo de mediciones.
Tipo Retorno	CINCAMIMIS		
Retorno	Normalmente, retorna un objeto CINCAMIMIS con sus componentes y relaciones informadas en el flujo de mediciones xml. Puede ocurrir alguna de las siguiente excepciones: • *JAXBException*: Si ocurriese algún inconveniente en el proceso de enlace.		

	• *XMLStreamException*: Si ocurriese alguna excepción durante el procesamiento del flujo que se está leyendo. • *UnsupportedEncodingException*: Si alguno de los caracteres informados dentro de las mediciones no es convertible a UTF-8.

Tabla 16. Clase CINCAMIMISProcessor. Método parseXmlTo.

Operación	***generateResponse(String idma,Integer readOK,Integer readBad)***		
Objetivo	Retorna una respuesta XML en base a C-INCAMI/MIS en formato de cadena de caracteres, informando el resultado de la lectura de un flujo de mediciones.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	Idma	ID del Adaptador de mediciones que ha enviado el flujo de mediciones y al cual se debe responder.
	Integer	readOK	Cantidad de mediciones leídas correctamente.
	Integer	readBad	Cantidad de mediciones con inconvenientes en su lectura.
Tipo Retorno	String		
Retorno	Normalmente, retorna una cadena de texto con formato XML en base al esquema C-INCAMI / MIS informando el resultado del procesamiento de un flujo de mediciones. Puede ocurrir alguna de las siguiente excepciones: • *JAXBException*: Si ocurriese algún inconveniente en el proceso de enlace. • *XMLStreamException*: Si ocurriese alguna excepción durante el procesamiento del flujo que se está generando. • *UnsupportedEncodingException*: Si alguno de los caracteres informados dentro de las mediciones no es convertible a UTF-8.		

Tabla 17. Clase CINCAMIMISProcessor. Método generateResponse.

4.4 Adaptador de Mediciones

La clase *MeasurementAdapter* implementa la funcionalidad del Adaptador de mediciones, cuyo objeto es administrar al gestor de fuentes de datos y a partir de las mediciones recolectadas por éste, es responsable de convertir las mismas al formato adecuado para concretar su transmisión.

De este modo el Adaptador de Mediciones implementa la interface *transmitionServiceConsumer* para poder llevar adelante la tarea de transmisión de las mediciones. Adicionalmente, presenta dependencia de las clases *DataSourceManager* y *CINCAMIMISProcessor.* La primera será responsable de la gestión y configuración de las diferentes fuentes de datos ya que la misma, previo a recolectar mediciones, debe llevar adelante los procesos de configuración mientras que la segunda, es responsable de implementar el mapeo entre los documentos XML estructurados bajo el esquema C-INCAMI / MIS y los objetos derivados del mismo, como puede apreciarse en la Ilustración 16.

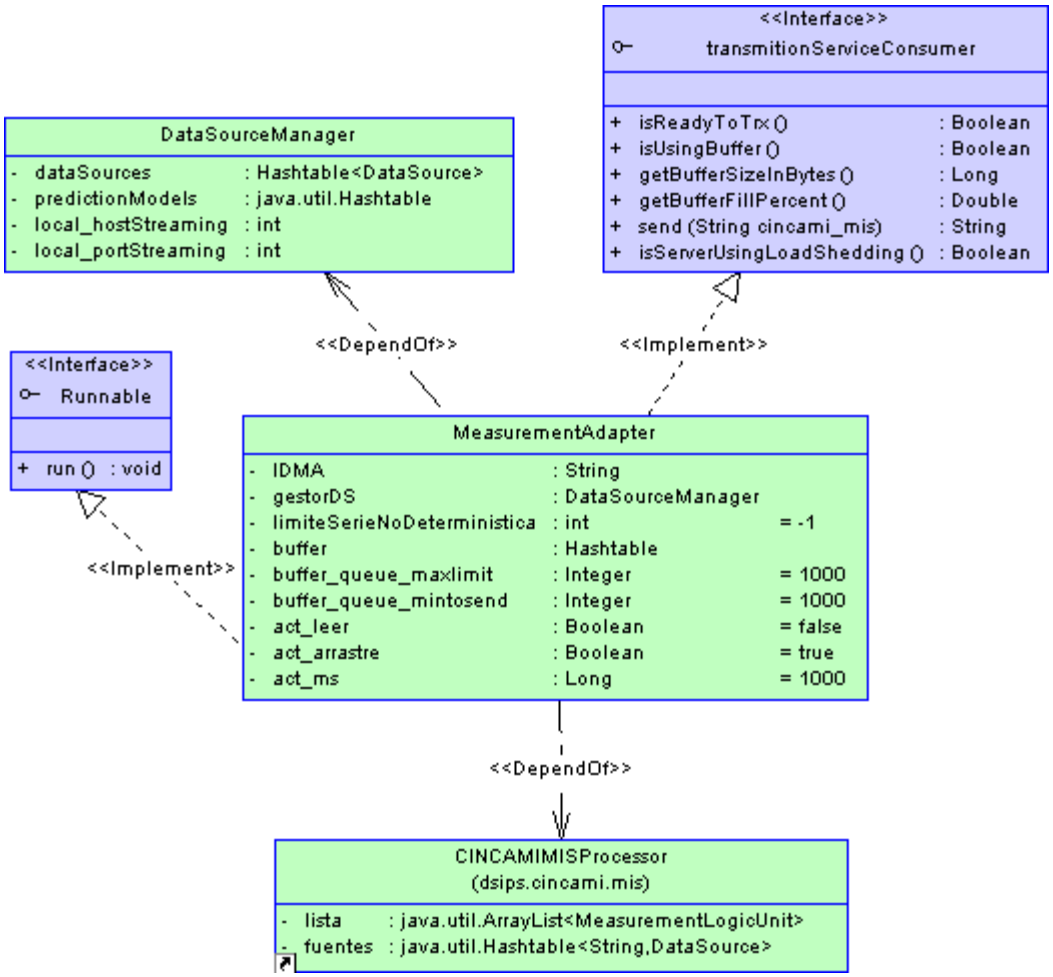

Ilustración 16. Vista Resumida (Sin Métodos) de las Dependencias de MeasurementAdapter

Si bien *DataSourceManager* es el responsable de llevar adelante la recolección de mediciones, ante un mecanismo de recolección por arrastre, no es quien determinará el momento en el cual recolectar las mismas, sino que dicha orden vendrá dada por el propio Adaptador de Mediciones. De este modo, el Adaptador se constituye en una clase activa dado que es posible que varios hilos dentro del mismo puedan coordinar la recolección a partir del DSM.

El adaptador de Mediciones implementa un buffer local mediante una tabla de dispersión, donde la función de dispersión es una función del ID de la fuente de datos. De dicho modo el buffer puede ser visto como un buffer jerárquico con dos niveles. El primer nivel permite el acceso al buffer particular de la fuente datos, mientras que el segundo nivel es el buffer particular a una fuente de datos en sí mismo, el cual se implementa como una lista ordenada con las restricciones de una cola como puede apreciarse en la Ilustración 17. Una vez dispersado el ID de la fuente de datos, se obtiene la raíz de la lista asociada con esa fuente de datos para poder acceder al buffer particular de la misma, en ella solo existen mediciones de dicha fuente.

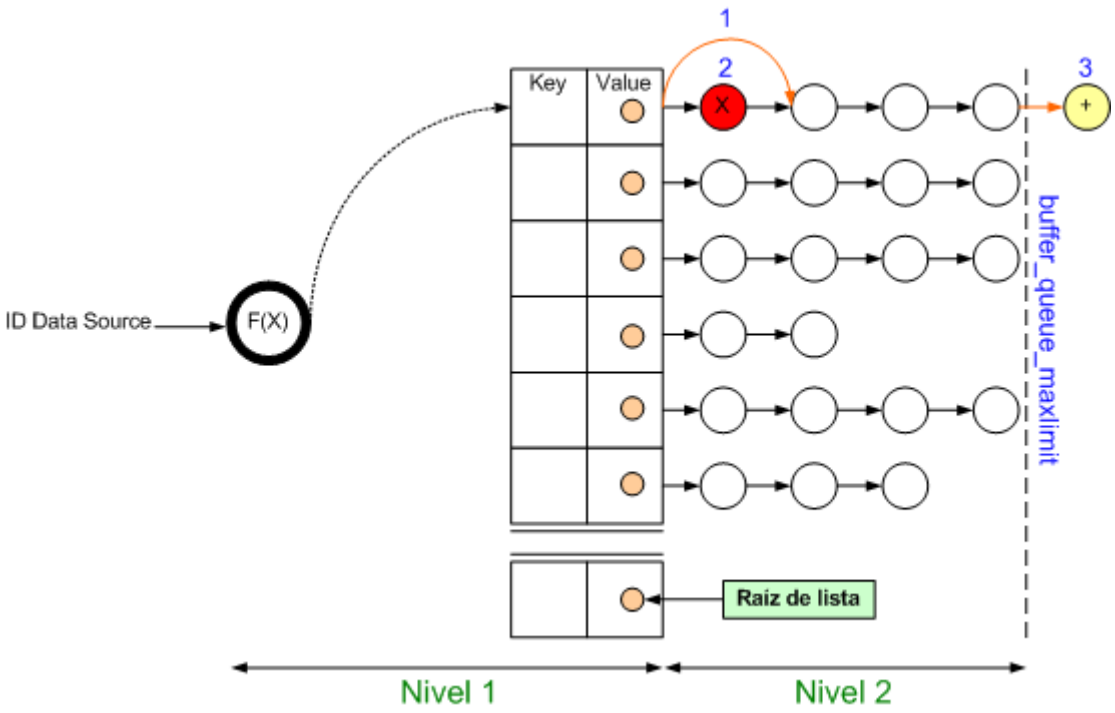

Ilustración 17. Esquema Representativo del Buffer Jerárquico de la clase MeasurementAdapter.

Existen dos propiedades de la clase *MeasurementAdapter* de particular importancia para la gestión del buffer, *buffer_queue_maxlimit* y *buffer_queue_mintosend.* La propiedad *buffer_queue_maxlimit* representa el límite máximo de mediciones que es posible almacenar para el buffer de una fuente de datos mientras que *buffer_queue_mintosend,* determina la cantidad de mediciones mínimas a transmitir por cada fuente de datos. De este modo, estas propiedades administran el volumen de mediciones a transmitir en el flujo que remita el adaptador de mediciones como así también la actualización de las mismas.

Si un buffer particular estuviese en el límite indicado por *buffer_queue_maxlimit,* al tratar de ingresar una nueva medición primero se redirecciona la raíz de la lista al segundo elemento, se descarta el primero y finalmente se incorpora la medición. De este modo ante un potencial exceso de buffer, *siempre* se descartan las mediciones más añejas no transmitidas como también puede apreciarse en lallustración 17. Es posible adaptar dinámicamente el parámetro *buffer_queue_maxlimit;* de incrementarse no existirá inconveniente en cuanto a las mediciones no transmitidas salvo por la verificación previa de suficiencia de memoria, pero de decrementarse, provocará que el adaptador de mediciones efectúe una revisión de cada buffer particular a cada fuente de datos para analizar si excede o no el nuevo límite, de hacerlo, descartará tantas mediciones añejas como sea necesario hasta sincronizar su tamaño.

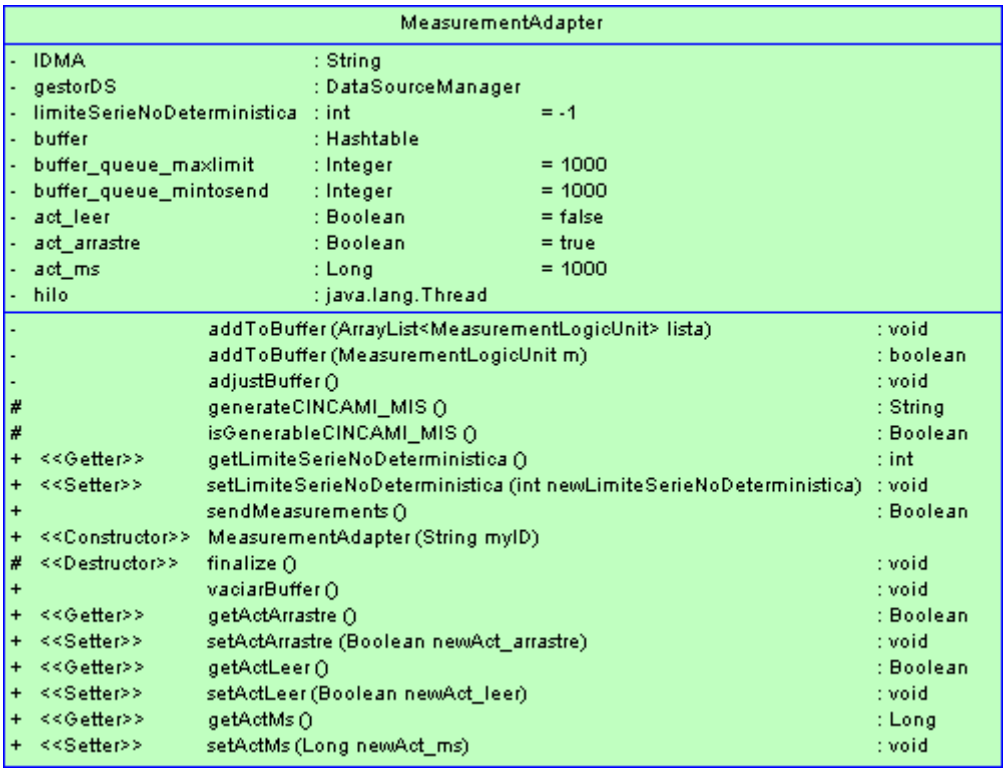

Ilustración 18. Clase MeasurementAdapter

Previo a analizar los métodos asociados a la clase *MeasurementAdapter*, se analizará a continuación, en la Tabla 18, cada una de las propiedades que la integran:

Tipo	Propiedad	Significado
String	IDMA	**Identificador Global del Adaptador de Mediciones**
DataSourceManager	gestorDS	**Gestor de las fuentes de datos a través de la cual es posible configurarlas y gestionarlas**
Integer	limiteSerieNoDeterministica	**Límite de valores para series no deterministas. Si una fuente de datos arroja una serie de valores no deterministas 'n', estos serán ajustados a los primeros '*limiteSerieNoDeterministica*'. Un valor -1 indica que no existe límite establecido por lo que se receptará la serie completa como válida.**
Hashtable	Buffer	**Tabla de dispersión cuya clave de acceso es el ID de la fuente de datos, mediante la cual se accede a una lista de las mediciones más recientes. El tamaño de dicha lista está regido por buffer_queue_maxlimit.**
Integer	buffer_queue_maxlimit	**Límite máximo de la lista de mediciones para cada fuente de datos en el buffer.**
Integer	buffer_queue_mintosend	**Cantidad mínima de mediciones a remitir desde la fuente de datos a la función de reunión. Debe ser menor o igual a buffer_queue_maxlimit. El límite aplica a cada cola, no confundir con cantidad de mediciones globales del buffer.**
Boolean	act_leer	**Indica si el MA leerá continuamente mediante mecanismo. Si act_arrastre está activado**

		entonces solicitará las mediciones a las fuentes de datos y las incorporará en el buffer, caso contrario la responsabilidad de llenar el buffer es de las fuentes. Mientras act_leer sea TRUE, se verificará, indistintamente si es por arrastre o empuje la recolección, si existen mediciones que transmitir. De existir se remiten y se vacía el buffer solo de las mediciones enviadas.
Boolean	act_arrastre	**Variable de control de la interfaz java.lang.Runnable de JAVA, que determina si culminar o no el ciclo de recolección de mediciones por arrastre. Si es FALSE el ciclo continúa, mientras act_leer sea TRUE, esperando mediciones en el buffer que sean informadas por las fuentes para su transmisión.**
Long	act_ms	**Tiempo en milisegundos cada el cual se producirá la lectura de mediciones mediante arrastre.**
java.lang.Thread	**Hilo**	**Hilo de control para lectura de mediciones**

Tabla 18. Propiedades de la clase MeasurementAdapter.

A continuación, se exponen cada una de los métodos específicos asociados a dicha *MeasurementAdapter*, organizados en forma de tablas a los efectos de explicitar el alcance total de su funcionalidad.

Operación	***MeasurementAdapter(String ID)***		
Objetivo	Constructor		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	ID	ID del Adaptador de mediciones.
Tipo Retorno	-		
Retorno	-		

Tabla 19. Clase MeasurementAdapter. Constructor.

Operación	***finalize()***
Objetivo	Detiene los hilos de recolección, vacía el buffer y libera la memoria.
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 20. Clase MeasurementAdapter. Método finalize.

Operación	***getLimiteSerieNoDeterministica()***
Objetivo	Retorna el valor limitante en series no determinísticas

Parámetros	-
Tipo Retorno	Integer
Retorno	Valor que limita la cantidad de valores no deterministas. Puede retornar -1 indicando que no existe límite prestablecido.

Tabla 21. Clase MeasurementAdapter. Método getLimiteSerieNoDeterministica.

Operación	***synchronized setLimiteSerieNoDeterministica(Integer limiteSerieNoDeterministica)***		
Objetivo	Define el tamaño máximo tolerado para series de valores no deterministas.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Integer	limiteSerieNoDeterministica	Límite para la serie no determinística
Tipo Retorno	-		
Retorno	-		

Tabla 22. Clase MeasurementAdapter. Método setLimiteSerieNoDeterministica.

