UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

INTERPRETACIÓN DE ALTO NIVEL DE SECUENCIAS DE VIDEO CONDUCIDA POR ONTOLOGÍAS EN PROBLEMAS DE SEGURIDAD

Héctor Fernando Gómez Alvarado

Diploma de Estudios Avanzados

Ingeniero en Informática

UNED

Departamento de Inteligencia Artificial

E.T.S.I Informática

Director: Rafael Martínez Tomás

DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL

E.T.S.I Informática

UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

INTERPRETACIÓN DE ALTO NIVEL DE SECUENCIAS DE VIDEO CONDUCIDA POR ONTOLOGÍAS EN PROBLEMAS DE SEGURIDAD

Héctor Fernando Gómez Alvarado

Diploma de Estudios Avanzados

Ingeniero en Informática por la UTPL-Ecuador

Director: Rafael Martínez Tomás

A mi esposa Susana

AGRADECIMIENTOS

Dedico este trabajo a mi esposa Susana, ya que sin su comprensión y ayuda no hubiera sido posible el culminar este trabajo doctoral.

Agradezco a Rafael Martínez Tomás, cuyo apoyo en el desarrollo de esta tesis, ha sido fundamental, sus revisiones y correcciones, me han ayudado mucho en el quehacer investigativo.

Agradezco también a la Universidad Técnica Particular de Loja, y a Senecyt – Ecuador, quien me apoyado para poder realizar las pasantías doctorales en la UNED.

Resumen IX

En esta tesis doctoral, proponemos un marco conceptual, metodológico y tecnológico que permite modelar e inferir situaciones de alto nivel semántico. Se parte del diseño mismo de las situaciones de interés, en las cuales interviene el experto y el ingeniero del conocimiento. El experto por su lado es el encargado de exponer el detalle de la situación mientras que el ingeniero del conocimiento se encarga de trasladar ese detalle al modelo semántico. Tratamos también en esta investigación con casos en los cuales el experto no tiene claro el detalle, y proponemos ayudar con algoritmos de modelado automático del conocimiento, con el fin de extraer información de un conjunto de casos y obtener detalles que le ayuden al experto a clarificar una situación.

A partir de una situación clara, el experto empieza a modelarla de forma semántica. Para ello, cuenta con el lenguaje de modelado de eventos de vídeo *VERL* (Nevatia & Hobbs, 2004) el cual le permite ir componiendo las actividades y situaciones. En el marco esta composición es jerárquica, ya que manejamos la abstracción por niveles semánticos para modelar actividades tal como lo recomienda (Martinez-Tomas & Rivas-Casado, 2009). Para la composición jerárquica en el marco, se puede hacer uso de *Conceptual Human Activity Ontology* (CHAO), la misma que a más de tener sus propias definiciones semánticas, combina los conceptos y relaciones de *Semantic Sensor* (W3C, Semantic Sensor Network Ontology, 2011), *BuildingArchitecture* (Hois, Bhatt, & Kutz,

2009), *Time* (W3C, Time Ontology in OWL , 2006), con el fin de inyectar semántica al modelado de eventos, actividades y situaciones en *VERL*.

Una vez modelada la situación, se utiliza el módulo de exportación del marco con el fin de generar un modelo tipo *OWL* (Bechhofer, y otros, 2004), que pueda ser exportado y manejado en cualquier herramienta semántica. En la exportación se crean clases y axiomas que permitirán inferir las situaciones de interés. Con todos estos componentes el marco queda operativo en espera de eventos de entrada para empezar la inferencia por niveles semánticos.

El modelo *OWL*, es la entrada para el módulo de inferencia del marco. Aquí las reglas y razonadores hacen uso de los axiomas con el fin de ir componiendo los niveles semánticos. Para instanciar el modelo, se hace uso de módulo de fusión sensorial de *Horus* (Castillo, Fernández, & López, 2011). *Horus* es un marco preparado para inferir eventos de bajo nivel semántico a partir de señales multisensoriales, en especial desde señales de vídeo. Esto nos dio paso a que nosotros probemos las herramientas de nuestro marco con señales que provienen desde videocámaras y de sensores (movimiento, RFID). El marco infiere situaciones de alto nivel semántico, a partir de eventos de bajo nivel semántico provenientes de *Horus*, por lo que toma el nombre de *High Horus* (*TH*). En esta tesis también desarrollamos herramientas que facilitan el modelado de actividades de alto nivel semántico y la exportación hacia el modelo *OWL*. La forma como ha sido diseñado *TH* hace que disponible para poder conectarse con cualquier herramienta de visualización de resultados, como ocurre con las desarrolladas

en esta tesis o como las que son producto de otros trabajos como (Rivas-Casado & Martínez-Tomás, 2011) (Rivas, Martinez-Tomás, & Fernández-Caballero, 2010).

Con el fin de probar el funcionamiento de *TH* experimentamos con casos de estudio, en donde pudimos inferir actividades de alto nivel semántico como hurto en supermercados y merodeo nocturno de pacientes con Alzheimer. El resultado de la experimentación es alentador y dio paso a las conclusiones y trabajos futuros que se exponen en este trabajo.

TH es una propuesta de marco para componer alto nivel semántico, por lo que consideramos que está en desarrollo, pero que si es una propuesta interesante para desarrollar sistemas en los cuales se necesite la abstracción semántica de una situación de interés, en especial en los escenarios de *Seguridad* y *Vigilancia,* como los que se detallan en este trabajo doctoral.

Contenido

Índice de Figuras

Índice de Tablas

1 Introducción

El término *Seguridad* "proviene de la palabra latina *Securitas* y se define como la ausencia de riesgo o también, a la confianza en algo o alguien" (Real Academia de la Lengua Española, 2010). El término *Vigilancia* se define como "el cuidado y atención exacta en las cosas que están a cargo de cada uno" (Real Academia de la Lengua Española, 2010). Se puede definir a *Seguridad y Vigilancia* (*SV*) como el seguimiento de personas y/o objetos de interés o bajo sospecha, con el fin de asegurar un entorno sin riesgos (Stevenson, 2007).

Un sistema de *SV* tradicional (generalmente analógico) es aquel en el que se le pide a un vigilante que esté atento a lo que sucede en la zona a la que se quiere brindar seguridad, en el que se utilizan sensores que generan alertas cuando receptan señales de anormalidad en la zona a ser vigilada. Un ejemplo clásico de este tipo de sistemas, son las alarmas anti-hurto instaladas en los domicilios que envían una señal relevante por línea telefónica o radio a una estación central, la misma que se encarga de tomar decisiones de llamar a la policía, bomberos, ambulancias o propietarios del inmueble.

La tecnología analógica de observación también se aplicó en circuitos cerrados de televisión, cuyos videos podrían grabarse y servían de prueba para el análisis de un delito (Berning Prieto, 2008). El inconveniente de este tipo de tecnologías era que los operadores humanos tenían que cambiar las cintas en los que se graba el video cada día. Esto se solucionó en la década de 1990, con la introducción de la multiplexación digital. Las unidades de multiplexor digital tenían características como lapso de tiempo y la

grabación solo se ejecutaba cuando existía algún movimiento en la zona de interés, esto permitía guardar una gran cantidad de espacio de cinta, además que se podía grabar en varias cintas simultáneamente. El próximo avance fue la digitalización, técnica que permitía capacidad de compresión y de bajo costo, y hace posible grabar videos de un mes de vigilancia en el disco duro. Además, las imágenes grabadas digitalmente son más claras y permiten la manipulación por medio de algoritmos informáticos de las imágenes con el fin de mejorar la calidad de resolución y obtener de forma automática la identificación de actividades humanas en las mismas –movimiento, detección de objetos–. Los acontecimientos del 11 de septiembre 2001 cambia la percepción del público de la videovigilancia. Los desarrolladores de *software* crearon programas que aumentan la capacidad de vigilancia a través de videocámara. En mayo de 2002, el *software* de reconocimiento facial se instaló en las cámaras de vigilancia de vídeo por ordenador en la Isla Ellis y la Estatua de La Libertad. Ese mismo año, el programa *SmartGate* se instaló en el Aeropuerto Internacional de Sídney en Australia. *SmartGate* es un sistema de cruce fronterizo automatizado para los miembros de la tripulación aérea. El sistema escanea las caras de los tripulantes, compara estas fotos con la de su pasaporte, y confirma la identidad en menos de diez segundos. En diciembre de 2003, el *Royal Palm Medio School* en Phoenix, Arizona instaló el *software* reconocimiento de rostros. Este es un programa piloto para el registro de delincuentes sexuales y el seguimiento de los niños desaparecidos. Un caso de éxito de sistemas de *SV* apoyados en algoritmos de análisis de secuencias de video, es el sistema de seguridad y vigilancia que permite detectar situaciones de alerta en las estaciones de trenes en Londres. A

partir de su puesta en marcha los delitos y robos disminuyeron en un 14% y 51% respectivamente (Berrier, 2008). La lucha antiterrorista y la lucha contra el crimen en general, promueven la investigación de este tipo de sistemas. Esos sistemas al estar basados en el análisis de escenas (videovigilancia), permiten identificar situaciones sospechosas, tal como lo hace un experto, y de forma no solamente reactiva, proactiva y eficiente, , sino también prospectiva (Albusac, 2008). Si bien la configuración exacta de un sistema de videovigilancia depende de cada escenario en particular, más y más empresas están encontrando un verdadero valor en instalar cámaras de seguridad y equipos para monitorear todos los puntos de entrada y salida, incluyendo los estacionamientos. Por ejemplo, las cámaras pueden disparar una alarma si detectan que alguien se brinca por alguna barda, o si abren alguna puerta o reja en momentos en que no está permitido hacerlo. Sistemas tipo *GeoVision*, son capaces de identificar personas y proveer detalles precisos del rostro de los sospechosos. Se trata de sistemas de *Monitorización* y videovigilancia que trabajan con redes multisensoriales (cámaras, sensores de movimiento, sensores de fuego) para controlar grandes áreas metropolitanas de forma inteligente. En este tipo de sistemas, las cámaras no solamente toman las imágenes, , sino que las modelan a través de modelos matemáticos. Los parámetros con los que trabajan los modelos matemáticos (la posición, tamaño o velocidad de los objetos), al ser datos, ocupan mucho menos ancho de banda, con lo cual se reduce también el coste de trasmisión. Las cámaras inalámbricas que mandan toda la información a una unidad central, en donde son procesados para identificar a las

personas o inferir el tipo de actividad que está realizando (SINC, 2012) (Albanece, Chellapa, Moscato, & Picariello, 2008).

(Collins, y otros, 2000) utilizan el término *Monitorización,* para conceptualizar el proceso de recolección y selección de actividades según la relevancia de la situación que se requiera identificar. Este proceso, parte del seguimiento de señales o imágenes que permiten caracterizar la situación de interés. Así, el objetivo general de esta *Monitorización* es identificar situaciones sospechosas, con el fin de asegurar que las actividades y situaciones son normales, informando sobre posibles anormalidades que pudieran presentarse (Aguas, 2010). La responsabilidad de la *Monitorización* de forma convencional está a cargo de un agente humano (vigilante). Pero, un vigilante no puede estar atento en todo momento, –solo durante veinte minutos puede *Monitorizar* cuatro cámaras simultáneamente, luego de eso; la tarea de *Vigilancia* deja de tener sentido (Honovich, Jhon, 2010) (Albusac, 2008)–. Esto pone de manifiesto una de las deficiencias de los sistemas de *SV,* la dependencia absoluta del operador humano, en donde factores como la fatiga producida tras varias horas de trabajo, reducen considerablemente la probabilidad de detectar todas las situaciones de interés. El que un sistema de *SV* proporcione un aviso de la ocurrencia de una actividad o situación, es tan importante como la selección de los elementos tecnológicos que permitieron captarlo. Los sistemas de *SV,* apoyados en tecnologías inteligentes han tenido un desarrollo acelerado en los últimos tiempos (Rantring, 2008) (Agrawal & Ryoo, 2011)–la detección e identificación de los números de la matrícula del coche, detección de

objetos estáticos en las vías y detección de peatones circulando por rutas no permitidas−.

La computación ubicua también apoya el desarrollo de sistemas de *SV*, por medio del reconocimiento de actividades de alto nivel semántico a partir de la *Monitorización* multisensorial, el mismo que va desde la captación de la señal hasta la interpretación de la misma. Para ello, a través de las distintas etapas se va abstrayendo la información proporcionada por los sensores y se la asocia con las actividades que suceden en el escenario−persona observa la televisión, persona toma su medicina, persona llama por teléfono−. Por lo general, este tipo de reconocimiento multisensorial ha sido aplicado para inferir actividades de la vida diaria de las personas (Chikhaoui, Wang, & Pigot, 2010).

El análisis de la información multisensorial requiere un alto grado de abstracción del bajo nivel semántico que no produce los detalles necesarios para comprender el detalle de lo sucedido en un escenario. Esta brecha semántica se identifica claramente en los sistemas de *SV* que procesan señales multisensoriales ya que pasan directamente desde la señal sensorial a interpretar la situación. Esta interpretación depende siempre de los conocimientos, la capacidad de expresión y el lenguaje específico del anotador. Algunas investigaciones proponen soluciones para eliminar la brecha semántica, la mayoría de ellas se basan en utilizar estructuras que parten desde el bajo nivel semántico y obtienen un alto nivel que permite descripciones de calidad que ayudan en la búsqueda y recuperación de actividades y situaciones en los sistemas de *SV* (Ayer & Chellapa, 2000) (Botia, Villa, & Palma, 2012) (Bredmod & Maillot, 2009). A pesar de todos los

esfuerzos de investigación, no se ha logrado integrar en una sola estructura funcional a los sistemas de *SV*, esta una idea que permitiría mejoras en la interpretación de situaciones en un escenario. El tener arquitecturas que agrupen sistemas multisensoriales, con el fin de ayudar al operador humano a tomar decisiones según se identifiquen una situación de interés, es con claridad una temática que debe desarrollarse desde la combinación tecnológica. Con esta temática el grupo *SIMDA* ha llevado a cabo proyectos que proponen la integración de diferentes tecnologías y la conceptualización semántica de las situaciones (SIMDA, 2008): AVANZA, CICYT 2004, CYCYT 2007, *INT3*. El aporte de *INT3* es fundamental para este trabajo, ya que obtuvieron a partir de él a *Horus* un marco multisensorial para monitorización y detección de actividades, que integra sistemas multisensoriales en una sola unidad de procesamiento, tal como se muestra en la Figura 1-1:

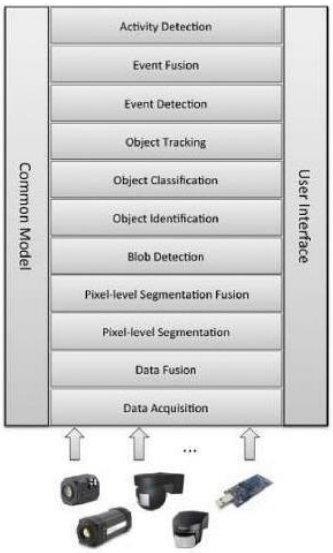


Figura 1-1. Sistema modular de *Horus*. Fuente: (Castillo, Fernández-Caballero, Serrano-Cuerda, & Sokolova, 2012)

Como se muestra en la Figura 1-1, *Horus* es una arquitectura modular para el manejo de entradas multisensoriales, que incorpora un modelo de conceptualización que permite compartir la información de interés entre múltiples escenarios. Las fuentes multisensoriales están relacionadas principalmente con sensores de imagen, ya que son los más extendidos para tareas de *Monitorización*; pero otras tecnologías de sensores, como redes de sensores inalámbricos (*WSN*), también están integrados objetos genéricos en *INT3–Horus*. El marco es distribuido e híbrido, los nodos remotos realizan el procesamiento de nivel inferior, así como la adquisición de datos, mientras que un nodo central se encarga de la recogida de la información y de su fusión. El marco incluye detección de objetos simples y; el seguimiento y detección de actividades (Castillo, Fernández, & López, 2011). Su tarea es ambiciosa dada la gran variedad de escenarios y actividades que pueden ser enfrentados. En su arquitectura, se establece una serie de niveles de operación, en los cuales se definen claramente la entrada / salida de las interfaces. Estos niveles son flexibles, ya que fácilmente puedan ser adaptados a un sistema final. Su infraestructura se basa en el paradigma Modelo-Vista-Controlador (*MVC*). Este paradigma divide una aplicación en: a) entidades, b) la definición de sus funciones principales, y c) las conexiones entre ellas. En los sistemas basados en eventos, el *MVC* provee de la información acerca de los cambios en la aplicación y proporciona una representación que se adapta a las necesidades del usuario. El modelo recibe las entradas a la aplicación e interactúa con ella, para actualizar los objetos y para representar la nueva información (Sokolava, Castillo, Fernández-Caballero, & Serrano-Cuerda, 2012) (Castillo, Fernández-Caballero, Serrano-Cuerda, & Sokolova, 2012).

Los módulos que se han desarrollado en *Horus* llegan hasta la fusión sensorial. Siendo nuestro trabajo colaborar con un marco general que se acople al módulo de fusión y obtenga el modelado de alto nivel semántico.

Proponemos trabajar con estructuras del conocimiento, que recojan generalidades y particularidades de las situaciones de interés, con el fin de identificarlas de forma automática en los escenarios *Monitorizados* (Bremond, Maillot, Thonnat, & Vu, 2004) (Bredmond, Corvee, Patiño, & Thonnat, 2008) (Town, 2006). Durante la última década las ontologías son utilizadas en aplicaciones para las áreas de procesamiento del lenguaje natural, *e-comerce*, integración de información inteligente, consulta de información, integración de bases de datos, bioinformática, educación y en la web semántica, entre otras. Dichas ontologías proporcionan un vocabulario y organización de conceptos que representan un marco de trabajo conceptual para el análisis, discusión o consulta de información de un escenario. Pero, existe la necesidad de realizar tareas de razonamiento, para lo cual; se debe integrar módulos o herramientas en un solo marco conceptual, metodológico y tecnológico. Estos módulos deben acoplarse al marco de *Horus*, con el fin de inferir actividades de alto nivel semántico (ver Figura 1-2).

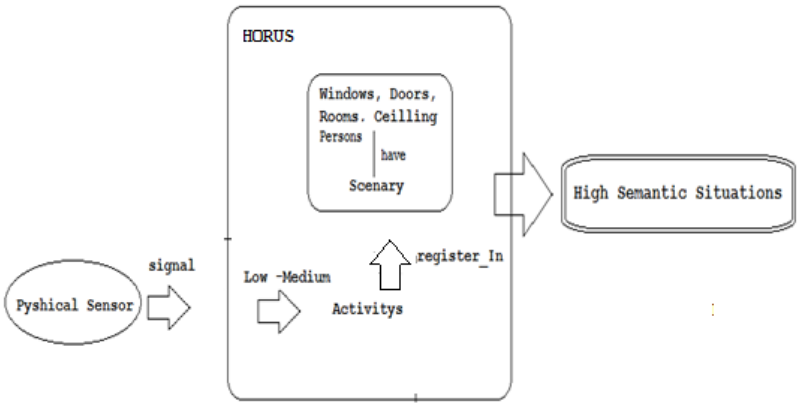


Figura 1-2. Modelo Semántico: La estructura del conocimiento se utiliza para modelar escenarios, actividades y situaciones.

La hipótesis a comprobar es que por medio del diseño de ontologías y tecnologías semánticas, fáciles de usar, reutilizar y modular; se puede inferir situaciones de alto nivel semántico y el rápido prototipo de sistemas de *SV*, con un nivel similar de abstracción del que tiene un agente humano. Para comprobar nuestra hipótesis nuestra ontología debe cumplir con los siguientes aspectos:

1. **Trabajar con señales multisensoriales y su integración**, con el fin de eliminar la dificultad que tienen los sistemas de *SV* para combinar múltiples dispositivos heterogéneos en una misma red de *Monitorización*. Conceptualizar los sensores y sus relaciones semánticas, con el fin de trabajar directamente con el sensor semántico antes que con el sensor físico.
2. **La representación del conocimiento**. La ontología debe ser capaz de conceptualizar los elementos del escenario y sus relaciones. La estructura de conocimiento debe poseer características que permitan trabajar con sistemas de *SV*

que son predictivos –*a priori*– o con sistemas de *SV* de análisis retrospectivo, es decir; que infieran o analicen situaciones después de que han ocurrido.

3. **Importar ontologías.** Adapta su estructura para combinarse con otras ontologías desarrolladas en distintos dominios. Esto con el fin de reutilizar las representaciones del conocimiento en diferentes áreas de la ciencia.
4. **Conceptualizar e inferir actividades**. Proceso en el cual se utiliza el conocimiento del experto para conceptualizar actividades, o; se aplican reglas o axiomas para inferencia a las actividades que se registran en el escenario. Nosotros nos preocupamos por inferir actividades de medio y alto nivel semántico dejando para *Horus* el nivel semántico bajo.
5. **Conceptualizar e inferir de situaciones.** Permitir que el vigilante o experto establezca las relaciones entre las actividades que se suceden en el mismo y diseñe reglas y axiomas semánticos para inferir una situación. De forma alternada, tener la capacidad semántica necesaria para adaptar su estructura al aparecimiento de nuevas conceptualizaciones de actividades y situaciones, producto del aprendizaje obtenido por algoritmos informáticos. En función del conocimiento de las situaciones de interés, se plantean dos tipos de tareas: a) conceptualizar y modelar el conocimiento del experto humano, cuando aquel exista (en función de las actividades básicas reconocibles desde los sensores o por el procesado de imágenes de vídeo) y b) conceptualizar y modelar situaciones, en dónde; ese conocimiento no existe; aunque sí es posible, encontrarlo en registros (bases de casos) de las situaciones que se pretenden identificar. En este caso, el proceso requerido es particularmente

complejo (Sunico, 2008) (Hu, W, Wang, & Maybank, 2004) y requerirá el uso de algoritmos inteligentes para su identificación. Los literales a) y b) se estudian en esta tesis, puesto que se trabaja con el conocimiento del experto cuando este puede describir con claridad los escenarios y situaciones, y; en escenarios en los que existe cierto conocimiento experto pero es poco preciso y se pretende encontrar de forma automática las situaciones de interés. Aquí las situaciones están compuestas por actividades que individualmente no son claramente sospechosas, pero que al ser analizadas en una determinada secuencia y repetición, sí reflejan serlo.

Al cumplir con estos aspectos, se pretende obtener diseños ontológicos que se acoplen a sistemas *SV,* con el fin de conceptualizar e inferir situaciones de interés con un alto nivel semántico. Y es que el uso de técnicas semánticas para modelar el conocimiento es de interés de las investigaciones actuales, que trabajan con el modelado de situaciones. De hecho la aplicación de las ontologías ha llegado a ser comparable con los algoritmos que permiten identificar eventos en video **(Riboni, Pareschi, Radaelli, & Bettini, 2011)**. Esto fundamenta aún más nuestra propuesta. En este trabajo nos centraremos en dar un aporte que ayuda a concebir el diseño ontológico, a través de la consecución de los siguientes objetivos:

- Diseñar una ontología necesaria, suficiente, adaptable y reutilizable que permita conceptualizar los componentes de un sistema de *SV*. Pretendemos ampliar las funcionalidades de los sistemas de *SV* actuales; conceptualizando su mecanismo de *Monitorización* con el fin de describir con un alto nivel semántico lo que sucede en un escenario.

- Conceptualizar las relaciones semánticas de los componentes de un sistema de *SV*. Trabajaremos con el sistema de *SV* como una sola unidad, y no como señales que provienen de distintos sistemas.
- Crear un marco conceptual, tecnológico y metodológico, con el fin de trabajar en niveles que permitan la inferencia de alto nivel semántico, a partir del procesamiento de la señal multisensorial, luego la inferencia de actividades y la inferencia de situaciones. Esto nos permite evitar la brecha de interpretación semántica de actividades existente en los *SV* tradicionales.
- Modelar situaciones y hacer uso de tecnologías para poner a prueba el aporte. Realizar experimentos que permitan observar la inferencia de actividades y situaciones por medio de las estructuras semánticas. Para ello, haremos uso de algoritmos informáticos y frontales que permiten mostrar el funcionamiento de nuestra propuesta.

Para cumplir nuestros objetivos, cobran especial importancia todas las soluciones basadas en ontologías que vayan orientadas a la conceptualización y modelado de escenarios, actividades y situaciones de interés (NCYT, 2012) (Botia, Villa, & Palma, 2012). Para implementar este comportamiento inteligente, una de las estrategias que se utiliza es el análisis semántico. En este trabajo diseñamos a *Conceptual High Level Human Activity Ontology* (*CHAO*), en la cual se escogió las conceptualizaciones necesarias provenientes de distintas ontologías para conceptualizar escenarios de *SV* y modelar e inferir situaciones de interés:

- Para conceptualizar a *SS* se utiliza la ontología *SSN*[1], que cumple con los estándares necesarios que le permiten conceptualizar la mayoría de sensores físicos. Emulamos el trabajo de *Horus* en los experimentos, partimos de la señal sensorial para inferir actividades de nivel medio semántico.
- Para conceptualizar el escenario se utilizó la ontología *BuildingArchitecture*, que conceptualiza los componentes de un escenario además de las relaciones espaciales (Hois, Bhatt, & Kutz, 2009). El término escenario se utilizará también como contexto.
- Para conceptualizar las relaciones temporales entre las actividades semánticas, se trabajó con *TIME*[2].

CHAO es una estructura de conocimiento que utiliza niveles jerárquicos semánticos con el fin de inferir situaciones de interés. El primer nivel que parte de *Horus*, consiste de anotaciones semánticas contextuales (bajo nivel), que provienen de recoger datos en bruto de los sensores físicos y conceptualizar e inferir actividades de bajo nivel semántico –camina en el cuarto, camina en la cocina–. En el nivel semántico medio, *CHAO* infiere actividades que resultan de la combinación de las actividades de nivel semántico bajo, pero que han interactuado con el escenario, –si la persona camina en el cuarto y luego camina en la cocina, implica que la persona cambió de zona–. En el nivel semántico alto, se infieren las situaciones de interés, apoyados en los niveles semánticos

[1] *SSN*: http://www.w3.org/2005/Incubator/ssn/, tomado el 22-10-2013.
[2] *Time* Ontology in OWL: http://www.w3.org/TR/owl-time/, tomado el 09-10-2013.

inferiores –si la persona cambia de zona pero repite los mismos lugares de visita en un ciclo de tiempo, probablemente la persona Merodea–.

Para modelar las actividades y situaciones, se utiliza el lenguaje Video Event Representation Language (*VERL*). La suma de las actividades individuales puede conducir a una nueva actividad (actividad compuesta). En este trabajo se utiliza a *VERL* para componer los niveles jerárquicos de las actividades (Nevatia & Hobbs, 2004). Si bien es cierto, *VERL* fue concebido modelar eventos de vídeo, en esta tesis; lo utilizamos también para modelar eventos que provienen de señales multisensoriales, desde *Horus*, por ejemplo.

Nuestro trabajo se deriva de la necesidad de desarrollar una estructura semántica que facilite la descripción, el modelado de escenas y situaciones que se basan en la terminología y la información que se obtiene durante el proceso de identificación y el seguimiento de situaciones de interés en un escenario. En ese sentido, proponemos que en un solo Marco conceptual, metodológico y tecnológico se combinen *CHAO* y *VERL* para modelar actividades de alto nivel semántico. El marco puede ser utilizado desde dos puntos de vista, para inferir situaciones a partir del conocimiento del experto o para ayudar a obtener dicho conocimiento. En el primer caso, en esta tesis diseñamos experimentos en los cuales se deja claro, la conceptualización de actividades y las relaciones con el escenario, además del nivel jerárquico que va desde el procesamiento de la información sensorial (*Horus*) hasta la inferencia de actividades y situaciones de alto nivel semántico. Se programó una aplicación que facilita el modelado de

actividades y situaciones desde *VERL* y exporta las conceptualizaciones semánticas a *OWL*, lenguaje que tiene limitaciones para expresiones semánticas que involucran el conceptualizar situaciones y actividades complejas. Para el marco también se aporta de una herramienta de modelado gráfico que parte de *VERL* pero que evidencia claramente el conjunto de conceptualizaciones que el diseñador de un prototipo para los sistemas de *SV* puede utilizar con el fin de lograr obtener un alto nivel semántico. A esta herramienta le hemos llamado *GVERL*. También se programó otra aplicación que permite realizar consultas al modelo cuando este está instanciado. Este sistema de consultas semánticas puede ser utilizado como fuente para un análisis retrospectivo de una situación –consultar en que imágenes las personas forman grupos, en que imágenes las personas llevan una mochila–.

