Especificando Fuentes de Datos en el Esquema Integrado de Procesamiento de Flujos

Mario Diván[1,2] , Luis Olsina[2]

[1] Facultad de Cs. Económicas y Jurídicas, UNLPam, Santa Rosa, La Pampa, Argentina
[2] Facultad de Ingeniería, UNLPam, General Pico, La Pampa, Argentina
{mjdivan, olsinal}@ing.unlpam.edu.ar

Abstract. El presente trabajo especifica el modelo de objetos asociado a la funcionalidad de configuración y transmisión como parte del enfoque integrado de procesamiento de flujos de datos. Se discute la necesidad de formalizar los objetos que intervienen en el acople de las fuentes de datos en el marco integrado de procesamiento, como así también los objetos cuya responsabilidad será la de generar los flujos CINCAMI/MIS a transmitir, en base a las mediciones y datos contextuales arribados desde dichas fuentes. Se ilustra la funcionalidad de los distintos objetos con un escenario de aplicación para la problemática de pacientes trasplantados ambulatorios.

1. Introducción

El enfoque integrado de procesamiento de flujos (EIPF) de datos surge como respuesta a la necesidad de abordar aplicaciones que requieren un procesamiento al vuelo de mediciones y datos contextuales, con el fin de permitir una corrección y análisis de datos tendientes a incrementar la consistencia y precisión en el proceso de toma de decisión. Desde el punto de vista de las mediciones y su contexto, el EIPF se sustenta en C-INCAMI (*Context – Information Need, Concept Model, Attribute, Metric and Indicator*) el cual establece una ontología y marco conceptual que incluyen los conceptos y relaciones necesarias para especificar datos y metadatos en términos de cualquier proyecto de medición y evaluación [7, 8]. El presente trabajo presenta el modelo de objetos asociado a las funcionalidades de configuración y transmisión de las mediciones, pertenecientes al proceso de recolección y análisis del EIPF de datos propuesto [5]. Los objetos intervinientes en la función de configuración, tendrán por finalidad permitir la integración de una fuente de datos al esquema de procesamiento, evaluar la pertinencia de la fuente para con la/s métrica/s que desea implementar y definir grupos de seguimiento para los casos en que se requieran. Los objetos intervinientes en la función de transmisión, serán responsables desde la transformación de las mediciones y datos contextuales en mensajes CINCAMI/MIS (*Measurement Interchange Schema*) [5] hasta la transmisión efectiva de dicho flujo de datos.

Este artículo se organiza en seis secciones. La sección 2 resume el subconjunto de clases asociadas al marco conceptual de medición y evaluación C-INCAMI como así también, una vista del esquema integrado de procesamiento de flujos. La sección 3

presenta la funcionalidad que deberá implementarse para integrar una fuente de datos al EIPF, especifica los objetos que intervienen en la configuración, como así también aquellos objetos que intervienen en la transmisión del flujo. La sección 4 plantea un caso de aplicación del modelo integrado empleando las funciones de configuración y transmisión, para el ente paciente trasplantado ambulatorio. La sección 5 expone los trabajos relacionados y por último, la sección 6 delinea conclusiones y trabajos a futuro.

2. C-INCAMI y Procesamiento de Flujos de Datos: Enfoque Integrado

El marco C-INCAMI (ver detalles en [7, 8]) es un marco conceptual que define los módulos y conceptos que intervienen en el área de medición y evaluación para proyectos de software. Se basa en un enfoque en el cual la especificación de requerimientos, la medición y evaluación de entidades y ulterior interpretación de resultados están orientadas a satisfacer una necesidad de información en particular. Esta sección esboza el subconjunto de clases C-INCAMI asociadas a métrica (Fig. 1) y muestra el EIPF de datos.

