Esquemas de representación ontológica para la integración de datos en los sistemas de información de planta

Fernando Roda[1], Estanislao Musulin[1,3] and Marta Basualdo[1,2,3]

[1] CIFASIS-CONICET, S2000BTP, Rosario, Argentina
<roda,musulin,basualdo@cifasis.gov.ar>
[2] UTN-FRRo, Zeballos 1341, S2000BTP, Rosario, Argentina
[3] Dpto. Control, FCEIyA-UNR, Av. Pellegrini 250, S2000BTP, Rosario

Abstract. La complejidad de los procesos productivos sumada a la falta de integración y consistencia en los datos hacen que los Sistemas de Información de Planta (SIP) sean sumamente dependientes de los expertos del proceso. Por este contexto, ha surgido un interés en sistemas de integración de datos basados en conocimiento. A diferencia de otros autores, que proponen soluciones de *mediación semántica*, en este trabajo se búsca explotar las capacidades deductivas del razonador siguiendo un enfoque de integración *dirigido por el conocimiento* (*knowledge-driven approach*). Conceptos propios de la ingeniería de procesos han sido implementados con éxito haciendo uso de los estándares y tecnologías propuestas recientemente por World Wide Web Consortium ($W3C$) en la construcción de $SemanticWeb$. Con el objeto de demostrar la potencialidad y el alcance de los esquemas de representación propuestos, se realizaron pruebas de razonamiento sobre un ejemplo de aplicación industrial.

1 Introducción

La infraestructura de los SIPs es muy variada y en general depende de los recursos disponibles, del grado de integración vertical y del sector productivo al que pertenecen. Es habitual la convivencia de un gran número de aplicaciones que cubren jerárquicamente todo el espectro de funcionalidad de la gestión empresarial [1]. Esta arquitectura facilita una implementación incremental que acompaña el crecimiento de la empresa pero exhibe problemas como: acoplamiento, redundancia e inconsistencia de datos. De hecho, se hace imprescindible la colaboración de personal especializado con gran experiencia en las operaciones de planta para analizar permanentemente las variables del proceso e interpretar las salidas del sistema. Estos problemas incentivaron el estudio de diversas tecnologías de integración de datos. Entre las más empleadas, se encuentran las basadas en servidores que centralizan la información, como los Data Warehouse. Sin embargo, estas tecnologías tienen una escasa capacidad para gestionar la semántica de los procesos, permaneciendo la lógica del negocio embebida en los módulos de las aplicaciones.

En el campo de la ingeniería del conocimiento, una ontología se define como "una especificación explícita de una conceptualización" [2]. Para establecer una representación ontológica se debe declarar un conjunto de conceptos, propiedades y axiomas que se relacionan entre sí.

La integración de datos basada en ontologías involucra el uso de una base de conocimiento para combinar datos y/o información proveniente de fuentes heterogéneas [3]. En el contexto industrial, estos sistemas tienen por objeto: a) Mejorar el aprovechamiento de las información distribuida; b) Lograr una representación unificada y consistente de los recursos disponibles; c) Integrar los subsistemas operacionales bajo un esquema que de soporte a personal no experto; d) Obtener trazabilidad sobre los cambios en el diseño y/o en los parámetros de la planta.

Entre otros trabajos, pueden citarse el de Schlenoff et al. [4], que propone una ontología en dos niveles con términos de fabricación, basándose en el lenguaje de especificación de procesos (PSL) y el de Hu et al. [5] donde se presenta una plataforma de integración basada en ontologías, construida de acuerdo al estándar IEC 61346 que modela un sistema técnico mediante objetos, aspectos y estructuras. Sin embrago, estos trabajos quedan acotados a la representación de un vocabulario especializado que captura la sintaxis pero no la semántica del dominio. En algunos casos, estas prácticas han llevado al desarrollo de *pseudo-ontologías* que no explotan las ventajas de integración y reutilización de las bases de conocimiento.