En el segundo caso, *CHAO* provee la secuencia de actividades para que el algoritmo *Generalized Sequential Patterns* (*GSP*) identifique patrones (*micropatrones*[3]) representativos de las situaciones de interés. Nos fiamos de los resultados de *GSP* puesto que ha dado buenos resultados en identificar patrones de situaciones humanas en escenarios de deportes y convivencia humana (Karikrishna, 2011) (Chikhaoui, Wang, & Pigot, 2010). El resultado permite la actualización de *CHAO*, ya que se generan nuevas relaciones entre los elementos del escenario.

[3] Realizamos la diferenciación entre patrones y *micropatrones*, puesto que en comportamiento humano, los patrones que se obtienen en escenarios de videovigilancia son más cortos (*micropatrones*) que en escenarios de compras en supermercados (Albusac, 2008).

Para probar a *CHAO* se infirieron situaciones en dos escenarios: a) Una casa de habitación[4], en la cual se han colocado sensores de distinto tipo. La situación a inferir es si una persona está teniendo algún problema mental –*Deambular* en el caso de Alzheimer– y b) Un supermercado, en el cual se han colocado cámaras en diferentes zonas. La situación a inferir está relacionada con el hurto en tiendas. En ambos casos, la generalidad de la estructura semántica es proporcionada por un experto. El resultado de los *micropatrones* también es sometido a análisis semántico, cuyo resultado es su modelado con *CHAO-VERL* con incremento de las relaciones semánticas.

CHAO y las tecnologías semánticas conforman el marco jerárquico de inferencia de situaciones de alto nivel semántico que proponemos en esta tesis. Y es que planteamos que este marco tiene todas las herramientas semánticas necesarias para poder llegar a un nivel alto semántico y con ello enlazarnos con la propuesta de *Horus*. Nuestra propuesta puede ser aplicada para modelar el conocimiento *apriori*, cuando la descripción de una situación está clara y la puede definir un experto, y también para la búsqueda del conocimiento que ayude a modelar una situación cuando esta no es lo suficientemente clara para ser modelada por un experto humano.

En la siguiente sección, se describen varios estudios que utilizan tecnologías semánticas para la inferencia de actividades, así como también el aporte del modelo semántico para la mejora de resultados de algoritmos de minería de datos. A continuación se describe la metodología que se utilizó en este trabajo, a través de la cual se obtuvo a *CHAO*. En

[4] Casas Project: http://ailab.wsu.edu/casas/datasets/index.html, tomado el 09-10-2013.

otro apartado se describirá la experimentación, las conclusiones y trabajos futuros resultado del trabajo realizado en esta tesis.

2 Trabajos relacionados

En este apartado se describen las principales investigaciones relacionadas al uso de señales multisensoriales y estructuras semánticas en la inferencia de actividades y situaciones de alto nivel semántico. Se detallan también los trabajos científicos que proponen marcos conceptuales, en los cuales las metodologías y técnicas utilizadas para inferir situaciones de interés, trabajan en unidad para el procesamiento semántico. El orden de los estudios que se detallan a continuación, está en relación a la propuesta realizada por (Agrawal & Ryoo, 2011), para clasificar a las distintas técnicas que permiten el obtener a los niveles de inferencia semántica de actividades.

2.1 Vigilancia y seguridad

El uso de sistemas *SV* basados en cámaras de Circuito Cerrado de Televisión (*CCTV*) ha crecido exponencialmente en la última década. La preocupación por la seguridad, especialmente como consecuencia del incipiente terrorismo internacional, hace que los expertos anticipen una difusión mayor de estos sistemas, así como su integración formando una red global de vigilancia remota. (Honovich, Jhon, 2009) analiza cuáles son los últimos avances en los sistemas de *SV* multisensoriales que tienen las empresas que producen este tipo de tecnología haciendo énfasis en su manufactura, valor agregado, diferencias con otros productos y su uso. Este análisis se centra en los sistemas *SV* basados en cámaras y sensores para la vigilancia. El resultado el análisis permite dar respuestas a preguntas como ¿Sería útil poder hacer un seguimiento de las personas en diferentes zonas y lugares? ¿Es posible comprobar falsas alarmas en

establecimientos, o simplemente vigilar un comercio desde la comodidad del hogar? (Anexo I). Aplicaciones de este tipo ya están comercialmente disponibles, permitiendo el acceso desde un único centro de control a las imágenes de los sistemas *CCTV* de varios entornos geográficamente distribuidos. Por ejemplo, el sistema de adquisición de video sincronizado desarrollado para interoperar con sistemas de *SV* con el fin de que actúen como servidor de trayectorias de objetos. Está compuesto por una serie de instrumentos de navegación que permiten la georeferenciación directa de cada uno de las imágenes capturadas por la videocámara en postproceso en un tiempo de referencia común para todos los instrumentos de navegación y para todos los sensores utilizados en el sistema de captura de video. La teledetección, permite tener información sobre un objeto o superficie a través del análisis y procesado de los datos suministrados por los diferentes sensores que están sincronizados. Además, asocia el tiempo y coordenadas de posicionamiento geográfico (*GPS*) con la imagen generada por el video (Calero & Wis, 2010). Como resultado se obtienen sistemas de capaces de analizar el video de diferentes subsistemas e interpretar lo que sucede en las imágenes. Ejemplos de aplicaciones que pueden tener los sistemas de adquisición y sincronización de video son la *Monitorización* en tiempo real del estado de tráfico, el control de incendios forestales, el seguimiento de catástrofes naturales, la proyección de video geo referenciado en máquinas virtuales públicas como es el caso *Google Earth* y *Virtual* (Calero & Wis, 2010).

Los sensores que se utilizan en los sistemas detallados en este apartado, pueden enlazar sus señales, componiendo redes multisensoriales, las mismas que se detallan a continuación.

2.2 Redes multisensoriales

Desde los años 90, las redes multisensoriales (conformadas por sensores de radio frecuencia (*RFID*), sensores de movimiento, videocámaras, etc.) han proporcionado la forma en que las personas pueden intercambiar información y coordinar procesos. Los sensores posibilitan la medición del entorno que nos rodea, capturando los datos y enviándolos a sistemas de *SV* para su procesado. La tecnología de redes multisensorial ha evolucionado en sistemas embebidos hasta el punto de proporcionar dispositivos que difícilmente pueden diferenciarse de los nodos computacionales, incorporando capacidades cognitivas y de comunicación con las que llegan a establecer verdaderas redes de información "inteligente". Sus características se encuentran detalladas en la investigación de (Los Santos Aransay, 2013).

Una de las aplicaciones de las redes multisensoriales se encuentra en computación ubicua –computación multisensorial–. El objetivo fundamental de la computación ubicua es la creación de escenarios inteligentes, lo cual se logra con un proceso que interpreta las señales multisensoriales de forma inapreciable para las personas, con el fin de proporcionarles información suficiente que los ayuden con el desarrollo de las tareas comunes de la vida diaria –muchas aplicaciones han sido desarrolladas a partir de este tipo de tecnologías, ayudar a la *Monitorización* de pacientes, ciudades ubicuas en los

cuales los sistemas de información multisensoriales están interconectados con el fin de dar seguridad en lugares públicos, oficinas y casas (Tapia, 2004)–.

El principal problema de la computación ubicua es que la representación de la información carece de expresividad y extensibilidad. Algunas investigaciones han tratado de resolver este problema, abordando la representación sensorial con el uso de metalenguajes tipo *XML*. Sin embargo, *XML* es incapaz de proporcionar el apoyo adecuado para la representación semántica, lo cual es esencial para el intercambio de conocimientos y el razonamiento (Coen, 1998) (Capra, Emmerich, & Mascolo, 2001) (Held, Buchholz, & Schill, 2002) (Chen, Finin, & Joshi, 2004). Otras soluciones proponen pasar de los sensores físicos a los sensores semánticos, capaces de integrase sin necesidad de cables con las redes de datos de una manera rápida y trasparente, gracias al uso de estándares abiertos ampliamente difundidos. Un sensor semántico, según el *IEEE 1451.2* un transductor inteligente; es aquel que proporciona más funciones de las necesarias para generar una correcta representación de la variable *Monitorizada*. Dichas funcionalidades están orientadas a facilitar la integración del transductor con las aplicaciones del escenario. Las ventajas del planteamiento de los sensores semánticos: 1) Aportan interoperabilidad entre aplicaciones de *software* independientemente de sus propiedades o de las plataformas sobre las que se instalen, 2) fomentan el uso de estándares y protocolos, lo que hace más fácil acceder a su contenido y entender su funcionamiento, 3) Permiten que servicios y *software* de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados y, 4) Permiten la interoperabilidad entre

plataformas de distintos fabricantes por medio de protocolos estándares (Morillo, Maciá, & Jorquera, 2013).

Pero, la fusión de datos de múltiples sensores se ha limitado, especialmente por la falta de normas en el intercambio de datos y descripción de los sensores (Council, 2000). Algunos estándares como *SensorML* y *O&M* han sido adoptadas por la propuesta de *Sensor Web Enablemente* (*SWE*) con el fin de mitigar estas deficiencias (Botts, Percival, Reed, & Davidson, 2008). *SWE* tiene como objetivo reunir a los sensores cuya información se encuentre disponible en Web, con el fin de tener la información concentrada en lo que ahora se conoce como *Web Semántica* (Berners-Lee, Hendler, & Lassila, 2001).

Es necesario enfatizar en que las señales que provienen de los sensores, deben pasar por un preprocesamiento antes que ser consideradas como limpias. La ciencia de la fusión de la información multisensorial, generalmente utiliza principios estadisticos que permiten filtrar la información y eliminar la mayor cantidad de ruido de la misma, con ello, también se intenta disminuir el error. Un método de estimación tradicional es el conocido como *Estimación con Mínimos Cuadrados* (*LSE*), el cual es coherente y eficaz en el manejo de datos de información multisensorial estándar. Sin embargo, la información en sistemas de *SV*, no es estándar no proviene de sensores estándares, , sino de diferentes tipos de sensores por lo que se pueden cometer errores relacionados con la mezcla de información y el manejo del ruido, por lo que no se puede seguir utilizando los algoritmos de función optima para el tratamiento de información multisensorial. Para resolver aquello (Wang J. , 2010) propone que los mecanismos de fusión de

información multisensorial, utilicen algoritmo que permita trabajar con factores de correlación en tiempo real de información multisensorial no estándar. Este algoritmo logra eliminar el error de la mezcla no lineal y disminuir el factor de incertidumbre de la información que proviene desde múltiples sensores. Además, contiene un mecanismo autoadaptativo para eliminar la dependencia local y la incertidumbre de los parámetros relacionados con distintos sistemas de *SV*. *Horus* que es nuestro marco base, aporta con mecanismos de fusión de datos, que permiten que nuestra propuesta trabaje con "señales limpias" (Castillo, Fernández-Caballero, Serrano-Cuerda, & Sokolova, 2012)que generan los eventos necesarios para que nosotros a partir de allí identifiquemos actividades y situaciones de interés.

Resuelto el tema de la fusión, queda otro problema. Actualmente, en los escenarios de *SV* los sensores semánticos no se encuentran integrados semánticamente entre sí, sus datos se procesan de forma individual, pero; la interpretación de las situaciones depende de la participación en conjunto de los mismos. Para solucionar este problema *Horus* aporta con la filosofía de la integración semántica, mientras que nosotros proponemos el marco necesario para la operatividad de la conceptualización jerárquica semántica de alto nivel, salvando también con ello el problema de la brecha de la interpretación semántica. Nuestra propuesta no es solamente la creación de la estructura del conocimiento, sino también contempla trabajar con tecnologías relacionadas que permitan identificar actividades y situaciones de interés. Estas tecnologías contemplan el uso de algoritmos computacionales, algunos de ellos se describen a continuación.

2.3 Tecnologías para seguridad y vigilancia.

Existen muchas aplicaciones utilizan, combinan y crean algoritmos para identificar actividades, situaciones y escenarios. En dominios multisensoriales, (Haigh, y otros, 2004) desarrollaron *I.L.S.A*, sistema destinado al cuidado de personas mayores. El objetivo principal de esta aplicación es determinar el modelo de situaciones humanas por medio del reconocimiento de actividades para con ello dar respuesta situaciones de alerta. El algoritmo de aprendizaje relaciona la activación de un sensor con una actividad, recuperando el orden y tiempo de ocurrencia. Cuando el orden o el tiempo no se corresponden con un patrón de actividades previamente establecido, el algoritmo genera una alerta. (IBM, 2013) desarrolló *PeopleVision*, un sistema de *SV* que permite identificar situaciones sospechosas, relacionadas con el tracking de objetos y la identificación de rostros, obtenidas a partir del procesamiento de señales multisensoriales.

(Perianu & Lombriser, 2008) utilizan los datos proporcionados por sensores electrónicos como entrada para algoritmos que utilizan la inferencia difusa para clasificar las actividades en dependencia del tiempo que dure cada una de ellas. La principal desventaja de este método es que existen actividades que al ser distintos pueden ser iguales en longitud temporal, por lo que estaría limitada al manejo de ciertas actividades independientes entre sí.

(Perse, Kristan, Pers, & Kovacic, 2008) diseñaron un algoritmo que procesa las señales multisensoriales en máquinas de estados y sistemas basados en reglas para reconocer el

movimiento de las personas –estos sistemas no obtienen buenos resultados cuando existen fallas de los sensores utilizados–. Estas aplicaciones tienen problemas con el número de ejemplos que necesitan para el aprendizaje, el cual tampoco es dinámico puesto que para situaciones particulares, el número de ejemplos es poco (robos, asaltos). Además, si aparece una nueva evidencia, el sistema tiene que ser entrenado nuevamente. El proceso de reentrenamiento, puede ser eliminado cuando se tiene un conocimiento *a priori*, como lo propone (Oliver, Rosario, & Pentland, 2000). Allí, un sistema basado en redes bayesianas recoge el conocimiento y la evidencia, y permite su funcionamiento aún con poca cantidad de información proporcionada por parte del experto –el *algoritmo acoplamiento de Markov* propuesto por (Nemanja, 2007) utiliza técnicas estadísticas de aprendizaje para enseñarle al sistema a reconocer actividades normales de las personas. El sistema posee un conocimiento *a priori* relacionadas con las actividades lo que evita que deba ser reentrenado en condiciones normales–. (Zhu, Yang, Yu, & Gong, 2009) (Charkraborty, Bagdanov, Gonzalez, & Roca, 2011) (Park, Lin, Metsis, Le, & Makedon, 2010) utilizan una función de puntuación probabilística, para calcular la similitud que tienen secuencias temporales de actividades humanas con patrones de actividades humanas definidos por expertos. El resultado sirve para identificar *Actividades de la Vida Diaria* (*AVD*) de personas en el hogar –leer, escuchar música–. En (Hongeng & Nevatia, 2001) usa a las *redes Bayesianas* que procesan señales multisensoriales para inferir actividades. El segundo nivel, es más complejo; pues necesita ordenar en el tiempo la secuencia en la que ocurren las actividades y de acuerdo a ello obtener un nivel semántico medio. En el tercer nivel, las actividades se

relacionan entre sí espacial y temporalmente, para producir otras actividades más complejas compuestas por la combinación de actividades de menor o igual nivel semántico.

En dominios de videovigilancia de forma inicial, los algoritmos se utilizan para procesar señales de video para identificar objetos y personas en una escena. Estos algoritmos segmentan las imágenes extraídas por las cámaras, identifican el fondo de la escena y el primer plano compuesto por objetos en movimiento. Los objetos detectados mediante estas técnicas son representados como *blobs* en el área de la imagen ocupada por el objeto. (Jhonson, Shotton, & Cipolla, 2013), utilizan la información temporal de los escenarios (descripción de las acciones de la escena, movimiento, cambios de toma, marcos), unido a la información espacial (relación entre los elementos de la escena, próximo a, sobre qué), para enriquecer los descriptores con un etiquetado semántico de la información del objeto. El etiquetado semántico proporciona una representación independiente de los acontecimientos con respecto a cambios espacio-temporales y cambios de escalas, diferencias de fondos y movimientos múltiples en un escenario. Los descriptores de video se utilizan como entrada para que el clasificador binario pueda ser entrenado con el fin de identificar clases de objetos en una imagen. El tiempo de ejecución de este tipo de algoritmos depende principalmente de tres factores: modelo de complejidad (variación de los objetos con respecto al fondo), tamaño de los videos analizados (dimensiones) y el rango de la escala de búsqueda seleccionado.

Se ha desarrollado algoritmos que trabajan con reglas difusas con el fin de clasificar objetos en vídeo. Dichos algoritmos comienzan el proceso de aprendizaje a partir de un

conjunto de entrenamiento en el que cada clase de objeto es descrito por un conjunto de ejemplos definidos por medio de un conjunto de características. En la etapa de aprendizaje, el algoritmo aprende el conjunto de reglas difusas que caracteriza a los diferentes tipos de objetos móviles a partir de los conjuntos de entrenamiento generados en la etapa anterior. El algoritmo tiene como objetivo aprender un conjunto de reglas lingüísticas que modelen la función S definida como S: V → O. Donde O es el conjunto de clases de objetos que se mueven en escenario vigilado y V es información obtenida en la etapa de segmentación: posición vertical (*v1*) y horizontal (*v2*) del objeto, tamaño del objeto *n* en número de macro bloques (*v3*) y en número de vectores de movimiento (*v4*), relación entre anchura y altura del objeto (*v5*) y una medida de dispersión del campo de vectores de movimiento pertenecientes al objeto (*v6*), es decir, V = {*v1*,*v2*, *v3*,*v4*,*v5*,*v6*}. Por su parte, el algoritmo de aprendizaje toma como entrada un conjunto de entrenamiento E={e_1, e_2, ...,e_n} donde cada ejemplo es *ei* = (*vi1*,*vi2*,..., *vi6*:o_i), siendo v_{ij} el valor del ejemplo *i* para la variable v_j y o_i el tipo del objeto detectado en la escena ($o_i \in O$). Una vez que se obtiene el conjunto inicial de centroides para cada partición de perspectiva, el algoritmo *K-Means* distribuye el conjunto de entrenamiento en *k* grupos modificando la representación de los grupos iniciales. Este algoritmo se obtiene los *clusters* a los que pertenece cada tipo de objeto y por ende identifican el tipo de escenario. La tercera fase tiene como objetivo reducir el conjunto de reglas iniciales para construir reglas más generales (Solana-Ciprés, C.J; Albuscac, J; Castro-Sanchez, J.J; Moreno-García, J; Rodriguez-Benítez, L, 2010).

(Barake & Saddik, 2008), diseñaron algoritmos basados en *Lógica difusa*[5] para controlar las cámaras de videovigilancia imitando al controlador humano. El objetivo que se persigue es obtener una imagen clara del rostro de los pasajeros, siendo su principal problema cuando existe la movilidad de los mismos, ya que la cámara necesita tiempo para el ajuste.

(Kang & Lee, 1998) desarrollaron algoritmos *Difuso-genéticos* que permiten distinguir a las personas de objetos en la imagen, lo cual conduce a que se puede detectar los eventos que registran las personas en vídeo. Existen algoritmos informáticos que en un vídeo pueden distinguir: cuando una persona camina, corre, forma grupos y cuando la persona toma un objeto. La base de estos algoritmos sigue siendo el procesamiento probabilístico y el filtrado de imágenes (Agrawal & Ryoo, 2011) (Chun, 2008) (Zhu, Yang, Yu, & Gong, 2009) (Sunico, 2008).

Los eventos que provienen del procesamiento de imágenes de video, son la entrada para que otros algoritmos infieran acciones y situaciones de interés. Estos algoritmos por lo general se basan en procesos probabilísticos.

(Cheng & Chen, 2007) proponen algoritmos probabilísticos que identifican las acciones que generan las personas en video que conducen a situaciones de alerta, –identifican que una persona ha tenido un accidente doméstico, procesando secuencias de imágenes en las cuales la persona estaba caminando, tropieza con una silla y luego esta inmóvil en

5 Sus siglas en ingles son FIS. Fuzzy inference Logic

el piso eventos–. Es necesario destacar que este ejemplo se habla de accidente doméstico porque el sistema está instalado en tal sitio, no porque el algoritmo ha sabido identificar el entorno en el cual se encuentra operando. Algoritmos basados en *Modelos Ocultos de Markov* (*HMM*), identificar situaciones relacionadas con peatones al cruzar una calle, –comportamiento ante una gran cantidad de tráfico, comportamiento cuando el semáforo cambia de color– (Kwon & Murphy, 2000). (Somboon, Bremond, & Nevatia, 2000) desarrollaron un algoritmo probabilístico que procesa señales de video con el fin de obtener eventos que permitan identificar situaciones violentas –a través del análisis de eventos relacionados con levantar los brazos, identifican la situación "agitación"–.

Los eventos, acciones, actividades y situaciones ocurren en un escenario, de hecho, su interpretación está directamente relacionada con este (Albusac, 2008). El trabajo con distintos escenarios es uno de los problemas a enfrentar en los sistemas de *SV*. Si se recoge la información generada en todo el escenario (actividades de todas las personas, movimiento de todos los objetos), esta pudiese resultar difícil de manejar y tediosa de analizar, proponiendo como solución, el filtrado de escenarios en el entorno o el reconocimiento de los mismos mediante algoritmos informáticos. Por ejemplo, en los videos en los cuales los algoritmos de visión identifican en un vídeo platos, personas, mesas, cubiertos en un vídeo, se puede concluir que es un restaurante. Mientras que si identifican en un vídeo coches, personas, semáforos, policía se puede decir que es una avenida o una calle (Town, 2006). Los algoritmos que se basan en *Redes de Petri* permiten identificar escenarios en vídeo y modelarlo en función de lugares y

transiciones (Cervantes, 2005). Sin embargo, es necesario señalar que las redes de Petri no pueden actualizar el conocimiento, no pueden ajustarse, es decir, no tienen la capacidad de aprender (Cervantes, 2005) (Albanece, Chellapa, Moscato, & Picariello, 2008). Por lo tanto, si deben reconocer otro escenario distinto para el que fueron concebidos, su funcionamiento es incorrecto. Las redes dinámicas bayesianas se utilizan en (Perzold & Pietzowski, 2005) para clasificar escenarios según las actividades humanas que allí se generen. El problema de este tipo de algoritmos es que son propietarios, es decir, solo pueden reconocer escenarios para los que fueron entrenados y no tienen un modelo general de inferencia.

Otro de los problemas que se deben enfrentar en escenarios de *SV* video-vigilados, es que actividades sospechosas pueden ser combinarse con situaciones normales (Aguas, 2010) (Gonzalez, 2009). Por ejemplo podríamos encontrar una persona que parece generar situaciones normales de comportamiento, cuando es *Monitorizada* en el transcurso de una ventana temporal, pero podrían tener otras intenciones posteriormente, como el deseo de robar mercancía de una tienda (Aguas, 2010). En algunas situaciones, lo que más importa son la secuencia de actividades que caracterizan la situación de interés dando la idea de la presencia de patrones (Rivas, Martinez-Tomás, & Fernández-Caballero, 2010), (Martínez-Tomas, Rincón, Bachiller, & Mira, 2008) (Simon & Zhuang, 2007), (Oliver, Rosario, & Pentland, 2000), (Chikhaoui, Wang, & Pigot, 2010) (Fern, Komireddy, & Burnnet, 2007). (Karikrishna, 2011) utiliza el algoritmo *GSP* para clasificar secuencia de imágenes (situaciones) en un partido de cricket registrado en vídeo, procesando secuencias de actividades relacionadas con las

secuencias de imágenes. (Chikhaoui, Wang, & Pigot, 2010) utilizan *GSP* para buscar patrones de actividades que generan las personas durante sus rutinas diarias grabadas en vídeo, el objetivo era distinguir conductas individuales. Los resultados muestran que existen diferencias claras entre las situaciones individuales y generales de las personas durante el desarrollo *AVD*. Es relativamente fácil de modelar las situaciones normales que generan las personas. Por ejemplo, las *AVD* de una persona en su casa en la mañana suele ser el mismo, y por lo tanto se puede modelar a priori (Chikhaoui, Wang, & Pigot, 2010). Cualquier situación diferente, es sospechoso (Williem, Vamsi, Boles, & Wageeh, 2008). En comparación, es difícil definir situaciones sospechosas de la gente en un supermercado, puesto que ese puede combinarse con situaciones normales. En este caso, sin embargo, se puede tratar de inferir perfiles que basados en patrones de situaciones humanas sospechosas, que sean de interés para el personal de seguridad del supermercado (Aguas, 2010).

(Hong, Lin, & Wang, 2006) explican que el problema de descubrir patrones secuenciales es encontrar las secuencias de actividades representativas para una situación particular

Si bien es cierto, los algoritmos de procesamiento de imágenes y multisensoriales logran identificar objetos, personas e inclusive actividades en los escenarios, mantienen el problema de la brecha semántica, que consiste en relacionar directamente los eventos con las situaciones de interés, sin especificar los niveles de abstracción semántica que permiten llegar hasta ellas. La existencia de esta brecha da paso a pensar que se deben

tener modelos de composición jerárquica de niveles semánticos, con el fin de inyectar el detalle semántico a la relación eventos – situación de interés. Para obtener esos modelos por lo general se hace uso de lenguajes que permiten convertir la descripción que realiza el experto de la situación a modelar, a un nivel de abstracción que luego puede ser utilizado por herramientas de inferencia semántica.

2.4. Modelado de la situación de interés a partir de lenguajes de descripción de eventos y actividades.

(Bai, Dick, & Dinda, 2009) desarrollaron microlenguajes que permiten modelar actividades de alto nivel semántico. El objetivo describir los eventos que provienen de cualquier *WSN* utilizando un lenguaje común, favoreciendo con ello la interoperabilidad entre diferentes *WSNs* (Ramirez, Royo, Olivares, & Roncero, 2010)

En un sistema de *SV* basado en señales de videocámaras, se desarrollaron lenguajes para consultar e indexar eventos vídeo, los cuales permiten componer consultas tanto a nivel de la imagen como a nivel de actividades y situaciones.

Ontology Web Language (*OWL*) ha sido utilizado por algunas investigaciones para modelar actividades de alto nivel semántico (Maditskos, Dasiopoulou, Efstathiou, & Kompatsiaris, 2013) (Bettini, y otros, 2010). Este lenguaje permite la conceptualización de elementos del escenario y las relaciones espacio temporales necesarias para componer los niveles semánticos. Este lenguaje puede trabajar con reglas que permiten la inferencia de las situaciones de interés. Para ello, generalmente hace uso del lenguaje

Semantic Web Rule Language (*SWRL*), el cual permite la generación de reglas de inferencia a partir del conocimiento estructurado en una *OWL* (Horrocks & Patel, 2009).

(Nevatia & Bredmond, 2001) (Nevatia & Hobbs, 2004), proponen la utilización de los lenguajes *Video Event Representation Language* (*VERL*) y *Video Event Markup Language* (*VEML*) en el dominio de la representación de actividades en un vídeo. El lenguaje *VERL* es utilizado para diseñar una ontología de eventos de vídeo en lenguaje natural. *VERL* tiene las mismas características de un lenguaje de programación tradicional (manejo de estructuras, expresiones, etc.). *VERL* añade a sus características el manejo de reglas de inferencia y el manejo de relaciones temporales. Esta última característica le permite a *VERL* conocer los eventos que se dan antes y después de un tiempo *t*, y con ello poder reconstruir escenarios en lenguaje natural a partir de un *t* predeterminado. Se conoce como escenario a la parte de un video que se considera importante para el análisis. La semántica que se obtiene a partir del uso del lenguaje *VERL*, ha permitido la descripción de la granularidad (número de personas que actúan en un vídeo), texto y audio de un vídeo. Estas descripciones son muy importantes en los dominios relacionados con videovigilancia. El lenguaje *VEML* es utilizado para dar una marca de importancia a ciertos eventos en el vídeo. Por ejemplo, si en un vídeo se trata de identificar el hurto de un objeto se usará *VERL* para representar a los eventos en general. *VEML* se usará para marcar al evento objeto no aparece en el entorno. La semántica de *VEML* permite alertar en lenguaje natural lo sucedido en el vídeo. En este trabajo utilizamos a la ontología de base de *VERL*, puesto que la misma se acopla a nuestra propuesta de la jerárquica de inferencia de situaciones y actividades. Nuestro

trabajo también complementa *VERL* adicionando relaciones semánticas durante el proceso de inferencia. Este es el lenguaje que utilizaremos para modelar las actividades de alto nivel semántico, pues como se ha descrito permite el modelo de actividades de forma jerárquica, permite el manejo de restricciones y parámetros necesarios para componer una situación de interés. De plano es el lenguaje que más se acerca a nuestra propuesta. Durante la experimentación, nosotros también añadimos funcionalidades a *VERL* pero sin modificar su estructura fundamental de modelado. Además, hemos comprobado que a pesar de que fue desarrollado para conceptualizar eventos y actividades de video, en principio; la señal de video puede ser cambiada sin inconvenientes por señales multisensoriales.