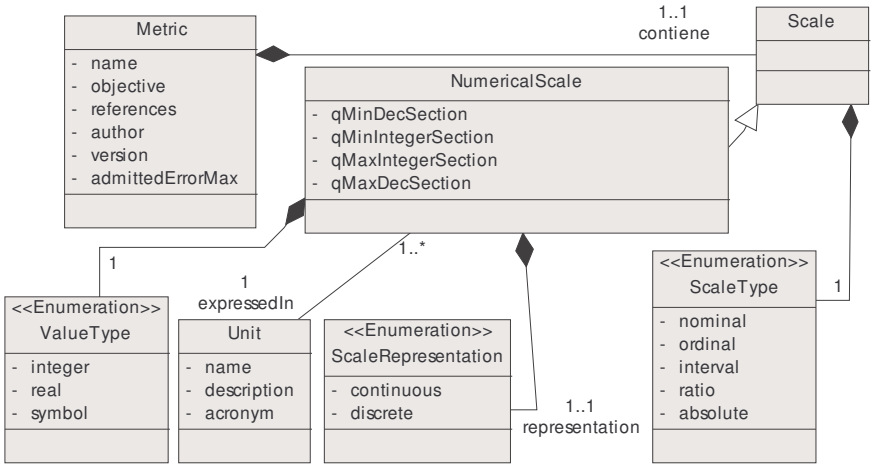


Figura 1. Principales clases de C-INCAMI para el módulo de medición centrado en métrica.

En la Fig. 1 se presenta una variación con respecto a C-INCAMI para *Metric* y *Scale*. La clase *Metric* ha desglosado su atributo *accuracy* en los atributos *qMinDecSection*, *qMinIntegerSection*, *qMaxIntegerSection* y *qMaxDecSection* ahora asociados a la clase *NumericalScale,* con la intención de definir requerimientos de mínima y máxima en términos de precisión decimal y entera para con las fuentes de datos. Adicionalmente, la clase *Metric* define su error máximo permitido en un instrumento de medición a través del atributo *admittedErrorMax*.

El EIPF de datos se sustenta en C-INCAMI a efectos de incorporar metadatos de

medición en forma conjunta con los datos, de modo de permitir una corrección y análisis consistente de datos, tendientes a incrementar la precisión en el proceso de toma de decisión basado además en el contexto de la medición. Básicamente éste consiste de tres procesos con sus funciones específicas:

- *Proceso de Recolección y Adaptación:* Permite acoplar a través del *Measurement Adapter* (MA) una o más fuentes de datos para el aprovisionamiento de datos y metadatos en forma conjunta hacia un procesador central. Dicho procesador central, denominado función de reunión $F^{rt}(d_{si})$, informa sobre el estado de procesamiento a los MA y recolecta las mediciones provenientes de los MA.
- *Proceso de Corrección y Análisis:* En función de los datos y metadatos arribados desde las fuentes de datos a través de los MA, este proceso permite dar consistencia al orden de procesamiento como así también suavizar los datos a través de análisis de outliers, ruido o ausencia de valor.
- *Proceso de Toma de Decisión:* Genera y actualiza incrementalmente un modelo de clasificación basado en árbol que, a partir de propiedades contextuales junto a datos de medición, permitiría mejorar la precisión en la toma de decisiones a los efectos de promover acciones proactivas.

3. Especificando la Fuente de Datos en el Enfoque Integrado

Toda fuente de datos que deba intervenir en el EIPF de datos contará con un objeto, posiblemente propietario, que se encargue de implementar la interface denominada *DataSource*. Se dice que posiblemente sea propietario, dado que muchos dispositivos no cuentan con APIs públicas de acceso a datos de precisión, medición, entre otras. A los efectos de facilitar la integración de dispositivos propietarios o no dentro del enfoque integrado, sin que ello implique vulnerar la tecnología embebida en el mismo, es que se plantea la interface *DataSource*. Esta interface tiene por finalidad definir una serie de funcionalidades que debe satisfacer la clase que la implemente, para interactuar con el esquema de procesamiento. Las principales funcionalidades son detalladas en la tabla 1.