En general, los trabajos asociados a esta temática se diferencian en el enfoque que emplean al aplicar ontologías en la integración de los datos: 1. En el enfoque de mediación semántica (Semantic Mediation approach) se procura el intercambio de información entre distintas aplicaciones usando un modelo integrado de conocimiento. 2. En el enfoque dirigido por el conocimiento (knowledge-driven approach), los esquemas y registros de datos son representados como instancias dentro de la propia ontología, proveyendo un sistema de conceptualización poderoso que aprovecha al máximo las capacidades deductivas del motor de inferencia.

En este trabajo se propone la incorporación de una capa semántica en la arquitectura de los SIPs. Ello involucra la creación de esquemas de representación ontológicos para conceptos propios de la ingeniería de procesos, pretendiendo que estos esquemas oficien como metacontenido de las bases de datos de los subsistemas operacionales (Fig. 1-a). Wiesner et al. [6] han hecho importantes avances en este sentido, proponiendo ontologías que capturan la semántica de los procesos químicos. La propuesta de Wiesner et al. sigue un enfoque de mediación semántica diseñado para razonar bajo la hipótesis de mundo abierto (open world assumption). Sin embargo, en [7] estos autores mostraron algunas dificultades para procesar eficientemente su ontología sobre los motores de inferencia disponibles. En el dominio industrial, donde aplican estos sistemas, la alta performance en el razonamiento y el procesamiento en tiempo real son requerimientos de diseño que no deben ser descuidados. Por tal motivo, este trabajo se enmarca en un ambicioso proyecto que sigue una estrategia de integracíon dirigida

por el conocimiento (knowledge-driven approach), que aprovecha eficientemente las capacidades de razonamiento para soportar las decisiones de supervisión y control. En este artículo se exponen los primeros avances de dicho proyecto, presentando la implementación de esquemas de representación para los equipos de planta, los sistemas de control y las variables del proceso. El razonador puede procesarlos rápidamente, soportando además la detección automática de fallas y la clasificación de eventos.

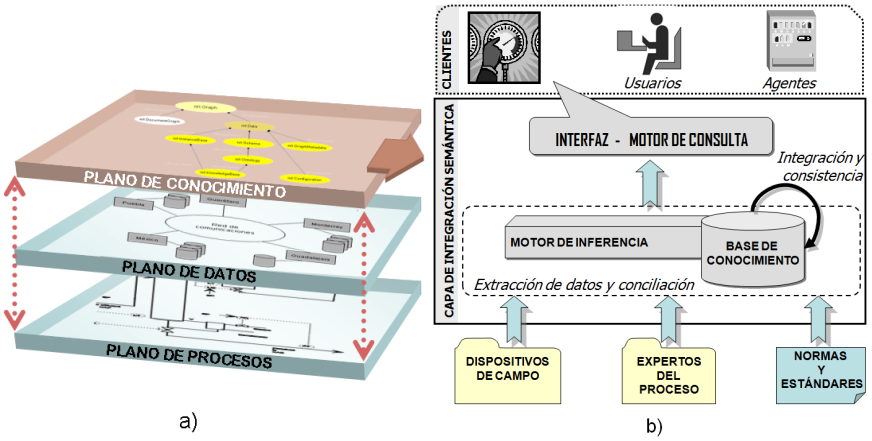


Fig. 1. Enfoque propuesto: a) Capa semántica en los SIPs b) Arquitectura de Integración

La Fig. 1-b) muestra un esquema general para la arquitectura de integración propuesta. En este sistema, además de la base de conocimiento y el motor de inferencia, se hace necesario implementar una estrategia para la carga y actualización de los datos generados por los dispositivos de campo, y una metodología para la adquisición del conocimiento experto. Cada vez que la base de conocimiento es actualizada, el razonador verifica la integridad y consistencia de los datos usando los axiomas y reglas almacenados en la propia ontología. Además, el sistema debe implementar interfaces de consulta para que sus clientes (usuarios u otros agentes autónomos) puedan extraer los hechos explícitos e implícitos almacenados.

El trabajo está organizado de la manera siguiente: en la Sec. 2.1 se presenta una jerarquía para la representación de la configuración de equipos. En las Secciones 2.2 y 2.3 se proponen esquemas de representación para conceptos de la configuración de los sistemas de control y la captura de eventos respectivamente. Por último, en la Sec. 3 se ilustran los resultados de la aplicación de este enfoque en un reactor CSTR que es controlado por un lazo en cascada.