Se ha detallado el lenguaje que permite modelar las situaciones de alto nivel semántico. En esta tesis, diseñamos herramientas que permiten a *VERL* utilizar las conceptualizaciones de *CHAO* para modelar la situación de alto nivel semántico. Esto se logra especialmente cuando se modela situaciones de conocimiento *a priori*, es decir; cuando el experto es conocedor del detalle suficiente como para describir la situación de interés. La utilización de las ontologías para modelar situaciones de alto nivel semántico se describe a continuación.

2.5. Uso de ontologías para el modelado de situaciones de alto nivel semántico.

Estructuras de conocimiento que conceptualizan diferentes tipos de sensores (radio frecuencia, videocámaras), con el fin de constituir redes de información sensorial han sido muy desarrolladas en los últimos años. Estas estructuras, carecen de herramientas que permitan gestionar y analizar la información que producen (Balanzinska, y otros, 2007). Para llenar esa carencia, se han diseñado estructuras semánticas que permiten la conceptualización los elementos de una *WSN*. Un ejemplo de este tipo de ontologías es *ONTOSENSOR*, diseñada sobre escenarios de vigilancia militar, con el fin de lograr una descripción semántica de los sensores ubicados en zonas de *Monitoreo* y la conceptualización de eventos de bajo nivel semántico (Maillot, Thonnat, & Boucher, 2004). La composición de eventos de bajo nivel semántico conduce a la conceptualización e inferencia de actividades (Akdemir, Turaga, & Chellapa, 2008). Una ontología que permite conceptualizar las actividades que se registran en un escenario es propuesta por (Fernández & González, 2007), –conceptualiza actividades como: caminar en la calle en la tarde, observando producto de una estantería, tomar objeto del escritorio–. Para modelar a las actividades humanas, es necesario diseñar una ontología que permita describir entidades (personas, objetos), escenarios, interacciones entidad-escenario, y sus relaciones espacio-temporales (Akdemir, Turaga, & Chellapa, 2008). Ejemplos de este tipo de ontologías son:

- *CAVIAR*: conceptualiza escenarios aplicada a plazas públicas y a un centro comercial (Town, 2006).

- *CARETAKER*: diseñada para conceptualizar las actividades ejecutadas por las personas en un banco (Bredmond, Corvee, Patiño, & Thonnat, 2008).
- *DOLCE*: diseñada para conceptualizar actividades de alto nivel semántico. Puede ser utilizada para relacionar los eventos directamente con la situación de interés (Gangemi, Guarino, Masolo, Oltramani, & Shneider, 2002).
- *SOUPA* fue diseñada por el grupo de investigación en Web Semántica[6] (*UbiComp Special Interest Group*) cuyo objetivo fue el compartir términos entre ontologías mas no importarlos directamente. Considerando la semántica de importación de ontologías propuesta por Bechhofer, *SOUPA* comparte términos con las siguientes ontologías[7]: *Friend-Of-A-Friend* (FOAF), *DAML-Time* y subontología temporal, Ontologías de manejo espacial(*BuildingArchitecture*), Ontologías para el cálculo de regiones *RCC*, *COBRA-ONT*, *MoGATU BDI* y con la ontología *Rei* para el manejo de políticas (Bechhofer, y otros, 2004) (Brickely & Miller, 2003) (Powers, 2003) (Hobbs, 2002) (Pan & Hobbs, 2004) (Douglas, Guha, & Guha, 1990).
- *OBAR* ontología que se utiliza para inferir eventos a partir de la fusión sensorial. El proceso para instanciar las clases de esta ontología empieza extrayendo información desde una base de datos en la cual se guarda la información de los sensores. Luego, los razonadores actúan sobre sus axiomas y obtienen los eventos de bajo nivel

[6] http://pervasive.semanticweb.org

[7] Una de las maneras como *SOUPA* comparte sus términos las ontologías nombradas anteriormente, es usando las propiedades *owl:equivalentClass* y *owl:equivalentProperty* definidas en el estándar *OWL* (Web Ontology Language, http://www.w3.org/TR/owl-features/)

semántico relacionados con las actividades de la vida diaria de las personas (Wongpatikaseree, 2012).

- *MODULAR ARCHITECTURE DESIGN*: Ontología especializada en conceptualizar planos arquitectónicos (Hois, Bhatt, & Kutz, 2009).
- *FOAF* permite conceptualizar la información de una persona y las relaciones interpersonales, y es útil para la creación de sistemas de información que apoyan a las comunidades en línea. (Dumbil, 2002).
- *TIME*[8] se ha diseñado para conceptualizar elementos temporales y las propiedades comunes a cualquier formalización del tiempo.

Las ontologías descritas anteriormente, pueden ser utilizadas para componer los niveles semánticos que permitan modelar la situación de interés, en donde el nivel bajo se corresponde con el modelado de los sensores, el nivel medio es la conceptualización de los eventos que provienen del modelado sensorial, y el alto nivel es la combinación de los dos anteriores. La teoría de composición de niveles ha sido descrita en (Castillo, Fernández-Caballero, Serrano-Cuerda, & Sokolova, 2012).

La composición de niveles semánticos, necesita también de ontologías que permitan inyectar semántica a partir de las relaciones espacio temporales que registran los eventos un un escenario (Martínez-Tomas, Rincón, Bachiller, & Mira, 2008). (Hois, Bhatt, & Kutz, 2009) desarrollaron una ontología que permite obtener la descripción

[8] *Time*.owl: http://www.w3.org/TR/owl-time/, tomado el 21-10-2013

espacial de un contexto así como las relaciones espaciales, tipo Región Conecting Calculus (*RCC*) que ocurren entre los elementos del mismo. (W3C, Time Ontology in OWL , 2006) (Hobbs, 2002) desarrollaron ontologías que permiten conceptualizar instantes, periodos y relaciones temporales. Estas ontologías pueden ser utilizadas para conceptualizar eventos y sus relaciones espacio temporales y a partir de allí componer los niveles de jerárquico semánticos.

En un sistema de *SV* centrado en el usuario, la información del escenario siempre girará en torno a este, desde la captación del estado del entorno para adaptarlo a las preferencias del usuario, hasta la observación del estado anímico del usuario para establecer un escenario favorable al mismo. Se han desarrollado plataformas de agentes inteligentes con una adaptación en tiempo real a los cambios en el escenario, para proporcionar una adaptación de sus servicios a la ubicación de los usuarios, sus preferencias o el estado de los dispositivos multisensoriales. Es decir, se utilizan las tecnologías de *Web Semántica* para permitir la comunicación entre la semántica del escenario y los procesos de razonamiento, con el fin de proporcionar una adaptación del entorno a las preferencias de los usuarios. Para que esto sea factible, los sistemas proporcionan, no solamente un sistema domótico cerrado con las funciones más o menos detalladas en un documento de especificación, , sino una plataforma abierta en la que cada servicio domótico venga suficientemente definido para llevar a cabo su funcionalidad, pudiendo colaborar con el resto de servicios del sistema –Aplicaciones domóticas para oficinas o despachos profesionales dentro del marco del proyecto Desarrollo de un Sistema Dinámico Colaborativo para el Control “Inteligente” de

Oficinas – OFIDOMO(Programa de Ayudas a la Trasferencia de Investigación de la Universidad de Granada, 2008), utilizan este tipo de propuestas (Rodriguez & Holgado, 2008).

Si bien es cierto, que existen aplicaciones que intentan combinar ontologías, y algoritmos informáticos, estas funcionan por separado. Nosotros proponemos que deben estar acopladas en un solo marco conceptual que facilite el intercambio de conceptualizaciones y permita su reutilización con el fin de lograr inferencias con alto nivel semántico. Trabajos que tienen relación con esta propuesta se detallan a continuación.

2.6. Diseño de Marcos con tecnologías semánticas

La interoperabilidad semántica de alto nivel no está resuelta debido a que falta un modelo conceptual común (Reed, Botts, Davidson, Percivall, & Collins, 2007), que agrupe a cada uno de los sistemas de *SV*. Algunos esfuerzos en *SV* se han desarrollado solamente para resolver el problema de realizar consultas sobre la semántica multisensorial (Lewis, Cameron, Xie, & Arpinar, 2006). El consorcio *GeoEspacial* (*OGC*)[9], trabaja sobre estándares abiertos de consultas semánticas en sistemas multisensoriales, siendo su principal problema; el tiempo de respuesta en especial cuando el volumen de datos aumenta, situación a la que dio arreglo *IRISnet* (Gibbons, Karp, Ke, & Seshan, 2003), la cual con su mecanismo de consultas jerárquico semántico

[9] Consorcio GeoEspacial: http://www.opengeospatial.org

logro mejorar el tiempo de respuesta de las consultas semánticas, realizadas sobre fuentes heterogeneras sensoriales, sin embargo; tiene problemas de reutilización de la estructura semántica, lo que dificulta su aplicación para diferentes escenarios. (Moodley & Simonis, 2005), diseño la arquitectura *SWAP* que trabaja con datos multisensoriales e infiere actividades de alto nivel. *SWAP* tiene tres niveles que comprenden, el sensor, el objetivo, y la decisión. Cada uno de los niveles es operado por un agente. La operabilidad semántica de *SWAP* permite que se pueda reutilizar sus conceptualizaciones en distintos sistemas de *SV*. Sin embargo, el trabajo con multiagentes es demasiado complejo lo dificulta la operabilidad de este sistema. (Konstantinou, Solidakis, Zoi, Zafeiropoulos , Stahopoulos, & Mitrou, 2007) desarrollaron *Priamos*, una arquitectura *middleware* que trabaja con datos multisensoriales y que trata de inferir situaciones de alto nivel semántico. Su facilidad de uso ha hecho que pueda ser utilizada para el diseño de interfaces específicas de sistemas de *SV*. Sin embargo, carece de un frontal para el manejo sencillo de sus operaciones por parte del usuario lo que hace que se haga difícil su portabilidad.

Una estructura semántica integral para la fusión de información multisensorial y procesamiento algorítmico, requiere de la relación con el escenario, ya que diferentes grupos conceptualizan y utilizan sensores de datos de diferentes maneras. Además, se debe planificar una estructura que permita a la información relacionarse entre sí con el fin que se pueda inferir actividades o situaciones según el tipo de escenario *Monitorizado*. Debe ayudar a escoger cuales son los sensores apropiados para poder inferir una actividad o situación determinada –la *Monitorizacion* de las funciones

cerebrales ayuda a identificar problemas de comportamiento humano, para procesarlas se puede escoger Electrocardiogramas o Resonancia Magnética (Parry, 2008) o para situaciones mas abstractas los sensores de movimiento combinados con los sensores de flujo de agua, ayudan a inferir *AVD* como desayunar, tomar medicinas, ver televisión (Hongeng & Nevatia, 2001)–.

En los últimos años, las redes de sensores inalámbricas han despertado un gran interés, dando lugar a numerosas oportunidades y líneas de investigación. La idea expuesta en (Colitti, Steenhaut, Descouvemont, & Dunkels, 2008) sobre *Satellite Sensor Networks* es una prueba del desarrollo de marcos de trabajo que buscan la expansión de las redes de sensores. El marco desarrollado allí, ayuda a los investigadores a interactuar con datos de otras redes en cualquier parte del mundo. Esta idea es más fácil de plasmar si la comparamos con el actual modelo de redes sociales con el que millones de personas interactúan en todo el mundo. Entidades individuales (o colectivas) comparten su información con el resto de la red, dando lugar a dos hechos: 1) el resto de entidades pueden acceder a estos datos y 2) la red, por sí sola, puede tratar estos datos y darles uso para acciones posteriores.

La combinación de redes sensoriales y ontologías han dado muy buenos resultados para el diseño e implantación de marcos. Un ejemplo de ello es el marco-arquitectura *Video-Sensor Distribuido* el cual se trabaja con la ontología *CORBA*, con el fin de desarrollar una herramienta distribuida basada en estándares, que permita la construcción de sistemas de *Monitorización* visual inteligente –a partir de una sola imagen obtenida del

video generado en distintos lugares y un Código de Identificación Personal (PIN), se puede verificar la identidad de la persona que aparece en ella–. *CORBA* define la especifica los estándares necesarios para trabajar en escenarios heterogéneos y un algoritmo basado en redes neuronales permite identificar a una persona a partir de su rostros (Guzmán & Cabello, 2013).

En aplicaciones multisensoriales de geoposicionamiento, la personalización de las aplicaciones *SIG* móvil se puede realizar mediante el uso de la tecnología de Web Semántica, que proporciona diferentes herramientas para almacenar información relacionada con las preferencias del usuario. Un ejemplo de ello son los marcos semánticos turísticos que muestran información turística personalizada basada en las preferencias del usuario. El uso de la semántica en estas aplicaciones aparece, ya que cuando se utilizan datos geográficos las bases de datos relacionales se vuelven insuficientes, para lo cual se dispone de una ontología espacial y mecanismos de consulta de datos de la misma que ayuda a resolver problemas de localización espacial cuando la primera alternativa ha fallado (Descamps-Vila, Casas, Conesa, & Pérez-Navarro, 2008).

El grupo *SIMDA* (SIMDA, 2008) en los cuales se ha desarrollo *Horus*. En este marco distribuido-híbrido los nodos remotos realizan el procesamiento de nivel inferior, así como la adquisición de datos, mientras que un nodo central se encarga de la recogida de la información y de su fusión. El marco incluye detección de objetos simples y el

seguimiento y detección de actividades. Para lograrlo se desarrollaron los siguientes niveles (Castillo, Fernández, & López, 2011):

Fusión de información sensorial: Este nivel es el encargado de la fusión de los datos de los sensores para mejorar la calidad de la información (más completa y precisa).

Localización y filtrado: Aísla los objetos de interés contenidos en las imágenes de entrada.

Localización y filtrado de fusión: Este nivel fusiona imágenes obtenidas en la localización y la etapa de filtrado ya que puede haber varios resultados de localización y métodos de filtrado que se ejecuta en el marco (por ejemplo, uno dedicado a las imágenes en color y otra para imágenes infrarrojas).

Detección de Blob: El nivel de detección *blob* filtra errores en las imágenes y los detecta correctamente en los niveles anteriores. Es el encargado de la extracción de la información asociada a los errores para permitir un análisis más eficaz de los objetos.

Identificación del objeto: Este nivel opera con objetos aislados de los errores. Esto mejora la captación de información y produce la cartografía de las coordenadas del objeto en el mundo real en lugar de simplemente operar con coordenadas de imagen.

Clasificación de objetos: Este nivel es especialmente importante, aquí se realiza un buen análisis de la actividad, ya que proporciona conocimientos sobre "qué" es el objeto. Además, ejecuta la clasificación de objetos y puede proporcionar información acerca de la orientación de los objetos.

Seguimiento de objetos: Este nivel es el encargado de trazar las coordenadas de los objetos de imagen en un escenario. Este nivel utiliza la información del modelo común que se refiere al mapa, la situación de los sensores y su rango de cobertura.

Detección de eventos: El nivel de detección de evento genera información semántica relacionada con el comportamiento de los objetos en el escenario.

Fusión Evento: En un sistema de seguimiento e interpretación multisensorial, en donde varios sensores monitorean un escenario común. Se explica aquí que los eventos generados a partir de diferentes fuentes por lo general no se corresponden. Por ello, es necesario unificar la información procedente de los diferentes datos sensoriales generados en el nivel anterior al nivel de fusión evento.

Detección de actividad: Este nivel final de la arquitectura está a cargo del análisis y la detección de las actividades ya asociados a las características temporales. Después de la fusión caso, el nivel actual tiene un mejor conocimiento de lo que está sucediendo en el escenario de acuerdo con los eventos detectados.

Modulo general: recoge toda la información de los diferentes niveles mientras que proporciona primitivas para acceder a la información.

Modelado Escenario: A pesar de que el modelado de escenarios no aparece como un nivel dentro del marco de definición, es un aspecto clave que permite trabajar con la información del sensor para situar los objetos en el escenario.

En esta tesis, *Horus* es nuestro director de orquesta, debido que tiene la metodología necesaria para modelar e inferir situaciones de interés a partir de niveles de composición jerárquica. Nuestro aporte en *Horus*, es un marco conceptual, metodológico y tecnológico que permitirá inferir situaciones de alto nivel semántico a partir de su módulo de fusión multisensorial. El marco descrito en esta tesis se compone de capas y módulos que permiten que: 1) el experto describir la situación de interés, 2) el ingeniero de conocimiento abstraiga detalles de la situación y la modele en niveles de composición jerárquico semántica y 3) Exportar los modelos a lenguaje *OWL* que permita trabajar con herramientas de inferencia semántica. Los detalles del marco propuesto en esta tesis se detalla en al apartado de metodología descrito a continuación.

3 Marco conceptual, metodológico y tecnológico para el modelado de alto nivel semántico

En este apartado, describimos el marco que facilita el establecimiento de una conexión semántica, entre las señales físicas multisensoriales y las situaciones de interés. El marco se utiliza con el propósito de componer actividades y llegar a la interpretación de lo que ocurre o ha ocurrido en el escenario, tal como lo haría un experto observador humano. En la Figura 3-1, se muestra un esquema del mismo:

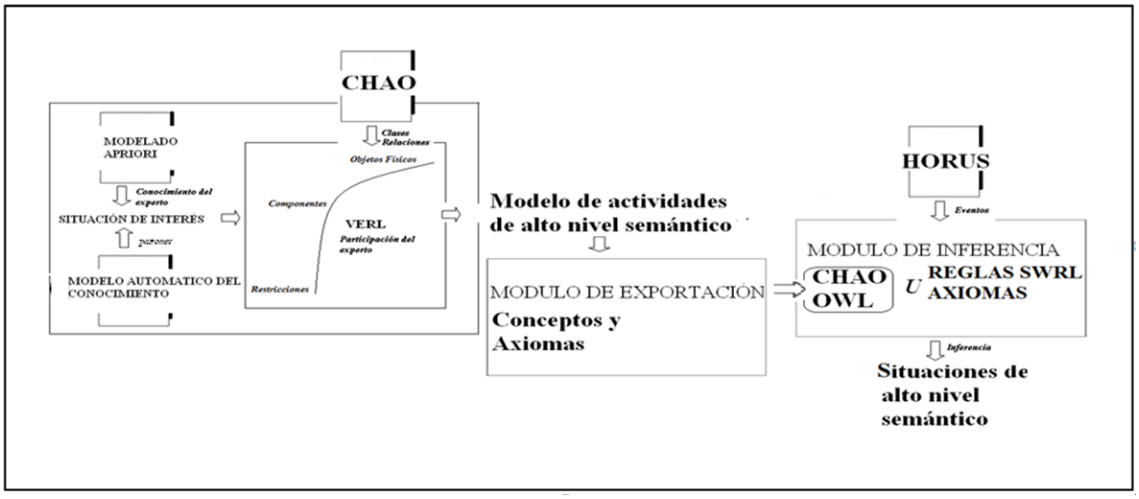


Figura 3-1. Marco conceptual, metodológico y tecnológico para el modelado del alto nivel alto semántico

En la Figura 3-1, se observan los componentes del marco propuesto en esta tesis. A través del mismo, modelamos el conocimiento *a priori* que viene de la descripción de un experto y; modelamos de forma automática el conocimiento cuando este no está claramente definido, por medio de *micropatrones* correlacionados con una situación de interés. El modelo de conocimiento, se utiliza como base para componer actividades de

alto nivel semántico en *VERL*. Esta composición es apoyada por las conceptualizaciones de *CHAO*.

En el módulo de exportación desarrollamos herramientas que permiten tomar los conceptos y restricciones modeladas en *VERL* y convertirlas en clases y axiomas en *OWL*. *CHAO* unida con reglas *SWRL*, axiomas y algoritmos permiten la inferencia de situaciones de alto nivel semántico.

El marco conceptual, tecnológico y metodológico, lo vamos a utilizar para modelar situaciones de alto nivel semántico. La composición de las situaciones es jerárquica, lo que nos permite trabajar con la abstracción por cada nivel. La descripción de la composición de situaciones se detalla a continuación.

3.1 Descripción de situaciones de interés

Para describir una situación de alto nivel semántico, se necesita la participación de experto, como se puede observar en la Figura 3-2:

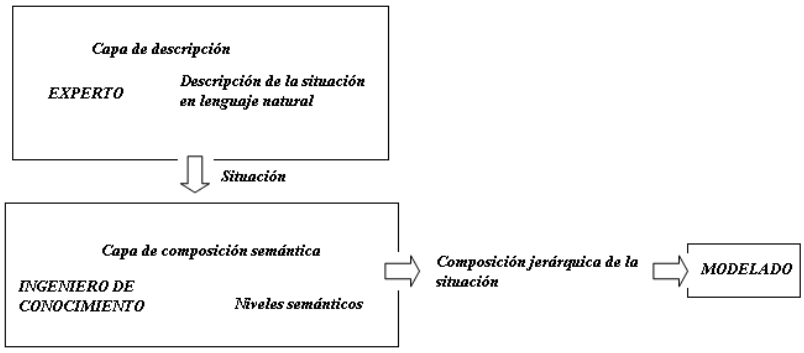


Figura 3-2. Capas para obtener el modelado de la situación de interés

La descripción de la situación es responsabilidad del experto y la conceptualización de la misma es responsabilidad del ingeniero de conocimiento. El proceso de descripción y conceptualización son manejados por capas las mismas que se puntualizan a continuación.

3.1.1 Capa de descripción

En dependencia del tipo de escenario y del conocimiento *a priori* de las situaciones de interés, existen dos posibilidades para describir una situación: 1) que existe conocimiento experto humano en una forma explícita y 2) que este conocimiento no existe *a priori*, aunque es posible, en principio, encontrar la información mediante el análisis de grabaciones anteriores o secuencias de vídeo. En este segundo escenario, se requiere un proceso particularmente complejo (Orten, 2005), (Tian, Brown, Hampapur, & Lu, 2008) (Masseglia, Poncelet, & Tesseire, 2009) para obtener datos adicionales o para modelar los procedimientos necesarios que permitan obtener el conocimiento adecuado de la situación de interés. El marco diseñado en esta tesis, nos sirve para modelar situaciones de alto nivel semántico en ambos casos. Para ello, la participación del experto es fundamental ya que es aquel que posee el conocimiento *a priori* o entiende los resultados del modelado del conocimiento automático convirtiéndolo en conocimiento claro fácil de manejar. Además, ayuda al ingeniero a estructurar el conocimiento de forma semántica. En esta capa el experto describe a la situación con el conjunto de detalles que permitan entender cómo se registra en el escenario. El resultado de la capa de descripción es la entrada para la capa de composición semántica en donde se obtiene un primer modelo de la situación de interés.

3.1.2 Capa de composición semántica

La descripción de la situación de interés, es codificada por el ingeniero de conocimiento en componentes semánticos. Estos componentes semánticos se utilizan para ir armando el modelado jerárquico necesario para componer los niveles de abstracción semántica. Por ejemplo, si la descripción de la situación de interés es la persona en la sala, toma el control remoto del televisor y luego se sienta en el sofá para ver una película. Una de las formas en que el ingeniero del conocimiento puede conceptualizar esta situación se muestra en la Figura 3-3:

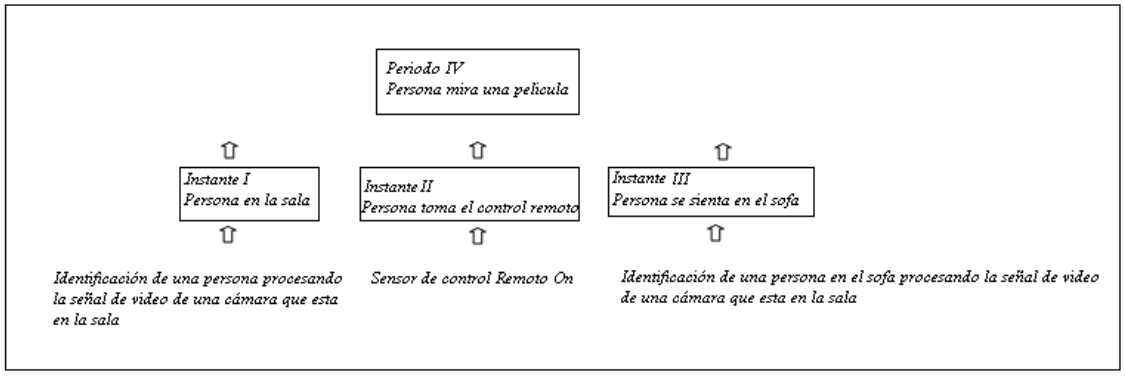


Figura 3-3. Composición semántica de una situación de interés

De acuerdo a lo que va indicando el experto, el ingeniero de conocimiento se va dando cuenta de que elementos necesita para conceptualizar la situación, a tal punto que vaya adaptándose, al marco de composición semántica. Cuando el experto describe la situación hace referencia sitios e intuitivamente instantes de tiempo en los cuales la persona registra un evento o actividad, pero es responsabilidad del ingeniero del conocimiento describir en un primer modelo semántico lo que el experto describe como situación de interés. La composición de situaciones es jerárquica, y es que es una de las

formas más adecuadas de modelar con criterio de abstracción (Martinez-Tomas & Rivas-Casado, 2009), tal como se puede observar en la Figura 3-4:

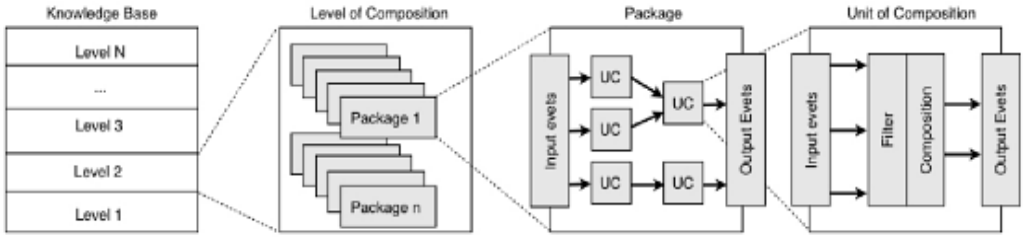


Figura 3-4. Arquitectura de composición jerárquica de eventos. Fuente. (Rivas-Casado & Martínez-Tomás, 2011)

Este trabajo, utiliza la arquitectura que se muestra en la Figura 3-4 a través de la cual se puede componer actividades a partir de las señales sensoriales, y situaciones a partir de actividades. Todo con un aporte de detalle semántico en el cual se puedan responder a preguntas de interés –en dónde ocurrió, qué características tenía el sitio, cuáles actividades desarrolló la persona–.

Nuestro trabajo para este apartado, se deriva de la necesidad de desarrollar un mecanismo que facilite la descripción y el modelado de escenarios, actividades y situaciones que se basan en la terminología y la información (objetos y actividades), que se obtienen durante el proceso de identificación y el seguimiento de objetos de interés en un escenario (Carmona, Rincon, Bachiller, Martínez-Cantos, Martínez-Tomás, & Mira, 2009), y en la identificación de los movimientos o actividades de bajo valor semántico, tales como; entrar en el campo de visión de una cámara y la identificación por *RFID* (identificación mediante radio frecuencia). Todos estos eventos están destinados a ser reconocible por las redes multisensoriales o sistemas de visión

artificial. Nuestro objetivo es mostrar las posibilidades que tiene el marco para la creación de un marco unificado, como la que se muestra en la siguiente figura.