Los grupos de seguimiento son reuniones lógicas de un conjunto de fuentes de datos que solicitan al procesador de flujos ser monitoreados en términos del grupo y no en general con respecto a las restantes fuentes de datos.

Una vez que se ha implementado la interface *DataSource* es necesario configurarla dentro del esquema de procesamiento. En este sentido, deberá implementarse según el rol a desempeñar, la interface *configurationServicesConsumer* para el consumidor y la interface *configurationServicesProvider* para el proveedor. Esto último permitirá establecer a la fuente de datos, a través de los servicios de configuración, las métricas que desea implementar y el grupo de seguimiento en el que desea participar o desvincularse.

Tabla 1. Principales funcionalidades de la interface 'DataSource'

Responsabilidad	**Descripción**
getIDGlobal	Retorna un identificador global único de la fuente de datos
isUtilizaBuffer	Indica si la fuente de datos emplea buffer para almacenar las mediciones en forma previa a su transmisión
evaluarPrecisionRequerida	Evalúa si la fuente de datos puede implementar las métricas indicadas como parámetro de acuerdo a las restricciones de precisión impuestas en las mismas dentro del proyecto.
evaluarErrorMaxTolerado	Evalúa si la fuente de datos posee un error inferior o igual al tolerado por cada métrica que desea implementar.
setMetrics	Asocia el conjunto de métricas que implementará la fuente de datos. Es responsabilidad del objeto que implementa la interface implementar la correspondencia entre la métrica y la medición que realice el dispositivo.
getMeasurements	Retorna mediciones asociadas a las métricas que implementa
isReadyToTrx	Retorna TRUE o FALSE de acuerdo a si la fuente de datos se encuentra con la medición lista para ser enviada al objeto que implementa la interface DataSource.
getTraceGroup	Retorna un obj. TraceGroup si la fuente de datos fueasociada a un grupo de seguimiento, <Nulo> en caso contrario

Tabla 2. Principales funcionalidades de la interface 'configurationServices'

Responsabilidad	**Descripción**
getActiveProjects	Retorna información de los proyectos de medición activos
getCalculableConcept	Retorna los conceptos calculables para un proyecto dado
getEntities	Retorna entidades que intervienen en proyecto indicado
getContexts	Retorna contextos definidos en el proyecto de medición indicado
getAttributes	Retorna el conjunto de atributos para una entidad dada
getContextProperties	Retorna las propiedades de contexto para un contexto dado
getMetrics	Retorna el conjunto de métricas asociadas a un atributo
getTraceGroupsByName	Retorna los grupos de seguimiento cuyo nombre contiene la expresión indicada como parámetro

En la tabla 2, se describen las principales responsabilidades involucradas en cada una de las funciones a implementar a partir de la interface *configurationServices*. Además, la interface *configurationServicesProvider* tendrá por objeto resolver cómo aprovisionar el servicio ante una demanda dada y la interface *configurationServiceConsumer* cómo acceder y consumir el servicio. La Fig. 2 muestra al gestor de fuentes de datos *DataSourceManager* (DSM), que es la clase que implementa a las interfaces de configuración desde el punto de vista del consumidor. Esta permitie vincular a las fuentes de datos dentro del esquema de procesamiento de flujos. La idea central del DSM, es que permita la registración de una fuente de datos con respecto a las métricas que desea implementar. A partir de esto, que sea

responsable de obtener las diferentes mediciones de las métricas por cada fuente de datos; indicar que la fuente de datos puede discontinuar el proceso de medición vinculado a alguna métrica con la que estaba asociada y solicitar que la fuente cree o sea incorporada a un grupo de seguimiento.