Operación	***getBuffer_queue_maxlimit()***
Objetivo	Retorna el límite máximo de mediciones tolerado para las mediciones provenientes desde la fuente de datos en su buffer.
Parámetros	-
Tipo Retorno	Integer
Retorno	Tamaño máximo de mediciones soportados por el buffer para cada fuente de datos

Tabla 23. Clase MeasurementAdapter. Método getBuffer_queue_maxlimit.

Operación	***synchronized setBuffer_queue_maxlimit(Integer buffer_queue_maxlimit)***		
Objetivo	Define el tamaño máximo de mediciones para cada fuente de datos. Debe ser un valor mayor o igual a 1. Este método provoca que se ajuste el tamaño del buffer automáticamente.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Integer	buffer_queue_maxlimit	Tamaño máximo del buffer para cada fuente de datos
Tipo Retorno	-		
Retorno	-		

Tabla 24. Clase MeasurementAdapter. MétodosetBuffer_queue_maxlimit.

Operación	*getBuffer_queue_mintosend()*
Objetivo	Define el Mínimo de mediciones que deben reunirse para transmitir las mismas a la función de reunión.
Parámetros	-
Tipo Retorno	Integer
Retorno	Cantidad mínima de mediciones que deben transmitirse a la función de reunión

Tabla 25. Clase MeasurementAdapter. Método getBuffer_queue_mintosend.

Operación	*synchronized void setBuffer_queue_mintosend(Integer buffer_queue_mintosend)*		
Objetivo	Establece la cantidad mínima de mediciones que deben transmitirse a la función de reunión. Debe ser una cantidad entre 1 y el tamaño máximo de mediciones por fuente de datos. Si el nuevo límite impuesto ocasiona que alguna cola satisface la restricción, las mismas son automáticamente transmitidas		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Integer	buffer_queue_mintosend	Cantidad mínima de mediciones a transmitir
Tipo Retorno	-		
Retorno	-		

Tabla 26. Clase MeasurementAdpater. Método setBuffer_queue_mintosend.

Operación	*run()*
Objetivo	Implementación del método de la interface Runnable. Permite la lectura periódica y activa multihilo.
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 27. Clase MeasurementAdapter. Método run implementado a partir de interface Runnable.

Operación	*synchronized addToBuffer(MeasurementLogicUnit m)*		
Objetivo	Agrega una medición con sus propiedades contextuales al buffer de la fuente de datos siempre que tenga su PK definida y memoria disponible. Controla el tamaño máximo de mediciones del buffer y en caso de que con la medición actual exceda el mismo, descarta las mediciones más antiguas para poder incorporar las nuevas al final.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	MeasurementLogicUnit	m	Unidad Lógica de Medición a incorporar en el buffer de la fuente de datos

Tipo Retorno	Boolean
Retorno	TRUE = Si ha podido agregarse la medición FALSE = en caso contrario

Tabla 28. Clase MeasurementAdapter. Método addToBuffer(MeasurementLogcUnit m)

Operación	***synchronized addToBuffer(ArrayList<MeasurementLogicUnit> lista)***		
Objetivo	Agrega un conjunto de mediciones al buffer efectuando los controles indicados en *addToBuffer(MeasurementLogicUnit m)*.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	ArrayList<MeasurementLogicUnit>	lista	Lista de unidades lógicas de medición a incorporar en el
Tipo Retorno	Boolean		
Retorno	TRUE = Si ha podido agregarse la medición FALSE = en caso contrario		

Tabla 29. Clase MeasurementAdapter. Método addToBuffer(ArrayList<MeasurementLogicUnit> lista)

Operación	***synchronized adjustBuffer()***
Objetivo	Ajusta cada una de las colas asociadas a una fuente de datos al máximo permitido
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 30. Clase MeasurementAdapter. Método adjustBuffer.

Operación	***synchronized isGenerableCINCAMI_MIS()***
Objetivo	Indica si con las mediciones que posee y las fuentes registradas posee información suficiente para generar el flujo de mediciones a transmitir
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE = Indica que tiene información suficiente para generar el flujo de mediciones, FALSE = en caso contrario

Tabla 31. Clase MeasurementAdapter. Método isGenerableCINCAMI_MIS().

Operación	***synchronized generateCINCAMI_MIS()***
Objetivo	A partir de las mediciones del buffer, genera un mensaje XML CINCAMIMIS, si no es posible retorna null.
Parámetros	-

Tipo Retorno	String
Retorno	Mensaje XML según esquema CINCAMI/MIS si es posible generarlo, caso contrario retorna null.

Tabla 32. Clase MeasurementAdapter. Método generateCINCAMI_MIS.

Operación	***synchronized vaciarBuffer()***
Objetivo	Vacía el buffer de mediciones completamente.
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 33. Clase MeasurementAdapter. Método vaciarBuffer.

Operación	***synchronized sendMeasurements()***
Objetivo	A partir de las mediciones del buffer, obtiene el XML CINCAMI/MIS y las remite mediante el método send, retornando TRUE si ha podido transmitir con éxito, FALSE en caso contrario. Transmitidas las mismas, se vacía el buffer y se informa por consola el resultado de la transmisión.
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE = si la transmisión ha podido efectuarse sin inconvenientes FALSE = en caso contrario

Tabla 34. Clase MeasurementAdapter. Método sendMeasurements.

Operación	***synchronized getAct_arrastre()***
Objetivo	Verifica si culmina o no el ciclo de recolección de mediciones en función de si emplea o no el mecanismo por arrastre.
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE = Se emplea lectura por arrastre desde el MA FALSE = No se utiliza el mecanismo de arrastre

Tabla 35. Clase MeasurementAdapter. Método getAct_arrastre.

Operación	*synchronized setAct_arrastre(Boolean act_arrastre)*		
Objetivo	Determina si leer o no mediciones desde las fuentes de datos. Para desactivar la lectura periódica se indica act_arrastre=FALSE mientras que para iniciarla se indica act_arrastre=TRUE. En caso de desactivar la lectura periódica, el MA hará caso quedará a la espera de las mediciones que informen las fuentes mediante los métodos addToBuffer.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Boolean	act_arrastre	Define si emplea o no el mecanismo por arrastre.
Tipo Retorno	-		
Retorno	-		

Tabla 36. Clase MeasurementAdapter. Método getAct_arrastre.

Operación	*synchronized getAct_leer()*
Objetivo	Indica que procederá a verificar si existen mediciones que transmitir una vez satisfecho la restricción temporal del período de lectura. Un valor FALSE indica que NO verificará si existen mediciones que transmitir y culminará el hilo de control.
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE = Indica que procederá a verificar si existen mediciones que transmitir FALSE = Culmina el hilo de control de mediciones

Tabla 37. Clase MeasurmentAdapter. Método getAct_leer.

Operación	*synchronized setAct_leer(Boolean act_arrastre)*		
Objetivo	Determina si *verificará o no* las mediciones en el buffer a los efector de su transmitión cada un período dado determinado por la propiedad act_ms		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Boolean	act_arrastre	Define si emplea o no el mecanismo por arrastre.
Tipo Retorno	-		
Retorno	-		

Tabla 38. Clase MeasurementAdapter. Método setAct_leer.

Operación	*synchronized getAct_ms()*
Objetivo	Consulta la frecuencia en milisegundos cada cuanto verifica el estado del buffer para transmisión y las fuentes si el mecanismo de arrastre está activado.

Parámetros	-
Tipo Retorno	Long
Retorno	Valor >= 1 expresado en milisegundos que determina el período de lectura

Tabla 39. Clase MeasurementAdapter. Método getAct_ms.

Operación	***synchronized setAct_ms(Long act_ms)***		
Objetivo	Define la frecuencia en milisegundos cada cuanto verificar el estado del buffer para transmisión y las fuentes si el mecanismo de arrastre está activdado.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Long	act_ms	Valor >= 1 expresado en milisegundos que determina el período de lectura
Tipo Retorno	-		
Retorno	-		

Tabla 40. Clase MeasurementAdapter. Método setAct_ms

4.5 Gestión de Mediciones Mediante Buffer Multinivel

Hasta el momento se ha analizado el cómo poder recolectar las mediciones, almacenarlas localmente en un buffer para lograr una cantidad mínima, luego poder transformarlas a un flujo de mediciones XML basado en el esquema C-INCAMI/MIS y transmitirlas. Se ha mencionado, incluso en la sub-sección 3.1 cuando se presentase el modelo conceptual del enfoque integrado de procesamiento de flujos de datos, que tales mediciones eran receptadas por la función de reunión e incluso se indicó que su objetivo consistía en receptar datos y metadatos desde el MA estructurados según el esquema C-INCAMI/MIS, para luego agruparlos si el objeto de medición fuese equivalente.

De este modo, el rol de la función de reunión será desempeñado por aquella clase que implemente la interface *transmitionServiceProvider*. Previo a efectuar el desglose de las propiedades y métodos de la clase que implementa tal interface, es necesario analizar cómo se gestiona el buffer de mediciones dentro de la función de reunión (ver Ilustración 19), dado que es el punto de encuentro entre lo que resulta del procesamiento paralelo de recolección y adaptación, para luego dejar el buffer organizado de acuerdo a las necesidades del proceso de corrección y análisis.

El buffer empleado en la función de reunión posee tres niveles anidados mediante tablas de dispersión y es implementado mediante la clase *bufferTG*. El primer nivel dispersa el ID de grupo de seguimiento para localizar un objeto del tipo *bufferMetric*, cuya función es administrar las mediciones vinculadas a métricas para ese grupo de seguimiento. El segundo nivel dispersa el identificador de la métrica para localizar la raíz de la lista que contiene las mediciones asociadas. Finalmente, el tercer nivel es el conjunto de mediciones específicas a la métrica y grupo de seguimiento dispersadas. Al igual que el buffer del MA, el buffer de la función de reunión posee parámetros que regulan su crecimiento, entre ellos *queue_max_limit* que trabaja en forma análoga a *buffer_queue_maxlimit* de MA.

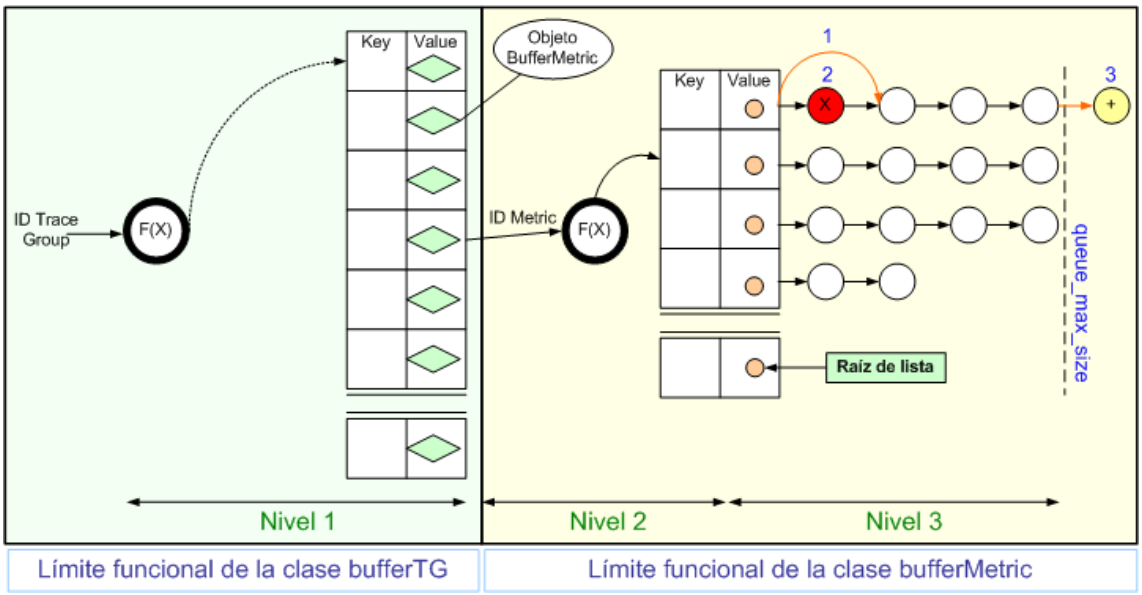

Ilustración 19. Esquema conceptual del buffer multinivel de la función de reunión

La clase *bufferTG (*ver Ilustración 20*)* será la responsable de ordenar las mediciones de acuerdo a los grupos de seguimiento en los que participe la fuente de datos que los originó. De este modo, si la fuente de datos participa en 'n' grupos de seguimiento, se dispersará tantas veces como grupos participe, para poder ubicar una copia de la medición en cada uno de ellos. Por otro lado, *bufferTG* contiene una colección de clases *bufferMetric* que administra las mediciones por grupo de seguimiento y las organiza por métricas. Así, cada grupo de seguimiento tendrá un único objeto del tipo *bufferMetric* que organizará la totalidad de las mediciones de acuerdo a cada una de las métricas involucradas en el mismo.

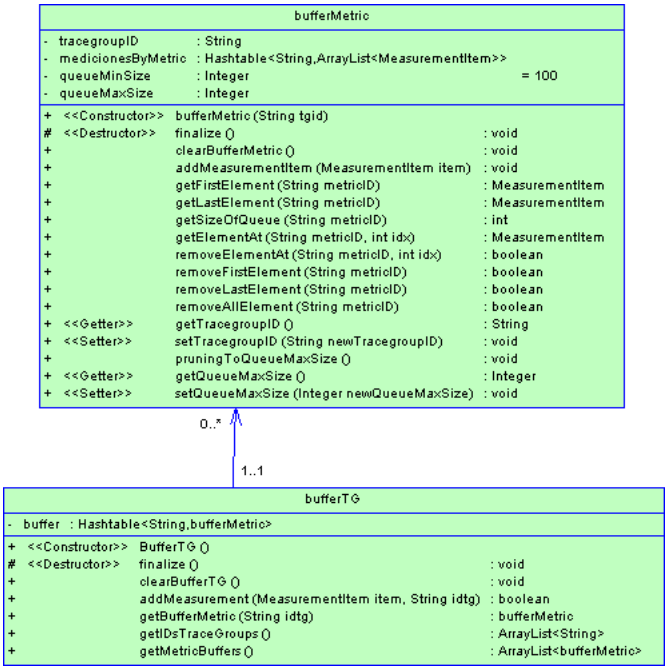

Ilustración 20. Clases bufferTG y bufferMetric. Responsables de la gestión del buffer multinivel.

Previo a analizar los métodos asociados a la clase *bufferMetric*, se analizará a continuación cada una de las propiedades que la integran:

Tipo	Propiedad	Significado
String	tracegroupID	**Identificador del grupo de seguimiento del cual administra las mediciones**
Hashtable<String, ArrayList<MeasurementItem> >	medicionesByMetric	**Tabla de dispersión con acceso mediante ID de métrica que contiene una lista con las mediciones organizadas por métrica.**
static final Integer	queueMinSize	**Longitud mínima reservada para las mediciones de una métrica dada en un grupo.**
Integer	**queueMaxSize**	**Longitud máxima permitida para las mediciones de una métrica dada en un grupo**

Tabla 41. Propiedades de la clase bufferMetric.

A continuación, se exponen cada una de los métodos específicos asociados a la clase *bufferMetric*, organizados en forma de tablas a los efectos de explicitar el alcance total de su funcionalidad.

Operación	***bufferMetric(String tgid)***		
Objetivo	Constructor		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	tgid	Identificador del grupo de seguimiento del cual administrará las mediciones
Tipo Retorno	-		
Retorno	-		

Tabla 42. Clase bufferMetric. Constructor.

Operación	***finalize()***
Objetivo	Destructor. Limpia completamente el buffer, liberando la memoria reservada.
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 43. Clase bufferMetric. Método finalize.

Operación	***synchronized clearBufferMetric()***
Objetivo	Limpia el buffer de mediciones completamente, descartando todas las colas asociadas a las distintas métricas e incluso la posición de dispersión de los ID de métricas.
Parámetros	-

Tipo Retorno	-
Retorno	-

Tabla 44. Clase bufferMetric. Método clearBufferMetric.

Operación	***synchronized addMeasurementItem(MeasurementItem item)***		
Objetivo	Agrega una nueva medición a en la lista correspondiente en función del metricID. Controla el tamaño máximo de cola y en caso de excederlo, descarta las mediciones viejas y las sustituye por las más recientes.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	MeasurementItem	item	Medición y factores contextuales a agregar
Tipo Retorno	-		
Retorno	-		

Tabla 45. Clase bufferMetric. Método addMeasurementItem.

Operación	***getFirstElement(String metricID)***		
Objetivo	Obtiene la primera medición asociada a la métrica dentro del grupo de seguimiento		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	metricID	Identificador de la métrica de la cual se desea obtener la primera medición asociada al grupo
Tipo Retorno	MeasurementItem		
Retorno	Primera medición asociada a la métrica dentro del grupo de seguimiento en caso de existir, null en caso contrario.		

Tabla 46. Clase bufferMetric. Método getFirstElement.

Operación	***getLastElement(String metricID)***		
Objetivo	Obtiene la última medición asociada a la métrica dentro del grupo de seguimiento		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	metricID	Identificador de la métrica de la cual se desea obtener la última medición asociada al grupo
Tipo Retorno	MeasurementItem		
Retorno	Última medición asociada a la métrica dentro del grupo de seguimiento en caso de existir, null en caso contrario.		