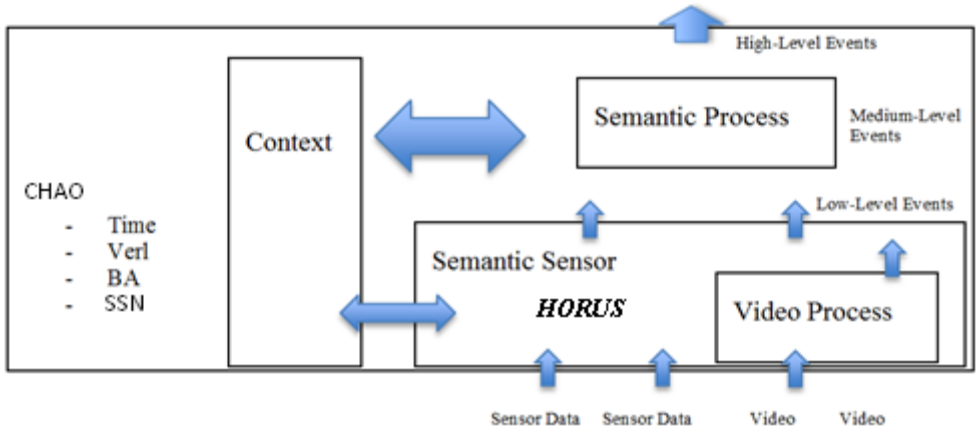


Figura 3-5. Marco unificado para inferir actividades y situaciones de interés

Como se observa en la Figura 3-5, nuestra contribución, así como las tecnologías semánticas disponibles, nos ayuda con dos procesos: 1) la información obtenida de los sensores y procesamiento de vídeo (eventos de bajo nivel) unificador. Con eventos semánticos de nivel medio, el proceso global puede ser visto como un sensor semántico que tiene una relación fija con el escenario (contexto). 2) El uso de la información semántica de los sensores: los eventos (actividades) de nivel medio se utilizan para crea (actividades) eventos de alto nivel semántico. En esta jerarquía, el nivel semántico tiene una relación fija con el escenario. Con actividades como el sensor se activa, se considera que es casi independiente (es decir, si tenemos en cuenta el hecho de que se encuentra dentro de una perspectiva espacio-temporal predeterminada en términos del escenario), pero donde la actividad de interés se considera que es mucho más independiente. Por otro lado, utilizamos la naturaleza abstracta del segundo nivel de ayuda para determinar el tipo de sensor que se utiliza. Y en este sentido, nuestra

propuesta es perfectamente reutilizable. En nuestro estudio, hemos demostrado que el uso de niveles semánticos basados en ontologías facilita el modelado de escenarios y de la inferencia de las actividades y situaciones de sospecha –que puede complementar los resultados expuestos (Mechsner, 2012) (Bickmore, Schulman, & Sidner, 2011) (Botia, Villa, & Palma, 2009). Además, nuestra ontología funciona con los sistemas multisensorial, con el objetivo de generar alertas. Por tanto, era necesario inferir situaciones que claramente representen o reflejen comportamientos sospechosos.

En resumen, se propone el uso del marco basado en los niveles semánticos para modelar el contexto y los niveles situación que ayudan a diferenciar los niveles que se han fijado las relaciones con los mecanismos de recogida de información de otros niveles que no tienen esta relación. En ambos casos, los mecanismos semánticos son reutilizables mediante el uso de la arquitectura mostrada en la figura anterior. Además con ello se pone en práctica el uso de las tecnologías semánticas a fin de permitir la inferencia de situaciones de alto nivel semántico.

Una vez que el ingeniero del conocimiento, tiene una idea clara y precisa de la situación, empieza a utilizar los elementos semánticos que dispone el marco para modelar la situación de interés. Uno de esos elementos es *VERL* el mismo que se describe a continuación.

3.2 Video Event Representation Language (*VERL)*

Para modelar actividades hacemos uso de *VERL*, puesto que ha sido utilizado con éxito en algunas investigaciones relacionadas al modelado de eventos (Nevatia & Bredmond, 2001) (Nevatia & Hobbs, 2004) (Bremond, Maillot, Thonnat, & Vu, 2004). Su ontología se describe a continuación.

3.2.1 Video Event Representation Language (Ontology)

En la Figura 3-6 se muestran las clases de *VERL*:

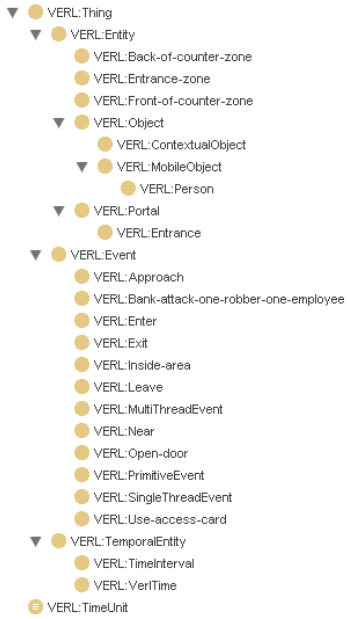


Figura 3-6. Ontología de *VERL*

En donde:

VERL:Entity es una clase que conceptualiza a los elementos del escenario en donde se registran las situaciones de interés.

VERL:Object conceptualiza a los objetos en el escenario. Estos pueden ser de dos tipos *contextuales* cuando pertenecen al escenario en sí mismo y carecen de movimiento propio, y *móviles* cuando pertenecen al escenario pero tienen movimiento propio.

VERL:Event conceptualiza a los eventos. Cada evento tiene parámetros, los cuales quedan expresados en la ontología como axiomas, como se muestra en la Figura 3-7:

Figura 3-7. Axiomas para componer eventos en *VERL*

El axioma *VERL:eventsArgs min 1*, es una restricción que fuerza a que cada evento tenga como mínimo un argumento. Este argumento puede ser un objeto físico, un estado u otro evento.

VERL:eventInterval min 1, restricción que hace referencia al período temporal en el que se espera la ocurrencia de un evento.

VERL:eventPredication min 0, restricción que indica que un evento puede o no estar relacionado con una acción. Esto es muy importante puesto que en ocasiones solo interesa modelar un evento independientemente de su acción. Por ejemplo, el evento persona camina puede ser modelado independientemente si la acción relacionada es Persona hace ejercicio. Todos estos axiomas son fundamentales a la hora de modelar una actividad, pues permiten modelar estados y eventos compuestos a partir de estados y eventos simples.

Para explicar de mejor manera los argumentos con los cuales se modela una actividad, en los siguientes apartados se describe el detalle los mismos.

3.2.2 Meta Conceptos para la descripción de objetos físicos

La clase *physical object* representa a todos los objetos del mundo real, que pueden ser identificados a través de una cámara o procesamiento sensorial.

La propiedad *atributes* son todas las características propias del objeto físico por ejemplo color, estatura, género y demás características determinantes.

La propiedad *liveliness* sirve para conceptualizar la movilidad y autonomía de un objeto físico.

La propiedad *current liveliness* es la característica observable de un objeto en un escenario en un determinado momento. Puede tomar dos valores: *mobile* y *contextual*. Cualquier objeto con *current liveliness = mobile*, se reconocerá como *mobile object*.

Un objeto será tomado como *mobile object* si en su estado inicial el objeto es identificado en movimiento. El tipo de movimiento del objeto sirve para asociarlo con su clase. Así pues, en función de su movimiento podemos deducir si son personas, puertas, juguetes, etc. La clase *agente* puede ser un tipo de *mobile object*, cuando la propiedad *current liveliness* tiene los siguientes estados: *remotely-movable*, *programable* y *autonomous*.

Un objeto con *current livelines=contextual* se reconocerá como *contextual object*, esto implica que no puede cambiar su posición en un escenario a menos que un objeto tipo

mobile object este desplazándolo. Un objeto tipo *contextual object* en un escenario está sujeto a los siguientes estados: *automatically-movable at the same position*, *fixed*, *static*, *movable at the same position*, *remotely-movable at the same position*, *displaceable*. Ejemplos de *contextual objects* son puertas, ventanas, muebles y árboles. El estado de un *contextual object* puede variar en el tiempo. Por ejemplo, si se tiene por ejemplo un carro sin conductor, el carro es un *contextual object*, mientras que si el carro tuviera conductor y estuviera en movimiento, el carro es clasificado como *mobile object*. La propiedad *state* representa al conjunto de todos los posibles estados de un objeto en un intervalo de tiempo. Esto permite identificar claramente a un objeto tipo *mobile object* o a un *context object*.

La granularidad es otro punto a considerar cuando se habla de tipos de objetos. Por ejemplo, si consideramos un grupo de personas cruzando la calle esto podría interpretarse como un *mobile object* único en lugar de múltiples *mobile objects*. Se debe considerar además, la complejidad de algunos objetos. Una persona por ejemplo es capaz registrar simultáneamente diferentes acciones con cada parte de su cuerpo. Según la acción que realiza cada parte del cuerpo puede considerarse como un objeto móvil; por lo tanto se debe tomar en cuenta que un número finito de objetos tipo *mobile object* pueden formar parte de otro objeto móvil, pero estos objetos deben considerarse como un ente único.

El *Rol* es una propiedad que representa el comportamiento del objeto basado en determinadas circunstancias; le da un contexto por así decirlo. Por ejemplo, el comportamiento de una persona no será el mismo en un día lluvioso que un día soleado.

Existen muchas otras propiedades (atributos) observables en objetos físicos como velocidad, color, posición y otros que se representan como *visual atributes*. Estos son de tres tipos: *position-base* (basados en posición), *global appearance* (apariencia global), *local appearance* (apariencia local). *Visual atributes* de tipo *position-base* pueden ser trayectoria, posición, dirección. *Visual atributes* de tipo global *appearance* pueden ser altura, color, tamaño. *Visual atributes* de tipo local *appearence* pueden ser: postura y rostro.

Los *objetos físicos*, se utilizan como parámetros para modelar actividades. La descripción de los metaconceptos para modelar las actividades se describen a continuación.

3.2.3 Metaconceptos para modelar actividades en *VERL*

Existen diferentes Metaconceptos que se utilizan para caracterizar las actividades y sus interacciones en un escenario: estado, evento (primitivo, compuesto y simple / multiagente compuesto) y la actividad. En donde:

La clase *primitive state* se conceptualiza a estados espacio-temporales de un objeto en un período de tiempo.

La clase *composite state* conceptualiza a un estado compuesto por la unión de *primitive state* y también de otros *composites states*. Se conoce como componentes a todos los subestados de un *composite state* y *constraints* a todas las restricciones de relación que se pueden dar entre los *physical objects*, los *primitive state* y los *compose states*.

La clase *event* conceptualiza los cambios de estado que pueden darse a lo largo de un intervalo de tiempo o entre dos instantes continuos de tiempo (*meet*).

Un cambio de estado se conceptualiza en la clase *primitive event*. La clase *composite event* conceptualiza la agrupación de *states* y *events*, con el fin de componer otros *composite states.*

Un *single agent event* es una clase particular de evento en la que interactúa solo un objeto tipo *mobile object* –un grupo de personas que cambia de lugar en el metro–.

Un *multi agent event* involucra al menos dos objetos del tipo *mobile object* y cada *physical object* tiene su propio rango de movimiento.

Activity es un grupo de eventos interrelacionados cuya combinación no está del todo definida. La combinación de eventos está sujeta a la participación de *physical objects* con las mismas restricciones.

Un *action* es una clase particular de evento relacionada al cumplimiento de una tarea específica.

A continuación se presentan, ejemplos de la descripción de un estado primitivo, un evento primitivo y un agente multiagente compuesto, utilizando la sintaxis de *VERL*.

```
Model:
PrimitiveState Inside_zone
PhysicalObjects:
(p:Person,z:Zone)
Constraints
(p in z)
Instance:
S1: PrimitiveState Inside_zone (Paul,
Entrance_zone)
```

Código 3-1. Estado Primitivo Fuente: (Bremond, Maillot, Thonnat, & Vu, 2004)

```
Model:
PrimitiveEvent Changes_zone
        PhysicalObjects:
                ( p : Person, z1 : Zone, z2 : Zone)
                Components :
                ( (c1: PrimitiveState Inside_zone (p, z1))
                ( (c2: PrimitiveState Inside_zone (p, z2))
        constraints :
                ( (distance(z1, z2) <= 1m)
                (c1 before c2) )
        Instance:
        e1: PrimitiveEvent Changes_zone (Paul, Entrance_zone,Front_counter)
```

Código 3-2. Evento Primitivo Fuente: (Bremond, Maillot, Thonnat, & Vu, 2004)

```
Model:
CompositeEvent Bank_attack_one_robber_one_employ
        PhysicalObjects:
                ( e : Person[employee], r : Person[robber] )
        Components:
                ( (c1 : PrimitiveState Inside_zone (e, "Back_Counter" ))
                ((c2:PrimitiveEvent Changes_zone (r,
                Entrance_Zone","Front_Counter" ))
                ((c3 : PrimitiveState Inside_zone (e, "Safe"))
                ( (c4 : PrimitiveState Inside_zone (r, "Safe")) )
        Constraints:
                ((duration-of(c3) >= 1 second)
                (c2 during c1)
                (c2 before c3)
                (c1 before c3)
                (c2 before c4)
                (c4 during c3) )
Instance:
e2: CompositeEvent Bank_attack_one_robber_one_employee (Paul,Robber1)
```

Código 3-3. Multievento compuesto. Fuente: (Bremond, Maillot, Thonnat, & Vu, 2004)

VERL además posee estructuras de control (if...then) y repetitivas (while, repeat) que le permiten modelar algoritmos que sirven de base para las reglas semánticas (Nevatia & Hobbs, 2004). Estos forman parte de estructuras *Rules* y *Process*. Un ejemplo de este tipo de estructuras es el siguiente:

```
RULE(IMPLY(facility(x), container(x)) // all facilities
// are containers
RULE(IMPLY(portal(p), AND(container(c),
portal-of(p, c))))
RULE(IMPLY(AND(portal-of(p, c), open(p)), open(c)))
// portal open => container
PROCESS(far(ent x, ent y), NOT(near(x, y)))
PROCESS(outside-of (ent x, ent y),
NOT(inside-of (x, y)))
PROCESS(approach(ent x, ent y),
cause(x, change(far(x,y), near(x, y))))
```

Código 3-4. Ejemplos de reglas y procesos en *VERL*. Fuente: (Nevatia & Hobbs, 2004)

El Código 3-4 muestra un ejemplo de reglas y procesos que se utilizan para inferir eventos en el ejemplo de tracking de situaciones descrito en (Nevatia & Hobbs, 2004). Nosotros también utilizaremos esta estructura, puesto que necesitamos de reglas de inferencia con el fin de inferir la situación de alto nivel semántico. A continuación se describen las ontologías que forman parte de *CHAO* y cuyas conceptualizaciones pueden ser utilizadas en VERL para modelar actividades de alto nivel semántico.

3.3 Ontología *CHAO*

En este trabajo, a partir de las características temporales y espaciales de un evento diseñamos a *Conceptual Human Activity Ontology* (*CHAO*), la misma que a más de tener sus propias conceptualizaciones se apoya en las siguientes ontologías para modelar las situaciones de interés:

- Para conceptualizar a *SS* se utiliza la ontología *SSN*[10], que cumple con los estándares necesarios que le permiten conceptualizar la mayoría de sensores físicos. Emulamos el trabajo de *Horus* en los experimentos, partimos de la señal sensorial para inferir actividades de nivel medio semántico.
- Para conceptualizar el escenario se utilizó la ontología *BuildingArchitecture*, que conceptualiza los componentes de un escenario además de las relaciones espaciales (Hois, Bhatt, & Kutz, 2009). El término escenario se utilizará también como contexto.
- Para conceptualizar las relaciones temporales entre las actividades semánticas, se trabajó con *TIME*[11].

A continuación se detalla la descripción de las ontologías que conforman a *CHAO*:

[10] *SSN*: http://www.w3.org/2005/Incubator/ssn/, tomado el 22-10-2013.

[11] *Time* Ontology in OWL: http://www.w3.org/TR/owl-time/, tomado el 09-10-2013.

3.3.1 Ontología del Sensor Semántico

La ontología de redes de sensores (*SSN*) se tiene las clases que se utilizan para modelar los eventos, que resultan de procesar señales multisensoriales (W3C, Semantic Sensor Network Ontology, 2011). En la Figura 3-8, se observan los distintos componentes de *SSN*.

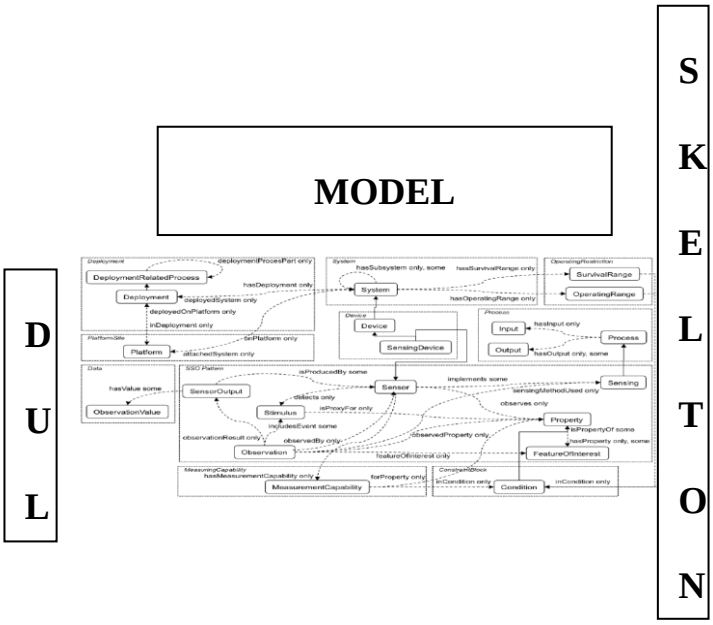


Figura 3-8. Semantic Sensor Network. Fuente: (W3C, Semantic Sensor Network Ontology, 2011) (Castillo, Fernández-Caballero, Serrano-Cuerda, & Sokolova, 2012)

SSN tiene tres módulos encargados de: a) relacionar una actividad con el evento sensorial (evento obtenido a partir del procesamiento de señales multisensoriales), b) detallar técnicas para procesar la señal, y c) detallar las configuraciones necesarias para la correcta instalación de los dispositivos sensoriales (en el Anexo II se encuentran las tablas en las que se muestran en detalle las clases de *SSN*). A continuación se describen cada uno de los módulos.

DUL

En este módulo se reutiliza las conceptualizaciones de la ontología DOLCE[12] Ultra Lite, para modelar actividades de alto nivel semántico a partir del procesado de la señal multisensorial. Es necesario aclarar, que en este módulo, el proceso de modelar no es jerárquico, sino que es directo, es decir; se vincula de forma directa el evento con su actividad (brecha semántica) (Arens & Nagel, 2003).

Una vez que se conoce cuál es la actividad que puede ser reconocida a partir del procesamiento de la señal multisensorial, el paso siguiente es modelar el proceso de funcionamiento del sensor, de eso está encargado el módulo SKELETON.

SKELETON

En este módulo se utilizan clases de *SSN* para conceptualizar al sensor. Además se obtienen los axiomas y reglas necesarias para su funcionamiento e interacción con otros elementos del escenario.

La clase *Stimulus* se utiliza para conceptualizar las características de la señal sensorial que da origen a un evento, por ejemplo, un sensor de movimiento se activa cuando la persona camina, esto conceptualizado en la clase *Stimules* es *SSN*:*Stimulus:signal_Type = Infrared, SSN*:*Stimulus:Sensor = Movement, SSN*:*Stimulus:signal_Sensor_State = ON*.

[12] DUL: http://www.loa.istc.cnr.it/DOLCE.html, tomado el 21-10-2013.

La clase *SensorDeployment* conceptualiza los elementos necesarios para la instalación y uso del sensor. Por ejemplo, un sensor de movimiento debe ser ubicado a 3 metros de altura y la señal debe ser recogida por un nodo central, esto conceptualizado en la clase *SensorDeployment* es *SSN:SensorDeployment:Sensor = Movement, SSN:SensorDeployment:heigthUbication=3, SSN:SensorDeployment:ubication_Units = meters, SSN:SensorDeployment:processing_Signal: CenterNode.*

Un sensor puede tener todo tipo de propiedades, que pueden o no estar relacionadas con los aspectos de interés para la inferencia de eventos, pero; estas no dejan de existir por no estar vinculados a un determinado punto de interés. Por ello, en este módulo se crean restricciones relacionadas con el escenario, dado que en un sistema multisensorial se pueden generar grandes cantidades de datos que pueden no ser relevantes para el proceso de inferencia.

En un sistema de *SV* no existe un solo sensor sino que muchos de ellos, los cuales por lo general conforman una red (Botia, Villa, & Palma, 2012). Se hace necesario contar con un módulo que recoja las señales multisensoriales y las fusione, con el fin que puedan ser utilizadas para inferir eventos. *Horus* se encarga del proceso de fusión de señales multisensoriales, y la salida del mismo son los eventos que utilizamos para componer actividades y situaciones de alto nivel semántico. A pesar de contar en nuestro trabajo con el proceso de fusión multisensorial, creemos conveniente también detallar el modulo que en *SSN* se encarga de ese proceso.

MODEL

Los diferentes nodos de una red de sensores deben integrarse a través de un sistema donde la información de cada nodo es procesada e interpretada. Aquí, se conoce como "sistema" a la entidad que usa como interfaces de adquisición de datos a los diferentes sensores de una red. Según el grado de complejidad y abstracción de un sistema este puede contener subsistemas que también son sistemas. La clase *System* se utiliza para conceptualizarlos.

La clase *Process* conceptualiza a todos los elementos que intervienen en el procesado de una señal multisensorial. Por ejemplo para sí ha detectado a una persona a partir de procesar una señal de video, la clase *Process* conceptualiza a:

Process:signal_Type:video;

Process:signal_Algorithm:Bayesian;

Process:signal_Time:eeriod_1.

La clase *Deployment* conceptualiza a los elementos necesarios para que los sensores puedan interactuar entre sí. Por ejemplo, se requiere de la señal de dos cámaras, la clase *Deployment* conceptualiza a:

Deployment:sensor:Camera1;

Deployment:sensor_Ubication:Zone1;

Deployment:sensor_Signal:Video;

Deplyment:sensor_Conecting:Camera2;

Deployment:sensor_Objective:Tracking

Dado que existen diversos procesos necesarios para el funcionamiento de una red multisensorial –instalación, configuración, mantenimiento, integración– se requiere una definición que agrupe a todos los procedimientos requeridos para su puesta en marcha. Para conceptualizar a los elementos que participan en esos procesos se utiliza la clase *DeploymentRelatedProcess*. Por ejemplo, se requiere *Monitorizar* el funcionamiento de un circuito cerrado de video en una tienda, la clase *DeploymentRelatedProcess* conceptualiza a:

DeploymentRelatedProcess:Process: Monitoring

DeploymentRelatedProcess:Sensors:Deployment

DeploymentRelatedProcess:Period:Period 1;

DeploymentRelatedProcess:summit_FunctionalProblems:Administrator

Una vez descrita la ontología que maneja el nivel semántico operativo del sensor, lo siguiente es describir la ontología que nos permite la conceptualización de los elementos arquitectónicos de un escenario, en el cual se registran los eventos que se infieren a partir del procesado de señales multisensoriales.

3.3.2 Ontología para la conceptualización de los elementos arquitectónicos de un escenario (*BuildingArchitecture*)

En esta tesis, el lugar físico en donde se registran las situaciones le llamamos "escenario". Los escenarios están compuestos por elementos físicos arquitectónicos –pisos, paredes puertas–. Conceptualizando estos elementos se puede obtener mayor detalle semántico, el instante en el que el escenario participa de la inferencia de situaciones de alto nivel semántico. El marco propuesto en esta tesis trabaja con ese tipo de conceptualizaciones. Para ello, se ha escogido a la ontología *Modular Ontologies for Architectural Design* (*MOAD*), la cual ha sido utilizada con buenos resultados para

conceptualizar los elementos de un plano arquitectónico (Hois, Bhatt, & Kutz, 2009) (Held, Buchholz, & Schill, 2002). Entre las ontologías que pertenecen a *MOAD* se tiene a *Building Architecture* (*BA*). En esta tesis nos centramos en explicar a *BA*, puesto que es la que está relacionada con los experimentos desarrollados en este trabajo, y sus conceptualizaciones son muy parecidas a las otras ontologías que forman parte de *MOAD* (Hois, Bhatt, & Kutz, 2009). *BA* tiene clases que permiten conceptualizar los elementos de un escenario y sus relaciones espaciales. En la Figura 3-9, se muestra las capas de *BA*. Las conceptualizaciones de cada capa se integran en el modelo de *Representación Integrada y Axiomas basado en E-Connections*, el cual consiste de un conjunto de restricciones semánticas que permiten obtener un correcto diseño semántico de un plano arquitectónico (Hois, Bhatt, & Kutz, 2009). La descripción de cada capa y los módulos de *BA* se detallan a continuación.

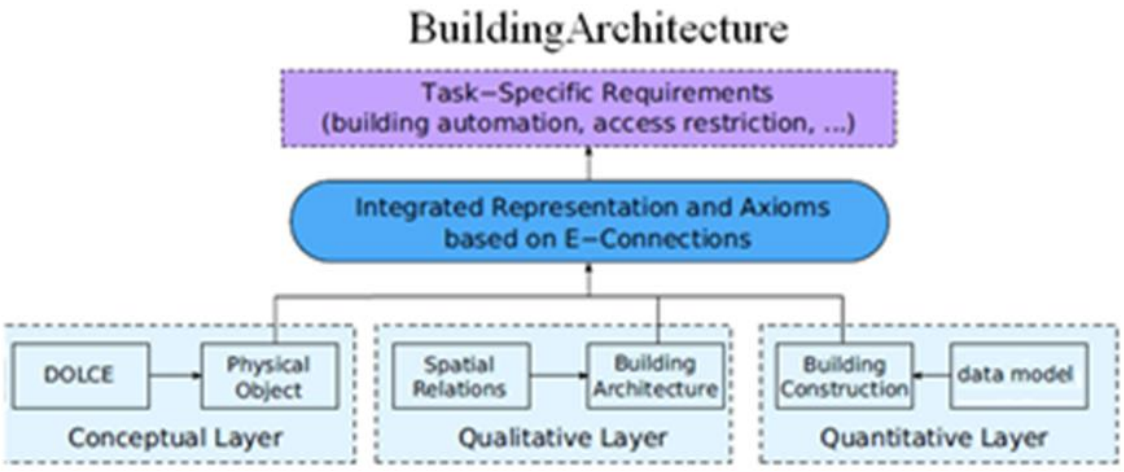


Figura 3-9. BuildingArchitecture. Fuente: (Hois, Bhatt, & Kutz, 2009)

Capa Conceptual

Utiliza clases y subclases que representan a los *physical objects* en un plano. Por ejemplo, la clase *Puerta* puede tener distintas subclases de *Puerta* batiente, giratoria o corrediza. La ontología DOLCE-Lite3 se utiliza en esta capa para conceptualizar detalles de las clases. Por ejemplo, el tipo de material con el cual está hecha la *Puerta*: madera, aluminio, plástico, etc. En la Tabla 3-1 se observa el detalle semántico incluido en la clase:

Tabla 3-1. Módulo Conceptual Fuente: (Hois, Bhatt, & Kutz, 2009)

Class	***Door***
SubClassOf	*Building_Construct*
DOLCE-Lite:has-quality	*exactly 1 Material(Wood)*

Capa Cualitativa

Utiliza características específicas para representar a los *physical objects* en un plano. Por ejemplo, la función de una *Puerta* puede ser: conexión entre habitaciones y corredores, adyacente a otros lugares proveer acceso a un lugar o, externamente conectada a una ventana o a otra puerta:

Tabla 3-2. Módulo Cualitativo. Fuente: (Hois, Bhatt, & Kutz, 2009)

Class	***Door***
SubClassOf	*Building_Structure*
rcc:externallyConnectedTo	*some Window, Door*

Siguiendo con el ejemplo, en un espacio arquitectónico las funciones de una puerta se ajustan a las características específicas, pero; existen diseños arquitectónicos en los cuales es necesario añadir mayor funcionalidad a una Puerta, y esto puede ser expresado con axiomas. Es posible encontrar diseños arquitectónicos en donde todas las puertas que conecten una habitación con un corredor deben tener por defecto el estado cerrado (axioma) mientras que todas las puertas que conecten una cocina con un corredor o cualquier otro espacio deberían permanecer abiertas (axioma).