Configuradas las fuentes de datos y sincronizadas con el DSM para la obtención de las mediciones, resta la transmisión de las mismas hacia la *función de reunión.* Para este fin y en forma análoga a las interfaces de configuración, se presenta la interface base *transmitionService* la cual se especializa de acuerdo a su rol, consumidor o proveedor. La interface *transmitionServiceProvider* provee la funcionalidad necesaria para receptar los flujos y tratar con la problemática de desborde. Entendiéndose por desborde a la situación en que la tasa de arribo de datos supera a la del procesamiento en la función de reunión y la cola de servicio de la función de reunión es insuficiente para almacenar la totalidad de los datos no procesados actuales más aquellos que arriban. En la tabla 3, se describe el objetivo de cada método de la interface *transmitionServiceProvider*.

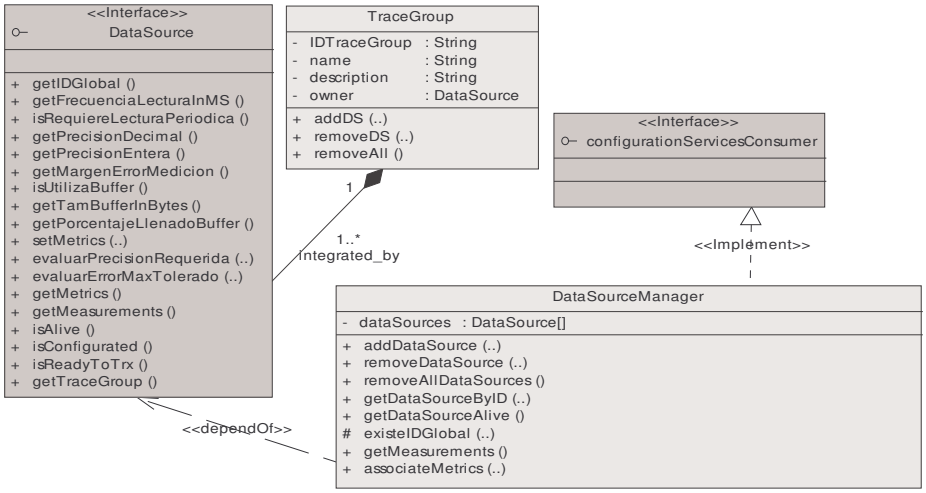


Figura 2. Principales dependencias y asociaciones de la clase DataSourceManager

Tabla 3. Funcionalidad a implementar para la interface transmitionServiceProvider

Responsabilidad	**Descripción**
isReadyToReceive	Indica si la clase que implementa la interface se encuentra en condiciones de recibir nuevos datos
Receive	Efectúa la recepción de los datos bajo el formato CINCAMI / MIS
isLoadSheddingEnabled	Indica si la clase que implementa la interface tiene activo algún algoritmo de descarte selectivo de datos no procesados en la cola de servicio (Load Shedding) ante situaciones de desborde [4]

Desde el punto de vista del consumidor del servicio de transmisión, es importante

diferenciar el buffer que puede poseer la clase que implementa la interface *transmitionServiceConsumer* del buffer que pueden o no disponer las fuentes de datos. Mientras que las fuentes de datos pueden implementar buffers por cuestiones particulares al dispositivo de medición, la clase que implemente la interface puede emplearlo a los efectos de optimizar el volumen de datos a transmitir, teniendo en cuenta el tiempo de arranque (start-up) de una transmisión versus el tiempo de transmisión de cada unidad de dato [13].

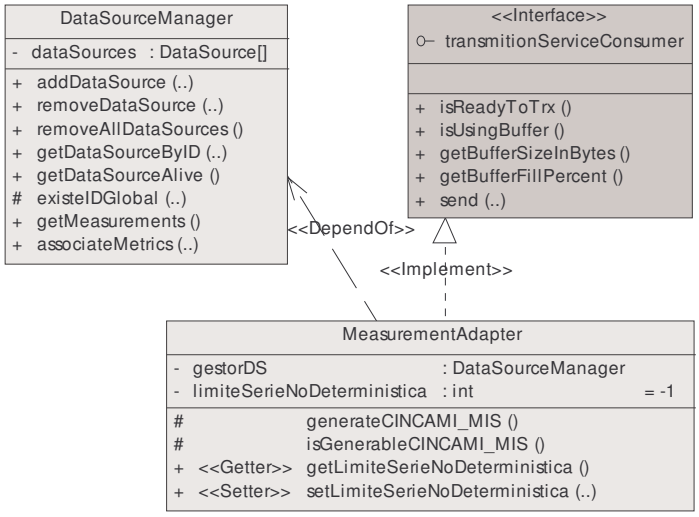


Figura 3. MeasurementAdapter. Dependencia e implementación de interface.