Tabla 47. Clase bufferMetric. Método getLastElement.

Operación	*getSizeOfQueue(String metricID)*		
Objetivo	Obtiene el tamaño de la lista asociada a la métrica dentro del grupo de seguimiento		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	metricID	Identificador de la métrica de la cual se desea obtener el tamaño de su lista de mediciones asociada al grupo
Tipo Retorno	Integer		
Retorno	Tamaño de la cola. El valor mínimo a retornar es 0 (cero) si se encontrase la lista sin elementos o vacía.		

Tabla 48. Clase bufferMetric. Método getSizeOfQueue.

Operación	*getElementAt(String metricID,int idx)*		
Objetivo	Obtiene la medición en la posición indicada asociada a la métrica dentro del grupo de seguimiento		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	metricID	Identificador de la métrica de la cual se desea obtener la medición en la posición indicada
	Integer	idx	Posición de la lista de mediciones de la métrica que se desea obtener. La contabilización de las posiciones comienza desde 0.
Tipo Retorno	MeasurementItem		
Retorno	Medición en la posición indicada de la lista vinculada a la métrica de existir, null en caso contrario.		

Tabla 49. Clase bufferMetric. Método getElementAt.

Operación	*synchronized removeElementAt(String metricID,int idx)*		
Objetivo	Remueve la medición asociada a la métrica en la posición indicada dentro del grupo de seguimiento y desplaza a la izquierda todas las mediciones que estuviesen a su derecha.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	metricID	Identificador de la métrica de la cual se desea remover la medición en la posición indicada
	Integer	idx	Posición de la lista de mediciones de la métrica que se desea remover. La contabilización de las posiciones comienza desde 0.
Tipo Retorno	Boolean		
Retorno	TRUE = Si ha podido remover la medición FALSE = Si la medición no ha podido ser removida		

Tabla 50. Clase bufferMetric. Método removeElementAt.

Operación	*synchronized removeFirstElement(String metricID)*		
Objetivo	Remueve la primera medición asociada a la métrica dentro del grupo de seguimiento y desplaza a la izquierda todas las mediciones que estuviesen a su derecha.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	metricID	Identificador de la métrica de la cual se desea remover la medición
Tipo Retorno	Boolean		
Retorno	TRUE = Si ha podido remover la medición FALSE = Si la medición no ha podido ser removida		

Tabla 51. Clase bufferMetric. Método removeFirstElement.

Operación	*synchronized removeLastElement(String metricID)*		
Objetivo	Remueve la última medición asociada a la métrica dentro del grupo de seguimiento y desplaza a la izquierda todas las mediciones que estuviesen a su derecha.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	metricID	Identificador de la métrica de la cual se desea remover la medición
Tipo Retorno	Boolean		
Retorno	TRUE = Si ha podido remover la medición FALSE = Si la medición no ha podido ser removida		

Tabla 52. Clase bufferMetric. Método removeLastElement.

Operación	*synchronized removeAllElement(String metricID)*		
Objetivo	Remueve todas las mediciones asociadas a la métrica dentro del grupo de seguimiento.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	metricID	Identificador de la métrica de la cual se desea remover la lista de mediciones
Tipo Retorno	Boolean		
Retorno	TRUE = Si ha podido remover las mediciones FALSE = Si las mediciones no han podido ser removidas		

Tabla 53. Clase bufferMetric. Método removeAllElement.

Operación	*getTracegroupID()*
Objetivo	Obtiene el grupo de seguimiento con el que se asocian el grupo de mediciones vinculadas.
Parámetros	-
Tipo Retorno	String
Retorno	ID del grupo de seguimiento con el que se asocian las mediciones

Tabla 54. Clase bufferMetric. Método getTracegroupID.

Operación	*setTracegroupID(String tracegroupID)*		
Objetivo	Asocia el ID del grupo de seguimiento con las mediciones administradas por el buffer.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	tracegroupID	Identificador del grupo de seguimiento con el que se asociarán las mediciones del buffer
Tipo Retorno	-		
Retorno	-		

Tabla 55. Clase bufferMetric. Método setTracegroupID.

Operación	*synchronized pruningToQueueMaxSize()*
Objetivo	Ajusta el tamaño de todas las listas de mediciones para todas las métricas del buffer, podándolas para que no superen el tamaño máximo definido en *queueMaxSize*, en la medida que el tamaño de la lista supere *queueMinSize*.
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 56. Clase bufferMetric. Método pruningToQueueMaxSize.

Operación	*Integer getQueueMaxSize()*
Objetivo	Obtiene el tamaño máximo permitido para la lista de mediciones
Parámetros	-
Tipo Retorno	Integer
Retorno	Tamaño máximo permitido para cada lista de mediciones asociada a una métrica dentro del grupo de seguimiento. El valor retornado será como mínimo *queueMinSize.*

Tabla 57. Clase bufferMetric. Método getQueueMaxSize.

Operación	*setQueueMaxSize(Integer queueMaxSize)*		
Objetivo	Define dinámicamente un nuevo tamaño máximo para la lista de mediciones. El tamaño máximo DEBE ser igual o superior a queueMinSize. Al cambiar el tamaño máximo, se ajustará el buffer automáticamente al nuevo tamaño en los casos que corresponda.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Integer	queueMaxSize	Nuevo tamaño que se intenta definir para la lista de mediciones
Tipo Retorno	-		
Retorno	-		

Tabla 58. Clase bufferMetric. Método setQueueMaxSize.

Previo a analizar los métodos asociados a la clase *bufferTG*, se analizará a continuación las propiedades que la integran:

Tipo	Propiedad	Significado
Hashtable<String,bufferMetric>	buffer	Buffer organizado por grupos de seguimiento que contienen en su interior un administrador de mediciones organizado por métricas.

Tabla 59. Propiedades de la clase bufferTG.

A continuación, se exponen cada una de los métodos específicos asociados a la clase *bufferTG*, organizados en forma de tablas a los efectos de explicitar el alcance total de su funcionalidad.

Operación	*bufferTG()*
Objetivo	Constructor
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 60. Clase bufferTG. Constructor.

Operación	*finalize()*
Objetivo	Destructor. Limpia completamente el buffer, liberando la memoria reservada.
Parámetros	-

Tipo Retorno	-
Retorno	-

Tabla 61. Clase bufferTG. Método finalize.

Operación	*clearBufferTG()*
Objetivo	Limpia el buffer de grupos de seguimiento completamente, descartando todas las colas asociadas a los distintos grupos e incluso la posición de dispersión de los ID.
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 62. Clase bufferTG. Método clearBufferTG.

Operación	*synchronized addMeassurement(MeasurementItem item,String idtg)*		
Objetivo	Agrega la medición al gestor de mediciones por métricas correspondiente.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	MeasurementItem	item	Medición a agregar
	String	idtg	Indica el grupo de seguimiento en el que la medición debe incorporarse
Tipo Retorno	Boolean		
Retorno	TRUE = si la medición ha podido ser incorporada FALSE = Si no se ha incorporado la medición		

Tabla 63. Clase bufferTG. Método addMeasurement.

Operación	*getBufferMetric(String idtg)*		
Objetivo	Retorna el gestor de mediciones asociado al grupo de seguimiento		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	idtg	Indica el grupo de seguimiento del cual desea recuperar el gestor de mediciones
Tipo Retorno	bufferMetric		
Retorno	Retorna el gestor de mediciones si está definido, null en caso contrario.		

Tabla 64. Clase bufferTG. Método getBufferMetric.

Operación	*getIDsTraceGroups()*
Objetivo	Retorna los ID de los grupos de seguimiento que tienen asociado en el buffer

	gestores de mediciones.
Parámetros	-
Tipo Retorno	ArrayList<String>
Retorno	Lista con los ID de los grupo de mediciones presentes en el buffer con gestores de mediciones.

Tabla 65. Clase bufferTG. Método getIDsTraceGroups.

Operación	***getMetricBuffers()***
Objetivo	Retorna todos los Gestores de Mediciones disponibles en el buffer. Recuerde que el gestor de mediciones conoce el ID del grupo de seguimiento con el que se asocia
Parámetros	-
Tipo Retorno	ArrayList<bufferMetric>
Retorno	Arreglo con todos gestores de mediciones presentes en el buffer.

Tabla 66. Clase bufferTG. Método getMetricBuffers.

4.6 Load Shedding

Los mecanismos de Load Shedding permiten abordar la problemática de cola de servicios ante eventuales desbordes, generados a partir de la potencial volatilidad de la tasa de arribo de datos desde los diferentes data streams que nutren de mediciones al esquema de procesamiento. En este sentido, su objeto fundamental es actuar proactivamente sobre la cola de servicios en situaciones donde la tasa de procesamiento de las mediciones es inferior a la tasa de arribo de datos, incorporando mecanismos de descarte selectivo con la finalidad de mantener el uso de recursos dentro de un marco limitado o finito, minimizando tanto como sea posible la pérdida de precisión (Chakravarthy & Jiang, 2009).

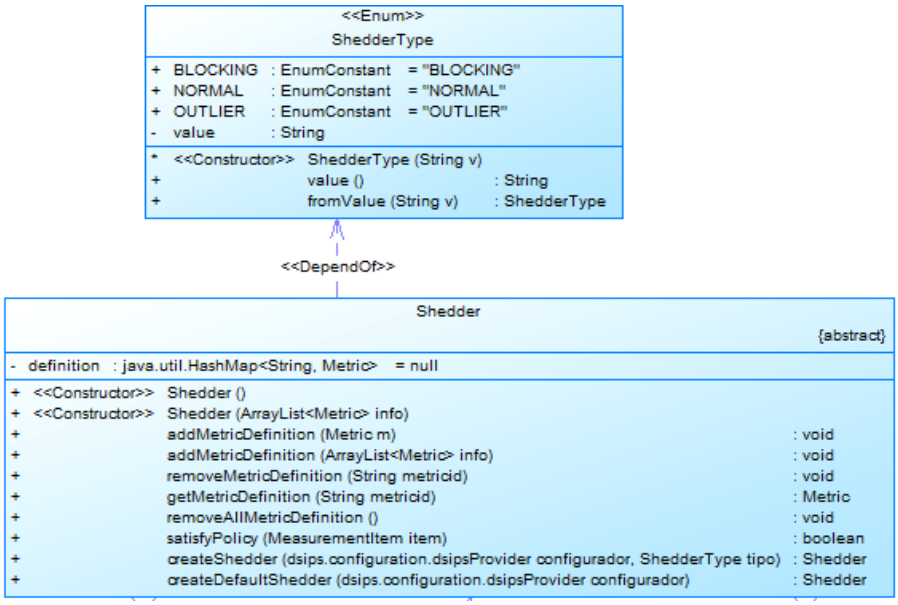

Ilustración 21. Clase Shedder

De este modo, uno de los principales temas a resolver dentro de los operadores de descarte adicionalmente a cómo eliminar el exceso de datos minimizando la pérdida de precisión, es dónde ubicarlos a los efectos de que el overhead que incorporan en el procesamiento sea mínimo en términos globales. En este sentido, dado que el buffer de gestión multinivel del esquema de procesamiento integrado, permite establecer que las mediciones en cada métrica dentro de un grupo de seguimiento dado se comporte como una cola donde, partiendo de un límite finito y configurable de la misma, cualquier exceso repercutirá en el descarte de 'n' mediciones antiguas para permitir la incorporación de las 'n' mediciones arribadas recientemente. De este modo, el buffer mantiene siempre mediciones recientes y evita el desborde, pero adicionalmente ante una situación de arribo masivo, se ha incorporado como mecanismo adicional de control un operador de load shedding especificado mediante la clase abstracta Shedder (ver Ilustración 21).

La clase abstracta Shedder debe ser aprovisionada con la especificación formal de la métrica dentro del proyecto de medición, esto ocurre durante la inicialización de la instancia y accede a la BD C-INCAMI mediante una instancia de la clase *dsipsProvider,* quien implementa la funcionalidad asociada a la configuración dentro del esquema de procesamiento. El hecho de contar con la definición formal de las métricas le permite interpretar las mediciones provenientes desde las diferentes fuentes de datos y determinar en principio si los valores son racionales a su definición, esto permite entre otras cosas detectar anomalías o ruido propio de la transmisión.

La clase Shedder plantea un método abstracto denominado *satisfyPolicy* el cual recibe como parámetro una instancia de *MeasurementItem*, el mismo tiene por objeto que cualquiera de las clases derivadas en base a los metadatos de la métrica en el proyecto de medición y a la unidad lógica de medición recibida, determine cuál será la política de descarte que empleará, si es que utiliza, dentro del esquema de procesamiento de flujos dotando de extensibilidad al mismo.

El operador Shedder es invocado desde el buffer de grupos de seguimiento en forma previa a la incorporación en los buffer a nivel de métrica, esto permite que el operador de descarte actué en forma anterior al buffer liberando de sobrecargas de eliminación e incorporación de mediciones si es que las mismas no se ajustan a la política vigente en el momento del arribo. La política de descarte puede ser acatualizada dinámicamente en tiempo de ejecución, informando a la instancia de bufferTG el objeto derivado de la clase Shedder con la nueva política mediante el método setMyShedder de bufferTG.

Actualmente el esquema de procesamiento soporta tres políticas de descarte selectivo: BLOCKING, NORMAL y OUTLIER. La política BLOCKING bloquea cualquier arribo de mediciones garantizando que el buffer permanecerá intacto hasta tanto se determine un cambio de política. La política NORMAL deja pasar solo las mediciones que no son considerados como outliers, es decir aquellos que se rigen en forma similar al comportamiento general de la serie pre-existente. Finalmente, la política OUTLIER permite solo el paso de mediciones cuyo comportamiento no se condicen con el comportamiento general de la serie.

Un aspecto fundamental a destacar en este sentido es que la activación y desactivación del operador de descarte es dinámica, es decir que si la instancia de bufferTG tiene una referencia nula en su propiedad myShedder, las mediciones serán distribuidas dentro buffer multinivel con el control de cola limitado al valor predefinido en la misma pero sin presencia de política de descarte. Por otro lado, si la propiedad myShedder tiene referencia a alguna implementación derivada de la clase Shedder con su política asociada, ésta se aplica inmediatamente la unidad lógica de medición arriba al buffer de grupos de seguimiento y determina si se descarta o procede a los buffer de mediciones asociados a las métricas. Vale destacar que exista política de descarte o no, el control de colas de mediciones sobre las métricas en cada grupo de seguimiento siempre es efectuado, con el objeto de garantizar la longitud de mediciones pre-definidas.

4.7 Recepción de Mediciones

La recepción de las mediciones y su organización es la funcionalidad principal de la función de reunión y es llevada cabo a través de la clase que implemente la interface *transmitionServiceProvider*.

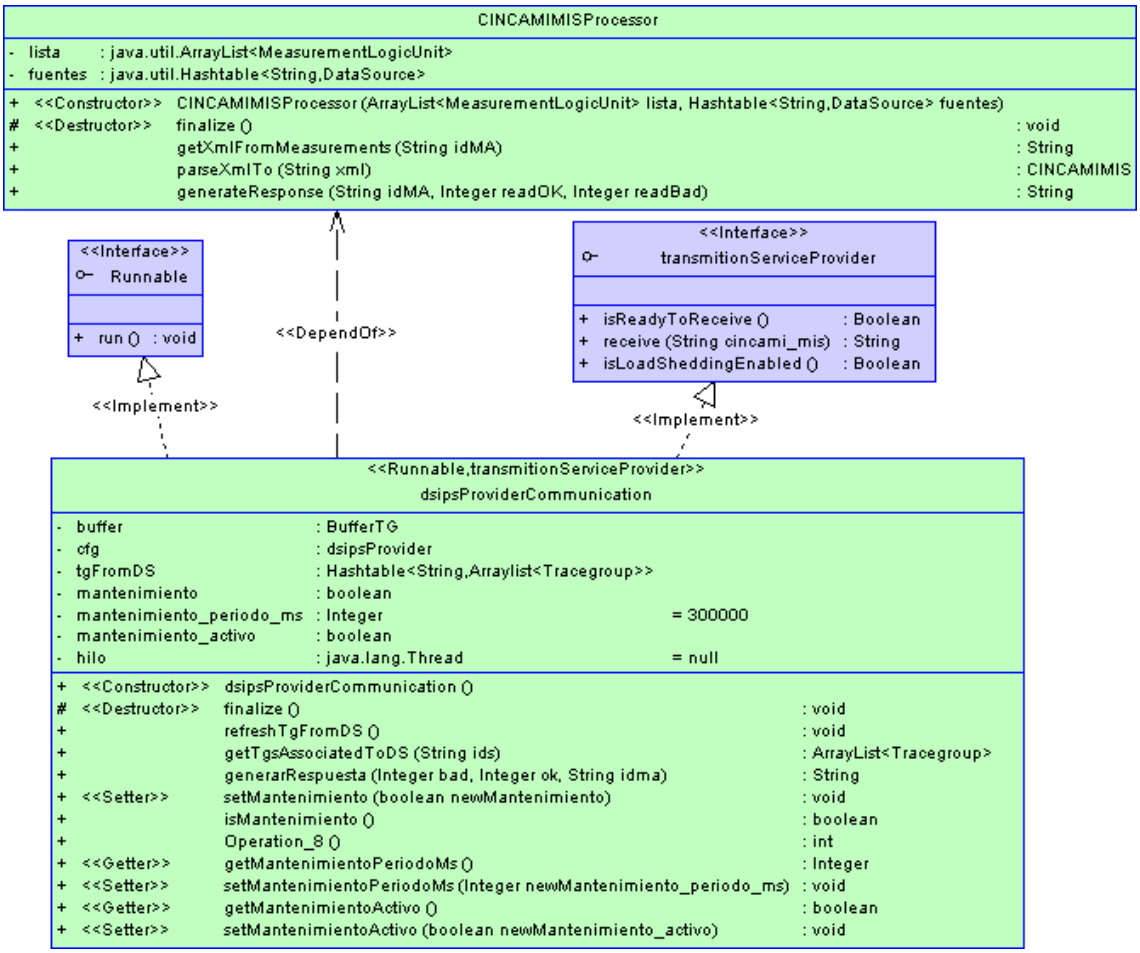

Ilustración 22. Clase dsipsProviderCommunication, dependencias e Interfaces Asociadas.