Capa Cuantitativa

La información contenida en el módulo cuantitativo hace referencia específicamente a la información dimensional del piso en construcción, que además puede ser extendido a todos los pisos que se construyan. En este módulo se conceptualiza la información de las medidas mínimas y máximas de los elementos de una construcción como baños, cocinas y dormitorios. Se puede adicionar información más específica referente a una entidad siempre que esta esté expresada en una métrica. Por ejemplo, pulgadas del vidrio, el ángulo de abertura de la puerta y la altura de la ducha del baño, valores que no son comunes a todos los objetos y que en cierta manera los caracterizan. Las características cuantitativas se conceptualizan en la clase *BuildingStructure*.

Los axiomas generados a este nivel podrían tener menor peso al momento de resolver inconsistencias debido a que, las relaciones descritas no aportan mayor especificación sobre la funcionalidad de un objeto.

Las capas conceptualizan los elementos de un plano. Estas conceptualizaciones se recogen en un solo módulo con el fin de obtener la conceptualización general del mismo

Módulo de representación integrada

Las conceptualizaciones de las capas cuantitativa, cualitativa y conceptual se integran en este módulo, con el fin de crear el modelo general del plano arquitectónico. El resultado de este módulo son dos clases generales conceptualizadas por DOLCE-Lite: *Functional_Structure* (cuarto, cocina, pasillo, ventanas, puertas) y *Architectural_Feature* (lugar, dimensiones), como se muestra en la Tabla 3-3:

Tabla 3-3. Representación Integrada. Fuente: (Hois, Bhatt, & Kutz, 2009)

DisjointClasses:	***DOLCE-Lite:particular***
buildingArchitecture	*Functional_Structure*
buildingConstruction	*Architectural_Feature*

Para ejemplificar el uso del módulo de integración, se propone el siguiente ejemplo. Un elemento de un plano es una instancia particular que posee una definición sobre sus características en el plano y una definición que indica sus aspectos funcionales. Por ejemplo, dada la clase *Puerta*, en el módulo cuantitativo se instancian sus dimensiones (características en el plano), en el módulo cualitativo se instancia su funcionalidad;

puede ser una puerta de ingreso a una habitación o a diferentes ambientes (cardinalidad). La Tabla 3-4 muestra una representación del ejemplo.

Tabla 3-4.Representación integrada con propiedades inversas. Fuente: (Hois, Bhatt, & Kutz, 2009)

ObjectProperty	***Compose***
Domain:	*Building_Construction:Architectural_Feature*
Range:	*Building_Architecture:Functional_Structure*
InverseOf:	*isComposedOf*
Class:	*buildingConstruction:Door*
SubClassOf:	*compose exactly 1 buildingArchitecture:Door*
Class:	*Building_Architecture:Door*
SubClassOf:	*isComposedOf exactly 1 buildingConstruction:Door*

Módulo de requisitos específicos de la tarea

Las tareas de este módulo especifican un propósito que debe ser cumplido –restringir el nivel de privilegios de un usuario, el tipo de dispositivo que puede tener acceso ha determinado servicio, el tiempo que el servicio puede estar activo y condiciones que deben ser cumplidas para la prestación de un servicio, tareas de monitoreo y vigilancia–. Estas tareas suponen la inclusión de diversas interfaces para cumplir la interacción con el medio. En escenarios de seguridad domiciliaria por ejemplo, implica

que determinadas tareas requerirán sensores de movimiento, cámaras de seguridad y alarmas.

Los requisitos específicos de una tarea en el contexto del diseño arquitectónico apuntan a crear ambientes inteligentes (*AmI)* escalables y de costo relativamente bajo. El proceso de construcción arquitectónica termina siendo manejado de una manera muy dinámica ajustándose mejor a la forma en que los diseños de constante cambio. Por ejemplo, una restricción a este nivel podría ser que todas las edificaciones tengan una terminal con inteligencia de navegación que puede orientar a los visitantes; tal requerimiento en el diseño tradicional implica desarrollar nuevos sistemas con sus propias definiciones y restricciones, pero con el manejo modular semántico, simplemente se agrega la restricción y la capa correspondiente, sin afectar el funcionamiento de las otras capas. La Tabla 3-5 se muestra el modelo de la restricción, propuesto en el ejemplo:

Tabla 3-5.Requisitos específicos de tareas. Fuente: (Hois, Bhatt, & Kutz, 2009)

Class	***buildingArchitecture:Building***
SubClassOf:	*rcc:inverseProperPartOf min 1 (buildingArchitecture:Display and (integratedRepresentation:isConceptualizedBy some physicalObject:NavigationTerminal))*

La restricción anterior puede contener errores, como también la conceptualización del plano. Para verificar los errores *BA* cuenta con un módulo especializado denominado *Modulo de corrección y eliminación de errores de diseño*. Este módulo tiene mucha

importancia para el diseñador de planos, pero no lo describimos en este apartado, ya que nos interesan más las conceptualizaciones de *BA* que diseñar un plano arquitectónico. Lo que tratamos es de incrementar el nivel semántico de una situación y no de un espacio funcional. Pero si hay interés sobre el mismo, hemos descrito en el Anexo III al módulo de *Corrección y eliminación de errores de diseño.*

Las clases y relaciones espaciales son conceptualizadas por *BA*, pero; las relaciones temporales tienen mucho que ver también a la hora de modelar una actividad de alto nivel semántico. Por ello, es necesario contar con una ontología que permita conocer el detalle semántico de las relaciones temporales lo que permitirá incrementar la expresividad conceptual de las situaciones de interés en un sistema de *SV* (Held, Buchholz, & Schill, 2002).

3.3.3 Ontología *Time*

Para poder conceptualizar eventos, acciones, actividades y situaciones, es necesario contar con las relaciones temporales que funcionen el marco. Para ello se utiliza la ontología *Time*, cuyos componentes principales se muestra en la Figura 3-10:

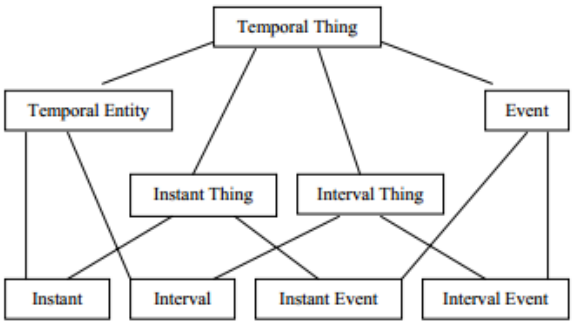


Figura 3-10. Ontología *Time*. Fuente: (Pan & Hobbs, 2004)

Time ha sido utilizada con excelentes resultados en algunas investigaciones que modelan eventos (Allen & Ferguson, 1994) (Hu, W, Wang, & Maybank, 2004), lo que nos da la pauta para poder utilizarla también en este trabajo doctoral. Con esta ontología se pueden responder preguntas relacionadas al ¿Cuándo ocurrió el evento o situación?, característica que incrementa el detalle del modelado semántico de alto nivel (Albusac, 2008). La clase *Event* no pertenece a esta ontología pero, siempre se hace referencia ella, puesto que allí se aplican las relaciones espaciales y temporales, un evento ocurre en un lugar y tiempo determinado. En los Anexos IV y V, se ilustran las relaciones temporales de *Time*, se utiliza la notación *N3* también conocida como *no-XML*; ya que es una forma más abstraída y de fácil entendimiento para expresar las relaciones temporales.

3.3.4 Clases y relaciones temporales

La clase principal de la ontología es *TemporalEntity*. Tiene dos subclases: *Instant* e *Interval*.

- *Instant* es un momento del tiempo sin longitud; es decir que su punto de inicio y fin son los mismos.
- *Interval* se define como dos puntos de tiempo separados por un intervalo temporal.

Para hacer referencia a un instante determinado de un intervalo de tiempo se tiene la propiedad *inDateTime*, definida de una manera muy parecida a la propiedad *hasDateTimeDescription*.

inDateTime debe ser usado particularmente para situaciones que no hacen referencia a un instante específico del tiempo; por ejemplo si se dice que una clase empieza a las 3 de la tarde, se hace referencia a cualquier instante entre las 3:00 y las 3:01 de la tarde, por lo tanto; no se refiere a un instante como tal.

Los eventos, acciones, actividades y situaciones tienen un principio y un final. Para conceptualizar estas relaciones se usan las propiedades *hasBeginning* y *hasEnd* respectivamente. Cuando el valor de la propiedad *hasBeginning* tiende al infinito positivo; se dice que no tiene fin; mientras que cuando su aproximación sucede hacia el infinito negativo, se dice no tiene comienzo.

La propiedad *Inside* se utiliza para representar que un instante se da en un intervalo. Esta propiedad no apunta a concebir un intervalo como una secuencia finita de instantes, sino más bien como una forma de describir la precedencia. La propiedad *Before* representa el orden de aparecimiento. Por ejemplo, si una situación *A* sucede antes de una situación *B*; el final de *A* sucede antes del comienzo de *B*. Las relaciones temporales pueden expresarse de una manera muy sencilla en términos de identidad. El manejo de granularidad temporal de zonas horarias y fechas se describe en el Anexo IV de esta tesis

En *Time* existen propiedades de los intervalos, que son necesarias para trabajar con actividades y situaciones de forma temporal, basándose en el *Algebra de Allen*:

Tabla 3-6. Propiedades de intervalos Fuente: (W3C, Time Ontology in OWL , 2006)

Propiedad	**Nombre**	**Inversa**
Igualdad	*Equals*	*After*
Precedencia	*Before*	*Meet*
Intervalo cumplido	*Meets*	*Overlaps*
Super-Posicion	*Overlaps*	
Inicio	*Starts*	*StartedBy*
Duración	*During*	*Contains*
Finalización	*Finishes*	*FinishedBy*

En los casos de estudio de esta tesis, los eventos tienen características temporales de aparición *inmediatamente antes* o *inmediatamente después*. Esto no coincide exactamente con las definiciones del álgebra de Allen. Por lo que recurrimos a completar la definición de la propiedad *Meets*, adicionándole un umbral con el fin de distinguir cuanto es el valor *inmediatamente*. Por ejemplo, una persona ingresa a una habitación, se mantienen en ella caminando, luego va a la sala y se mantiene en ella caminando, luego va al... y por último va al baño. Si se aplica la propiedad *Before* la persona antes de ir al baño estuvo en la habitación, en la sala, en la..., es decir; la propiedad registra todos los lugares por los cuales estuvo la persona. La propiedad *Meet* registra los lugares visitados inmediatamente antes o coincidentes. Por ejemplo, la persona está en la habitación desde las 10:00 hasta las 10:10:20, luego a las 10:10:25 está a la cocina y a las 10:10:30 está en el pasillo. Estos eventos no son temporalmente coincidentes de forma exacta, ya que tienen una diferencia de 5 segundos entre ellos. Si

a la propiedad *Meet* le añadimos ese umbral temporal, creamos la propiedad *inmediatamente antes o después* que necesitamos en nuestros casos de estudio, en dónde; se necesita de eventos coincidentes más que la reunión de todos los eventos que se han registrado en un escenario.

VERL apoyado por *CHAO*, permite modelar situaciones de alto nivel semántico. Pero, nosotros trabajamos con herramientas semánticas y lo que hace necesario que los modelos se exporten hacia el lenguaje *OWL*. El módulo de exportación se describe en el siguiente apartado.

3.4 Módulo de exportación

Los modelos de las situaciones de alto nivel semántico desarrollados en *VERL*, son útiles cuando pueden ser utilizados por herramientas semánticas. De allí que este módulo se encarga de exportar los Metaconceptos utilizadas en el modelo (*physical objects, states, events)* y los *constraints* hacia el lenguaje *OWL* como clases, propiedades y axiomas. El resultado de la exportación permite a los razonadores semánticos, las reglas y algoritmos el poder inferir situaciones de alto nivel semántico, cuando el marco entra en operación en sistemas de *SV*. La operatividad del marco, empieza recogiendo los eventos provenientes de Horus, para luego de acuerdo con el modelo semántico inferir acciones, actividades y situaciones de interés. El módulo de inferencia del marco se describe a continuación.

3.5 Módulo de inferencia

El módulo de inferencia entra en operación, cuando recibe los eventos que provienen del módulo de fusión sensorial de *Horus*. Estos eventos se instancian en la clases que utilizo el ingeniero de conocimiento para modelar la situación de interés. Luego, los razonadores utilizan los axiomas que se compusieron a través de restricciones en *VERL* para componer los niveles que conduzcan al alto nivel semántico. Todo esto ocurre en tiempo de ejecución. Para facilitar el trabajo de modelar situaciones de interés y la operatividad del marco, hemos desarrollado herramientas que ayudan al ingeniero del conocimiento a modelar el alto nivel semántico. Estas herramientas se describen a continuación.

3.6 Diseño de herramientas para el modelado y consulta de situaciones

En este apartado se describen las herramientas que se crearon a partir de las ideas que resultaron de las aplicaciones de nuestro marco para la inferencia de distintas situaciones de alto nivel semántico. El punto es presentar al agente humano las herramientas necesarias que faciliten tanto para el modelado de eventos, acciones, actividades y situaciones como su consulta. Es necesario destacar que estas herramientas se encuentran en desarrollo y son una propuesta para del uso del marco en sistemas de *SV*.

3.6.1 Herramienta para modelar situaciones

La herramienta se basa en la estructura propuesta por *VERL* para modelar situaciones. En la Figura 3-11, se observa dichos componentes:

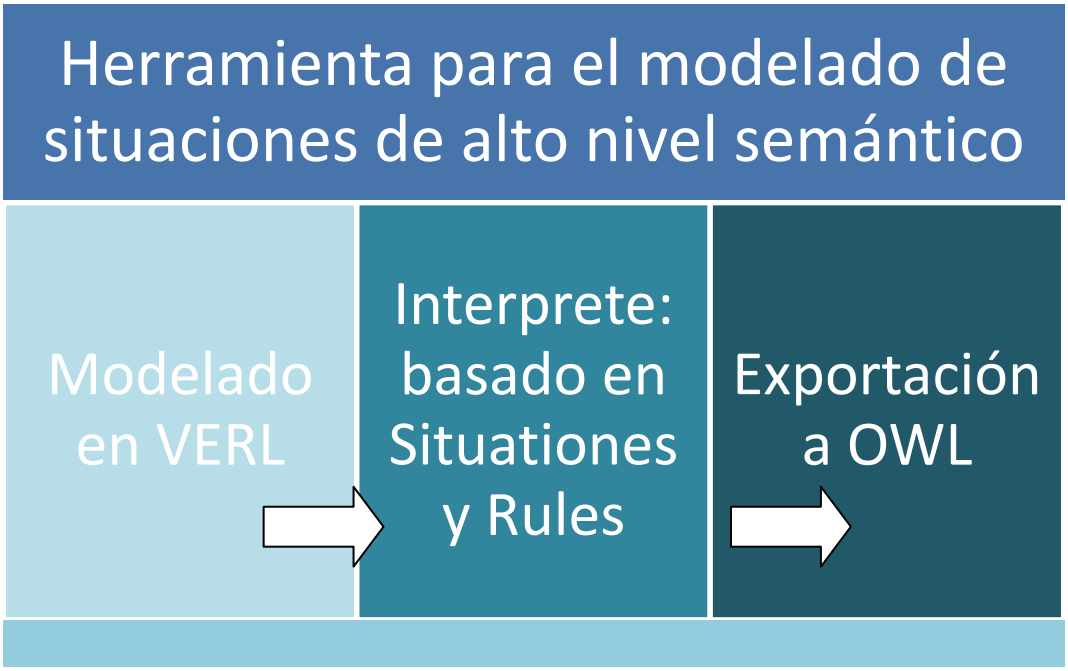


Figura 3-11.Componentes del marco de situaciones

La herramienta está diseñada para tomar las conceptualizaciones de las ontologías *SSN*, *BA* y de la ontología *Time*. Además, en la herramienta se pueden componer eventos desde de distinto nivel jerárquico, como se observa en las Figuras 3-12 y 3-13:

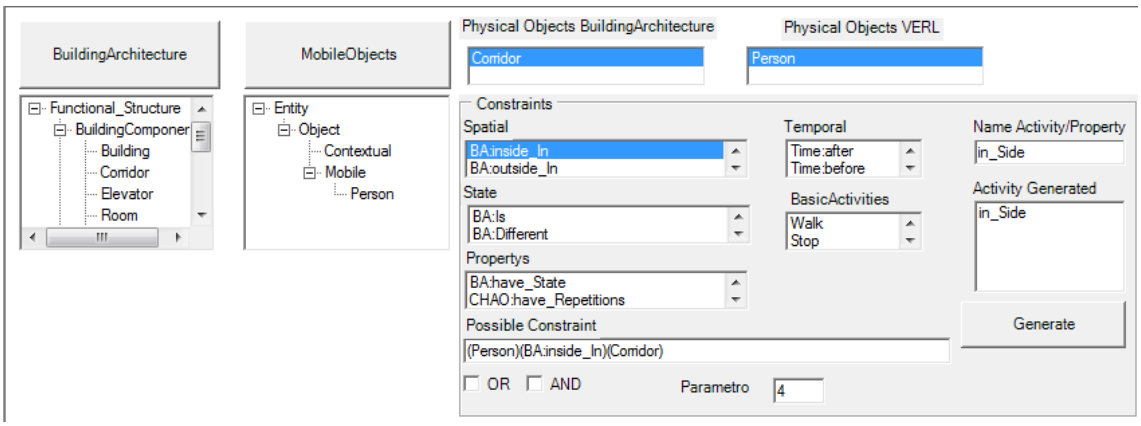


Figura 3-12. Person Inside

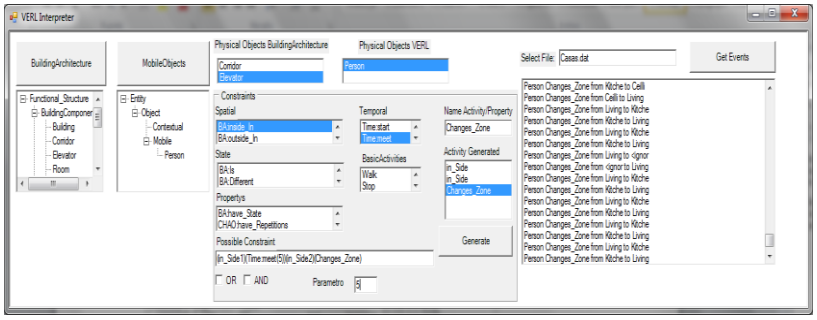


Figura 3-13. Inferencia del evento Changes_Zone

En las figuras anteriores se observa la creación del evento *Changes_Zone((Person, BuildingComponent) Meet(5) (Person,BuildingComponent2)).* Este evento se crea con la propiedad temporal (*Meet(5)*) de la situación (*Person Inside BuildingComponent*) (Nevatia & Hobbs, 2004). Aquí, a la persona se lo toma como objeto móvil, ya que forma parte de los *Mobile Objects* en *VERL* (Nevatia & Hobbs, 2004). Una vez que ha sido modelado el evento *Changes_Zone*, este modelo puede ser exportado a *OWL* con el algoritmo 3-1:

```
while ((text = reader.readLine()) != null) {
        //System.out.println(text.charAt(2));
         for (i = 0; i < text.length(); i++) {
                        if (text.charAt(i)=='('){
                                i++;
                                clase = "";
                                while (text.charAt(i)!=')') {
                                        clase = clase + text.charAt(i);
                                         i++;
                                }
                        }
        if (clase.equals(Restricciones)){
                p = model.createObjectProperty(relationshipUri+clase);
                propiedad = clase;
                model.write( System.out, "RDF/XML-ABBREV" );
         }else{
                j++;
                if (j == 1){
```

```
                        myClass = model.createClass( relationshipUri+clase );}
                        if (j==2){
                                myClass2 = model.createClass( relationshipUri+clase );
                        }
                        if (j==3){
                                myClass3 = model.createClass( relationshipUri+clase );
                                myClass2.addProperty(p, myClass3);
                                model.write( System.out, "RDF/XML-ABBREV" );
                                p.addRange(myClass3);
                                p.addDomain(myClass2);
                                model.write( System.out, "RDF/XML-ABBREV" );
                                p2 =
model.createObjectProperty(relationshipUri+"haveParameters");
                                p2.addRange(myClass3);
                                p2.addRange(myClass2);
                                p2.addDomain(myClass);
                                model.write( System.out, "RDF/XML-ABBREV" );
                                Set rangeSet = Jena.set(p2.listRange());
                                RDFList list = model.createList(rangeSet.iterator());
                                IntersectionClass intersectionClass =
model.createIntersectionClass(null,list);
                                p2.setRange(intersectionClass);
                                model.write( System.out, "RDF/XML-ABBREV" );
                                restriccion =
model.createMinCardinalityRestriction(relationshipUri, p, 1);
                                restriccion.addSubClass(myClass);
                        }
                }
        }
}
```

Algoritmo 3-1. Interprete de *VERL* a OWL

El Algoritmo 3-1, lee las líneas de código de *VERL* que se encuentran almacenadas en un fichero. Luego según va encontrando las clases las va creando en formato *OWL*. Verifica también las existencias de restricciones espaciales y temporales. De la misma

manera deja explicita en *OWL*[13] los parámetros y los axiomas que necesita la actividad o situación que se está analizando. En el ejemplo la ontología en *OWL* es:

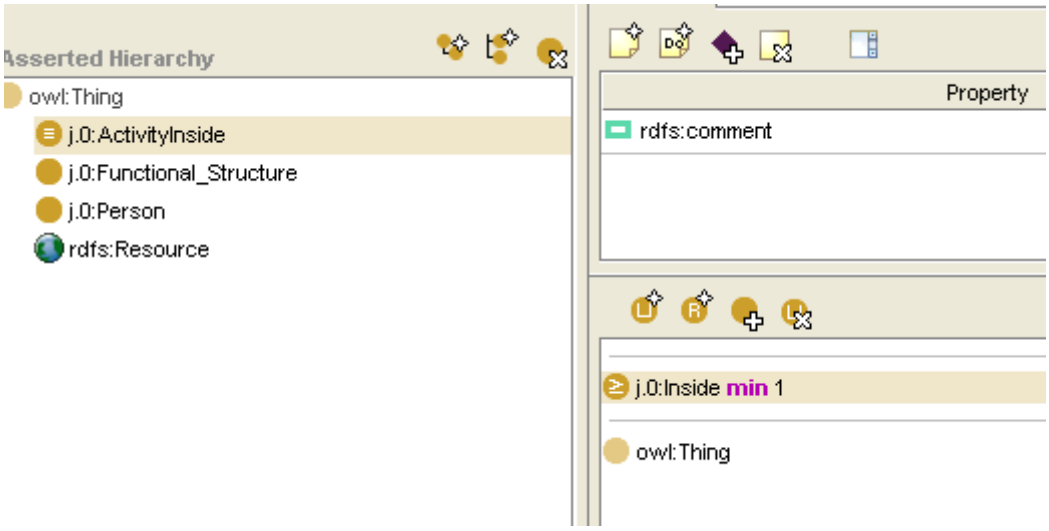


Figura 3-14. *Activity_Inside* en OWL

En la Figura 3-14, se muestra que la actividad *Activity_Inside* tiene el axioma *Inside min 1*. Este axioma sirve para trabajar con la ontología con el fin de inferir la actividad siempre y cuando al menos una persona cumpla con *Persona inside Functional_Structure*.

[13] Se trabaja con Protégé para mostrar el uso de OWL: http://protege.stanford.edu/, tomado el 20-10-2013.

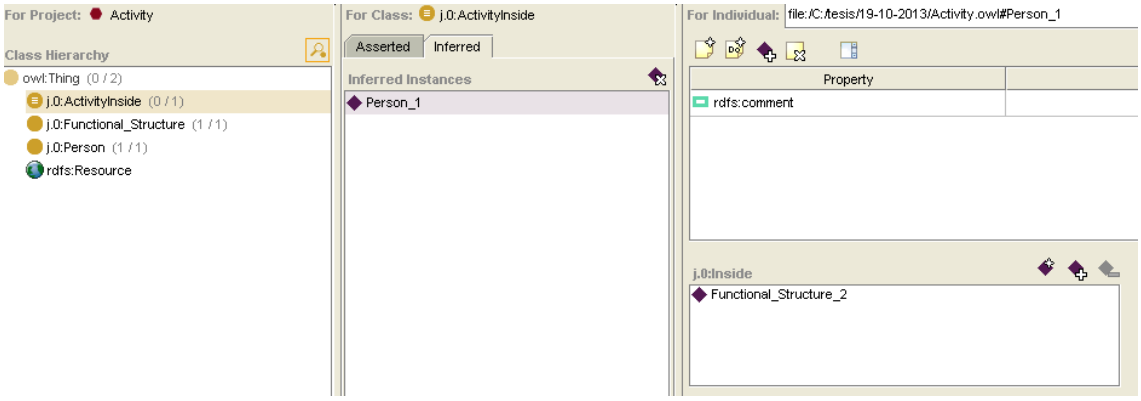


Figura 3-15. Inferencia de la actividad Activity_Inside

En la Figura 3-15, se muestra como la actividad *Activity_Inside* ha sido inferida a través del razonador a partir de los axiomas. En el ejemplo, se ha necesitado que la *Persona_1* se encuentre en *Functional_Structure_2* para que la actividad sea inferida. Los parámetros que necesita la actividad se muestran a continuación:

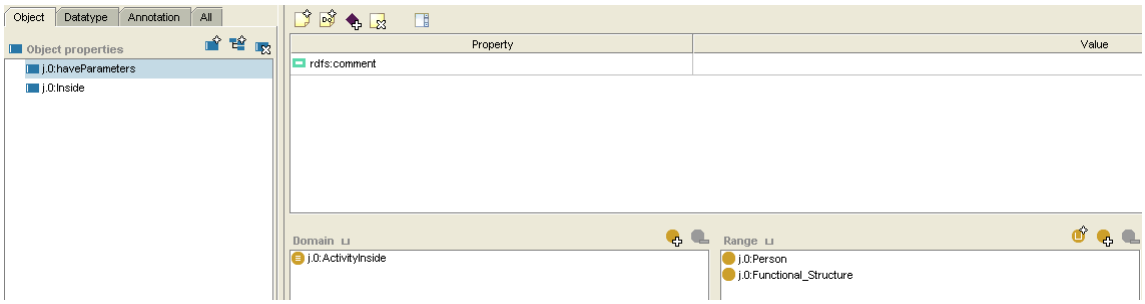


Figura 3-16. Parámetros de Activity_Inside

En la Figura 3-16, se muestran los parámetros que necesita *Activity_Inside* para su inferencia automática en *OWL*. El frontal también puede ser utilizado para crear situaciones. Continuando con el ejemplo, el evento *Changes_Zone* en *OWL* se observa en la siguiente figura:

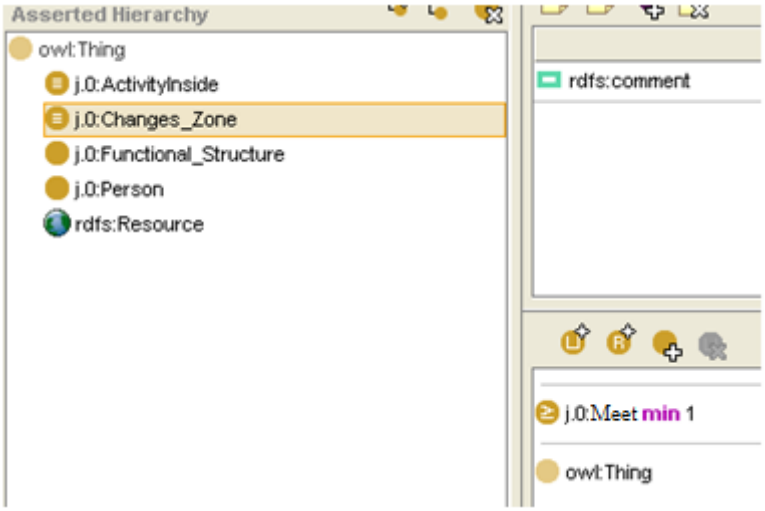


Figura 3-17. Parámetros de Changes_Zone

En la Figura 3-17, se muestra a *Changes_Zone* en Protégé. Se muestra el axioma *Meet min 1*, el cual indica que deben existir dos eventos Inside coincidentes temporalmente. No se puede especificar el umbral de coincidencia en un axioma, pero se lo tomará en cuenta en las reglas o algoritmos en donde sea necesario utilizarlo para inferir o consultar alguna actividad de interés. Continuando con el ejemplo, se modela la situación *Wandering* en la herramienta, como se muestra en la Figura 3-18:

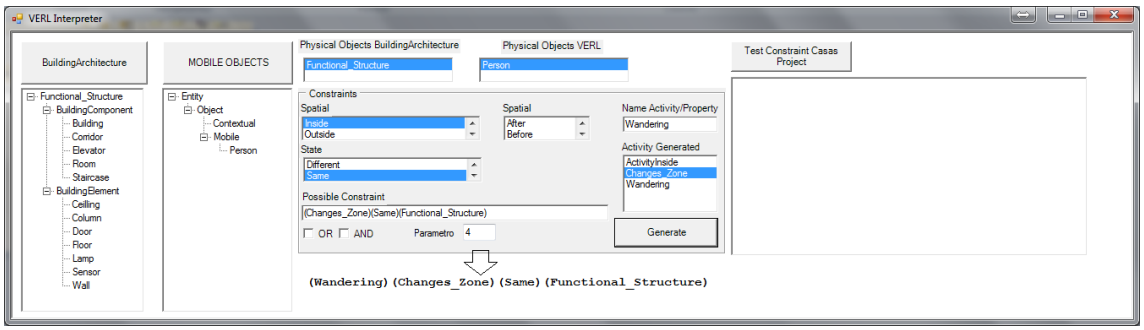


Figura 3-18. Modelado de la actividad Wandering

La situación *Wandering* se ha modelado a partir de que la situación *Changes_Zone* en la cual se repite las mismas *Functional_Structure*. Con esto se quiere decir que la persona,

está en un lugar *Activity_Inside* luego cambia a otro lugar, y vuelve al lugar de partida, repitiendo los lugares de visita.