2 Representación del conocimiento

Los esquemas de representación del conocimiento fueron formalizados y luego implementados utilizando las herramientas y estándares propuestos por W3C

en torno a la creación de *Semantic Web*. Esta utiliza ontologías para dotar de semántica al contenido de los sitios web tradicionales, permitiendo que el conocimiento allí almacenado sea compartido y reutilizado por muchas aplicaciones. Dicho movimiento ha impulsado el desarrollo de herramientas y lenguajes de especificación de ontologías que promete extenderse con éxito al dominio de las aplicaciones de negocios. En este sentido, la utilización de dichas herramientas en este trabajo tiene como propósito adicional su validación en el dominio de las aplicaciones industriales.

Las ontologías fueron implementadas utilizando el lenguaje $OWL2$ (Web Ontology Language). Además, se emplearon los entornos de desarrollo provistos por las aplicaciones *Protégé* y *Jena*. En la especificación de los esquemas propuestos, se utilizaron un total de 133 clases, 77 propiedades y 279 axiomas distribuidos en tres áreas de dominio (Equipos, Control y Eventos). También se han utilizado reglas $SWRL$ (Semantic Web Rule Language) que aumentan la capacidad deductiva del motor de inferencia. Las reglas SWRL son expresadas como cláusulas de Horn utilizando los términos de OWL.

2.1 Conceptualización de la Configuración de Equipos

Los conceptos asociados a los elementos físicos de la planta han sido representados como una taxonomía jerárquica de equipos, siguiendo las normas ISA-95 e ISA-S88. Muchos estándares que sirven como metadata han sido formalizados utilizando ontologías, como por ejemplo las normas ISO 199115 para información espacial [8], o la norma ISO10303-11 para la representación de datos de productos [9]. ISA-95 [10] define una jerarquía que abarca desde la empresa hasta las celdas de trabajo, pero no incorpora conceptos de control. Para procesos por lotes, existe una normativa similar de más bajo nivel, la norma ISA-S88 [11] que permite una especificación más detallada desde el punto de vista de los sistemas de control. En este contexto, es posible hasta cierto punto unir ambas especificaciones para tener una normativa completa. El inconveniente es la existencia de una diferencia notable entre el control de procesos por lotes y de procesos continuos. En los procesos por lotes, los procedimientos y equipos de control están altamente ligados a una unidad de producción en particular. Este no es siempre el caso en procesos continuos, donde los lazos de control pueden estar distribuidos a través de todo el proceso. Por este motivo, se siguió la especificación de las normas ISA-95/ISA-S88 hasta el nivel de equipo y se incorporaron los instrumentos que se enlazan a los sistemas de control (Fig. 2). Como se puede observar, todas las relaciones son de agregación/ensamble; de hecho para el caso de *Equipment*, la propiedad se vuelve recursiva, pudiendo un equipo componerse de otros. Las relaciones de este tipo no poseen una sintaxis diferencial en OWL y los razonadores no son capaces de realizar inferencias a través de ellas. Sin embargo, pueden ser implementadas con éxito definiendo una jerarquía de propiedades transitivas (disponibles en OWL 2). La Fig. 2-b muestra la solución propuesta para la relación *hasPhysicalElement*.

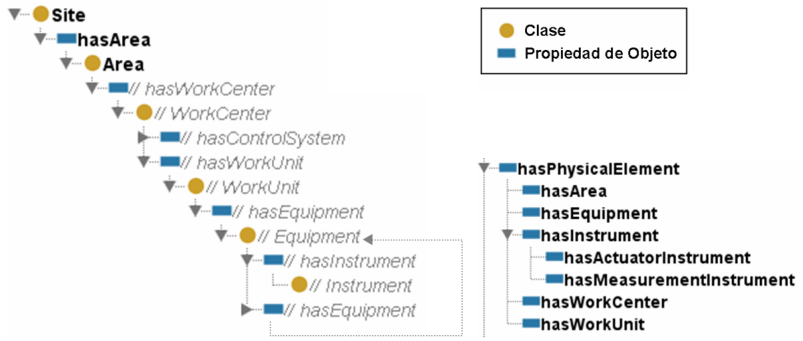


Fig. 2. Conceptualización de componentes físicos basado en ISA-S88/ISA-95 : a) Jerarquía de los equipos de planta b) Jerarquía de propiedades para hasPhysicalElement.