La clase *dsipsProviderCommunication* implementa la interface *transmitionServiceProvider* y *java.lang.Runnable,* de este modo la clase se constituye en la función de reunión como puede observarse en la Ilustración 22.

Previo a analizar los métodos asociados a la clase *dsipsProviderCommunication*, se analizará a continuación cada una de las propiedades que la integran:

Tipo	Propiedad	Significado
bufferTG	buffer	**Gestor del buffer de mediciones**
dsipsProvider	Cfg	**Gestor de configuración. Brinda acceso a la BD C-INCAMI para la recuperación de datos y metadatos.**
Hashtable<String, ArrayList<Tracegroup>>	tgFromDS	**Almacena en memoria los grupos de seguimiento asociados con cada fuente de datos**
Boolean	mantenimiento	**Bandera que indica si el proveedor de servicios de comunicación está en mantenimiento o no.**
Integer	mantenimiento_periodo_ms	**Ciclo en milisegundos cada cuanto efectuara mantenimiento.**
Boolean	mantenimiento_activo	**Bandera de control que indica si continua activo el Thread de mantenimiento o lo culmina.**
java.lang.Thread	**hilo**	**Hilo al que se le pasará el actual objeto Runnable para que controle su ejecución mediante el método *run* y efectúe tareas de mantenimiento sobre el proveedor de servicios. El hilo es iniciado con el constructor y cada vez que se indica la finalización del mantenimiento.**

Tabla 67. Propiedades de la Clase dsipsProviderCommunication.

A continuación, se exponen cada una de los métodos específicos asociados a la clase *dsipsProviderCommunication*, organizados en forma de tablas a los efectos de explicitar el alcance total de su funcionalidad.

Operación	***dsipsProviderCommunication()***
Objetivo	Constructor. Inicializa el buffer, el gestor de configuración y el hilo de control.
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 68. Clase dsipsProviderCommunication. Constructor.

Operación	***finalize()***
Objetivo	Destructor. Detiene el hilo de control, limpia el buffer, libera su memoria, finaliza el gestor de configuración y limpia la lista de grupos de seguimientos asociados con las fuentes de datos.
Parámetros	-
Tipo Retorno	-

Retorno	-

Tabla 69. Clase dsipsProviderCommunication. Método finalize.

Operación	***synchronized refreshTgFromDS()***
Objetivo	A través del gestor de configuraciones, accede a la BD C-INCAMI y recupera todos los grupos de seguimientos asociados con cada una de las fuentes de datos que están transmitiendo. Recuperada la información, actualiza la copia en memoria estructurada mediante tabla Hash.
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 70. Clase dsipsProviderCommunication. Método refreshTgFromDS.

Operación	***getTgsAssociatedToDS(String ids)***		
Objetivo	Obtendrá una lista con los grupos de seguimientos vinculados al ID de la fuente de datos indicada.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	String	ids	ID de la fuente de datos de la cual se desea consultar los grupos de seguimiento asociados
Tipo Retorno	ArrayList<Tracegroup>		
Retorno	Lista de grupos de seguimientos asociados con el ID de la fuente de datos. Puede retornar null si el ID indicado es inexistente o sin valor.		

Tabla 71. Clase dsipsProviderCommunication. Método getTgsAssociatedToDS.

Operación	***generarRespuesta(Integer bad,Integer ok,String idma)***		
Objetivo	Generará un mensaje C-INCAMI / MIS de estado tipo a partir de las mediciones que han podido leerse y cuales no.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Integer	Bad	Cantidad de mediciones con inconvenientes en la lectura
	Integer	Ok	Cantidad de mediciones correctamente leídas
	String	Idma	Identificador del MA que remite las mediciones
Tipo Retorno	String		
Retorno	Mensaje C-INCAMI / MIS de estado tipo a partir de las mediciones que han podido leerse y cuales no..		

Tabla 72. Clase dsipsProviderCommunication. Método generarRespuesta.

Operación	*isMantenimiento()*
Objetivo	Indica si el proveedor actualmente se encuentra efectuando tareas de mantenimiento
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE = Si actualmente se realizan tareas de mantenimiento FALSE = No hay tareas de mantenimiento en curso

Tabla 73. Clase dsipsProviderCommunication. Método isMantenimiento.

Operación	*setMantenimiento(boolean mantenimiento)*		
Objetivo	Indica al proveedor de servicios si comenzar ó finalizar el mantenimiento.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Boolean	mantenimiento	TRUE = Inicia tareas de mantenimiento FALSE = Finaliza las tareas de mantenimiento en curso
Tipo Retorno	-		
Retorno	-		

Tabla 74. Clase dsipsProviderCommunication. Método setMantenimiento.

Operación	*synchronized getMantemiento_periodo_ms()*
Objetivo	Obtiene el período en milisegundos cada el cual efectúa tareas de mantenimiento
Parámetros	-
Tipo Retorno	Integer
Retorno	Período en milisegundos cada el cual efectúa tareas de mantenimiento.

Tabla 75. Clase dsipsProviderCommunication. Método getMantenimiento_ periodo_ms.

Operación	*setMantemiento_periodo_ms(int mantemiento_periodo_ms)*		
Objetivo	Modifica el período en milisegundos cada el cual efectúa tareas de mantenimiento.		
Parámetros	*Tipo*	*Nombre*	*Concepto*
	Integer	Mantenimiento_periodo_ms	Período expresado en milisegundos cada el cual se efectúan tareas de mantenimiento. Debe ser un valor igual o superior a 1000 ms.
Tipo Retorno	-		
Retorno	-		

Tabla 76. Clase dsipsProviderCommunication. Método setMantenimiento_periodo_ms.

Operación	*synchronized isMantenimiento_activo()*
Objetivo	Consulta si el hilo de mantenimiento está activo
Parámetros	-
Tipo Retorno	Boolean
Retorno	TRUE = Si actualmente el hilo de control está activo FALSE = Si no existe hilo de control con tareas de mantenimiento asignadas

Tabla 77. Clase dsipsProviderCommunication. Método isMantenimiento_activo.

Operación	*synchronized setMantenimiento_activo(boolean mantenimiento_activo)*
Objetivo	Establece como activas/inactivas las tareas de mantenimiento
Parámetros	-
Tipo Retorno	-
Retorno	-

Tabla 78. Clase dsipsProviderCommunication. Método setMantenimiento_activo.

Planteados las funcionalidades de la clase *dsipsProviderCommunication*, la cual implementa las responsabilidades de la función de reunión, se expone a continuación un diagrama de estado asociado al mismo.

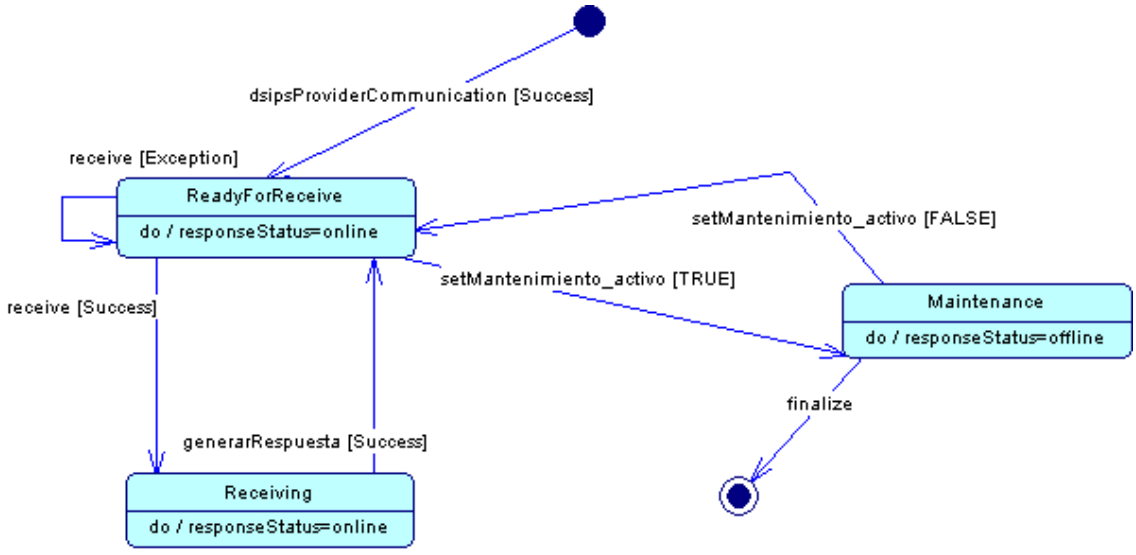

Ilustración 23. Diagrama de Estado de la Clase dsipsProviderCommunication (Función de Reunión)

Básicamente a la función de reunión se le asocian tres estados (Ver Ilustración 23), los cuales son:

- ***ReadyForReceive***: La función de reunión se encuentra lista para receptar mediciones desde cualquier adaptador de mediciones. Desde aquí podrá recibir nuevas mediciones sin inconvenientes para lo que transita al estado '*Receiving*', pero sin en cambio ha ocurrido algún inconveniente en el inicio de la recepción, se mantendrá en el presente estado. Adicionalmente, puede iniciar tareas de mantenimiento *únicamente* desde este estado.

- ***Maintance***: La función de reunión o proveedor de servicios, como se le prefiera mencionar, se encuentra actualmente en tareas de mantenimiento y por ende, imposibilitada de recibir mediciones. Al finalizar el mantenimiento, retornará al estado *ReadyForReceive* donde podrá retomar la recepción de mediciones. Únicamente desde el estado de mantenimiento el objeto puede ser destruido, ya que aquí se garantiza que no hay recepción de ninguna medición y por ende, no se interfiere con ningún flujo.

- ***Receiving***: La función de reunión se encuentra actualmente recibiendo mediciones, exista un error o se reciba correctamente, siempre generará una respuesta indicando el estado de la recepción.

Ahora bien, conocido el espectro de recolección en forma holística y explicados los componentes, tiene sentido ahora abordar los estados posibles por los cuales puede transitar el adaptador de mediciones.

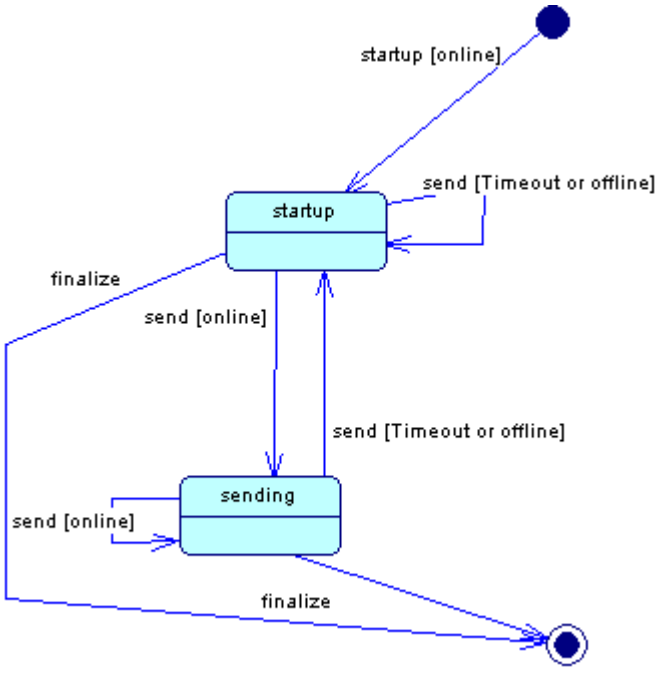

Ilustración 24. Diagrama de Estado de la Clase MeasurementAdapter

Al adaptador de mediciones se le asocian básicamente dos estados (Ver Ilustración 24), los cuales son:

- ***Startup***: El MA se encuentra listo para transmitir pero aún no ha comenzado ninguna transmisión. Si inicia una transmisión y falla, sea porque el proveedor de servicios se encuentra fuera de línea o efectúa un timeout, se mantiene en el estado.

- ***Sending***: El MA ha iniciado la transmisión del flujo sin inconvenientes y se mantendrá en el estado hasta culminar la misma. Puede que al culminar un envío tenga cola de espera de mediciones por transmitir, con lo cual puede intentar desde el mismo estado seguir transmitiendo. Si el inicio del nuevo envío procede correctamente se mantiene en el estado, caso contrario transita al estado *startup.*

En cualquiera de los dos estados puede recibir la señal de finalización del objeto. Esto es así porque el MA se encuentra en un dispositivo móvil cercano a las fuentes de datos y podría ocurrir, entre otros tantos inconvenientes, que el dispositivo se quede sin energía con lo cual *debe* garantizarse un estado de consistencia sea que se esté transmitiendo o no. En caso de estar transmitiendo y el MA cortar abruptamente el flujo, la función de reunión procesará solo hasta las mediciones receptadas correctamente y generará su respuesta con el resumen de la comunicación. Esto último implica que el proveedor de servicios garantiza la consistencia de los datos recibidos, descartando aquellos que son incoherentes o no responden a la definición del esquema C-INCAMI/MIS

4.8 Paralelismo y Concurrencia. Implementación en EIPFD

La presente sub-sección explica el modo en que el enfoque integrado de procesamiento de flujos de datos centrado en metadatos de mediciones incorpora el paralelismo y la gestión de concurrencia de modo natural a los efectos de resguardar la consistencia de los datos para su posterior análisis estadístico e inferencial.

4.8.1 Paralelismo en EIPFD

La lógica funcional del enfoque integrado de procesamiento de flujos de datos se puede separar en dos áreas: la primera está asociada a la entidad bajo análisis, desde la cual debemos recuperar las medidas y transmitirlas, y la segunda, se asocia a la recepción y procesamiento de las mediciones correspondientes a la entidad bajo análisis.

La lógica asociada a la entidad bajo análisis, en general es desplegada sobre dispositivos móviles los cuales se conectan directamente a los sensores (termómetros, marcapasos, etc.) para recolectar la medida. Sobre dichos dispositivos móviles, se ejecuta la pieza de software que implementa la funcionalidad del adaptador de mediciones (Ver Ilustración 4) y por ende recepta/organiza localmente, previo a la transmisión a la función de reunión, las medidas desde los sensores. El adaptador de mediciones, previo a cualquier transmisión y solo durante su inicialización, debe registrar ante los servicios web de configuración de EIPFD, qué métricas implementará cada sensor al que se asocia y a qué entidad bajo análisis medirá. Esto último puede verse gráficamente mediante la Ilustración 7 en el caso de aplicación, donde una cantidad de sensores de marcas/modelos determinados se asociaban a paciente 1 y éste al dispositivo Palm Treo 750. Una vez configurado, el MA se encuentra listo para informar las mediciones a la función de reunión.

La segunda área funcional se asocia a la función de reunión, la cual debe receptar los flujos de datos, reunirlos y remitirlos a un buffer central. Dado que los MA se encuentran distribuidos geográficamente recolectando mediciones desde sus entidades bajo análisis asociadas, la forma de receptar los flujos ha sido implementada mediante servicios web desplegados sobre un servidor de aplicaciones Apache Tomcat 6.0.20 con balance de carga. Esto permite, receptar los flujos en paralelo desde los adaptadores de mediciones y dar instrucción de cómo procesar los mismos ante el buffer en función de sus metadatos. Claro que la clase responsable de dar instrucción de procesamiento en base a los metadatos es propietaria del EIPFD y se denomina *dsipsProviderCommunication*. La misma, presentada en la sub-sección 4.7, corresponde a una clase activa invocada por los servicios web y brinda acceso al buffer a través de su gestor (cuya implementación corresponde a las clases bufferTG y bufferMetric analizadas en la sub-sección 4.5) ante cada invocación de los servicios web por parte de un MA.