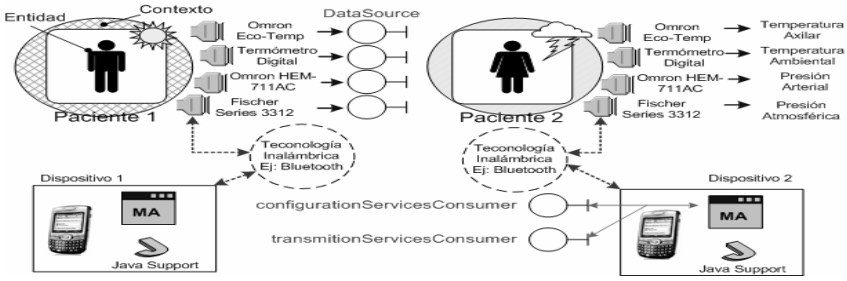
La clase *MeasurementAdapter* (MA) representa la funcionalidad del adaptador de mediciones dentro del EIPF. Esta solicitará al DSM las mediciones de cada una de las métricas vinculadas a las fuentes de datos registradas, transformará las mismas bajo el esquema CINCAMI/MIS, para luego, a través de la implementación de la interface *transmitionServiceConsumer* enviarlas a la función de reunión. Ante el caso en que no exista un valor determinista para una medición dada [1] el MA puede configurar un volumen determinado de pares (valor, probabilidad) a transmitir. De este modo el MA permite abastecer a la función de reunión tanto de valores deterministas como así también de una distribución de probabilidad ante la imposibilidad de determinar un valor para la medición. La Fig. 3 expone la dependencia y realización de la clase *MeasurementAdapter* para con el gestor de fuentes de datos y la mencionada interface.

4. Un Caso de Aplicación

Hasta el momento se han analizado las principales interfaces y clases necesarias para incluir fuentes de datos dentro del esquema integrado de procesamiento de flujos de datos esbozado en la sección 2. Esta sección tiene por finalidad ilustrar cómo interactuan dichas clases e interfaces a los efectos de la transmisión de mediciones en

forma conjunta con sus metadatos y con qué finalidad lo hacen. Como prueba de conceptos y siguiendo con el caso expuesto en [5], supondremos que dos pacientes trasplantados serán dados de alta de la unidad médica y ésta, requiere que durante el desarrollo de su vida cotidiana, se les monitoree la presión y temperatura ambiental como así también la presión arterial y temperatura axilar. De este modo, el personal de la unidad médica primero configurará los sensores del paciente (fuentes de datos) con el dispositivo de transmisión para que el mismo quede en condiciones de transmisión continua de mediciones desde las fuentes configuradas.

Como puede apreciarse en la Fig. 4, cada paciente posee un contexto posiblemente diferente del otro y dispondrá de los dispositivos necesarios para medir cada uno de los parámetros de monitoreo propuestos por los especialistas en forma previa a su alta. Por ejemplo, el dispositivo *Omron HEM-711AC* [9] sirve para medir la presión arterial. Este dispositivo debe implementar la interface *DataSource* para poder interactuar en el EIPF. En forma análoga, cada dispositivo deberá implementar la interface *DataSource* a los efectos de poder interactuar con el *Measurement Adapter* (MA) propuesto (ver Fig. 2 y 3). Implementada la interface el personal técnico de la unidad médica brindará al paciente algún dispositivo con conectividad, a través del cual se podrá efectuar la configuración de las fuentes, receptar las mediciones desde las mismas y transmitirlas a la función de reunión. El dispositivo con conectividad, ejecutará el *Measurement Adapter* el cual implementa las interfaces *configuationServicesConsumer* y *transmitionServicesConsumer* permitiendo de este modo configurar y transmitir las mediciones provenientes desde las fuentes de datos.