2.2 Conceptualización de Sistemas de Control

Los conceptos de control fueron organizados siguiendo una jerarquía de clases enlazada a los módulos de equipos y eventos (Fig. 3). En la cima de la jerarquía se encuentra la clase *ControlElement*, al cual agrupa todos los conceptos que configuran el sistema de control de la planta. Los sistemas de control (clase *ControlSystem*) han sido definidos como ensambles de lazos de control (clase *ControlLoop*).

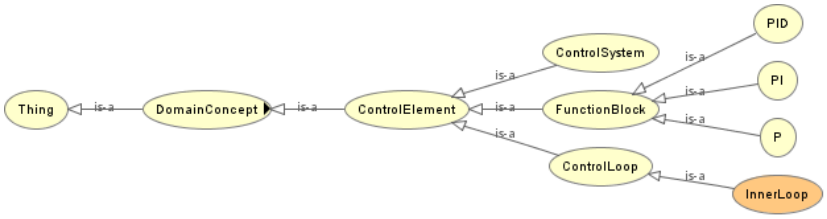


Fig. 3. Jerarquía de clases de elementos de control

En la Fig. 4-a se muestra el árbol de propiedades para un lazo de control clásico: este posee un bloque funcional (*FunctionBlock*), un valor de referencia (*SetPoint*) y algunos instrumentos de medición y actuación (las clases *MeasurementInstrument* y *ActuatorInstrument* respectivamente). Estos últimos son subclases de la clase *Instrument* y se relacionan con las variables medidas y manipuladas en cada lazo. Un actuador puede ser asociado a dos variables: *hasManipulatedProperty* corresponde a la variable que está siendo manipulada, mientras que *hasOutput* representa la magnitud que es realmente operada por el dispositivo para afectar a la primera. Aquí, tanto *hasManipulatedProperty* como

hasOutput y *hasMeasuredProperty*, han sido definidas como subpropiedades de *hasProperty*. El concepto *Functional Block* involucra a las funciones de control presentes en el lazo, las que son a su vez clasificadas en sub categorías como PI, PD o PID. Este esquema también admite el uso de bloques funcionales de múltiples entradas y salidas para representar sistemas de control más complejos.

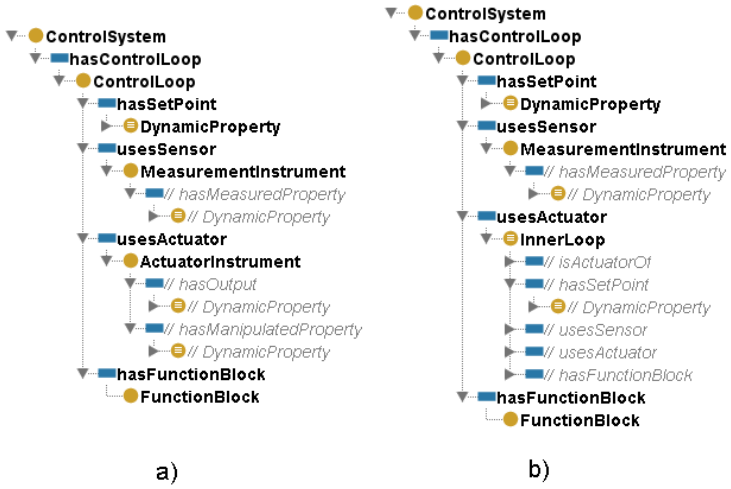


Fig. 4. Especificación de los lazos de control: a) Control clásico b) Control cascada

Para definir un lazo de control en cascada, la propiedad *usesActuator* es utilizada para establecer la conexión con el lazo de control interno (Fig. 4-b). *InnerLoop* es una *clase definida*, con la condición necesaria y suficiente: "*ControlLoop and (usedAsActuatorBy some ControlLoop)*". De esta forma, el razonador detecta automáticamente los lazos internos y los clasifica en la clase correspondiente.