Ahora bien, el hecho de permitir el paralelismo en la recepción mediante servicios web, abre un interrogante previo a la cuestión de la concurrencia ¿Se podrá regular de algún modo que ante incrementos abruptos en las tasas de arribo, la tasa de procesamiento no se vea desbordada? La respuesta en EIPFD fue abordada mediante dos mecanismos complementarios:

a) **El buffer local de cada métrica incorpora un comportamiento de cola sincronizada en cada grupo de seguimiento**. Dado que tanto para las actividades de minería de datos como para las técnicas predictivas, el mayor peso relativo se asocia con la historia reciente, cada métrica en el buffer (la que luego se convertirá en variable bajo análisis en los análisis estadísticos posteriores y/o técnicas de minería de datos) incorpora una cola sincronizada con comportamiento dinámico y límite máximo de crecimiento. Esto implica que la cola tiene una tamaño finito pre-configurado (el cual puede ser dinámicamente modificado si se desease), el cual al llenarse, comienza a descartar en forma sincronizada tantos datos antiguos como nuevo deseen incorporarse. De hecho, suponiendo que por algún motivo se disminuya dinámicamente el tamaño de cola para la métrica, la cantidad de datos que excedan al nuevo tamaño, serán descartados a partir de los datos más antiguos.

b) **Load Shedding**. La idea de la cola sincronizada supone que el gestor de buffer a nivel de grupo de seguimiento no ha sido desbordado, por lo que al incorporar la solución del inciso anterior se incorpora también un cuello de botella en el gestor de buffer a nivel de grupo de seguimiento (Ver sub-sección 4.5). Para abordar esta problemática se incorpora la técnica de load shedding con descarte selectivo a nivel buffer de grupo de seguimiento. Dado que toda medición, previo a derivarse al buffer a nivel de métrica debe primero atravesar el buffer a nivel de grupo de seguimiento, la técnica de load sheeding basada en metadatos C-INCAMI, permite priorizar las métricas a preservar en caso de desborde y a través de los metadatos en el flujo (Ver sub-sección 4.1), determina si la medición es o no prioritaria a los efectos de su descarte o permitir su avance al siguiente nivel del buffer. De este modo, se logra regular la tasa de servicio ante incrementos abruptos en la tasa de arribo a nivel de buffer de grupo de seguimiento.

La siguiente cuestión es, si la clase *dsipsProviderCommunication* es quien informa al buffer a nivel grupo de seguimiento las mediciones y éste último mediante load shedding regula la tasa de arribo vs la tasa de servicio, ¿Cómo evito el desborde a nivel servicios web? Es una duda razonable por cuanto lo mismo ocurrió en el buffer a nivel de métrica cuando se incorpora la regulación mediante cola sincronizada, se trasladó la problemática a nivel buffer de grupo de seguimiento y allí se aplicó load shedding para resolverlo, por ende ahora si crece la cantidad de MA que desean transmitir simultáneamente ¿Cómo resolverlo? Ello fue lo que motivó a implementar los servicios web con balanceo de carga. De esto modo, ante una situación de potencial desborde de todos los nodos Tomcat ejecutándose, basta con agregar los nodos dinámicamente para incrementar la capacidad de balance de carga sin que sufra riesgo la aplicación del EIPFD.

4.8.2 Concurrencia en EIPFD

Una vez receptados los flujos mediante los servicios web, la responsabilidad de organización de las mediciones es trasvasada el gestor de buffer a nivel de grupo de seguimientos, instancia de la clase bufferTG (Ver sub-sección 4.5). Dichas instancias del buffer intentarán acceder al mismo en forma concurrente ante cada petición de los servicios web y la cuestión es ¿Cómo lograr la unificación de mediciones, evitando corrupción de datos y sin caer en la serialización? Es por ello que el buffer fue organizado multinivel. Dado que cada MA invocador por excelencia de los servicios web, transmite flujos de mediciones en nombre de un grupo de seguimiento, cada instanciación de servicio web arrojará mediciones para un determinado grupo. Si en paralelo, otro MA invoca el servicio web para transmitir mediciones, seguramente corresponderán a otro grupo de seguimiento. De este modo, si se percata la organización del buffer expuesto en la sub-sección 4.5, no se dará conflicto en las mediciones por cuanto la clase bufferTG implementa una tabla hash para localizar en tiempo constante el gestor de métricas para el grupo de seguimiento, localizado éste se informa dicho gestor al servicio web para que prosiga con el aprovisionamiento de mediciones. De este modo, la única serialización posible a nivel buffer de grupo de seguimiento, es el asociado a la localización de la instancia de bufferMetric dentro de la tabla hash, lo cual es un tiempo constante y reduce notablemente el riesgo de serialización.

Como se ha mencionado anteriormente, dado que la cola a nivel buffer de métrica se encuentra sincronizada, la única forma posible de serialización radicaría en que el mismo MA invoque por segunda vez los servicios web (a los efectos de transmisión de mediciones), cuando aún el buffer a nivel de métrica de su grupo de seguimiento no haya culminado de procesar la primera invocación. De ser esa la situación, se pone en funcionamiento los mecanismos de load shedding bajo la política configurada pero suponga adicionalmente, que la métrica es prioritaria y por ende atraviesa el mecanismo de load shedding inundando el buffer a nivel de métrica ¿se caería en una serialización? No, porque como se mencionó anteriormente lo que prevalece en EIPFD es la historia reciente, por lo que ante una situación de este tipo, la sincronización buscaría evitar corrupción del dato (Ejemplo: que la misma métrica para igual grupo de seguimiento posea dos medidas deterministas diferentes en igual instante de tiempo) y aquí en cambio se descartaría automáticamente de la cola (sin mediar

análisis) un volumen de datos antiguos igual al que desea ingresar, efectuando directamente su reemplazo en igual espacio de memoria.

5 Simulación del Caso de Aplicación

El capítulo anterior discutió C-INCAMI/MIS como esquema de intercambio de mediciones basado en el marco formal de medición y evaluación C-INCAMI. Seguido, se plantearon las interfaces necesarias de implementación para tornarse en fuente de datos plausible de transmisión y responsabilidades del procesador central o función de reunión. Luego, se analizo el procesador de esquema C-INCAMI/MIS, el cual permitía la serialización y deserialización de las mediciones dentro del flujo. Se presentaron las responsabilidades del adaptador de mediciones como origen vinculado directamente a los sensores de datos y sus responsabilidades para gestionar un buffer local que permita optimizar y regular las transmisiones por la red. Se han analizado políticas de abordaje de load shedding, embebidas dentro del enfoque integrado de procesamiento de flujos de datos mediante la función de reunión y finalmente, se analiza cómo se implementa la recepción de mediciones mediante la función de reunión y su vínculo con el buffer central multinivel.

El presente capítulo, plantea la arquitectura sobre la que ejecutará la simulación acotado al caso de aplicación presentado en la sub-sección 3.6, en forma conjunta con su objetivo y planificación. Seguido, se analizan las métricas involucradas en la simulación, para efectuar luego un análisis descriptivo de las variables del estudio, un análisis de los factores que incorporan variabilidad al sistema junto con un análisis de correlación. Finalmente, se analiza el incremento en el tiempo de procesamiento y su relación con las variables bajo estudio.

5.1 Arquitectura

La simulación se desarrolló sobre un sistema de memoria compartida con procesador Athlon Dual Core de 64 bits corriendo sobre plataforma Windows Home Premium. Para ejecutar el enfoque integrado de procesamiento de flujos de datos (EIPFD) se emplea JRE 1.6.0_20 comunicada, mediante multihilo, con el software estadístico R mediante el paquete Rserve.

5.2 Objeto de la Simulación

Analizar el comportamiento del EIPFD en cuanto al tiempo requerido para procesar, analizar estadísticamente la ventana de datos, suavizarla y disparar las alarmas pertinentes si fuese el caso, a medida que varía la cantidad de variables y la

cantidad de mediciones asociadas a cada una de ellas en las ventanas de procesamiento.

5.3 Planificación de la Simulación

Se pretende definir dos variables para la prueba:

- cantidad de variables intervinientes en una ventana de transmisión/procesamiento
- cantidad de mediciones informadas dentro de la ventana de transmisión/procesamiento asociada a cada variable

Las variables son representadas por las métricas directas que definen qué medir y en la misma se tienen en cuenta tanto las métricas vinculadas a atributos de la entidad como aquellas vinculadas a propiedades del contexto de la entidad. De este modo la cantidad de variables entiende a los atributos de la entidad como así también a las propiedades contextuales.

La prueba variará desde un mínimo de 3 variables hasta 99 como así también variará desde 100 mediciones hasta 1000 asociado a cada una de ellas. Se parte de 3 variables como mínimo para que tenga sentido realizar el análisis de correlación y el de componentes principales dado que el objeto final es medir tiempo. Si en la práctica la cantidad de variables fuese inferior a 3, desde el punto de vista del procesamiento sería beneficioso ya que el volumen de datos se ve reducido. Se fija un máximo de 99 variables ya que si bien en la vida real pueden existir proyectos con mayor cantidad, esta magnitud dado el rango que define, nos permitiría aproximar una tendencia en cuanto al consumo marginal de tiempo. En cuanto a las mediciones, se parte de 100 mediciones para simular un tamaño mínimo de una ventana de mediciones pero sin ningún aspecto especial, si fuese menor se reitera que beneficia al procesamiento ya que el objeto en este sentido es medir tiempo insumido por el análisis estadístico. Como cota máxima se toma 1000 mediciones para cada variable involucrada. El número es escogido racionalmente desde el punto de vista de los data stream, ya que el flujo de datos se caracteriza por transmitir flujos tan pronto como sea posible y si fuese el caso ideal en forma continua sin interrupciones, el tamaño de la ventana tendería a disminuir hasta 1 medición por variable. No obstante, se planteo la variación hasta 1000 mediciones previendo la posible incorporación de transmisiones asíncronas a ráfagas.

Téngase en cuenta que el caso máximo analizado considera 99 variables con 1000 mediciones cada una, lo que representa en el análisis estadístico de 99000 mediciones en forma conjunta para una ventana dada (1 único envío desde el MeasurementAdapter). La idea de establecer el máximo de mediciones por variable en 1000 es a los efectos de analizar la variabilidad del consumo de tiempo marginal a medida que se incrementa el requerimiento de procesamiento, por lo que nos permitirá analizar una tendencia en este sentido ante el supuesto caso de que se desee a posteriori analizar 1500, 2000 o más mediciones por variables.

Téngase en cuenta que cada medición será analizada como Double de JAVA, por lo que el requerimiento mínimo de memoria, en caso de desear reproducir esta prueba en otra arquitectura, será 3*(tam del Double en la Arquitectura)*100. En forma análoga el tamaño máximo de memoria requerido será ((99*(tam del Double en la Arquitectura)*1000) + Memoria para alojar programa Java + Memoria para alojar R y Rserve).

5.4 Definición de Métricas para la Simulación

En la prueba se definieron las siguientes métricas directas:

- Tiempo en milisegundos insumido en la apertura de la conexión a Rserve (Nombre de la variable: *stratup*)
- Tiempo en milisegundos insumido en la preparación del conjunto de datos aleatorio (Nombre de la variable: *gendato*)
- Tiempo en milisegundos insumido en el Análisis Descriptivo de los datos (Nombre de la variable: *andesc*)
- Tiempo en milisegundos insumido en el Análisis de Correspondencias (Nombre de la variable: *cor*)
- Tiempo en milisegundos insumido en el Análisis de Componentes principales (Nombre de la variable: *pca*)
- Tiempo en milisegundos insumido en el proceso total (Nombre de la variable: *total*)Los datos en bruto arrojados durante la prueba se adjuntan en forma anexa dentro del archivo denominado stressR.dat.

5.5 Análisis Descriptivo

A partir de las mediciones de la prueba efectuada sobre el EIPFD se ha obtenido, como puede apreciarse en la Tabla 79, que:

	startup	gendato	andesc	cor	pca	total
Mínimo	0.00000	0.00000	15.00000	0.00000	0.00000	15.00000
Primer Cuartil	0.00000	0.00000	94.00000	15.00000	16.00000	125.00000
Mediana	0.00000	0.00000	203.00000	16.00000	31.00000	250.00000
Media	0.03561	0.23640	268.60000	24.57000	50.40000	343.00000
Tercer Cuartil	0.00000	0.00000	453.00000	31.00000	63.00000	546.00000
Máximo	47.00000	16.00000	936.00000	110.00000	234.00000	1092.00000
Desv.Standard	1.29363	1.97301	204.16967	23.56094	46.33493	265.65655

Tabla 79. Resumen Descriptivo. Valores expresados en milisegundos (ms).

Las variables *startup* y *gendato* prácticamente no inciden en el tiempo total, esto puede deducirse a partir del tiempo máximo insumido por cada una de ellas y si adicionalmente, se compara sus desviaciones estándar contra cualquiera de las variables involucradas en el análisis estadístico propiamente dicho, la relación es de al menos 1 a 11,94 (Surge de 23,56094/1,97301).

Del cuadro surge que la variable *andesc* tiene una alta incidencia en el tiempo total y le seguiría en orden descendente las variables *pca* y *cor*. De igual modo, andesc es la que mayor desvío estándar presenta (204,16967), seguido por *pca* (46,33493) y *cor* (23,56094), lo que considerando las medianas, medias, máximos y mínimos estaría indicando un desplazamiento de la media hacia valores superiores.

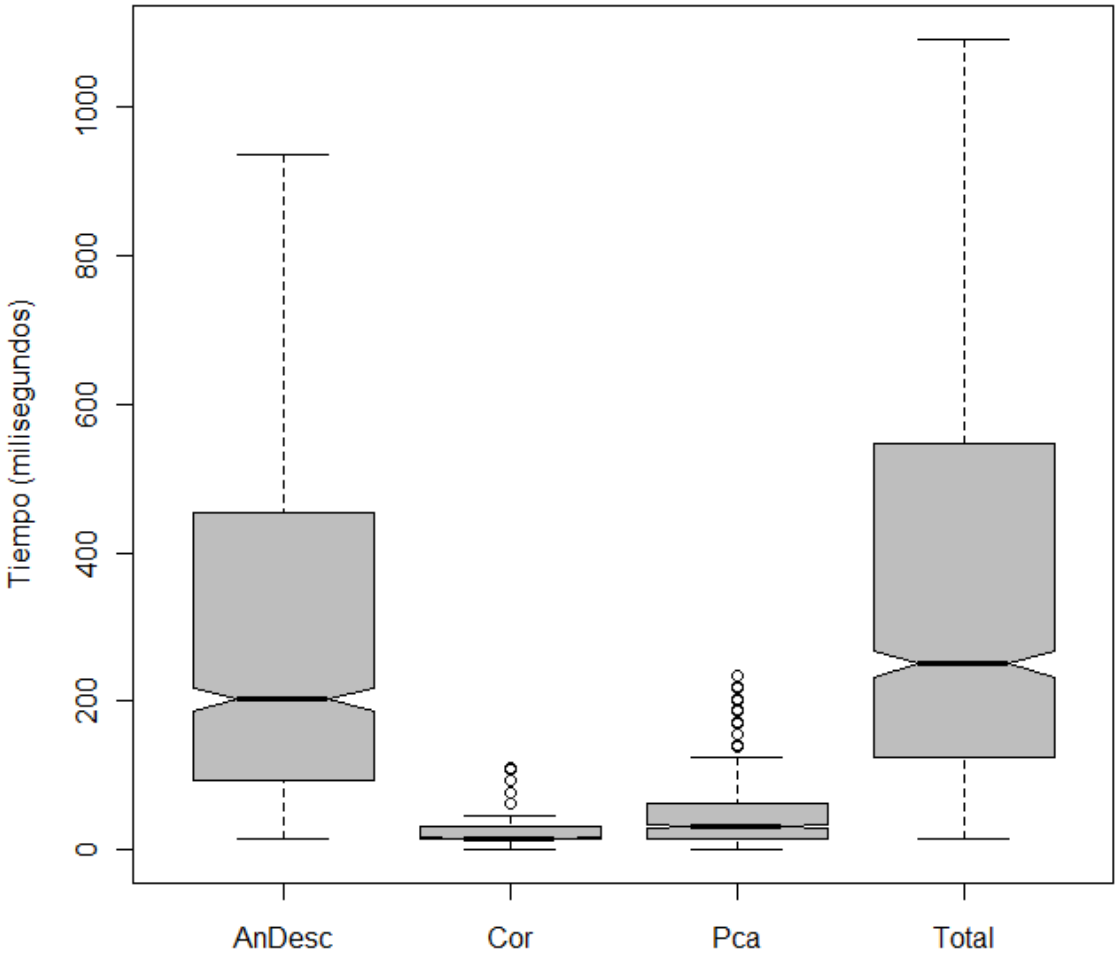

Ilustración 25. Boxplot de las variables andesc, cor, pca y total

El boxplot de la Ilustración 25, permite visualmente identificar como los máximos de cada variable empujan la media hacia valores superiores, principalemente en las variables andesc y total. Adicionalmente a los mencionado, notar que las variables pca y cor presentan outliers superiores (Representados con círculos en el boxplot), lo cual indicaría posibles demoras adicionales a la normal, que podrían asociarse con el crecimiento de la ventana de procesamiento.

La Ilustración 26 permite visualizar el comportamiento conjunto de la evolución en la cantidad de mediciones, la cantidad de variables y el tiempo total del proceso en milisegundos. Desde la perspectiva de las mediciones puede observarse que a medida que se incrementan, la evolución del tiempo total del proceso no es significativa. No ocurre lo mismo en absoluto si se analiza la perspectiva de la cantidad de variables como expone la Ilustración 27.