Figigura 4. Esquema de implementación de las interfaces en el escenario de aplicación.

La Fig. 4 muestra una de las formas mediante la cual una fuente de datos puede conectarse con el MA y como éste, mediante un dispositivo celular con soporte a conectividad, interactúa con los servicios Web que implementan las interfaces *configurationServicesProvider* y *transmitionServicesProvider*. Dichos servicios Web residirían localmente en la unidad médica con servidores propios, los que constantemente recibirían las mediciones de sus pacientes bajo monitoreo. De este modo el *DataSourceManager* (ejecutado dentro del dispositivo) recolecta las mediciones desde las fuentes de datos que implementan la interface *DataSource*; éstas son a su vez transformadas al formato CINCAMI/MIS por el MA, el cual las transmitirá en forma conjunta con sus metadatos a la función de reunión, la cual despliega su funcionalidad a través de servicios Web.

Supongamos ahora que el técnico de la unidad médica desea configurar una fuente de datos, por lo que ingresa a la pantalla donde se detectan las fuentes de datos

(aquellas que implementan la interface *DataSource* –Fig. 5a-), selecciona una fuente de datos y presiona el botón "Next". La siguiente pantalla (Fig. 5b), solicita a la interface *configurationServiceProvider* (CSP) los proyectos activos, para lo que el técnico elige "*Monitoreo de Pacientes Trasplantados*". Al seleccionar un proyecto, puede solicitarse a la interface CSP que informe sobre las entidades vinculadas al mismo si se selecciona dicha opción, o bien, puede solicitarse los contextos definidos si se eligiera la segunda opción. El técnico ha seleccionado, según la pantalla 8b, la entidad "*Paciente Trasplantado Ambulatorio*" para lo que, hecho esto, se solicita a la interface CSP informe las métricas vinculadas a la entidad. De dichas métricas el técnico puede seleccionar una o más a los efectos de intentar asociar las mismas a la fuente, y luego presionar el botón "Registrar" para efectuar la solicitud a la interface CSP. En caso de que la evaluación de la fuente de datos para con la métrica que desea implementar sea compatible, se registrará la misma y se enviará un mensaje de retorno similar a "*Fuente Asociada Correctamente*", en caso contrario se informará el motivo de la no registración, por ejemplo: "*La precisión decimal de (1) Omron Eco-Temp es insuficiente para la métrica Valor de la Temperatura Axilar*". La operatoria de las restantes funcionalidades se lleva adelante en forma análoga a la expuesta y escapan al objeto de una descripción detallada del presente caso de aplicación.

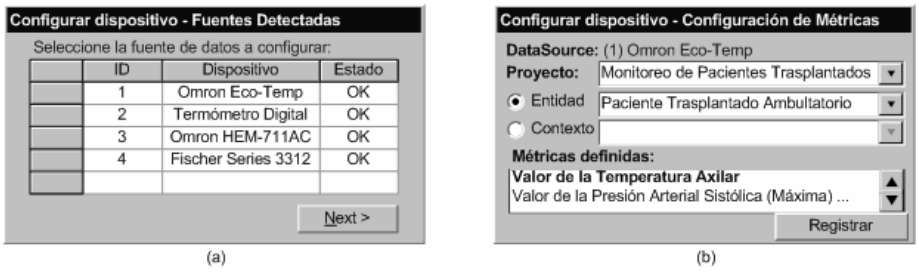


Figura 5. Ejemplo de configuración de una fuente de datos para con una métrica del proyecto