Figura 3-19. Wandering en Protégé

En la Figura 3-19, se observa a la situación *Wandering* en *Protégé*. Se ha creado automáticamente el axioma *Same min 4*, con el fin de indicar que si la persona repite su visita al mismo lugar por cuatro ocasiones se considera que existe *Wandering*.

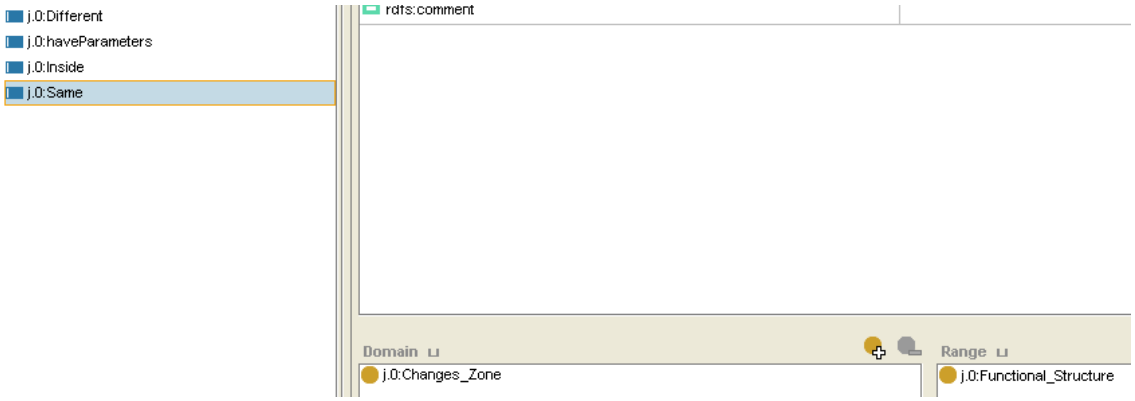


Figura 3-20. Propiedad Same diseñada en el marco

En la Figura 3-20, se muestra que se ha creado automáticamente la propiedad *Same* que sirve para inferir a *Wandering*. Hemos visto como con el frontal podemos modelar situaciones y estas exportarlas al lenguaje *OWL*. Esto nos da la idea de que los componentes del marco los podemos utilizar de formar fácil y rápida para modelar situaciones de alto nivel semántico. El manejo de lenguaje necesita conocer cuales la estructura del mismo, en el siguiente apartado explicamos el uso de una herramienta visual para el manejo del modelado con *VERL*, aplicación que la hemos desarrollado con el fin de facilitar a los diseñadores de sistemas de *SV* el modelado de situaciones de

alto nivel semántico. Esta aplicación lógicamente también forma parte de las tecnologías semánticas relacionadas que se necesitan para modelar las situaciones de interés. La aplicación también puede exportar reglas que forman parte de *PROCESS* en *VERL*. Un extracto del código de exportación es el siguiente:

```
Dim nFileNum As Integer, sText As String, sNextLine As String, lLineCount As Long
'================================
Private Sub Form_Load()
' Get a free file number
nFileNum = FreeFile
nFileNum2 = 4
' Open a text file for input. inputbox returns the path to read the file
Open "C:\tesis\17-11-2013\VERL.txt" For Input As nFileNum
Open "C:\tesis\17-11-2013\SWRL.txt" For Output As nFileNum2
Dim cadenarule As String
lLineCount = 1
' Read the contents of the file
Do While Not EOF(nFileNum)
  Line Input #nFileNum, sNextLine
  'do something with it
  'add line numbers to it, in this case!
  sNextLine = sNextLine & vbCrLf
  sText = sText & sNextLine
  If InStr(sNextLine, "Presence") Then
      cadenarule = "is_In(?Presence,?Place)"
      Print nFileNum2, cadenarule
      cadenarule = "have_Time(?Presence,?occur_In)"
      Print nFileNum2, cadenarule
      cadenarule = "j.2:inXSDDateTime(?occur_In,?Date)->"
      Print nFileNum2, cadenarule
  End If
  If InStr(sNextLine, "While") Then
      cadenarule = "sqwrl:concat(?SecuencePlaces,?Place)"
      Print nFileNum2, cadenarule
      cadenarule = "sqwrl:select(?Presence,?Place,?occur_In,?Date)"
      Print nFileNum2, cadenarule
      cadenarule = "sqwrl:orderBy(?Fecha)"
      Print nFileNum2, cadenarule
  End If
Loop
'Text1.Text = sText
' Close the file
Close nFileNum
Close nFileNum2
End Sub
```

Código 3-5. Exportación de reglas de *VERL* hacia *SWRL*

En el siguiente apartado se describe, la herramienta visual del manejo de *VERL*. Esta herramienta también facilita el modelado de situaciones de alto nivel semántico.

3.6.2 Herramienta visual para modelar situaciones

El modelado de situaciones de alto nivel semántico necesita de herramientas metodológicas que faciliten la tarea. Y es que ese también es un principio fundamental del marco de manejo semántico que planteamos en este trabajo, el facilitar el modelado y la reutilización de las herramientas semánticas. De allí que hemos desarrollado una herramienta de manejo gráfico que hace uso de la capacidad lógica de *VERL*. Para facilitar la explicación de la herramienta de *software* desarrollada, a continuación se explica un ejemplo en el cual se modela gráficamente evento de bajo nivel semántico y se compone actividades de alto nivel semántico.

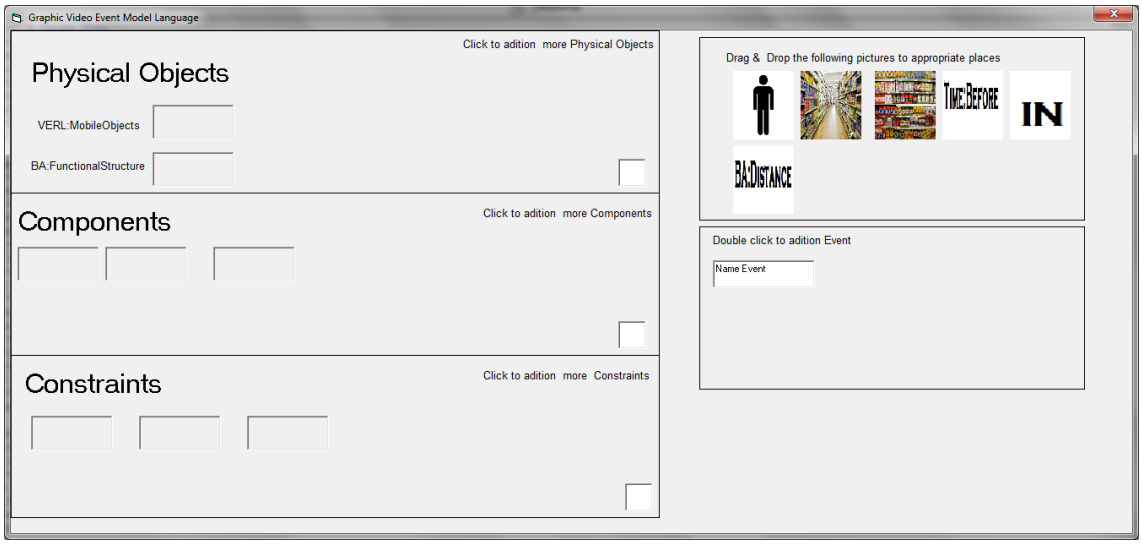


Figura 3-21.Graphic Video Event Model Language

GVERL es la versión gráfica de *VERL*, de hecho, se puede observar en la Figura 3-21, que tiene la misma estructura–*Physical Objects*, *Components* y *Constraints*–, solo que se le ha incrementado la capacidad descriptiva que proviene de *CHAO*. Se hace uso de las conceptualizaciones semánticas provenientes de distintas ontologías–*Time:meet*,

BA:distance−para incrementar el detalle semántico del modelado. El primer evento que vamos a modelar es *Inside_zone*, tal como se propone en (Bredmod & Maillot, 2009), como se muestra en la Figura 3-22:

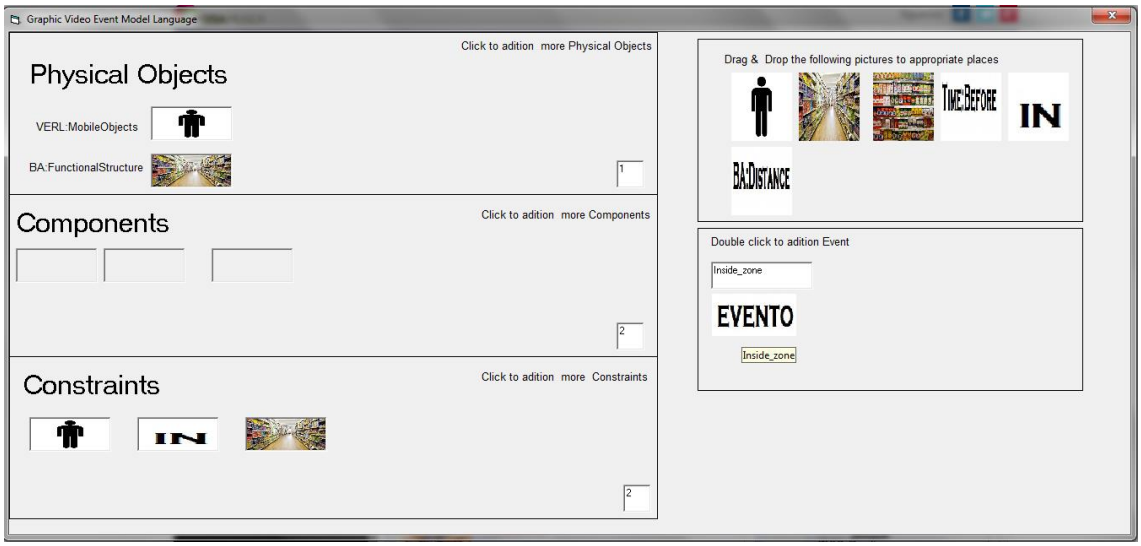


Figura 3-22.*Graphic Video Event Model Language*: Model State *Inside_zone*

El funcionamiento del *GVERL* es muy simple, en el que existen las conceptualizaciones de cada una de las ontologías y lo que hacemos simplemente es arrastrar y pegar las conceptualizaciones en los lugares en que se necesite de las mismas. En este caso, se arrastra la conceptualización de la Persona hacia *VERL:Mobile_Objects* y la conceptualización de una zona hacia *BA:FunctionalStructure*. Y es que esto es lo importante del marco, el relacionar todas las herramientas semánticas lo que le permite al diseñador de modelos obtener el detalle adecuado para con ello obtener un alto nivel semántico. Las *constraints* son fáciles de diseñar puesto que de igual manera solamente se arrastran las conceptualizaciones disponibles en la herramienta. En este caso se utiliza a la persona y zona, y se los relaciona espacialmente con la propiedad "*IN*".

Dado que este es un estado primitivo, no tiene componentes. A continuación, modelamos un evento que está compuesto por estados.

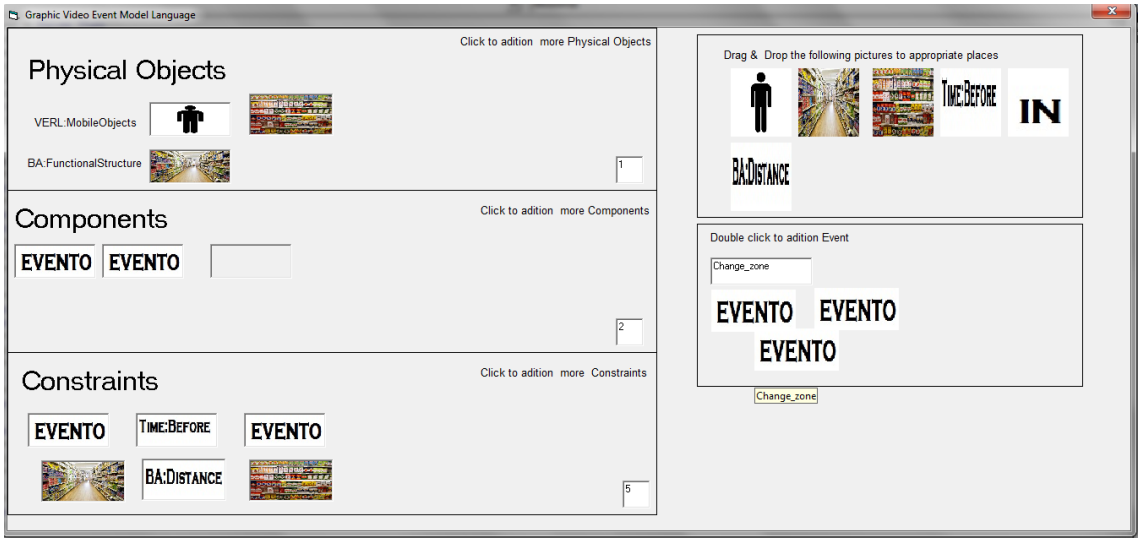


Figura 3-23. ***Graphic Video Event Model Language: Model Event Changes_zone***

Como se muestra en la Figura 3-23, el evento *Changes_Zone* se compone de dos estados tipo *Inside_zone* por ello es que se los arrastró hacia la zona de componente. Al mismo tiempo, tiene las restricciones conformadas por *Time:Before*, que indica que un *Inside_zone* que ocurre en la *zona1* se registró antes que el *Inside_zone* que ocurre en la *zona2*. De igual forma se puede modelar gráficamente que debe existir un valor de distancia mínimo entre la *zona1* y la *zona2* para indicar que se ha producido el evento que estamos modelando gráficamente. Y es que este lenguaje gráfico nos presta toda la facilidad del caso para modelar eventos y estados, con gran detalle semántico se especifica en el mismo de donde vienen las conceptualizaciones necesarias y útiles durante este proceso. Al final, por cada evento que se crea la herramienta puede reproducir de forma exacta el trabajo literal de *VERL*:

```
"Model"
"Primitive State Inside_zone"
        "Physical Objects"
                "(Person,Zone1)"
        "Components"
                "(,,)"
        "Constraints"
                "(Person,Constraint:In,Zone1)"
"Model"
"Primitive State Inside_zone"
        "Physical Objects"
                "(Person,Zone2)"
        "Components"
                "(,,)"
        "Constraints"
                "(Person,Constraint:In,Zone2)"
"Model"
"Primitive Event Change_zone"
        "Physical Objects"
                "(Person,Zone1,Zone2)"
        "Components"
                "(Inside_zone,Inside_zone,)"
        "Constraints"
        "(Inside_zone,Time:Before,Inside_zone,Zone1,BA:Distance,Zone2)"
```

Código 3-6.Exportación desde *GVERL* hacia *VERL*

El código 3-6 demuestra la fortaleza de *GVERL* para obtener el código *VERL* a partir de un modelado gráfico, y además que indica claramente cuáles son las conceptualizaciones útiles para el modelado, dando las indicaciones claras al diseñador de los sistemas *SV* de cuáles son los conceptos útiles para crear prototipos que puedan inferir situaciones de alto nivel semántico. Con esto logramos enfatizar que nuestro marco está dispuesto a acoplarse a las necesidades de los diseñadores y por supuesto a lo propuesto por en el marco general *Horus*. A continuación, vamos a detallar en cambio un frontal que permite las consultas de situaciones, lo cual es muy importante cuanto queremos utilizar el marco como ayuda para el análisis retrospectivo es decir detallar lo que ha sucedido en el escenario.

3.6.3 Herramienta para el análisis retrospectivo

En este apartado se presenta el desarrollo de un marco que permite realizar consultas *SPARQL* en las ontologías para *SV*, en especial para la que hemos desarrollado en este trabajo, *CHAO*. En la actualidad hay pocas herramientas que sean amigables para el usuario, en sí y solo existe un conjunto disperso de soluciones que en algunos casos son limitadas y/o de pago. Por eso la necesidad de crear una arquitectura general y aplicación web para gestionar las búsquedas en las ontologías de *SV*. Con el uso del marco se persigue:

- Implementar una arquitectura general que permita realizar consultas *SPARQL* sobre las ontologías que se requieran.
- Presentar los resultados de la consulta *SPARQL* lo más parecido al lenguaje natural de las personas. **"***SPARQL* se puede utilizar para expresar consultas que permiten interrogar diversas fuentes de datos, si los datos se almacenan de forma nativa como *RDF* o son definidos mediante vistas *RDF* a través de algún sistema middleware. *SPARQL* contiene las capacidades para la consulta de los patrones obligatorios y opcionales de grafo, junto con sus conjunciones y disyunciones. *SPARQL* también soporta la ampliación o restricciones del ámbito de las consultas indicando los grafos sobre los que se opera. Los resultados de las consultas *SPARQL* pueden ser conjuntos de resultados o grafos *RDF*" (Pastor Sánchez & Díaz Ortuño, 2008).
- Subir y mostrar en pantalla las imágenes correspondientes, en el caso de que la ontología a analizar así lo requiera.
- Permitir la administración de los usuarios que empleen la aplicación.

- Permitir subir las ontologías como un archivo "*.owl*" o especificarlo desde una dirección web que contenga el archivo "*.owl*".

Las nuevas tecnologías como la web semántica, el *Linked Data*, lenguajes de consulta *SPARQL*, entre otras, posibilita dar un mayor orden a la información en el internet, así como una nueva forma de entenderla más fácilmente.

Una de las tecnologías para ordenar la información en la web y hacerla accesible al público en general son las ontologías, las cuales permiten catalogar la información de forma más exacta evitando información innecesaria. El marco ontológico va a permitir consultar la información guardada en las ontologías de *SV* y mostrarla de una forma ya previamente diseñada para ser dinámica adaptándose a las diferentes fuentes de datos mediante consultas *SPARQL* también ya diseñadas para adaptarse a las diferentes ontologías en general, facilitando su comprensión.

La Figura 3-24 ejemplifica una de las tres posibles resultantes de este módulo, cuyas indicaciones son:

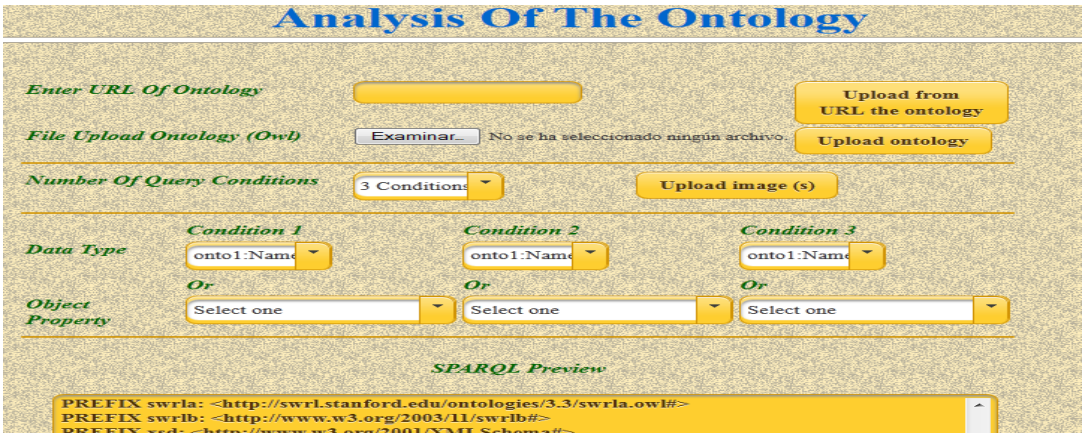


Figura 3-24. Diseño de consulta *SPARQL*

1. Seleccionar el número de condiciones para la consulta *SPARQL*, como se observa en la Figura 3-25.

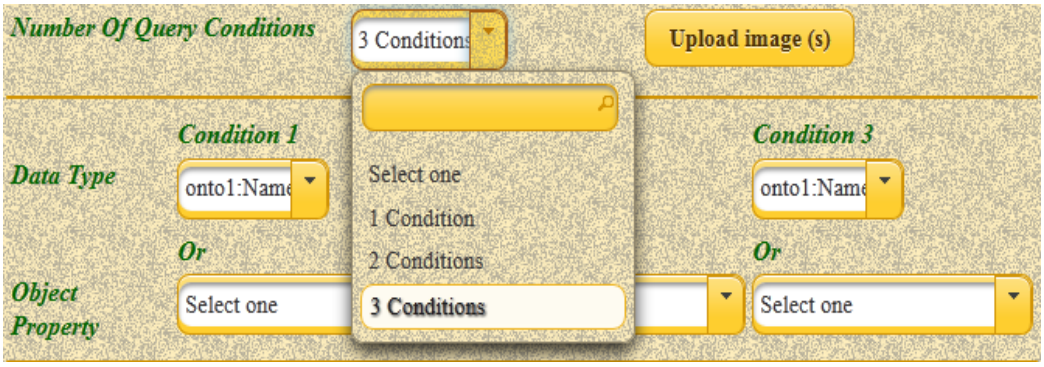


Figura 3-25. Selección de número de condiciones

2. En el caso de seleccionar una condición se habilitan las cajas de *Data Type* y *Object Property* correspondientes a la condición 1.
3. Seleccionar el campo deseado listado en los componentes de tipo caja disponibles para cada condición, como se visualiza en Figura 3-26.

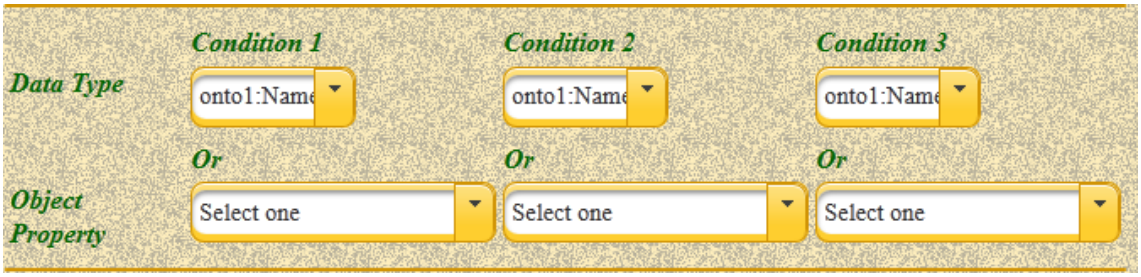


Figura 3-26. Componentes tipo caja para una condición

4. En el caso de seleccionar 2 condiciones se habilitan las cajas de *Data Type* y *Object Property* correspondientes a la condición 1 y 2.
5. Seleccionar el campo deseado listado en los componentes de tipo caja disponibles para cada condición, como se observa en la Figura 3-27:

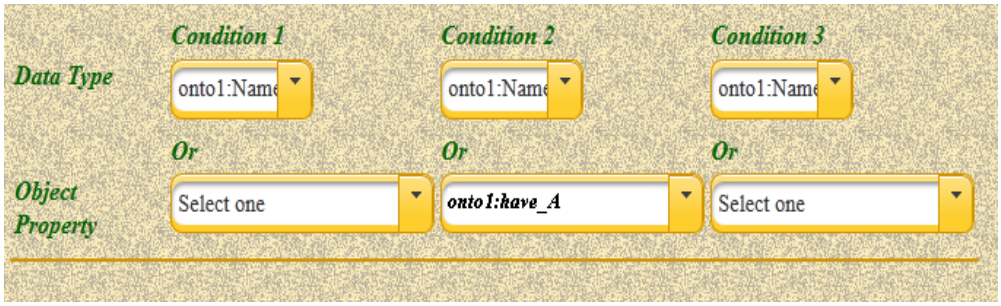


Figura 3-27. Componentes tipo caja para dos condiciones

6. En el caso de seleccionar 3 condiciones se habilitan las cajas de *Data Type* y Object Property correspondientes a la condición 1, 2 y 3.
7. Seleccionar el campo deseado listado en los componentes de tipo caja disponibles para cada condición, como se observa en la Figura 3-28.

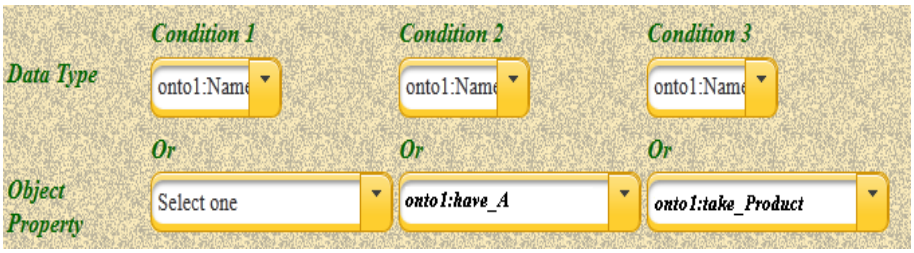


Figura 3-28. Componentes tipo caja para tres condiciones

Ejecución de consulta *SPARQL*

Los pasos para ejecutar el caso de uso del módulo son:

1. Presionar botón *Query* para ordenar ejecutar la consulta *SPARQL*, como observa en la Figura 3-29.

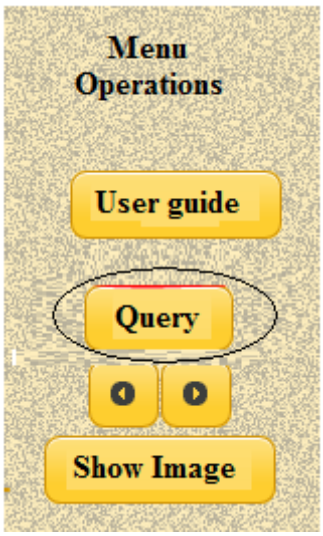


Figura 3-29. Ejemplo del botón *Query*

2. Los datos resultantes de la consulta *SPARQL* sobre la ontología son recibidos por la aplicación.
3. Los datos son salvados en una lista, sin importar si dio o no datos la consulta *SPARQL*.
4. Presenta los datos ordenados lo más parecido al lenguaje natural humano, presentando primero el sujeto seguido del predicado, y por último el objeto, añadiendo entre cada uno un espacio en blanco. Un ejemplo de resultados se muestra en la Figura 3-30:

Figura 3-30. Salida de consulta *SPARQL*

3.7 Principales herramientas de *software* utilizadas para el desarrollo de las herramientas *API Jena*[14]

Existen diferentes formatos de ontologías para la representación de información en la web semántica. Existen desde los más expresivos, *OWL Full*, a los menos expresivos, *RDF-S*. Por medio de esta *API* de gestión de ontologías, se da al usuario una interfaz de programación (*API*) con diferentes funcionalidades para crear de aplicaciones que requieren del uso de ontologías, indiferentemente del esquema de ontología que use en el programa (The Apache Software Foundation).