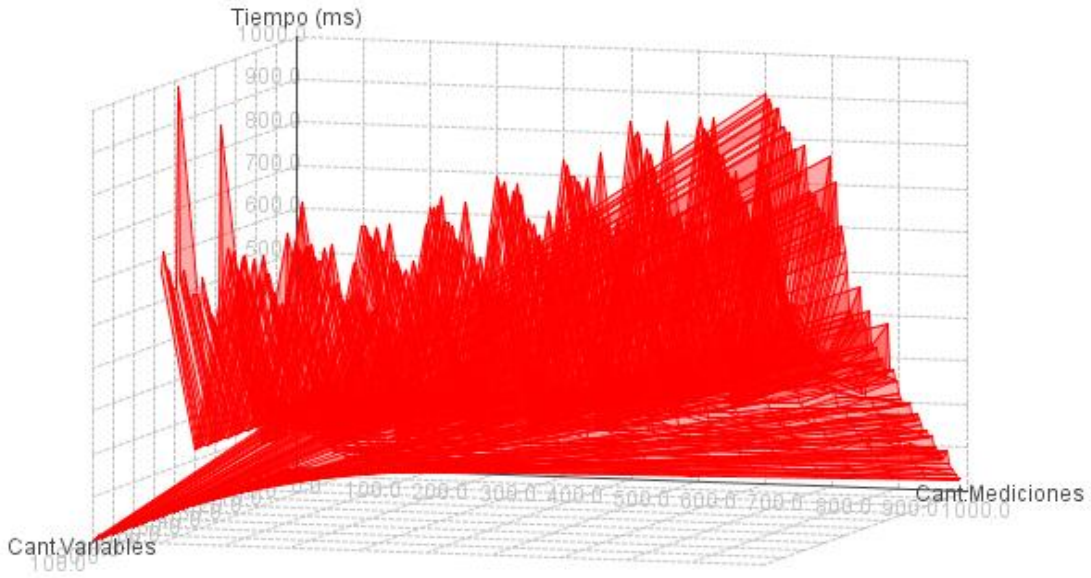

Ilustración 26. Evolución Tiempo, cantidad de variables y cantidad de mediciones. Vista Tiempo-Mediciones.

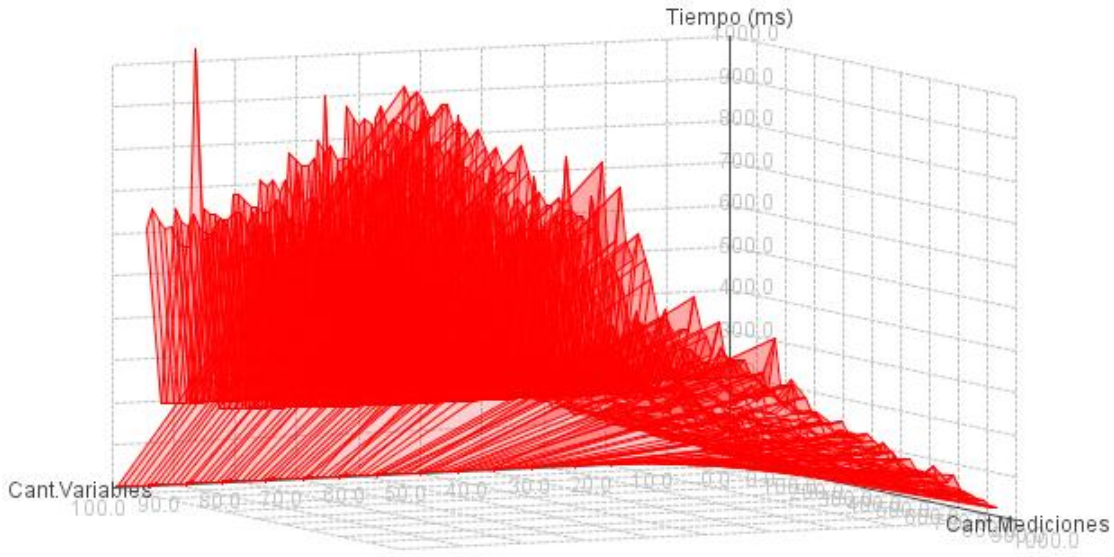

Ilustración 27. Evolución Tiempo, cantidad de variables y cantidad de mediciones. Vista Variables-Mediciones.

Los incrementos temporales que produce el incremento en la cantidad de variables es visualmente superior al que produce el incremento en la cantidad de mediciones. Adicionalmente, se puede observar desde una perspectiva superior de la gráfica, analizándola desde el punto de vista temporal, ver Ilustración 28, que a medida que la cantidad de variables se incrementan comienzan a aparecer picos de procesamiento.

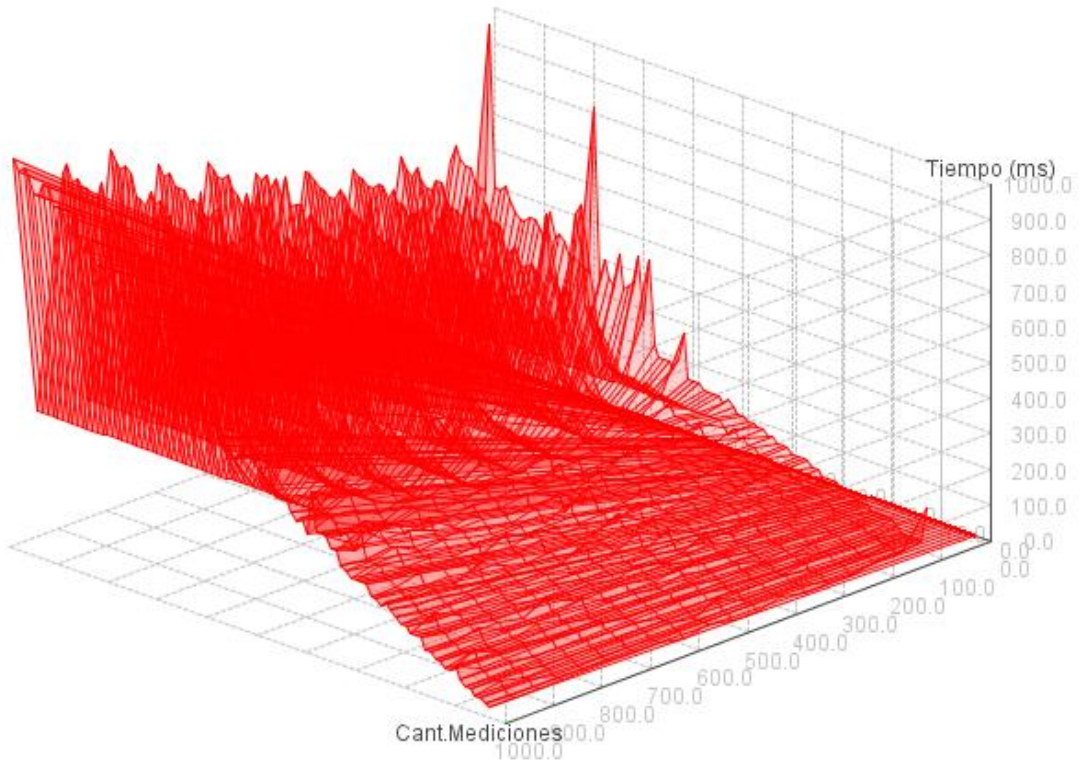

Ilustración 28. Evolución Tiempo, cantidad de variables y cantidad de mediciones. Vista Superior Mediciones-Tiempo.

En este último sentido, puede diferenciarse claramente dos zonas visuales, una que podría denominarse “serena“, donde el incremento en el número de variables y observaciones no afectan el normal crecimiento al tiempo global y otra que podríamos denominar “Picada” (por su forma) donde el incremento de las variable incorporan cierto grado de volatilidad dado por la aparición de outliers en el tiempo global de procesamiento.

5.6 Variabilidad del Sistema

A los efectos de analizar las variables que más aportan a la variabilidad global del sistema se aplica análisis de componentes principales (PCA) mediante utilización de matriz de correlación obteniendo en la Ilustración 29, la explicación de la desviación:

Componente	sd	var	% Explicada	Acumulada
1	2,1546760750	4,6426289882	58,03%	58,03%
2	1,0553707930	1,1138075107	13,92%	71,96%
3	0,9974267030	0,9948600279	12,44%	84,39%
4	0,9893999600	0,9789122808	12,24%	96,63%
5	0,3768626410	0,1420254502	1,78%	98,40%
6	0,3235947290	0,1047135486	1,31%	99,71%
7	0,1518239450	0,0230505103	0,29%	100,00%
8	0,0012977970	0,0000016843	0,00%	100,00%
		8,0000000010		

Ilustración 29. Variabilidad Explicada por Cada Componente Principal

Note que el primer componente principal (PC) explica él solo el 58% de la variabilidad del sistema. De este modo, es sumamente interesante analizar la composición de los aportes del primer componente principal, el cual se expone en la Tabla 80, como salida de R:

```
> pca$loadings

Loadings:

Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8

qVar -0.428 0.302 0.374 -0.202 0.731

meds -0.144 -0.868 -0.165 0.113 0.384 -0.188

Startup 0.213 -0.970 0.110

GenDato 0.104 0.132 0.983

AnDesc -0.449 0.189 0.314 -0.537 -0.602

Cor -0.434 -0.141 -0.738 -0.487

Pca -0.435 -0.182 -0.223 0.822 0.167 -0.137

Total -0.460 0.103 0.137 -0.379 0.783
```

Tabla 80. Salida de R para PCA

PCA nos confirma lo mencionado sobre el tiempo de startup y generación de datos aleatorios, prácticamente no participan en PC 1 pero tener precaución, no por ello descartarla porque es la variable de mayor aporte en PC 3 que explica el 12,44% de la variabilidad del sistema.

La PC 1 confirma lo esquematizado visualmente en el análisis descriptivo, las mediciones inicialmente no aportan demasiado mientras que sí lo hace de un modo relativamente homogéneo *qvar, andesc, total, pca y cor.* Al igual que el párrafo anterior no debe descartarse a *meds* solo por la PC1, ya que es la variable que más aporta en PC2 la cual explica el 13,92% de la variablidad del sistema.

De este modo, PCA nos permite determinar tres ejes claros de variabilidad, PC 1 el cual se nutre principalemente de las variables *qvar, andesc, total, pca y cor* y explica el 58,03%. PC 2 el cual se nutre principalmente de *meds* y explica el 13,92% y finalmente PC 3 el cual se nutre principalmente de *startup* el cual explica el 12,44%. De este modo los tres primeros componentes principales explican el 84,39% de la variabilidad del sistema.

5.7 Análisis de Correlación

Dado que se ha detectado la presencia de outliers en alguna de las variables bajo estudio, es conveniente emplear la matriz de correlación para eventualmente detectar posibles relaciones lineales entre las variables.

```
> cor(matriz)
                qVar          meds       Startup      GenDato      AnDesc
qVar     1.000000e+00  8.942024e-05 -0.039881269  0.124507229  0.97169273
meds     8.942024e-05  1.000000e+00 -0.044269866  0.034513644  0.12442329
Startup -3.988127e-02 -4.426987e-02  1.000000000 -0.003413529 -0.02989642
GenDato  1.245072e-01  3.451364e-02 -0.003413529  1.000000000  0.13021685
AndDesc  9.716927e-01  1.244233e-01 -0.029896421  0.130216851  1.00000000
Cor      7.853603e-01  3.998799e-01 -0.028730492  0.100511248  0.84501302
Pca      7.735025e-01  4.491646e-01 -0.011535517  0.106564187  0.84588770
Total    9.520086e-01  2.094318e-01 -0.022693096  0.134735993  0.99177132
               Cor          Pca       Total
qVar     0.78536035  0.77350250  0.95200862
meds     0.39987990  0.44916456  0.20943178
Startup -0.02873049 -0.01153552 -0.02269310
GenDato  0.10051125  0.10656419  0.13473599
AndDesc  0.84501302  0.84588770  0.99177132
Cor      1.00000000  0.88982053  0.89385578
Pca      0.88982053  1.00000000  0.90409854
Total    0.89385578  0.90409854  1.00000000
```

Ilustración 30. Salida del Análisis de Correlación de R

En principio existiría, según se puede ver en la Ilustración 30, una fuerte correlación entre *qvar* (cantidad de variables) para con *andesc, cor, pca y total* pero no así con la cantidad de mediciones. Esto último ratifica analíticamente lo que se exponía visualmente con respecto a los incrementos en el tiempo total.

Se puede observar correlaciones entre *total* con respecto a *pca, cor y andesc* lo cual es así dado que total se compone de la sumatoria de los anteriores y como pudo observarse en el análisis descriptivo son quienes definen el valor final del tiempo total de procesamiento. De este modo se podría indicar que el PC 1 es en realidad explicado por *qvar* y *total*.

De este modo, si aplicase PCA solo sobre meds, qvar y total debiera arrojar un resultado similar al indicado en el PCA original (ver Ilustración 29 y Ilustración 31).

Componente	sd	var	% Explicada	Acumulada
1	1,4052728000	1,9747916424	65,83%	65,83%
2	0,9999812000	0,9999624004	33,33%	99,16%
3	0,1588897000	0,0252459368	0,84%	100,00%

Ilustración 31. Componentes Principales Sin Variables Correlacionadas.

En este caso inclusive, entre la primera y segunda componente se explica prácticamente la variabilidad completa del sistema. De este modo es de suma importancia analizar su composición como expone en la Tabla 81.

> pca$loadings
Loadings:
Comp.1 Comp.2 Comp.3
qVar **0.691** 0.215 0.691
meds 0.152 **-0.977** 0.152
Total **0.707** -0.707

Tabla 81. Análisis PCA. Composición de Componentes Basados en Variables No Correlacionadas.

Es importante destacar en esta línea de razonamiento que quienes vuelven a explicar la variabilidad del sistema son *qvar* y *total* a través de PC 1, siendo la variable *meds* la de mayor aporte en PC 2. De este modo se decanta que la principal variabilidad del sistema dentro del contexto de las pruebas realizadas va de la mano con la cantidad de variables y en menor medida por la cantidad de mediciones.

5.8 Incremento en el Tiempo de Procesamiento

Por lo expuesto anteriormente visual y analíticamente, se sabe que la mayor incidencia en el tiempo de procesamiento estaría influida, por la evolución del número de variables más que por el incremento en la cantidad de mediciones en cada ventana de procesamiento. De este modo, el objeto a partir de ello es visualizar el comportamiento del tiempo a medida evoluciona la cantidad de variables a los efectos de ratificar visualmente lo expuesto analíticamente en el análisis de correlación.

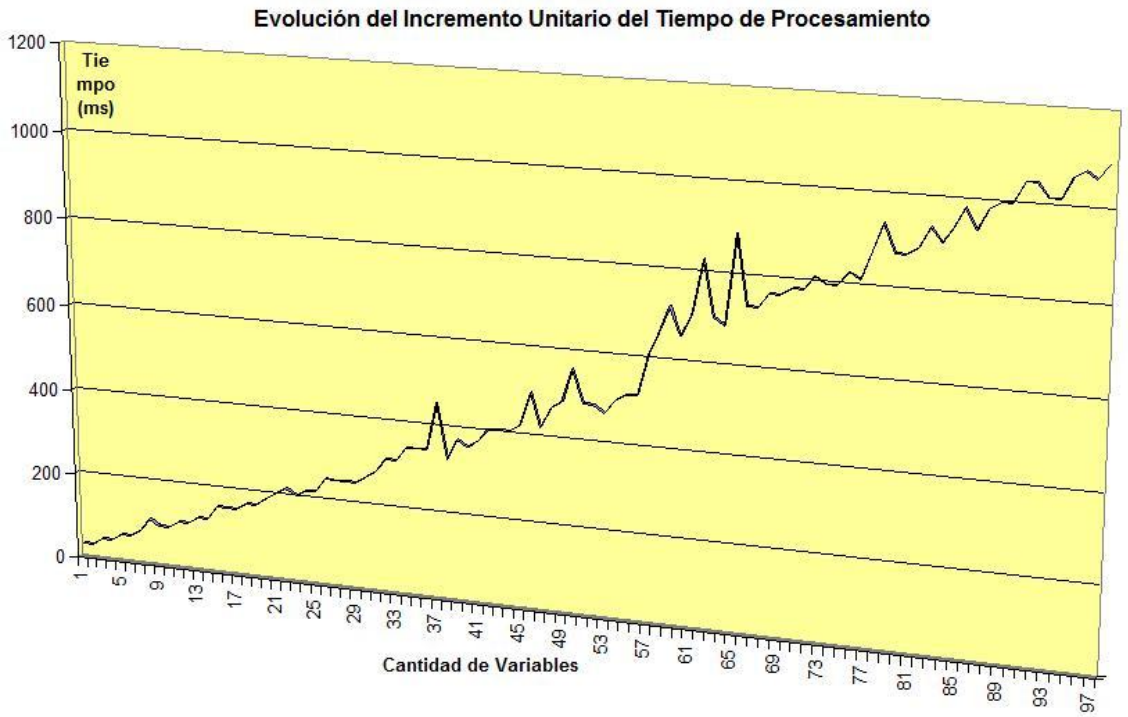

Ilustración 32. Evolución del Tiempo de Procesamiento.

La Ilustración 32 se ha construido, fijando en 1000 las mediciones (variable meds) y obteniendo los datos del tiempo y cantidad de variables sobre sus respectivos ejes, esto

representa en realidad, un corte en el plano sobre gráfico de superficie anteriormente expuesto a los efectos de visualizar los ejes tiempo vs cantidad de variables. Aquí puede apreciarse claramente que la variable *total* presentaría un comportamiento aproximadamente lineal tal lo mencionado en el análisis de correlación entre la variable *qvar* y *total,* donde se indicaba un coeficiente de correlación de 0,952. A modo de afianzar estadísticamente el coeficiente de correlación (Basado en Pearson) expuesto, a continuación se expone el intervalo de confianza para las variables mencionadas y el coeficiente dado con un nivel de confianza de 0.95:

Pearson's product-moment correlation

data: parte[, 1] and parte[, 2]

t = 112.9214, df = 1318, **p-value < 2.2e-16**

alternative hypothesis: true correlation is not equal to 0

95 percent confidence interval:

0.9466804 0.9568162

sample estimates:

cor

0.9520086

Ilustración 33. Intervalo de Confianza para el Coeficiente de Correlación

La probabilidad arrojada por el test es claramente inferior a 0,05; por lo que existiría correlación y su intervalo de confianza marca claramente que no se corre riesgo de incorporar al 0 dentro del mismo lo que podría darse incluso aunque se dispusiese de un coeficiente de correlación alto. En este último, sentido quedo claramente demostrado que el coeficiente de correlación entre la variable *qvar* y *total* es confiable y expondría una fuerte relación lineal entre ellas.