Finalmente, una vez que el técnico ha configurado cada una de las fuentes de datos, el paciente está en condiciones de desarrollar su vida cotidiana permitiendo al centro de salud o unidad médica monitorear automáticamente los parámetros definidos por sus especialistas e incluso, con la capacidad de actuar en forma preventiva a partir de que las mediciones dentro del esquema de procesamiento, abastecen un modelo de clasificación el cual, además de trabajar con los datos, procesa sus metadatos y los factores contextuales que lo influencian. Esto último, permite ajustar la precisión de los modelos de clasificación del esquema de procesamiento y posiblemente, mejorar el comportamiento proactivo ante situaciones de riesgo para el paciente, las cuales serían prevenidas mediante mecanismos de alarma disparadas por recomendación del modelo de clasificación a partir de las mediciones.

5. Trabajos Relacionados

En cuanto al procesamiento de flujos de datos existen trabajos relacionados tales como los llevados adelante en el proyecto STREAM de la Universidad de Stanford [2,

3,11], y el proyecto Aurora desarrollado en colaboración entre el MIT y las Universidades de Brown y Brandeis [6,10,12]. 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 flujos de datos (data streams). En este sentido los enfoques descriptos en ambos proyectos son útiles en las funciones de reunión de datos dentro del proceso de corrección y análisis del modelo integrado, como así también en el proceso de recolección aquí analizado, máxime en los aspectos de desborde de la capacidad de procesamiento donde debe efectuarse un descarte selectivo de los datos no procesados que aún residen en la cola de servicio (load shedding) [4, 3].

Si bien existen enfoques puntuales en las áreas específicas como los mencionados, nuestra propuesta incorpora e integra a la medición un conjunto de metadatos sustentados en la base ontológica definida en el marco C-INCAMI [7, 8]. Estos metadatos permiten incorporar información precisa sobre las definiciones de métricas, tipos de escala, unidades, métodos, entre otros conceptos, que agregan valor al dato, permitiendo a la función de reunión discernir claramente su significado, ayudando notoriamente a que el proceso de corrección y análisis sea efectuado de un modo consistente. El proceso de recolección claramente puede discernir, a partir de la configuración inicial, la procedencia de las mediciones como así también paralelizar el monitoreo de las mismas en base a grupos de seguimiento.

Dado que existen piezas de software propietarias con acceso a dispositivos de medición, como se ha esquematizado en el caso de aplicación, nuestro aporte en tal sentido es que cualquier fuente con la capacidad de implementar una interfaz pública (*DataSource*) pueda participar del esquema de procesamiento de flujos haciéndolo de este modo lo más participativo posible. El hecho de incorporar consistencia de datos y metadatos desde el propio proceso de recolección, evaluando por ejemplo la compatibilidad de la fuente con respecto a la métrica que desea implementar, repercutirá necesariamente y positivamente en la función de decisión $G(d^t_{si})$ y en la función de sustitución/ajuste $G'(d^t_{si})$ lo que, para diferentes condiciones contextuales, produciría modelos de análisis y toma de decisión posiblemente más robustos y precisos.

6. Conclusión

El presente trabajo ha propuesto al marco conceptual C-INCAMI como un mecanismo útil para la formalización de proyectos de medición y evaluación con el fin de dar soporte al procesamiento automático de mediciones, planteado éste en términos del modelo de procesamiento integrado de flujos de datos. Éste último, se sustenta en la base ontológica de C-INCAMI, lo que permite incorporar significado a las mediciones y su contexto, con el objeto de lograr un proceso de corrección y análisis de datos más consistente. Dicha consistencia, permitiría que los procesos de toma de decisión se encuentren mejor ajustados al contexto de medición favoreciendo la precisión para distintas situaciones.