La *API* de *Jena* es pensado para que sea lo más neutral posible. Por ejemplo los nombres de clase de *Java* no son específicos del esquema de ontología a usar, es decir; la clase *java* "*OntClass*" puede ser una clase *OWL* o clase *RDFS* (The Apache Software Foundation).

Por tal motivo a cada esquema de ontología se asigna un *perfil*, que señala los constructores, los nombres de las clases y propiedades de las que hace uso (The Apache Software Foundation).

Para dar un mayor entendimiento se da el caso del *perfil* de *OWL*, en dónde; la función *owl:ObjectProperty* genera un dato usable y el *perfil RDFS* es *null* ya que el *RDFS* no se definen propiedades objeto (The Apache Software Foundation).

[14] http://jena.apache.org/download/index.html

El *perfil* se une a un modelo de ontología, que es una versión extendida de *Jena* de la clase *Model*. La base de *Model* permite el acceso a las declaraciones de un conjunto de datos *RDF*. (The Apache Software Foundation).

Otra clase *java*, *OntModel* expande esto agregando soporte para los tipos de construcciones que se esperaría en una ontología *OWL*: clases (en una jerarquía de clases), propiedades (en una jerarquía de propiedades) y los individuos. (The Apache Software Foundation).

Cuando se trabaja con una ontología en *Jena*, toda la información permanece codificada como tripletas *RDF* (accesible en *Jena* como *Statements*) almacenados en el modelo *RDF*. Una de las características del *API* de *Jena* es que no cambia la representación *RDF* en las ontologías. Lo que sí hace es añadir un conjunto de clases con utilidades y métodos que hacen que sea fácil manipular las tripletas *RDF*. (The Apache Software Foundation).

Para facilitar el aprendizaje del *API* los nombres de predicados definidos en el lenguaje de ontologías corresponden a los métodos de acceso a las clases de *Java*. Por ejemplo, una la clase *java OntClass* tiene un método para listar sus superclases, lo que corresponde a los valores del método *subClassOf* propiedad perteneciente al esquema RDF. (The Apache Software Foundation).

3.7.1 API Pellet[15]

Pellet es un *API* gratuito y de libre distribución el cual ofrece las características de deducir información por medio del razonamiento lógico, utilizando ontologías *OWL*. (Clark & Parsia). Las ontologías que soporta son del formato *OWL FULL*, *DL* y 2 *E* (Clark & Parsia). *Pellet* también complementa las falencias de *Jena* respecto a ciertos tipos de datos que no soporta.

3.7.2 NetBeans IDE

El *NetBeans IDE* es un producto libre y gratuito sin restricciones de uso, el cual puede ser descargado desde su sitio web[16]. *NetBeans IDE* proporciona un amplio soporte de primera clase para las últimas tecnologías *Java* y las últimas mejoras de *Java* previamente a otros *IDE*. Es el primer *IDE* que proporciona soporte para *JDK 7, Java EE 7* y *JavaFX 2.* (Oracle). El editor de *NetBeans* soporta lenguajes como de *Java*, *C/C++*, *XML*, *HTML*, *PHP*, *Groovy*, *Javadoc*, *JavaScript* y *JSP*. Debido a que el editor es ampliable, se puede adaptar a otros lenguajes. (Oracle).

3.7.3 Arquitectura de red

El sistema implementa la arquitectura cliente / servidor, para comunicar al usuario remoto con la aplicación en el servidor local. Mientras tanto en el servidor internamente, se emplea la arquitectura distribuida en tres capas para procesar las peticiones del cliente.

---

[15] http://clarkparsia.com/pellet/download
[16] https://netbeans.org/downloads/

3.7.4 Aplicación servidor

La aplicación servidor tiene la tarea de dar respuesta a las peticiones de los clientes realizadas por medio de los navegadores web al servidor. Los componentes de este son:

- Base de Datos: Se recurre al servidor *PostgreSQL* de base de datos para la gestión y almacenamiento de los datos que utiliza el sistema Buscador de datos en Ontologías (*BDO*).

- Servidor Web: Se utiliza el servidor web *Apache Tomcat* para la publicación en internet del sistema web *BDO* estando disponible al público en general.

3.7.5 Aplicación cliente

La aplicación cliente es la interfaz por la cual se interactúa con el sistema *BDO*, se envían las solicitudes al servidor y se recibe las respuestas. Esta aplicación hace uso del navegador web para interactuar con el usuario. La aplicación da a los usuarios la posibilidad de registrarse (o ingresar cuando tengan usuario), subir una ontología (ya sea desde el equipo local o desde el internet), con la única condición de ser extensión "*.owl*", proceder a realizar las consultas a la ontología en estándar *SPARQL*, visualizar los resultados, y subir imágenes si son necesarias en la ontología.

3.7.6 Esquema de base de datos

La aplicación *BDO* emplea *PostgreSQL* como servidor de base de datos para gestión y almacenamiento de información. En la base de datos solo existe una tabla la cual maneja la información de los usuarios.

A continuación se indica la única tabla empleada por la aplicación como su función dentro de la misma:

Usuario: Esta tabla cumple 2 funciones la primera es, almacenar los usuarios registrados para usar la aplicación, y la segunda es verificar a los usuarios ya ingresados en la aplicación web evitando conflictos en el uso de espacio de disco duro en el servidor web.

En la Figura 3-31, se indica la única tabla que compone el esquema de base de datos de la aplicación *BDO*.

Figura 3-31.Representación de base de datos del BDO.

A continuación se van a desarrollar ejemplos que utilizan las herramientas del marco propuesto en esta tesis. Estos empiezan describiendo el uso de la conceptualización del sensor hasta el modelado de situaciones de alto nivel semántico, cuando estas pueden ser descritas con claridad por el experto, es decir; aquí se modela el conocimiento *apriori*. Además, se detalla los algoritmos que dentro del marco permiten obtener el conocimiento adecuado cuando el conocimiento del experto no está claro referente a la

situación de interés y es necesario utilizar algoritmos de aprendizaje para obtener el conocimiento-detalle de la misma.

A continuación se describe algunos ejemplos sencillos de uso del marco propuesto en esta tesis. Se utilizan las herramientas de modelado así como se expone el uso de reglas y algoritmos para consultar e inferir situaciones de interés.

3.8 Conceptualización de las actividades compuestas.

En este apartado se modelan un ejemplo de modelado de actividades que conducen a situaciones de alto nivel semántico. En el marco tenemos las herramientas suficientes que nos permiten lograr este cometido. El lenguaje de modelado a utilizar es *VERL*, y utilizamos nuevamente los datos del proyecto *CASAS* con el fin de describir a detalle el modelado de una actividad compuesta de alto nivel semántico.

En *CASAS* las actividades que se realizan entre las 12:00 y 12:45 son para preparar el almuerzo:

Realizar una llamada por teléfono para anotar una receta. Los participantes caminan por el comedor con el teléfono en la mano, buscan un número telefónico en la guía telefónica, marcan el número y escuchan un mensaje en el teléfono. Los mensajes grabados corresponden recetas de cocina que los participantes anotan en un cuaderno.

- Lavarse las manos: Los participantes caminan en la cocina, y se acercan al fregadero para lavarse las manos. Usan jabón y secan sus manos con una toalla de papel.
- Cocinar: Los participantes cocinan una sopa de avena de acuerdo la receta. Para ello los participantes vierten agua en una olla de acuerdo a la medida, ubican la avena en la olla y luego agregan azúcar y pasas.
- Comer: Los participantes se sirven la sopa de avena y una medicina en el comedor.

- Limpiar: Los participantes toman los platos y los llevan al fregadero, para lavarlos con agua y jabón.

VERL conceptualiza la actividad tal y como se muestra en el Código 3-7. Los niveles de detalle de las zonas pueden ser obtenidos mediante *Building_Architecture.* En el código se hace uso de restricciones que trabajan con los estados de los sensores. Por ejemplo la restricción *M is ON* implica que hay un sensor de movimiento en la zona. Esto debe haber sido conceptualizado con la clase Sensor a través de la propiedad *BA:isCompose*.

```
Model:
ComposeEvent Lunch_Activities
        PhysicalObjects:
                ( p : Person,z1: Zone, z2:Zone)
                Components :
                ((c1: PrimitiveState Walk (p,z1))
                ((c2: PrimitiveState Write_Receipt (p,z1))
                ((c3: PrimitiveState Washing_Hands (p,z1))
                ((c4: PrimitiveState Cook_Meal (p, z2))
                ((c5: PrimitiveState Eating (p, z1))
                ( (c6: PrimitiveState Cleanning (p, z2))
constraints :
        (t Time:during 12:h00 – 12:45)
        (c1 Time:meet(5) c2)
        (c2 Time:meet(10) c3 )
        (c3 Time:meet(10) c4 )
        (c4 Time:meet(10) c5 )
        (c5 Time:meet(10) c6 )
Instance:
e1: ComposeEvent Lunch_Activities (VERL:Paul, BA:Dinner_Room,BA:Kitchen)
Model:
PrimitiveStates Walk
        PhysicalObjects:
                (p : Person,z1: Zone)
        constraints :
        (SSN:M is ON ) /*movement sensor
Instance:
e1: PrimitiveState Walk (VERL:Paul, Dinning_Room)
```

```
PrimitiveEvent Write_Receipt
        PhysicalObjects:
                (p : Person,z1: Zone)
        constraints :
         (SSN:* is ON ) /* * is phone sensor
Instance:
e1:  PrimitiveEvent  Write_Receipt  (VERL:Paul,
BA:Dinning_Room)
Model:
PrimitiveEvent Washing_Hands
        PhysicalObjects:
                (p : Person,z1: Zone)
        constraints :
         (SSN:WC1 is ON ) /*water sensor
Instance:
e1: PrimitiveEvent Washing_Hands (VERL:Paul, BA:Kitchen)
Model:
PrimitiveEvent Cook_Meal
        PhysicalObjects:
                (p : Person,z1: Zone)
        constraints :
         (SSN:KC1 is ON ) /*kitchen sensor
Instance:
e1: PrimitiveEvent Cook_Meal(VERL:Paul, BA:Kitchen)
Model:
PrimitiveEvent Eating
        PhysicalObjects:
                (p : Person,z1: Zone)
        constraints :
         (SSN:MC1 is ON ) /*medicine Sensor
Instance:
e1: PrimitiveEvent Eating (Paul, BA:Dinning_Room)
Model:
PrimitiveEvent Cleanning
        PhysicalObjects:
                (p : Person,z1: Zone)
        constraints :
         (SSN:WC1 is ON ) /*water sensor
Instance:
e1: PrimitiveEvent Cleanning (VERL:Paul, BA:Kitchen)
```

Código 3-7. Conceptualización de la actividad Preparar el almuerzo con *VERL*

En este ejemplo se utiliza la propiedad temporal *meet*. Esta propiedad viene de *Time*, pero nosotros utilizamos un umbral temporal, ya que los evento no son exactamente coincidentes, pero para modelar este tipo de actividades si necesitamos que un evento tome en cuenta el inmediatamente anterior. Aquí definimos un tiempo de monitoreo

(12:00 hasta 12:45) en el cual se registran los eventos. Este tiempo tiene relación directa el periodo en el cual se realiza las tareas para preparar el almuerzo. Es decir, que nos encontramos frente a que un período temporal tiene relación directa con la situación que se está *Monitorizando*, y que los eventos antes o después de ese periodo o se toman en cuenta porque no son relevantes para dicha actividad. De allí que la definición de las relaciones temporales, tiene que ser clara en el modelado de una actividad. En el ejemplo que tratamos "Actividades para preparar el Almuerzo", la propiedad temporal *meet*, no actúa sola, sino en relación con un período temporal. Esto es válido para describir la situación de interés, pero no funcionaría para describir la situación si el período temporal va desde las 08h00 hasta las 10h00, porque allí no se registran eventos válidos para el ejemplo. Utilizando las herramientas de modelado, se puede exportar el código *VERL* a *OWL*. En las siguientes Figuras se muestra el modelo en Protégé:

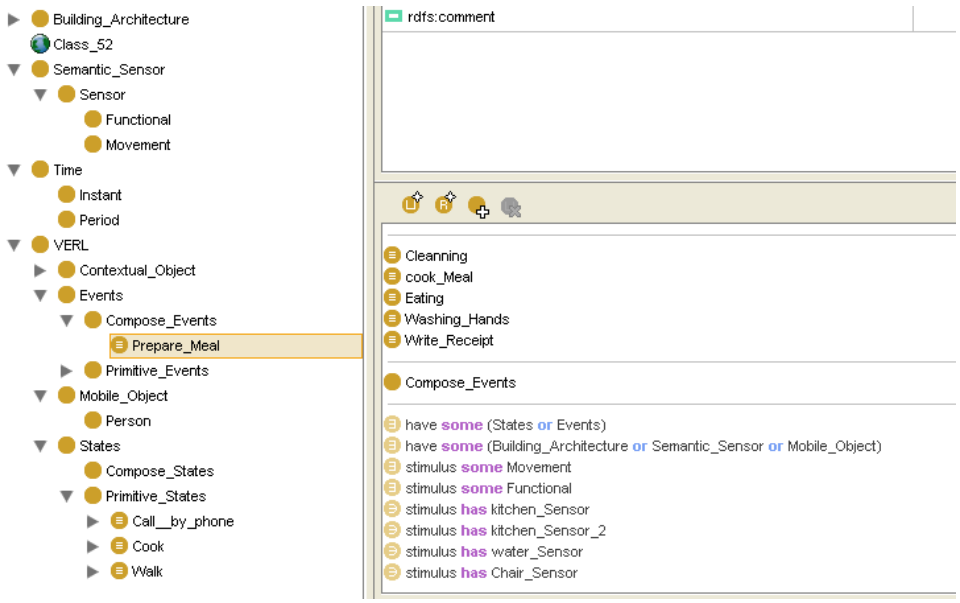


Figura 3-32.Resultado de la composición de actividades

En la Figura 3-32 se observa el resultado de la exportación del modelo de VERL. Las tareas para preparar el almuerzo, se registran según se van activando los sensores. En la Figuran 3-33 se observa los instantes de activación sensorial, y el uso de la propiedad *Meet*:

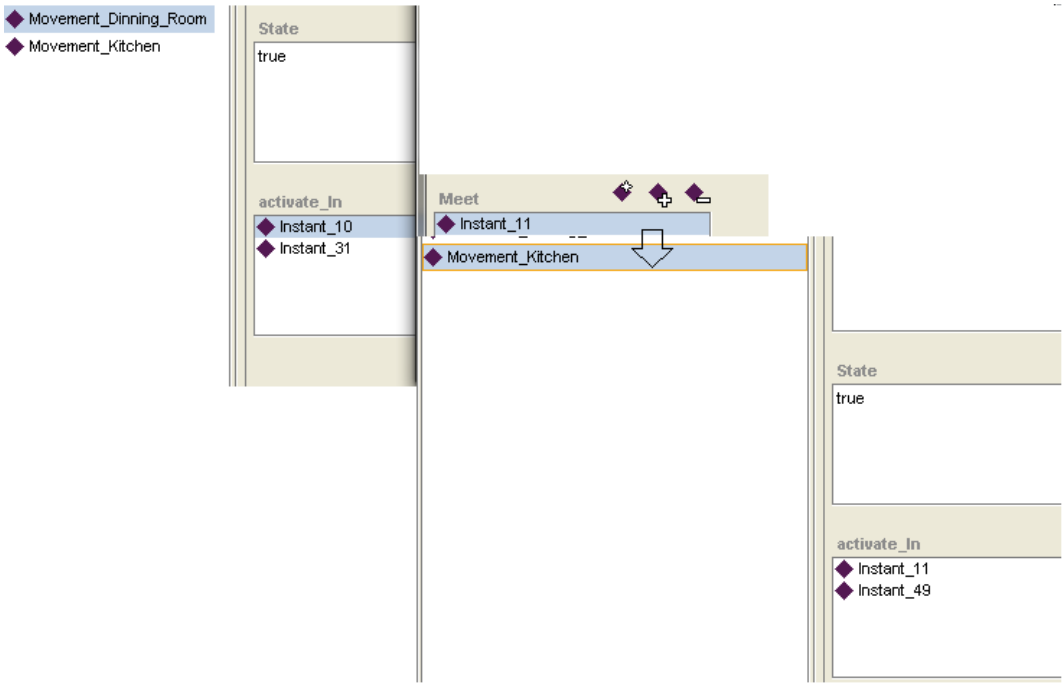


Figura 3-33. Instantes de activación del sensor y *Meet*

A través de la propiedad *Meet*, se agrupan los instantes en que se activan los sensores. La regla SWRL que al ejecutarse relaciona a los sensores con la propiedad Meet es:

Sensor (?vSensor) ∧ activate_In (?vSensor,?vInstant) ∧ sqwrl:Difference(?vSensor,?vInstant,?Umbral) → meet(?VSensor,?VInstant) **R 3-1**

La regla anterior utiliza las instancias de los sensores y su tiempo de activación, para comprobar que si la diferencia entre los instantes de activación es menor al umbral (5 minutos para nuestro caso) entonces se considera que los instantes son continuos y se

instancia la propiedad *Meet*. Como cada sensor está ubicado en un lugar (Dinning Room, Kitchen), se infiere la tarea que se está ejecutando.

Se ha modelado una situación a partir de sus componentes, actividades de alto nivel semántico, y estas a través de eventos de acciones de bajo nivel semántico. A continuación se describe el modelado de situaciones más complejas de modelar y es que en este tipo de modelados, se observa con mayor claridad la participación del marco. Se tiene en cuenta que existe el conocimiento *a priori* de experto para modelar las situaciones de interés y otras circunstancias en el cual este tipo de conocimiento no existe o no es lo suficientemente claro como para producir un modelo adecuado de la situación de interés. En este caso es necesario utilizar algoritmos de aprendizaje que también se forman parte del marco para obtener el conocimiento necesario que permita modelar situaciones con un alto nivel semántico. Los casos de uso analizan comportamiento humano sospechoso, puesto que para los sistemas de *SV,* los registros de las actividades sospechosas de las personas son lo más principal, a pesar; de que también se puedan *Monitorizar* objetos o zonas para otro tipo de experimentaciones (Aguas, 2010) (Albusac, 2008).

4 Modelado de situaciones a partir de conocimiento *a priori*

En este apartado presentamos los casos de estudio, en los cuales el experto conoce la situación y puede detallarla. En cada uno de ellos, se trata de modelan las situaciones de alto nivel semántico *Deambular_nocturno*, y *Hurto_con_mochila* aprovechando las potencialidades del marco propuesto en esta tesis. Estas situaciones están directamente relacionadas con la situación *Merodear* que ocurre cuando una persona visita varias veces los mismos lugares en diferentes momentos como si caminara sin rumbo (Aguas, 2010). *Merodear* se convierte en *Deambular_nocturno* si ocurre en una casa de habitación y en la noche. La situación *Hurto_con_mochila* también tiene entre sus componentes a *Merodear*, ya que una persona regularmente antes de cometer un hurto, merodea por el lugar. A continuación, vamos a detallar el modelado de las situaciones de interés. Dado que *Merodear* es común para los casos de estudio, empezamos modelándola en el marco.

4.1 Situación *Merodear*

(Nathaniel, Masoud, Papanikopoulos, & Issacs, 2005) desarrollaron un algoritmo que infiere la situación *Merodear* en una parada de autobús. Allí la persona permanece en la parada durante un período prolongado de tiempo sin un propósito específico (definición de experto). En escenarios de compras en supermercados, *Merodear* se define como la situación en que una persona ingresa a un supermercado, camina por los pasillos y visita ciertas zonas repetitivamente (Aguas, 2010). En la Figura 4-2, se muestra un ejemplo de este tipo de situación:

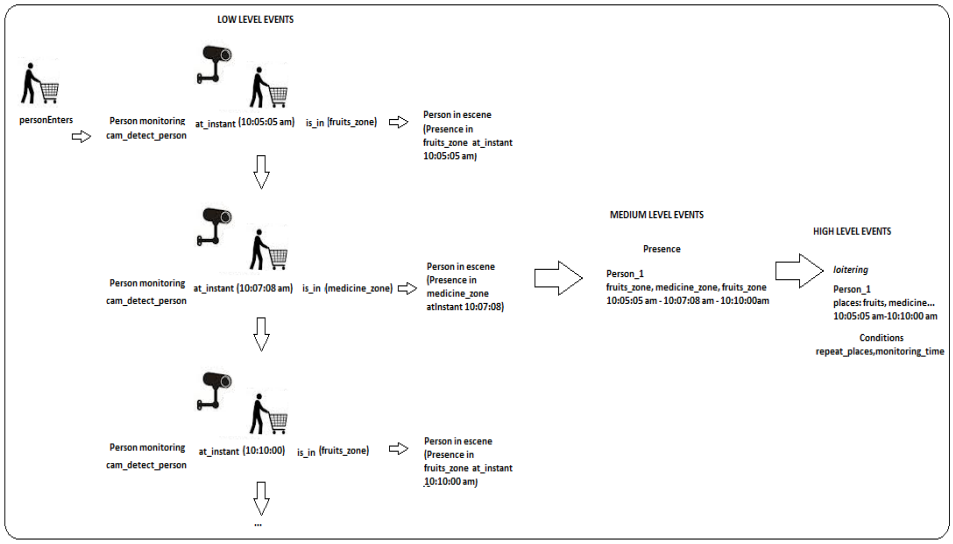


Figura 4-1. Pasos para inferir la situación Merodear: La cámara detecta la presencia de la persona en el supermercado. Axiomas que han sido dispuestos en los sistemas de *SV* infieren las actividades de alto nivel semántico.

Como se puede ver en la Figura 4-1, la persona repite los lugares que va visitando (*Fruits_zone* por ejemplo) y eso hace que sea considerada como *Merodeador*. Las definiciones anteriores, provienen del experto (Aguas, 2010). La tarea del ingeniero del conocimiento es trasladar estas definiciones a *VERL*. Para ello utilizamos las herramientas del marco:

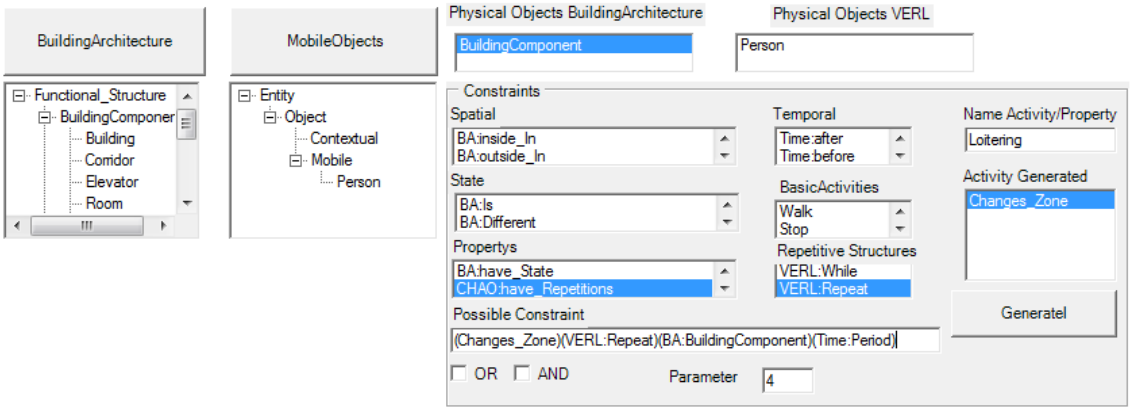


Figura 4-2. Modelado de la situación *Merodear*

Model:

Primitive_Event Loitering(p,z1,z2,t)

PhysicalObjects:

(p=VERL:Person,z1=BA:Building_Component, z2=BA:Building_Component,t=Time:Period)

Components :

(Changes_Zone(p,z1,z2,t))

constraints

VERL:Repeat(BA:BuildingComponent),t)(4)

Código 4-1. Situación *Merodear* modelada en *VERL*

En la Figura 4-2, se muestra el modelado de la situación *Merodear* en *VERL*. El módulo de Exportación (botón Generate), obtiene el modelado en *VERL*. La restricción VERL:Repeat(BA:BuildingComponent),t)(4) indica que la persona debe repetir al menos cuatro veces la visita a un lugar para considerarse como merodeadora (Aguas, 2010). Esta restricción se exporta como axioma en *OWL*, tal como se muestra en las siguientes Figuras

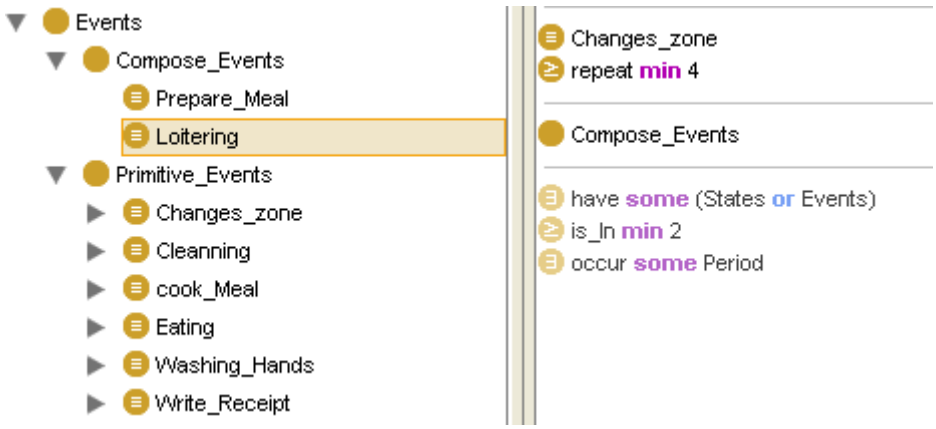


Figura 4-3. Modelo de la situación Loitering en Protégé

En la Figura 4-3, se observa un parámetro de repetición (*repeat min 4*). Este modelo le sirve al ingeniero de conocimiento para construir las reglas necesarias que nos permitan inferir la situación. Estas reglas son las tecnologías semánticas a fines con las que se trabaja en el marco, como la Regla 4-1 que permite la inferencia de la situación.

```
cam_Detects_Person(?camDetects)                    ∧
camera(?camDetects, ?videoCamera)                  ∧
at_Instant(?camDetects,?atInst)                    ∧
person_At(?camDetects, ?Person)                    ∧
is_In(?videoCamera, ?place)                        ∧
swrlx:createOWLThing(?Presence, “Presence”, ?atInst)
->
presence(?Presence)                                ∧
person_At(?Presence, ?Person)                      ∧
at_Instant(?Presence, ?atInst)                     ∧
in_Place(?Presence, ?place)
```

R 4-1. Inferencia de la actividad *presence* a partir de cam_Detects_Person

La regla 4-1 infiere la actividad *presence*, la cual ocurre cuando una persona entró en el lugar que estaba siendo *Monitorizado* por la cámara. Esto también ocurre cuando la persona cambia su lugar (*Changes_zone)*. En este caso, se registra una nueva instancia de la *presence*. Aclaramos, que en este caso estamos emulando el funcionamiento de modulo sensorial de *Horus.* La siguiente regla verifica que la persona está en el área *Monitorizada* (*presence)*, y que visita repetidamente algunos lugares. La regla obtiene el número de veces (*?countVisits*) que una persona visita el mismo lugar (?*in_Places*). Además, compara si este número de visitas puede ser considerado normal (*?normalVisit*). De allí, que sí *?countVisits* es mayor que *?normalVisit*, entonces la situación *Merodear* es inferida, con la regla R 4.2. Todo esto ocurre en un Time:period (?monitoring_time).

```
presence(?presence)                                    ∧
person(?presence, ?person)                             ∧
in_place(?presence, ?inPlace)                          ∧
at_instant(?presence, ?atInst)                         ∧
visited_places(?visited)                               ∧
in_place(?visited, ?inPlace)                           ∧
in_place(?visited, ?person)                            ∧
count(?visited, ?countVisits)                          ∧
sqwrlb:add(?countVisits,?countVisits,1)                ∧
normal_visits(?normalVisits)                           ∧
in_place(?normalVisits,?inPlace)                       ∧
normal_count(?normalVisits, ?Count)                    ∧
sqwrl:greatherThan(?countVisits, ?Count)               ∧
time:during(?atInst, ?monitoring_time)                 ∧
swrlx:createOWLThing(?loitering, “loitering”, ?atInst1)
->
loitering(?loitering)                                  ∧
person_At(?loitering, ?Person)                         ∧
in_place(?loitering,?inPlace)                          ∧
at_instant(?loitering,?atInst1)
```

R 4-2. Situación *Merodear* desde actividades de nivel medio

La definición anterior de *Merodear* se basa en contar el número de veces que una persona visita un lugar. Esta situación puede convertirse en *Deambular* cuando la persona forma ciclos con los lugares de visita. Por ejemplo, si la persona va del dormitorio, va al baño, luego va a la cocina, luego al pasillo, luego al baño, dormitorio y cocina, está formando ciclos (Park, Lin, Metsis, Le, & Makedon, 2010). La situación deambular se modela en *VERL* de la siguiente manera:

```
PROCESS PRESENT_LIST_PLACES
Presence=Horus_event
Place=BA:Functional_Structure
Occur_in=Time:Instant
Date=Occur_in(Date_time);
While not Presence.EOF
CREATE_LIST LIST(Presence,Place,Occur_in,Date)
Wend
LIST = LIST.SORT
While not  LIST.EOF
        System.println(LIST.Presence,LIST.Place,LIST.Ocurr_in,.LIST.Date)
Wend
END PROCESSS
```

Código 4-2. Presenta en pantalla una lista ordenada con los lugares de visita

⇩

is_In(?Presence, ?Place) ∧
have__Time(?Presence, ?occur_In) ∧
j.2:inXSDDateTime(?occur_In, ?Date)
→ sqwrl:concat(?SecuencePlaces,?Place) ∧ sqwrl:select(?Presence, ?Place, ?occur_In, ?Date) ∧ sqwrl:orderBy(?Fecha)

R 4-3. Codificación en SWRL

⇩

SQWRLQueryTab | Rule-2

?Evento	?Place	
Presence_10	Baño	2013-11-16T03:24:52
Presence_13	Dormitorio	2013-11-16T03:25:11
Presence_16	Pasillo	2013-11-16T03:25:22
Presence_19	Cocina	2013-11-16T03:25:29
Presence_22	Baño	2013-11-16T03:25:38
Presence_24	Dormitorio	2013-11-16T03:25:47
Presence_26	Pasillo	2013-11-16T03:25:54
Presence_28	Cocina	2013-11-16T03:26:01

Figura 4-4. Ejemplo en Protégé que muestra en pantalla los lugares de visita

El código 4-2 muestra el modelado en VERL de un procedimiento que nos permite presentar los lugares de visita de forma ordenada. El evento Presence, se utiliza para

representar que la persona fue identificada en un sitio de la casa. Este evento viene de Horus. Se utiliza el módulo de exportación para convertir el modelo en la regla SWRL R 4-3. Al ejecutar esa regla en Protégé se muestran los lugares de visita de ordenados. Al observar los resultados en la Figura 4-4, nos damos cuenta que la persona empieza a formar ciclos con los lugares que visita (los lugares Baño, Dormitorio, Pasillo y Cocina se repiten). En la misma regla con *sqwrl:concat*(?SecuencePlaces,?Place), formamos la cadena SecuencePlaces, en donde se encuentran todos los lugares de visita.