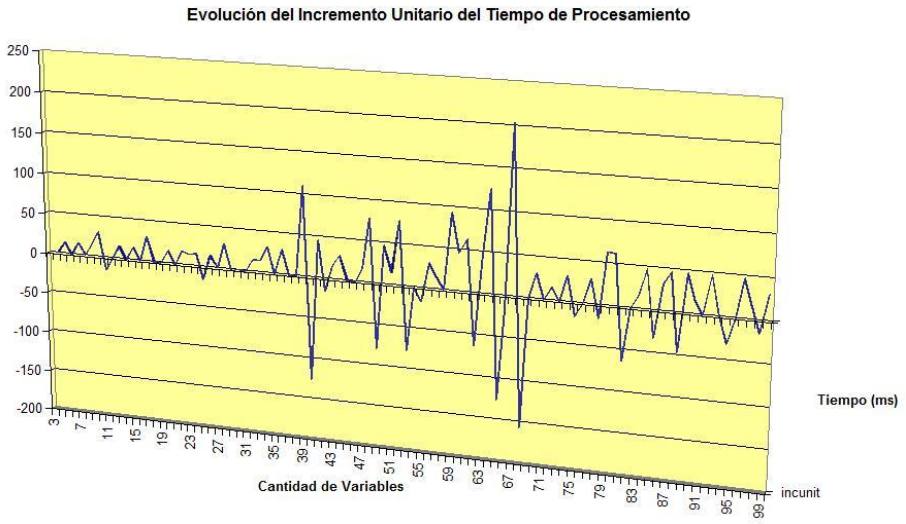

Ilustración 34. Evolución del Incremento Unitario del Tiempo de Procesamiento

La figura expone el incremento unitario de tiempo en milisegundos a medida que evoluciona la cantidad de variables, de este modo se puede apreciar que el incremento de las variables estaría contenido en mas menos 50 milisegundos adicionales por cada variable incorporada, de hecho la serie convergería dentro de dichos límites. Si es importante destacar que entre 35 y 70 variables presenta un comportamiento atípico, el mismo que se indicase en el gráfico de superficie el cual posiblemente se manifieste por la transición en el volumen de datos teniendo en cuenta la cantidad de variables y su localización en los caches primarios, secundarios y memoria principal del ordenador.

5.9 Análisis del Paralelismo y la Concurrencia en EIPFD

El incremento progresivo en la cantidad de variables ha permitido analizar el comportamiento del procesamiento paralelo en el EIPFD, dado que a medida que se incorporaba un MA en la invocación de servicios web, se incrementaba en tres la cantidad de variables hasta llegar a las 99 variables de la simulación. De este modo se ha pretendido probar el EIPFD desde la situación en que se iniciaba el procesamiento con 3 variables (3 variables por cada MA, por ende el procesamiento inicia con un único MA) hasta llegar a 99 variables (33 MA transmitiendo en paralelo) de modo de analizar los tiempos de procesamiento con la evolución de los MA involucrados. Adicionalmente la evolución en la cantidad de mediciones por variable ha pretendido simular situaciones de incremento en la tasa de servicio dentro de la cola a nivel de métrica, con vistas a validar el comportamiento concurrente en el buffer.

Si se analiza la Ilustración 27, se puede observar que el incremento en la cantidad de variables incrementa el tiempo global de procesamiento en mayor magnitud al incremento en la cantidad de mediciones. Esto último se debe en primer lugar, a que el hecho de incrementar la cantidad de variables incorpora en los análisis estadísticos mayor interacción en las mismas haciéndolo más complejo. En segundo lugar, cada nuevo trío de variables que se incorpora con el MA, incrementa la complejidad del buffer a nivel de grupo de seguimiento en términos de tamaño en memoria (un nuevo casillero en la tabla hash) como de sincronización en el acceso al mismo desde los servicios web. Finalmente, con 33 MA transmitiendo en paralelo solo se llegó a 1 segundo de tiempo total de procesamiento para 1000 mediciones en paralelo en cada métrica, es decir un total de 99000 mediciones en paralelo arribando al buffer mediante los servicios web. Note sin embargo, que el incremento en la cantidad de mediciones expuesto en la Ilustración 26, no ha comprometido prácticamente la tasa de servicio de los procesadores, esto valida inicialmente y sobre el caso de aplicación analizado, los mecanismos de balanceo de carga, load shedding y sincronización de colas dentro del buffer.

Adicionalmente, el incremento en la cantidad de mediciones ha perseguido poner a prueba el control de consistencia sobre los buffer, ante una situación en que 33 MA están intentando con 3 métricas cada uno actualizar su buffer de métrica, por lo que se estaría produciendo una congestión de acceso a nivel de buffer de grupo de seguimiento. En los hechos, el mecanismo de acceso a bufferTG desde servicios web no presento mayor complicación en términos de desborde, permitiendo acceso paralelo y actuando correctamente la cola sincronizada a nivel de métrica. Dicha consistencia, se ve reflejado en los tiempos de los análisis estadísticos posteriores (Análisis descriptivo, de correlación y de componentes principales), dado que si hubiese existido corrupción del buffer los análisis

posteriores no se hubiesen llevado a cabo y el tomador de decisiones (Ver sub-sección 3.4) hubiese disparado las alarmas del caso, dado que esa es justamente una de sus responsabilidades.

6 Conclusiones del Trabajo Final

El presente capítulo aborda las conclusiones a las que se arriban luego de desarrollado el trabajo, como así también, en función de la labor realizada, propone las líneas de investigación futuras vinculadas a la temática.

6.1 Conclusiones

En materia de sistemas de gestión de flujos de datos y considerando los principales exponentes a nivel académico como como MaxStream, STREAM, Borealis, entre otros; el EIPFD propone la gestión de datos basando su comportamiento en los metadatos incorporados con los propios datos. Esto desde el punto de vista del procesamiento, constituye una gran ventaja por cuanto permite desglosar el mismo en función del significado del dato a los efectos no solo de la paralización de la carga de procesamiento sino en cuanto a la organización del buffer central y su acceso concurrente posterior. Este efecto no solo ha quedado demostrado en los aspectos de transmisión y recepción, sino también en los tiempos de procesamiento obtenidos a partir de la simulación efectuada sobre el caso de aplicación propuesto.

El EIPFD incorpora consistencia en términos de análisis, permitiendo realizar análisis predictivos en línea, dado que C-INCAMI no solo permite la incorporación de los metadatos asociados a las métricas, sino que permite modelar y medir el contexto de la entidad bajo análisis. Esto, en términos del proceso de descubrimiento de conocimiento, es sumamente importante por cuanto permite conocer el hábitat de medición y en el peor de los casos, de no serlo, no altera el resultado final y solo acarrea un mínimo overhead de procesamiento. En este último sentido, debe indicarse que si bien puede tomarse ‘n’ mediciones de una entidad, la situación del contexto para un instante de tiempo dado es el mismo, por cuanto no requiere más que una única transmisión que describa el contexto mediante tales métricas para conocer el mismo.

Mediante el adaptador de mediciones y la función de reunión, las cuales hacen uso de CINCAMIProcessor, se ha probado que es posible incorporar a los datos, metadatos descriptivos de un proyecto de medición y evaluación que permita luego guiar el procesamiento posterior. Tal procesador, se basa en procesos de serialización y deserialización bien conocidos como JAXB en base a un esquema definido como

CINCAMI/MIS. En función de la simulación, ha podido observarse que tanto la serialización como la deserialización no representan tiempos significativos en términos del global de procesamiento y pueden considerarse viables al momento de implementar estrategias de medición y evaluación a través de data streams.

Como ha podido observarse mediante la interface transmitionServicePovider, quien implementa en definitiva la función de reunión del EIPFD, el procesamiento de los data streams o fuentes de datos es netamente paralelizado mediante hilos, lo cual denota una ganancia en términos temporales notable frente a estrategias serializadas. Adicionalmente, la propia función de reunión, a partir de los metadatos incorporados en el flujo, determina el modo de distribución de las mediciones directas junto con las propiedades de contexto en el buffer central. Téngase en cuenta, que a partir de dicho buffer central, existe un componente denominado StatisticalAnalyzer, el cual desarrolla en línea y a partir de los datos en el buffer, un análisis descriptivo completo, análisis de componentes principales y análisis de correlación absolutamente automático y guiado por la definición del proyecto de medición y evaluación a través de sus metadatos. Este último componente, si bien incluido a los efectos de la medición en la simulación planteada, no ha sido analizado en detalle dado que escapa al presente trabajo y corresponde a mi tesis doctoral como parte de la función de suavización de EIPFD.

Los mecanismos de load shedding guiados por metadatos, constituyen una alternativa de aplicación interesante para evitar el desborde en la cola de servicios ante incrementos abruptos en las tasas de arribo con respecto a las de procesamiento. De hecho, de los resultados de la simulación puede observarse que el incremento en la cantidad de mediciones no afecta la estabilidad del EIPFD como así tampoco compromete los tiempos de procesamiento totales.

De la simulación, pudieron obtenerse resultados interesantes, como que el EIPFD se ve mayormente afectado por el incremento en la cantidad de variables o métricas del proyecto en lugar de que se incremente el volumen de las mediciones. Esto, en efecto, tiene su explicación desde el punto de vista de que al incrementar las variables se incrementa el universo de las mismas con lo que la combinación de interacciones entre las mismas aumenta considerablemente, comprometiendo los tiempos del análisis de correlación, descriptivo y de componentes principales, no así los modelos de decisión on line mediante árboles. Estos últimos no abordados en el presente trabajo pero presente en el prototipo mediante el proyecto MOA de la Universidad de Waikato.

La capacidad de comunicar vía TCP/IP al EIPFD y fundamentalmente a la función de reunión paralelizada, con el software estadístico R mediante el CRAN RServe, posibilitó agilizar y paralelizar los análisis estadísticos posteriores a la organización del buffer central, mediante la capacidad multithreading de R, RServe y EIPFD.

Mediante la incorporación de los metadatos sujetos a una estricta ontología en la que se basa C-INACMI, ha permitido regular las funciones de load sheeding en base al significado del dato y no de modo aleatorio o sintáctico. Esto permite incrementar la eficiencia del funcionamiento lógico y priorizar los datos que se desean resguardar y procesar.

La simulación de EIPFD se ha desarrollado suponiendo la evolución de las cantidades de variables y mediciones conjuntamente y sobre un caso de aplicación testigo. Tal simulación no hubiese sido posible de no haberse implementado prototípicamente EIPFD sobre JAVA, permitiendo probar en ámbito de laboratorio las funcionalidades y tiempos asociados con cada una de ellas. Adicionalmente, se ha vinculado el prototipo mediante RServe con el software estadístico R, lo que abre un gran abanico de aplicaciones futuras.

El caso de aplicación ha permitido, entre todo lo mencionado, fijar un umbral testigo. Esto es, que para procesar 99 variables con 1000 mediciones cada una se requirió 1 segundo sobre un equipo común y accesible en el mercado. Esto permite inicialmente delinear posibles campos de aplicación, aunque por supuesto dependiendo el ámbito de aplicación a aplicar el prototipo requerirá mayores pruebas sobre otro tipo de dominios y/o casos de aplicación.

La capacidad de balanceo de cargas en los servicios web implementado mediante el servidor de aplicaciones Apache Tomcat 6.0.20, ha permitido simular progresivamente mediante el caso de aplicación presentado en la sub-sección 3.6, la situación desde la que 1 MA iniciaba su transmisión con 3 métricas y 100 mediciones, hasta el hecho de que 33 MA transmitían en paralelo 1000 mediciones por cada métrica, poniendo a prueba el EIPFD ante el arribo en paralelo de 3000 mediciones por MA y accediendo al buffer central con 99000 mediciones en un mismo instante de tiempo. Gracias a los metadatos incorporados en el flujo (Ver sub-sección 4.1), se posibilitó una estructuración y acceso paralelo por niveles al buffer, evitando un primer punto de congestión, y en donde la sincronización de la cola por métrica, posibilitó no solo mantener la consistencia de los datos sino que también, dado que prevalece la política de historia reciente en los datos, evitar la serialización mediante el descarte automático de los mismos ante el arribo de nuevos datos.

Los mecanismos de load shedding junto con el balanceo de carga han expuesto su efectividad en la recepción y reunión de mediciones ante la paralelización en los puntos de ingreso de datos, dado que el peor caso logrado sobre un entorno arquitectural ordinario, ha sido de 1 segundo en el tiempo global de procesamiento.

Es importante destacar que ante una situación en que 99000 mediciones han intentado ingresar en paralelo al buffer, la consistencia de los datos se vio resguardada. Ello se baso en que los metadatos C-INCAMI a través de la estructuración multinivel del buffer, posibilitaron desglosar los puntos de congestión y adicionalmente, el nivel de métrica con cola sincronizada, al basarse en la política de historia reciente y priorización de métricas a través de las técnicas de load shedding en el buffer de grupo de seguimiento, ha permitido seleccionar qué datos métricas descartar a nivel de grupo de seguimiento, cuales preservar y de las preservadas, descartar sus datos históricos para mantener los recientemente ingresados sin mayor análisis. En este último aspecto de historia reciente, pareciera un tanto impactante el hecho de mencionar “descartar sin mayor análisis”, pero por ejemplo dado el caso de aplicación, si las 3 métricas por paciente fuesen presión arterial, frecuencia cardíaca y temperatura ambiente, y un hospital se encontrase monitoreando pacientes trasplantados ambulatorios la pregunta es ¿De qué me sirve la historia de estas métricas si en realidad tengo que velar porque el paciente se encuentre en una situación de control en este momento? En este tipo de contexto es fundamental resguardar la situación más actualizada posible del

paciente y almacenar solo el lapso de historia que pueda influir de algún modo en el estado actual del objeto bajo medición (paciente), por ello la cola sincronizada del buffer a nivel de métrica si bien posee una longitud finita la misma es ajustable dinámicamente.

6.2 Trabajo a Futuro

Dentro de las líneas a futuros en las cuales se pretende trabajar, se encuentra:

- **Incorporar Java Native Interface (JNI) para optimizar el procesamiento intra-hilo en la función de reunión**: Como se mencionó, el prototipo está desarrollado completamente en JAVA y no es precisamente un lenguaje destinado a rendimiento en procesamiento. No obstante, es posible enlazar Java con rutinas en C, las cuales a su vez pueden hacer uso de MPI, para mejorar los tiempos de procesamiento de cada hilo y/o incluso abriendo el juego a una arquitectura de cluster.

- **Llevar el EIPFD a entornos Grid**: EIPFD está pensado para abordar proyectos de medición y evaluación, nutriéndose de fuentes de datos heterogéneas que implementen la interface DataSource. Ahora bien, debiera poder abrirse el concepto de fuentes de datos, para que los usuarios de la grilla hagan uso del procesador de mediciones, permitiendo reconfigurar dinámicamente los proyectos de medición y evaluación en los lapsos de tiempo en los que tienen asignados los recursos.

- **Paralelizar las implementaciones de algoritmos de clasificación on line**: EIPFD basa la operatoria de clasificación en el proyecto Massive On Line Analysis (MOA) de la Universidad de Waikato. MOA posee la totalidad de los algoritmos implementados en JAVA, pero dado que el objeto de este tipo de algoritmos es procesar cada dato o medición del flujo ante su arribo, pareciera una alternativa para avanzar tales implementaciones a entornos basado en lenguaje C con uso de librerías MPI para balancear la carga de trabajo y el load shedding.

- **Paralelizar la recolección de medidas a nivel MA:** Actualmente, el MA recolecta mediante mecanismo POP las medidas a partir de los sensores para luego efectuar su transmisión a la función de reunión (servicios web). Como trabajo futuro se prevé que la política de recolección de MA cambie a un mecanismo de empuje (PUSH) desde los propios sensores, incorporando paralelismo a nivel MA por cada sensor originante del dato (Análogo al flujo de datos con respecto a la función de reunión) y tomando los mismos recaudos en términos de concurrencia y paralelismo que la función de reunión pero a nivel MA.

- **Pool de hilos a nivel de la función de reunión.** La función de reunión supone la instanciación de un hilo al momento de ser invocado el servicio web y luego acceder al buffer central. En la simulación, se ha expuesto el caso en que 33 MA transmitan al mismo tiempo en paralelo, pero la cuestión es ¿siempre estarán los 33 o 'n' cantidad transmitiendo al mismo tiempo? La idea en este sentido es plantear a nivel de función de reunión un pool de hilos cuyo tamaño pueda ser ajustado dinámicamente y analizar

con respecto a la simulación efectuada en este trabajo final, cuál de las estrategias efectúa un mejor aprovechamiento de los recursos variando la carga de trabajo.

Particularmente, creo que las líneas de trabajo enunciadas constituyen áreas interesantes de abordaje futuro, inclusive también podrían serlo para quien quiera avanzarlo para lograr el título de Magíster en la presente disciplina.