En particular, las funcionalidades de configuración y transmisión de las mediciones propuestas, disponen de metadatos claramente definidos, contribuyendo

esto a la capacidad de identificar el significado de los datos y su contexto asociado. El caso de aplicación expuesto, muestra cuán importante es identificar el metadato asociado a una medición, permitiendo esto enriquecer al EIPF dado que dispondrá de mayor información a los efectos de analizar la consistencia de la medición recibida. Esto potencialmente mejorará la actualización incremental del modelo de clasificación, quien en definitiva es responsable de la automatización de decisiones tales como "*Avisar al centro de salud para atender al paciente Juan Fernández el cual podría encontrar su salud en riesgo dado la evaluación de las mediciones arribadas a partir de los parámetros definidos por los especialistas*".

Por último, al presente estamos implementando las funcionalidades e interfaces de configuración y transmisión aquí abordadas con el fin de generar un prototipo capaz de recolectar las mediciones desde las fuentes de datos, incorporar los metadatos de medición y transmitirlos hacia la función de reunión bajo el esquema definido en CINCAMI/MIS. Como trabajo futuro inmediato se implementará la funcionalidad de la función de reunión y se incorporará al prototipo a los efectos de construir incrementalmente el modelo integrado de procesamiento de flujos de datos.

Referencias

1. Abajo Martínez, N.: ANN quality diagnostic models for packaging manufacturing: an industrial data mining case study. In proc. of 10th ACM SIGKDD. Seattle, USA (2004)
2. Babcock, B., Babu, S., Datar, M., Motwani, R. & Thomas, D. (2004). Operator Scheduling in Data Stream Systems. VLDB Journal, Vol. 13:4, pp. 333-353.
3. Babcock, B., Datar, M. & Motwani, R. Load Shedding for Aggregation Queries over Data Streams. In proc. of IEEE ICDE, Int'l Conference on Data Engineering. Boston, US. (2004).
4. Chaudhry, N., Shaw, K., Abdelguerfi, M. (Eds.): Stream Data Management. Springer (2005)
5. Diván, M., Olsina, L.: Enfoque integrado para el procesamiento de flujos de datos: Un escenario de uso. XII Conferencia Iberoamericana de Ingeniería de Requisitos y Ambientes de Software. Medellín, Colombia, pp. 374–387, (2009)
6. Hwang, J.-H., Balazinska, M., Rasin, A., Çetintemel, U., Stonebraker, M. and Zdonik, S.. High Availability Algorithms for Distributed Stream Processing. In proc. of the 21st IEEE ICDE. Tokio, Japón. (2005)
7. Molina H, Olsina L. Towards the Support of Contextual Information to a Measurement and Evaluation Framework. In 6th Int'l Conference on the Quality Information and Communications Technology (QUATIC), IEEE CS Press, pp.154–163. Lisbon, Pt. (2007)
8. Olsina L., Papa F., Molina H.: How to Measure and Evaluate Web Applications in a Consistent Way. In: Springer, HCI Series, Rossi, Pastor, Schwabe & Olsina (Eds.) Web Engineering: Modelling and Implementing Web Applications. pp. 385–420. (2007)
9. Omron. http://www.omronhealthcare.com/product/1119-186-blood-pressure-monitors-advanced-bp-monitor-hem-711ac (Último acceso: 7-Jul-09)
10. Ryvkina, E., Maskey, A. S., Cherniack, M. and Zdonik, S. Revision Processing in a Stream Processing Engine: A High-Level Design. In the 22nd IEEE ICDE. Atlanta, US. (2006).
11. Srivastava, U. and Widom, J. Flexible Time Management in Data Stream Systems. In proc. of ACM PODS (Principles of Database Systems), 2004. Paris, Francia. (2004).
12. Tatbul, N. and Zdonik, S. Window-aware Load Shedding for Aggregation Queries over Data Streams. In proc. of the 32nd ACM VLDB. Seoul, Korea. (2006).
13. Tinetti, Fernando; "Rendimiento de una red de interconexión de estaciones de trabajo"; Reporte técnico PP003-01, Laboratorio de investigación y desarrollo de software, Facultad de Informática, UNLP, La Plata, Argentina. (2001).