Es importante destacar que se han utilizado características recientemente incluidas en *OWL2* para incrementar la capacidad deductiva del razonador. En efecto, la propiedad *hasControlSystem* y su inversa, *isControlSystemOf*, han sido implementadas como *Property Chain* (cadena de propiedades). Un axioma *Property Chain*, incluido en la relación entre dos individuos, por ejemplo A y B, expresa que estos estarán vinculados por dicha relación si A puede ser conectado con B a través de una cadena específica de propiedades.

Las siguientes cláusulas escritas en la sintaxis de *Turtle* para la propiedad *hasControlSystem* permiten que los sistemas de control queden automáticamente enlazados con los centros de trabajo gracias a la ubicación de los instrumentos involucrados.

```
:hasControlSystem rdf:type owl:ObjectProperty ;  owl:inverseOf :isControlSystemOf ;
                  owl:propertyChainAxiom
                  (:hasWorkUnit :hasEquipment :hasInstrument :UsedBy :isControlLoopOf).
```

2.3 Conceptualización de Eventos de Planta

Para poder obtener información de las propiedades dinámica de la planta, es de vital importancia establecer un esquema eficiente de gestión de eventos. Los datos relacionados a dichos eventos representan el mayor volumen de información que el sistema debe manipular. Por lo tanto, en el diseño de la ontología se buscó una representación adecuada que contenga los datos básicos para su identificación. La clase *Event* involucra a todos los eventos generados por los instrumentos, con sus valores, time-stamps y calidad de medición. Como se puede observar en la Fig. 5-a , nuevamente se hizo uso de un axioma *property chain* para especificar la asociación entre eventos e instrumentos a través de sus variables. La propiedad *isPropertyOf*, que aparece en dicha figura, ha sido definida como inversa de la propiedad *hasProperty*, citada en la sección anterior.

Para representar las variables y los parámetros del proceso se propone un esquema simple, conformado por una jerarquía de dos niveles que clasifica a los individuos de acuerdo a la naturaleza de la medición. La clase *Property* posee dos subclases: *ConstantProperty* y *DynamicProperty*. A su vez, esta última es especializada en categorías disjuntas como *Temperature*, *Pressure* o *Flow*; declaradas utilizando el axioma *AllDisjointClasses* de OWL2.

El concepto *Deviation* describe los patrones de comportamiento que cada variable puede adoptar y que en ciertos casos pueden ocasionar situaciones de riesgo. La Fig. 5-b muestra las propiedades de los desvíos. Un desvío queda identificado por la variable dinámica (*Parameter*) y por una palabra-guía o *Guide Word*. Se incluyen además atributos para indicar el límite superior e inferior que admite cada desvío. Las *Guide Words* son palabras reservadas utilizadas en el estudio de HAZOP. La clase *Guideword* posee una representación basada en el estándar IEC 61882:2002. Según esta norma, se establecen una serie de términos de propósito general (como ser *MORE, LESS o NONE*) que son utilizados para descubrir los posibles desvíos.

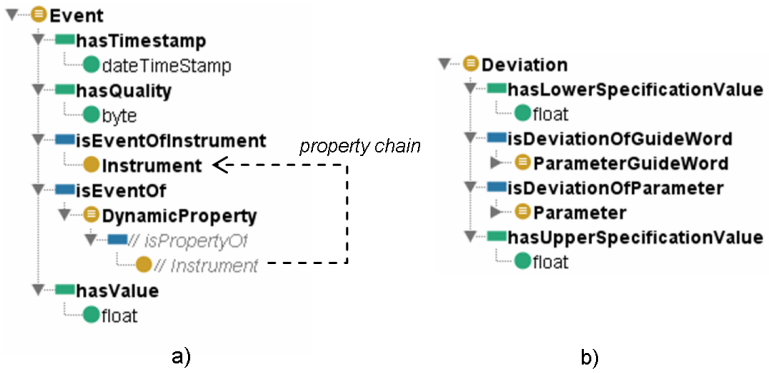


Fig. 5. Árboles de especificación de eventos (Event) y desvíos (Deviation)

Al analizar los tipos de magnitud que son manipuladas por el proceso, estas expresiones son refinadas para ajustarse a cada caso. Es por esto que cada miembro de *Guideword* fue clasificado con una estructura de herencia múltiple. Como ejemplo, considérese la siguiente especificación del concepto *Vacuum* (vacío).

```
:Vacuum rdf:type :None , :PressureGuideWord , owl:NamedIndividual .
```

De esta forma, *vacuum* queda definida como una Guide Word de *Presión* y a la vez, como una Guide Word de ausencia (*None*).