7 Bibliografía

Abadi, D., Ahmad, Y., Balazinska, M., Cetintemel, U., Cherniack, M., Hwang, J., y otros. (2005). The Design of the Borealis Stream Processing Engine. *Conference on Innovative Data Systems Research (CIDR)* (págs. 277-289). Asilomar, CA: CIDR.

Abajo Martínez, N. (2004). ANN quality diagnostic models for packaging manufacturing: an industrial data mining case study. *ACM SIGKDD* (págs. 799-804). Seattle, USA.: ACM.

Aleman-Beza, B., Halaschek, C., Arpinar, B., & Sheth, A. (2003). Context-Aware Semantic Association Ranking. *In proc. Semantic Web and Databases Workshop*, (págs. 33-50). Berlin.

Ali, M. A. (2005). NILE-PDT: A Phenomenon Detection and Tracking Framework for Data Stream Management Systems. *VLDB*, (págs. 1295-1298). Trondheim, Norway.

Ali, M., Aref, W., Bose, R., Elmagarmid, A., Helal, A., Kamel, I., y otros. (2005). NILE-PDT: A Phenomenon Detection and Tracking Framework for Data Stream Management Systems. *VLDB*, (págs. 1295-1298). Trondheim, Norway.

Andrews, G. (1999). *Foundations of Multithreaded, Parallel and Distributed Programming.* Addison Wesley.

Arasu, A., Babu, S., & Widom, J. (2006). The CQL Continuos Query Language: Semantics Foundation and Query Execution. *VLDB Journal , 15* (2), 121-142.

Arasu, A., Chaudhuri, S., & R., K. (2008). Transformation-based Framework for Record Matching. *International Conference on Data Engineering (ICDE)* (págs. 40-49). Cancun, Mexico.: IEEE.

Arasu, A., Ganti, V., & R., K. (2006). Efficient Exact Set-Similarity Joins. *ACM VLDB* (págs. 918-929). Seoul, Korea: ACM.

Babcock, B., & Chaudhuri, S. (2005). Towards a Robust Query Optimizer: A Principled and Practical Approach. *ACM SIGMOD* (págs. 119-130). Maryland, USA: ACM.

Babcock, B., Babu, S., Datar, M., Motwani, R., & Thomas, D. (2004). Operator Scheduling in Data Stream Systems. *VLDB Journal , 13* (4), 333-353.

Babcock, B., Datar, M., & Motwani, R. (2004). Load Shedding for Aggregation Queries over Data Streams. *IEEE International Conference on Data Engineering* (págs. 350-361). Boston, USA.: IEEE.

Babu, S. &. (2001). Continuous Queries over Data Streams. *ACM SIGMOD Record* , 109-120.

Babu, S., & Duan, S. (2007). Processing Forecasting Queries. *ACM SIGMOD* (págs. 711-722). Beijing, China: ACM.

Babu, S., & Widom, J. (2001). Continuous Queries over Data Streams. *ACM SIGMOD Record* , 109-120.

Babu, S., Bond, C., Chandramouli, B., & Yang, J. (2007). Query Suspend and Resume. *ACM SIGMOD* (págs. 557-568). Beijing, China: ACM.

Basili, V., Caldiera, G., & Rombach, D. (1994). The Goal Question Metric Approach. En *Encyclopedia of Software Engineering.* Wiley.

Becker, P., & Olsina, L. (2010). Towards Support Processes for Web Projects. *International Conference on Web Engineering (ICWE)* (págs. 102-113). Vienna, Austria: Springer.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New Ensemble Methods For Evolving Data Streams. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (págs. 139-148). Paris, France: ACM.

Botan, I., Alonso, G., Fischer, P., Kossmann, D., & Tatbul, N. (2009). Flexible and Scalable Storage Management for Data-Intensive Stream Processing. *Extending Database Technology (EDBT)* (págs. 934-945). Saint Petersburg, Russia: ACM.

Botan, I., Cho, Y., Derakhshan, R., Lindar, N., Gupta, A., Haas, L., y otros. (2010). A Demonstration of the MaxStream Federated Stream Processing System. *International Conference on Data Engineering (ICDE)* (págs. 1093-1096). California, USA: IEEE.

Box, G., & Jenkins, M. (1991). *Time Series Analysis.* Prentice Hall.

Chakravarthy, S., & Jiang, Q. (2009). *Stream Data Processing: A Quality of Service Perspective.* Springer.

Chakravarthy, S., & Jiang, Q. (2009). *Stream Data Processing: A Quality of Service Perspective.* Springer.

Chakravarti, L. &. (1967). *Handbook of Methods Applied Statistics, Volume I.* John Wiley and Sons.

Chakravarti, Laha, & Roy. (1967). *Handbook of Methods Applied Statistics, Volume I.* John Wiley and Sons.

Chaudhry, N., Shaw, K., & Abdelguerfi, M. (Edits.). (2005). *Stream Data Management.* Springer.

Congreso de la República Argentina. (1969 con modificaciones de las resoluciones 84/09 y 709/09 de la Secretaría de Políticas, Regulación e Institutos y SAGPyA respectivamente). Capítulo 3: De los Productos Alimenticios, Condiciones Generales. En *Código Alimentario Argentino.* Argentina: Congreso de la República Argentina.

Dey, A. (2001). Understanding and Using Context. *Journal of Personal and Ubiquitous Computing , 5* (1), 4-7.

Diao, Y., Liu, A., Peng, L., & Sutton, C. (2009). Capturing Data Uncertainty in High-Volume Stream Processing. *Biennial Conference on Innovative Data Systems Research (CIDR).* California, USA.

Diván, M. (2006). *Construcción de un modelo de centro de distribución celular para acopio de miel en la cooperativa de Doblas. Tesis de Maestría en Administración de Negocios: UTN-FRC* (Primera ed.). (Divsar, Ed.) Santa Rosa, Argentina: Divsar.

Diván, M. (2005). *Introducción a la Tecnología de la Información para Profesionales de Ciencias Económicas.* Santa Rosa, La Pampa: Facultad de Ciencias Económicas y Jurídicas (UNLPAM).

Diván, M. O. (2009). Enfoque Integrado para el Procesamiento de Flujos de Datos: Un Escenario de Uso. *Cibse*, (págs. 374-387).

Diván, M. O. (2009). Especificando Fuentes de Datos en el Esquema Integrado de Procesamiento de Flujos. *CACIC.* San Salvador de Jujuy: Universidad Nacional de Jujuy.

Diván, M., & Olsina, L. (2009). Enfoque Integrado para el Procesamiento de Flujos de Datos: Un Escenario de Uso. *Cibse*, (págs. 374-387). Medellín, Colombia.

Diván, M., & Olsina, L. (2009). Especificando Fuentes de Datos en el Esquema Integrado de Procesamiento de Flujos. *CACIC.* San Salvador de Jujuy: Universidad Nacional de Jujuy.

Diván, M., Molina, H., & Olsina, L. (2008). Hacia un Modelo Integrado de Procesamiento de Flujos de Datos. *Congreso Argentino de Ciencias de la Computación (CACIC). Workshop WISBD.* Chilecito, La Rioja, Argentina.: Red UNCI.

Dolin, R., Alschuler, L., Beebe, C., Biron, P., Lee Boyer, S., Essin, D., y otros. (2001). The Practice of Informatics. Review: The HL7 Clinical Document Architecture. *Journal of the American Medical Informatics Association (JAMIA)*, *8*, págs. 552-569.

Domingos, P. (2000). A Unified Bias-Variance Descomposition and its Applications. *International Conference on Machine Learning (ICML)*, (págs. 231-238). California, USA.

Domingos, P., & Hulten, G. (2000). Mining High-Speed Data Streams. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (págs. 71-80). Boston, USA.: ACM.

Duin, R., Tortorella, F., & Marrocco, C. (2008). Maximizing the area under the ROC curve by pairwise feature combination. *Pattern Recognition , 41* (6), 1961-1974.

Fan, W., Chu, F., Wang, H., & Yu, P. (2002). Pruning and Dynamic Scheduling of Cost-Sensitive Ensembles. *National Conference on Artificial Intelligence* (págs. 146-151). Alberta, Canada.: ACM.

Foundation, R. S. (2010). *R Software.* Vienna, Austria: The R Foundation for Statistical Computing.

Frank, E., & Witten, I. (2005). *Data Mining. Practical Machine Learning Tools and Techniques.* Morgan Kaufmann.

Gama, J., Rocha, R., & Medas, P. (2003). Accurate Decision Tree for Mining High-Speed Data Streams. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (págs. 523-528). Washington, USA.: ACM.

Getta, L., & Vossough, E. (2003). Optimization of Data Stream Processing. *ACM SIGMOD Record , 33* (3), 34-39.

Golab, L., & Özsu, M. (2003). *Data Stream Management Issues –A Survey.* Technical Report, School of Computer Science, University of Waterloo.

Han, J., & Lamber, M. (2001). *Data Mining. Concepts and Techniques.* San Francisco, USA: Morgan Kaufmann.

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining Time-Changing Data Streams. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (págs. 97-106). California, USA.: ACM.

Hwang, J., Balazinska, M., Rasin, A., Çetintemel, U., Stonebraker, M., & Zdonik, S. (2005). High Availability Algorithms for Distributed Stream Processing. *IEEE International Conference on Data Engineering (ICDE)*, (págs. 779-790). Tokio, Japón.

Jahnke, J. (2005). Toward Context-Aware Computing in Clinical Care. *Object-Oriented Programming, Systems, Language and Applications (OOPSLA).* California, USA.

Jin, R., & Agrawal, G. (2003). Efficient Decisión Tree Construction on Streaming Data. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (págs. 571-576). Washington, USA.: ACM.

Johnson, D. (1998). *Métodos Multivariados Aplicados al Análisis de Datos.* Thomson Editores.

Johnson, D. (2000). *Métodos Multivariados Aplicados al Análisis de datos.* México: Thomson Editores.

Kaplan, R., & Norton, D. (2000). *Cómo Utilizar el Cuadro de Mando Integral.* Buenos Aires, Argentina: Editorial Gestión 2000.

Kaplan, R., & Norton, D. (1993). Putting the Balanced Scorecard to Work. *Harvard Business Review , 71* (5), 134-147.

Kaplan, R., & Norton, D. (1992). The Balanced Scorecard – Measures That Drive Performance. *Harvard Business Review , 70* (1), 71-79.

Kaplan, R., & Norton, D. (1996). Using the Balanced Scorecard as a Strategic Management System. *Harvard Business Review , 71* (1), 75-85.

Kirkby, R. (2007). *Improving Hoeffding Tree.* PhD Tesis of Computer Science, University of Waikato, Hamilton, New Zealand.

Koudas, N., & Srivastava, D. (2005). Data Stream Query Processing: A Tutorial. *21st International Conference on Data Engineering (ICDE)* (pág. 1145). Tokyo, Japan: ICDE.

Krishnamurthy, S. C. (2003). TelegraphCQ: An Architectural Status Report. *IEEE Data Engineering Bulletin , 26*.

Krishnamurthy, S., Chandrasekaran, S., Cooper, O., Deshpande, A., Franklin, M., Hellerstein, J., y otros. (2003). TelegraphCQ: An Architectural Status Report. *IEEE Data Engineering Bulletin , 26*.

Langsam, Y., Augenstein, M., & Tenenbaum, A. (1997). *Estructuras de Datos con C y C++* (Segunda ed.). Prentice Hall.

Li, M., Ganesan, D., & Shenoy, P. (2006). PRESTO: Feedback-driven Data Management in Sensor Networks. *Symposium on Networked Systems Design & Implementation.* California, USA.

Lindar, N., Güc, B., Lau, P., Özal, A., Soner, M., & Tatbul, N. (2009). DejaVu: Declarative Pattern Matching over Live and Archived Streams of Event. *ACM SIGMOD* (págs. 1023-1026). Rhode Island, USA: ACM.

Marrocco, C., Duin, R., & Tortorella, F. (2008). Maximizing the area under the ROC curve by pairwise feature combination. *ACM Pattern Recognition* , 1961-1974.

Medhat Gaber, M., Zaslavsky, A., & Krishnaswamy(b), S. (2005). Resource-aware Mining of Data Streams. *Universal Computer Science , 11* (8), 1440-1453.

Medhat Gaber, M., Zaslavsky, A., & Krishnaswamy, S. (2005). Mining Data Streams: A Review. *ACM SIGMOD Record , 34* (2), 18-26.

Molina, H., & Olsina, L. (2007). Towards the Support of Contextual Information to a Measurement and Evaluation Framework. *QUATIC* (pp. 154–163). Lisboa, Portugal: IEEE.

Namit, J., Gehrke, J., & Balakrishan, H. (2008). Towards a Streaming SQL Standard. *VLDB.* Auckland, New Zealand: ACM.

Olsina L, P. F. (2007). How to Measure and Evaluate Web Applications in a Consistent Way. En P. S. Rossi, *Ch. 13 in Web Engineering* (págs. 385–420). Springer.

Olsina, L., & Martín, M. (2004). Ontology for Software Metrics and Indicators. (R. Press, Ed.) *Web Engineering , 3* (4).

Olsina, L., Molina, H., & Papa, F. (2005). Organization-Oriented Measurement and Evaluation Framework for Software and Web Engineering Projects. *In proc. ICWE. 3579/2005*, págs. 351-361. Sydney: Springer Berlin / Heidelberg.

Olsina, L., Papa, F., & Molina, H. (2007). How to Measure and Evaluate Web Applications in a Consistent Way. En P. S. Rossi, *Ch. 13 in Web Engineering* (págs. 385–420). Springer.

Ort, E., & Mehta, B. (01 de 03 de 2003). *Oracle Sun Developer Network.* Recuperado el 22 de 03 de 2010, de Java Architecture for XML Binding: http://java.sun.com/developer/technicalArticles/WebServices/jaxb/

Oza, N., & Rusell, S. (2001). Online Bagging and Boosting. *International Workshop on Artificial Intelligence and Statistics* (págs. 105-112). Florida, USA: Morgan Kaufmann.

Pérez López, C. (2005). *Métodos Estadísticos Avanzados con SPSS.* Madrid: Thomson.

Petersa, R., Becketta, N., Forettea, F., Tuomilehtoa, J., Ritchiea, C., Waltona, I., y otros. (2009). Vascular risk factors and cognitive function among 3763 participants in the Hypertension in the Very Elderly Trial (HYVET): a cross-sectional analysis. *International Psychogeriatrics (Cambridge Journals) , 21* (2), 359-368.

R Software Foundation. (2010). *R Software.* Vienna, Austria: The R Foundation for Statistical Computing.

Rundensteiner, W., Mani, M., & Wei, M. (2008). Utility-driven Load Shedding for XML Stream Processing. *International World Wide Web* (págs. 855-864). Beijing, China: ACM.

Ryvkina, E., Maskey, A., Cherniack, M., & Zdonik, S. (2006). Revision Processing in a Stream Processing Engine: A High-Level Design. *IEEE International Conference on Data Engineering (ICDE)*, (pág. 141). Atlanta, USA.

Singh, S., Vajirkar, P., & Lee(b), Y. (2003). Context-aware data mining framework for wireless medical application. *Lectures Notes in Computer Science of Springer , 2736*, 381-391.

Singh, S., Vajirkar, P., & Lee, Y. (2003). Context-Based Data Mining using Ontologies. *Lecture Notes in Computer Science. 2813*, págs. 405-418. Springer Berlin / Heidelberg.

Srivastava, U., & Widom, J. (2004). Flexible Time Management in Data Stream Systems. *ACM PODS (Principles of Database Systems)*, (págs. 263-274). Paris, Francia.

Stamou, G., & Wallace, M. (2002). Towards a Context Aware Mining of User Interests for Consumption of Multimedia Documents. *IEEE International Conference on Multimedia & Expo. 1*, págs. 733-736. Lausanne, Switzerland: IEEE.

Stephens, M. (1974). EDF Statistics for Goodness of Fit and Some Comparisons. *Journal of the American Statistical Association* , 730-737.

Street, W., & Kim, Y. (2001). A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (págs. 377-382). California, USA.: ACM.

Tatbul, N. (2010). Streaming Data Integration: Challenges and Opportunities. *ICDE. International Workshop on New Trends in Information Integration (NTII)* (pág. 155). California, USA: IEEE.

Tatbul, N., & Zdonik, S. (2006). Window-aware Load Shedding for Aggregation Queries over Data Streams. *ACM VLDB*, (págs. 799-810). Seoul, Korea.

The Stream Group. (2003). *STREAM: The Stanford Stream Data Manager.* Stanford.

Toman, D. (2009). Data Expiration and Aggregate Queries. *Alberto Mendelzon International Workshop on Foundations of Data Management (AMW).* Arequipa, Peru.

Tranh, T., Sutton, C., Cocci, R., Nie, Y., Diao, Y., & Shenoy, P. (2009). Probabilistic Inference over RFID Streams in Mobile Environments. *International Conference on Data Engineering (ICDE)* (págs. 1096-1107). Shanghai, China: IEEE.

Wang, H., Fan, W., Yu, P., & Han, J. (2003). Mining Concept-Drifting Data Streams using Ensemble Classifiers. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (págs. 226-235). Washington, USA.: ACM.

Wei, M. R. (2008). Utility-driven Load Shedding for XML Stream Processing. *International World Wide Web* (págs. 855-864). Beijing, China: ACM.