Además, en la conceptualización se buscó un esquema que le permita al razonador establecer automáticamente si un evento representa un desvío. En ese caso, un hecho implícito será agregado para vincular al evento y el desvío usando la propiedad de objeto *isRecognizedAsDeviation*. A continuación se incluye la regla SWRL que establece el valor de dicha propiedad de acuerdo a una simple verificación de límites.

```
hasPossibleDeviation(?e, ?g), hasLowerSpecificationValue(?g, ?lv),
 hasUpperSpecificationValue(?g, ?hv),hasValue(?e, ?v),greaterThanOrEqual(?v, ?lv),
 lessThanOrEqual(?v, ?hv) -> isRecognizedAsDeviation(?e, ?g)
```

En esta expresión, la variable *'?e'* será instanciada con un miembro de la clase *DynamicalProperty*, *'?g'* con una posible *Deviation* y *'?lv'*, *'?hv'* y *'?v'* con valores de punto flotante. La propiedad *hasPossibleDeviation* ha sido definida como inversa de *isDeviationOfParameter* mostrada en la Fig. 5-b. Con el fin de identificar estos eventos, se ha creado una subclase definida llamada *DeviatedEvent* con las siguientes condiciones suficientes y necesarias: ***Event and (isRecognizedAsDeviation some Deviation)***.

Por otro lado, para detectar qué desviaciones están involucradas en los eventos actuales, se ha creado una subclase de *Deviation*, llamada *CurrentDeviation*, con las siguientes restricciones de equivalencia: ***Deviation and (hasAssociatedEvent some DeviatedEvent)***. Aquí, la propiedad *hasAssociatedEvent* está definida como inversa de *isRecognizedAsDeviation*.

3 Ejemplo de Aplicación

Con el fin de demostrar la aplicabilidad de los esquemas propuestos, se ha creado e instanciado una base de conocimiento que describe el proceso de la Fig. 6. Este se compone de un reactor de mezcla completa de agitado continuo con camisa refrigerante (en ingles CSTR). Existe un lazo de control en cascada $LC1$ que regula el nivel del reactor $L1$ manipulando el set point $F.SP$ del lazo interno $LC2$. Este último controla el caudal de salida $F.PV$ mediante la válvula *Vf*.

En una aplicación real, la base de conocimiento deberá actualizarse con nuevas instancias transferidas desde los dispositivos y aplicaciones de la planta. Estas actualizaciones podrán ser ejecutadas en forma periódica o disparadas por eventos, dependiendo de la dinámica del proceso y el grado lo criticidad de sus operaciones.

Siguiendo las prácticas propuestas por $W3C$, las instancias fueron separadas de las representaciones conceptuales y distribuidas en dos archivos OWL de

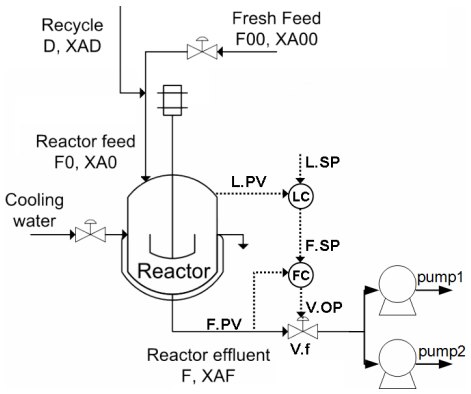


Fig. 6. Reactor CSTR

acuerdo a su tasa de actualización. Se identificaron dos grupos: A) Los datos más estables están relacionados con los conceptos de equipos y control. B) Por otro lado, los conceptos relacionados a las variables de proceso conforman la parte más dinámica de los datos. Esta distribución de instancias facilita el uso de estrategias de razonamiento incremental, que operan solo sobre los nuevos datos, aumentando notablemente la velocidad de procesamiento. Gracias a esta técnica y a una correcta implementación de la ontología, el razonamiento puede hacerse inmediatamente después de cada carga, generando hechos inferidos que en pocos segundos están disponibles para los clientes del sistema.

En este caso de estudio, se utilizó el razonador Pellet 2 ya que posee algunas características no ofrecidas por otro razonadores, como la posibilidad de operar sobre tipos de datos complejos y procesar reglas $SWRL$. Las pruebas de razonamiento fueron realizadas en una computadora personal con un procesador Intel I7 y 8 Gb de memoria RAM. En este entorno, Pellet demandó en promedio 3,1 s. en finalizar el procesamiento completo de la ontología. En la primera etapa del proceso, el razonador verifica la consistencia de la base de conocimiento utilizando métodos de resolución lógica que actúan sobre los axiomas almacenados. Durante las pruebas, Pellet pudo detectar que un sistema de control se encontraba incorrectamente asociado a un centro de trabajo utilizando la ubicación de los instrumentos de medición. En los sistemas de integración de datos, esta verificación es muy importante, ya que asegura la consistencia e integridad de los datos de fuentes heterogéneas.

La Fig. 7 muestra las deducciones arrojadas por el razonador en relación a un evento del sensor *Fmeter*. Se puede apreciar cómo el razonador detecta un desvío en la variable F.PV, para luego clasificar al evento $e1$ como *DeviatedEvent*.

En relación a la explotación de conocimiento, ésta puede ser implementada de tres formas distintas: navegación exploratoria, búsqueda (exploración + objetivo) o consultas. En el entorno de trabajo propuesto, las consultas a la ontología requieren de un lenguaje que reconozca la sintaxis de las sentencias RDF. Se ha utilizado el lenguaje de consulta $SPARQL$ ya que posee una rica sintaxis y

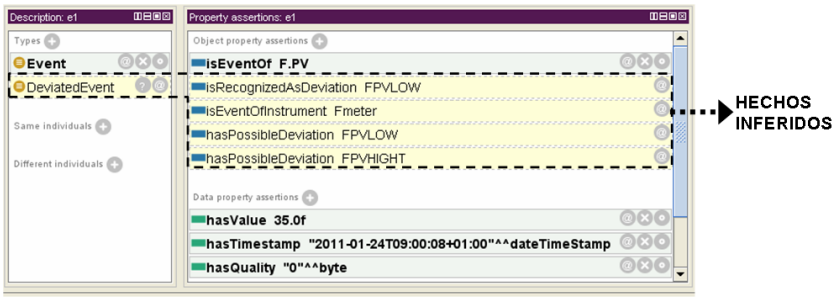
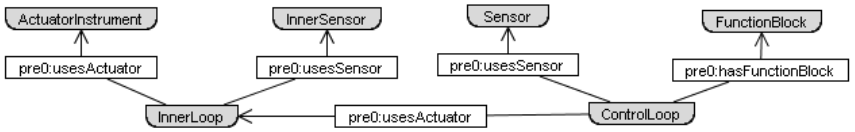


Fig. 7. Inferencia del razonador para un evento del sensor *Fmeter*

(a)



(b)

```
PREFIX onto:<http://www.semanticweb.org/PSOntology.owl>
SELECT ?FunctionBlock ?ControlLoop ?ActuatorInstrument ?Sensor ?InnerLoop ?InnerSensor
WHERE
  { ?InnerLoop onto:usesActuator ?ActuatorInstrument .
    ?ControlLoop  onto:usesActuator ?InnerLoop .
    ?InnerLoop  onto:usesSensor ?InnerSensor .
    ?ControlLoop  onto:hasFunctionBlock ?FunctionBlock ;
              onto:usesSensor ?Sensor . }
```

(c)

Results

?ControlLoop	?Sensor	?FunctionBlock	?InnerLoop	?InnerSensor	?ActuatorInstrument
pre0:LC1	pre0:Lrsensor	pre0:PID1	pre0:LC2	pre0:Fmeter	pre0:Vf

Fig. 8. Consulta de lazos cascada. (a) Patrón RDF. (b) consulta SPARQL. (c) Resultados arrojados por el motor

es implementado por una gran cantidad de motores de consulta. En este caso se utilizó el motor provisto por *OWL3 Query Tab* para consultas *SPARQL DL* que puede ser montado sobre *Protege* y *Pellet*. En él, las consultas *SPARQL DL* pueden ser expresadas como patrones gráficos *RDF* (fragmento de la red semántica) que el motor *equipara* con los axiomas almacenados e inferidos. A modo de ejemplo, en la Fig.8 se muestra una consulta para las configuraciones de control en cascada. El motor SPARQL resolvió satisfactoriamente la consulta, detectando el lazo de control inmediatamente.

4 Conclusiones y Trabajos Futuros

Con el objetivo de resolver los problemas que se plantean en la explotación de conocimiento de los SIPs, se han mostrado avances en torno al desarrollo de una arquitectura de integración de datos basada en conocimiento. Sobre las bases de esta propuesta, se ha presentado una ontología que respeta un enfoque de

integración *dirigido por el conocimiento*, aprovechando así las capacidades del razonador. Conceptos propios de la ingeniería de procesos han sido implementados con éxito mediante los estándares y tecnologías propuestos recientemente por $W3C$ en la construcción de $SemanticWeb$. Se ha mostrado como éstos son aplicados en la conceptualización de un proceso simple, compuesto por un reactor CSTR controlado por un lazo en cascada. Las pruebas de razonamiento fueron exitosas, arrojando tiempos de procesamiento aceptables y generando inferencias correctas. Además, las capacidades de razonamiento fueron aprovechadas tanto para la verificación de consistencia de la información, como para la clasificación de lazos de control y la detección de desvíos en las variables del proceso.

En trabajos futuros, se buscará ampliar el modelo de conocimiento con la conceptualización de patrones temporales que representen mejor la dinámica del sistema. Por otro lado, se avanzará en el diseño de la infraestructura y se propondrán métodos para implementar procesos de extracción, transformación y carga ($ETL's$) que mantengan actualizada la base de conocimiento.

References

1. Kalogeras, A., Gialelis, J., Alexakos, C., Georgoudakis, M., Koubias, S.: Vertical integration of enterprise industrial systems utilizing web services. IEEE Transactions On Industrial Informatics **2**(2) (2006) 120–128
2. Gruber, T.: A translation approach to portable ontology specifications. Knowledge acquisition **5**(2) (1993) 199–220
3. Wache, H., Vgele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., Hbner, S.: Ontology-based integration of information - a survey of existing approaches. (2001) 108–117
4. Schlenoff, C., Ivester, R., Knutilla, A.: A robust process ontology for manufacturing systems integration. In: Proceedings of 2nd International Conference on Engineering Design and Automation, Maui. (1998) 7–14
5. Hu, Z., Kruse, E., Draws, L.: Intelligent binding in the engineering of automation systems using ontology and web services. IEEE Trans. Syst., Man, Cybern. C **33**(3) (2003) 403–412
6. Wiesner, A., Morbach, J., Marquardt, J.: Information integration in chemical process engineering based on semantic technologies. Computers chem. Engng. **35** (2011) 692–708
7. Morbach, J., Wiesner, A., Marquardt, W.: Onto cape-a (re-) usable ontology for computer-aided process engineering. Computers and Chemical Engineering **33** (2009) 1546–1556
8. ISO: ISO 2004/TC211 Geographic Information Metadata Standard. (2004)
9. Zhao, W., Liu, J.: Owl/swrl representation methodology for express-driven product information model. Computers in Industry **59**(6) (2008) 580 – 589
10. ISA: ANSI/ISA-95.00.01 - 2000, Enterprise/Control System Integration Part 1: Models and Terminology (2000.)
11. ISA: ANSI/ISA-S88.01 - 1995, Batch Control, Part I: Models and Terminology. (1995)