Para obtener los ciclos de repetición de lugares, se modela en *VERL* otro proceso que obtiene una subcadena de la lista de nombres de los lugares de visita:

```
RULE SUBSTRING
        SubString(CycleSecuences,LIST,4)
END RULE
```

Código 4-3. Proceso para obtener la subsecuencia en *VERL*

```
swrlb:substring(?SequencePlaces, ?CycleSequences, 1, 4)
```

R 4-4. Regla en SWRL que obtiene la subsecuencia

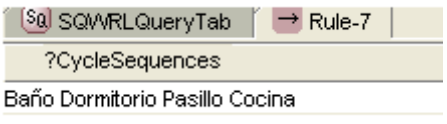


Figura 4-5. Ejemplo en Protégé

El Código 4-3 es el proceso en *VERL* para obtener una subsecuencia. Con el módulo de exportación se obtiene la regla *SWRL* R 4-4, que permite obtener la subsecuencia. La subsecuencia (*CycleSequences*) va a tener cuatro lugares a buscar, tal como se recomienda (Arens & Nagel, 2003) (Chun, 2008) (Haigh, y otros, 2004). La repetición de la *CycleSequences* en la lista total de eventos, indica que la persona forma ciclos con los lugares de visita y que probablemente deambula. El proceso en *VERL* y la regla que permite inferir la situación se muestran a continuación:

```
RULE FIND_SECUENCE
String.Found(CycleSecuences,LIST,4)
END RULE
```

Código 4-4. Proceso en *VERL* para verificar ciclos

```
sqwrl:Find(?CycleSequences, ?SequencePlaces, 0, 30)
```

R 4-5. Regla para identificar ciclos de visita en *SWRL*

Con las reglas en *SWRL* se ha podido inferir que la persona *Deambula*. Se aclara aquí que esta situación está compuesta por varios eventos *Presence*. Estos eventos constituyen el nivel medio semántico. Le reunión de varios *Presence*, nos permitió el obtener los lugares de visita, verificar si existen ciclos y luego inferir la situación de alto nivel *Deambular*. Para probar la eficacia del módulo de inferencia, se utilizó 40 vídeos etiquetados manualmente (actividades marcadas manualmente bajo nivel), con

situaciones de *Merodeo* y 40 vídeos en los que no ocurrió la situación de interés[17]. Los valores de *precisión* igual a 0,71 y la *recall* igual a 0,92 se obtuvieron del experimento. Estos resultados demuestran que *TH* puede inferir la situación de interés (sin perder los aspectos positivos y sin generar falsas alarmas exageradas).

4.2 Situación Deambular_Nocturno

La enfermedad de *Alzheimer* puede borrar la memoria de los lugares conocidas por una persona, así como hacer que le sea difícil su adaptación a un nuevo escenario. Como resultado, las personas que tienen la enfermedad de *Alzheimer* pueden alejarse de sus hogares o centros de atención y perderse, ponerse nerviosos y quedar desorientados - a veces muy lejos de donde empezaron su caminata (Albusac, 2008) (Botia, Villa, & Palma, 2009) (Park, Lin, Metsis, Le, & Makedon, 2010). Para este experimento se va a hacer uso del sensor semántico *SSN* y de *BA* para identificar que una persona ingresa a un lugar o habitación. Un ejemplo de *Deambular_Nocturno* se muestra en la Figura 4-3:

[17] Videos tomados de youtube: [17] Dinero.com, in http://www.youtube.com/watch?v=bv_VTJXy7Ow, 2012.

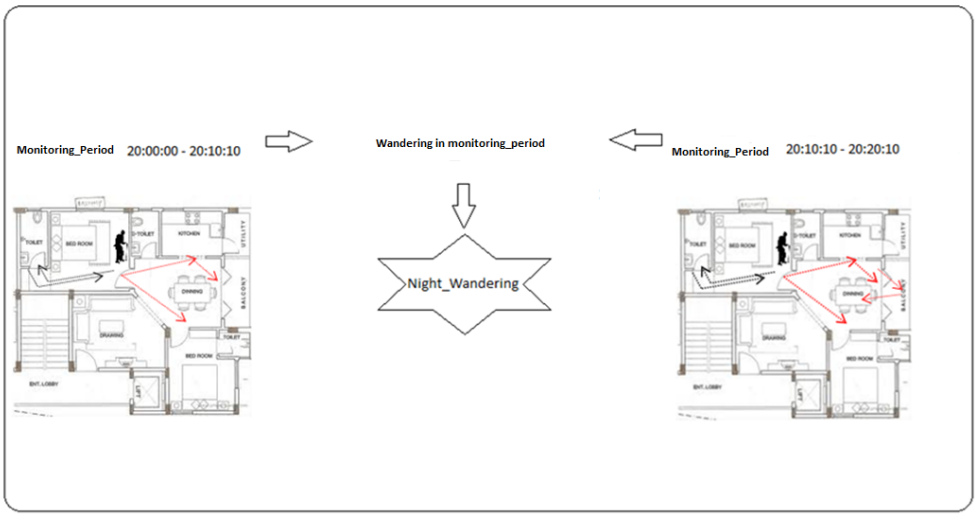


Figura 4-6. Inferencia de la situación deambular en la noche. La persona visita los mismos lugares (dormitorio, cocina, comedor) muchas veces.

La Regla 4.3 se utiliza para deducir este caso son los mismos que por *Merodear*, teniendo en cuenta que el evento de bajo nivel ahora viene de un *RFID*. Una vez más esta regla desacopla el nivel medio del dispositivo de detección físico.

sensor_Detect_Person(?person_enters) ∧ in_Sensor(?person_enters, ?video_camera) ∧ at_Instant(?person_enters, ?inst) ∧ person(?person_enters, ?Person) ∧ is_In(?video_camera, ?place) ∧ swrlx:createOWLThing(?Presence, “Presence”, ?inst) -> presence(?Presence) ∧ person_At(?Presence, ?Person) ∧ at_instant(?Presence, ?inst) ∧ in_place(?Presence, ?place)

R 4-6. Inferencia del evento *presence* a partir de la señal RFID.

La inferencia de la situación *Deambular_nocturno* parte de la inferencia de *Merodeo* con la siguiente restricción: Si se codifica para las reglas que el tiempo de *Monitorización* sea nocturno, las mismas reglas para *Merodeo* sirven para inferir la situación *Deambular_Nocturno*. Para probar la eficiencia de *CHAO* y sus reglas para

inferir esta situación, se trabajó con los registros *PL1A1* de (Ye, Coyle, Dobson, & Nixon, 2007) y con los registros de *Cairo.dat* y *Aruba.dat* que forman parte del proyecto CASAS. Los resultados de este experimento son alentadores (F1Score = 0,71[18]), los valores de precisión y la recuperación se discuten en la sección de conclusiones de este apartado.

4.3 Situación Hurto con Mochila

Esta situación se constituye en un tipo de hurto en el cual la persona esconde los productos de un supermercado en su ropa o en una mochila (*backpack)*. El patrón que cumple esta persona, es que primero analiza el lugar (*Merodear*), escoge su producto (*take_product, moves_object)* y luego comete el hurto. Todo esto ocurre en un corto periodo de tiempo (*Time:during*), lo cual tiene que estar coordinado porque en muchas ocasiones existen solapamientos entre el tiempo que la persona toma el producto (?*takingPeriod*) y el tiempo en que el producto desaparece del escenario (*?oclussionPeriod*) (Zhao, Wang, & Chong-Qing, 2002). El análisis de oclusión lo realiza *Horus* por medio del módulo de fusión multisensorial. La Figura 4-5, muestra la forma como se modela la situación en el marco:

[18] Métrica F1Score: http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html, tomado el 14-10-2013

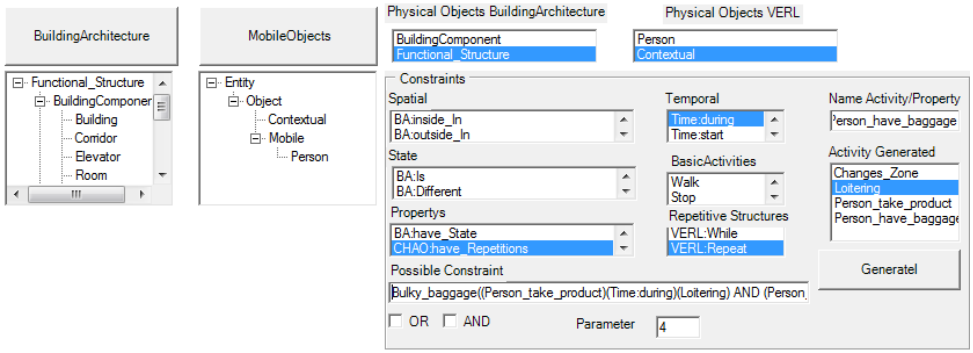


Figura 4-7. Modelado de la situación Bulky_baggage

Model:
ComposeEvent Bulky_baggage(p,pp,z1,z2,t)
PhysicalObjects:
(p = VERL:Person,pp=VERL:Contextual_object,z1=BA:Functional_structure, z2=Functional_structure, t=Time:Period)
Components :
((c1:ComposeEvent(Loitering(p,z1,z2,t)))
((c2:PrimitiveEvent(Person_Take_Product(p,pp,t)))
((c3:PrimitiveEvent(Person_have_baggage(p,Bulky,t)))
((c4:PrimitiveEvent(Product_Oclussion(p,pp,t)))
Constraints :
(c2 Time:during c1)
(c3 Time:during c1)
(c4 Time:during c1)

Model
PrimitiveEvent Person_take_product (p,pp,t)
PhysicalObjects:
(p =VERL:Person,pp:VERL:Contextual_object, t=Time:Period)
constraints :
(SSN:Deployment_sensor (Person_take_product(p,pp,t)) /* sensor

Model
PrimitiveEvent Person_have_baggage (p,pp,t)
PhysicalObjects:
(p =VERL:Person,pp:VERL:Contextual_object, t=Time:Period)
constraints :
(SSN:Deployment_sensor (Person_have_baggage(p,pp,t)) /* sensor

Model
PrimitiveEvent Product_Oclussion(p,pp,t)
PhysicalObjects:
(p =VERL:Person,pp:VERL:Contextual_object, t=Time:Period)
constraints :
(SSN:Process (Oclussoin(p,pp,t)) /* sensor

Código 4-5. Modelo de la situación Bulky_baggage en *VERL*

En los Códigos 4-2, pp=VERL:Contextual_object, se refiere a un producto que se encuentra en el supermercado. Se utiliza la propiedad SSN:Deployment_sensor para indicar que existen algoritmos que son capaces de identificar cuando una persona toma un producto y cuando tienen una mochila. El proceso de identificación se realiza desde el módulo de fusión de *Horus*. Con este modelo en *VERL* se diseñó la siguiente regla que permite inferir la situación:

```
loitering(?loitering)                                          ∧
at_Instant(?loitering, ?inst-1)                                ∧
take_Product(?takeProduct)                                     ∧
person_Take(?takeProduct, ?Person)                             ∧
product_Take(?takeProduct, ?product)                           ∧
in_place(?takeProduct, ?atPlace)                               ∧
oclussion (?product)                                           ∧
period_take_product(?product, ?takingPeriod)                   ∧
period_Oclussion_Product(?product, ?oclussionPeriod)           ∧
time:during(?inst-1, ?takingPeriod)                            ∧
time:during(?takingPeriod, ?oclussionPeriod)                   ∧
swrlx:createOWLThing(?bulky, "bulky_baggage", ?oclussionPeriod)
 ->
bulky_Baggage(?bulky)                                          ∧
person_Is(?bulky,?Person)                                      ∧
in_Place(?bulky,?inPlace)                                      ∧
 at_Period(?bulky,?oclussionPeriod)
```

R 4-7. Inferencia de la situación *Hurto con Mochila* a partir de la situación *Merodear*

Para probar la eficacia de nuestra propuesta, *CHAO* se instanció con secuencias de actividades que se registraron en vídeos en los que *Hurto con Mochila* apareció y 40 en los que no apareció[19]. Los valores de *precisión* y de *recall* para este experimento, parecen demuestran que *TH* en tiempo de ejecución puede inferir la situación (sin

[19] http://www.youtube.com/watch?v=bv_VTJXy7Ow, 2012. http://www.youtube.com/results?search_query=compras+en+supermercados&oq=compras+en+supermercados&gs_l=youtube.3..0.201.2351.0.2498.24.13.0.8.8.1.190.981.7j5.12.0...0.0...1ac.YxIZlm0lM5M, 2012.

perder los aspectos positivos), sin embargo, debemos tener cuidado con las falsas alertas que se generan. Los resultados de la etapa de experimentación muestran que podemos inferir situaciones basado en el reuso de la conceptualización y reglas semánticas. La Tabla 4-1, muestra los resultados comparativos.

Tabla 4-1. Resultados comparativos

Situación	***TP***	***TN***	***FP***	***FN***	***Precision***	***Recall***	***F1Score***
Merodeo	37	15	25	3	0,71	0,92	0,80
Deambular_Nocturno	81	45	55	19	0,64	0,81	0,71
Hurto_con_Mochila	24	16	24	16	0,6	0,6	0,6

Para la situación *Merodeo*, los resultados muestran mejoras en relación con el estudio (Williem, Vamsi, Boles, & Wageeh, 2008), en donde no se puede diferenciar entre una situación normal y la situación de *Merodeo*. Para *Hurto_con_Mochila*, los resultados muestran que la situación se puede inferir. La inferencia de esta situación se basa en los avances en algoritmos de visión artificial (Masoud & Papanikolopoulos, 2001) (Isard & Blake, 2008). El resultado de *F1Score* igual al 64% para *Deambular_Nocturno*, tiene su explicación ya que se trabaja con conjuntos de datos en los que esta situación puede ser confundida con otras situaciones diarias (Cheng & Chen, 2007). Sin embargo, creemos que este tipo de inferencia puede ser utilizado con éxito para ayudar en el control de las situaciones en pacientes con *Alzheimer*, u otras enfermedades con efectos similares.

4.4 Conclusiones del modelado *a priori*

Los resultados obtenidos en la fase de experimentación, no se puede decir que sean definitivos. Por el tipo de situaciones en las que trabajamos, es particularmente importante encontrar una débil sospecha y no se trata de la identificación de un hurto menor o un robo mayor, sino más bien; sobre la obtención de sospecha previa, y en el establecimiento de un comportamiento que es lo suficientemente sospechoso. Por ello, no es necesario que la especificidad sea alta, pero sí es importante en estos casos de estudio que los falsos positivos sean pocos, ya que en procesos de *Monitorización* es muy importante que no se pase por alto la detección de una situación de interés. Por ejemplo, en el proceso de *Monitorización* de personas con Alzheimer, se puede aceptar que existan algunos falsos positivos, pero no falsos negativos, ya que esto haría que se pueda dejar a una persona deambulando constantemente sin la atención adecuada (Park, Lin, Metsis, Le, & Makedon, 2010) (Stevenson, 2007). Se demostró que nuestra propuesta puede modelar diversas situaciones y la reutilizar los componentes del marco y el uso de reglas de inferencia que trabaja con una arquitectura jerárquica de niveles semánticos.

Nuestro trabajo en este artículo comenzó con la necesidad de completar *Horus* con el reconocimiento de situaciones. De hecho, hemos hecho algunas contribuciones importantes en este sentido. Por un lado, era metodológicamente conveniente diferenciar entre eventos semánticos de mediano y alto nivel de los que eran de bajo nivel. Por lo tanto, sobre la base de esta teoría no nos limitamos a un sistema físico de control. Por otra parte, hemos sido capaces de trabajar hacia la obtención de sensores

semánticos con el procesamiento inteligente de imágenes. Por otro lado, desde el punto de vista teórico, el uso de las tecnologías semánticas facilita los prototipos de vigilancia modelado y la implementación de los escenarios y situaciones, para los tres niveles jerárquicos descritos. Tanto la metodología descrita, así como las tecnologías semánticas utilizadas, permiten su reuso y corroboran los resultados de la etapa de experimentación. En *CHAO* se logró integrar ontologías de escenarios específicos, que son coherentes con la metodología.

Para ejemplificar la propuesta, dos escenarios se modelaron: un escenario de hogar que fue monitoreado por sensores *RFID*, y un supermercado escenario que contiene situaciones sospechosas en los campos de la videovigilancia. Tres situaciones de alto nivel fueron modelados (inferidos) a través del seguimiento de las actividades registradas por la persona. Hemos trabajado en la premisa de que nuestro estudio facilitaría la labor de este agente humano. Ayudamos a centrar su atención no solamente en un escenario en el que se produjo un delito, sino también; por un accidente o una actividad repentina y donde sea posible el preaviso de situaciones sospechosas (que son normalmente actividades no súbitas). Este es precisamente el caso de las situaciones *Merodeo* y *Deambular_nocturno*: se refiere a la identificación de un comportamiento que es lo suficientemente sospechoso. Y por esta razón no es necesario que la precisión que sea alta, pero suficiente para que el agente humano no se sature con falsos positivos. Si este número puede ser relativamente alto como hemos mencionado anteriormente, el número de falsos negativos deben ser también considerablemente bajas. Por otra parte, siempre debemos tener en cuenta la posibilidad de identificar la sospecha previa de un acto ilegal real como el robo. De esta manera, los resultados que se obtienen se

ajustarán a estas necesidades. Para ser más concretos, estos experimentos demuestran que la propuesta facilita la semántica modelación de escenarios y actividades, con el fin de inferir las actividades sospechosas en la vigilancia multisensorial *ad hoc*.

Además, se ha demostrado con claridad que *TH* sirve para poder obtener secuencias de alto nivel semántico, con conocimiento explicito, es decir; *a priori*, que el experto puede modelar y explicar con facilidad. A continuación, vamos a tratar de obtener conocimiento mediante un sistema que permite obtener perfiles de una situación de interés. Este es el caso en cambio, cuando el experto no tiene claro los eventos ni su agrupación, y es necesario entonces contar con un sistema de aprendizaje que le ayude a clarificar la descripción de la situación *Monitorizada.*

5 El modelo del aprendizaje en el marco propuesto

5.1 Introducción

En nuestro trabajo, hasta ahora se ha podido modelar situaciones *a priori*, en los cuales el conocimiento del experto hace que las mismas puedan detallarse con expresividad. Dependiendo del tipo de escenario y del conocimiento *a priori* de las situaciones de alerta, existen dos posibilidades: 1) que existe conocimiento del experto humano en forma explícita y 2) que este conocimiento no existe *a priori*, aunque es posible, en principio, encontrar el conocimiento en secuencias de vídeo. En este segundo escenario, se requiere un proceso particularmente complejo (Orten, 2005), (Tian, Brown, Hampapur, & Lu, 2008) (Masseglia, Poncelet, & Tesseire, 2009) para obtener datos adicionales o para modelar los procedimientos necesarios que permitan alcanzar el conocimiento adecuado de la situación de interés. En este apartado, trabajamos bajo el supuesto de que es posible desarrollar un sistema que emula la capacidad de un experto para reconocer una situación de interés en un escenario de *SV*. En este tipo de escenarios por lo general, se busca identificar patrones de situaciones sospechosas, las cuales no se muestran con claridad en los registros de vídeo (Aguas, 2010) (Fern, Komireddy, & Burnnet, 2007) (Chun, 2008). Hemos diseñado dentro del marco, un sistema que aprende de un conjunto de casos sospechosos, y conformamos con ello un *perfil* de comportamiento de la persona *monitorizada* a partir del análisis de *micropatrones* (patrones cortos, formando secuencias de acciones repetidas lo suficientemente como para ser representativas). El sistema que plantemos es dinámico, puede seguir

aprendiendo de acciones y conductas sospechosas humanas en tiempo de ejecución. El detalle del sistema se describe a continuación.

5.2 TH-GSP

En el marco *TH*, se trabaja con el algoritmo *Generalized Sequential Patterns* (*GSP*) para obtener los *micropatrones*. En (Karikrishna, 2011) (Chikhaoui, Wang, & Pigot, 2010), el *GSP* fue también aplicado con éxito a los patrones de comportamiento humano en otros campos como los deportes y actividades de la vida diaria. (Srikant & Agrawal, 1996) utilizan *GSP* para obtener patrones secuenciales a partir de datos sobre los hábitos de compra de los consumidores en los supermercados. En nuestra propuesta, hemos considerado cambiar las secuencias de transacciones del planteamiento inicial (Srikant & Agrawal, 1996) por secuencias de etiquetas de actividades de interés que registra la persona *Monitorizad*a. Estas secuencias se utilizan para entregar a *GSP*.

5.2.1 Entrenamiento de *GSP*

Las Figuras 5-1 muestra la utilización de *GSP* para obtener los *micropatrones*

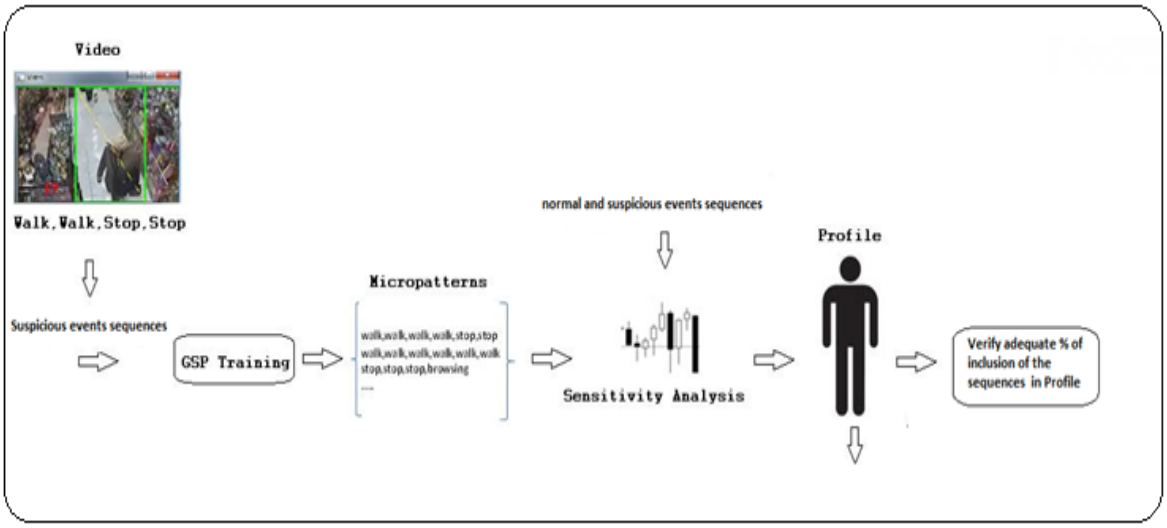


Figura 5-1. GSP está entrenado con las secuencias de actividades (eventos) sospechosas, con el fin de obtener micropatrones. Después de probar con nuevos casos de comportamientos normales y de hurto, a través del análisis de sensibilidad correspondiente, los patrones más característicos son seleccionados y agrupados en lo que llamamos un perfil

Para entrenar a *GSP* se utiliza como un fichero de secuencias etiquetadas de actividades de vídeo. Los pasos para entrenar a *GSP* son:

- Se toman todas las actividades individuales (1 - secuencias) de mayor frecuencia.
- Durante la segunda etapa, un conjunto candidato (2 – secuencias) se forman tomando en cuenta los más frecuentes (1 - secuencias). Las 2- secuencias frecuentes (subsecuencias) se utilizan para generar el candidato de (3- secuencias), y este proceso se repite hasta que no se encuentran secuencias más frecuentes.
- La generación de candidatos, se realiza teniendo en cuenta el conjunto de frecuentes (k - 1) - frecuentes secuencias de F (k - 1), los candidatos para la siguiente pasada se generan mediante la unión F(k - 1) consigo misma. Una fase de poda elimina cualquier secuencia, es decir, donde al menos una subsecuencia

no se considera frecuente, debido a que no supera un valor de umbral conocido como soporte mínimo (*min_support*).

El resultado de *GSP* son los *micropatrones* más representativos de la situación de interés. Después de esto, un análisis de sensibilidad se aplica a todos los *micropatrones*. Entonces, los más fiables se utilizan para identificar para la situación objetivo, estos conforman lo que conocemos como *perfil* de comportamiento.

5.2.2 Análisis de sensibilidad de los *micropatrones*

El análisis de sensibilidad de los *micropatrones* se lleva a cabo de la siguiente manera:

- Reconocer (*matching*) un *micropatrón p* en una nueva secuencia (*s*) implica la extracción de subsecuencia *s'* de *s*, que es de la misma longitud que p, y calcular la distancia de *Levenshtein* (*L*) entre los *p* y *s'*. Se llama distancia de *Levenshtein*, distancia de edición, o distancia entre palabras, al número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Se entiende por operación, bien una inserción, eliminación o la sustitución de un carácter (Cáceres, 2010). Si *L* es menor que el umbral α[20], el resultado positivo y por lo tanto s pertenece a un comportamiento sospechoso. Esto se repite para todos los *s'* que forman parte del *s*.

[20]Se define un umbral de distancias tal como lo hace (Iglesias, 2010) (en ese caso para distinguir estrategias de futbol). El uso de este umbral se justifica, ya que en una secuencia de actividades de video, los *micropatrones* pueden aparecer en cualquier lugar, y no es necesario que una secuencia tenga todos los *micropatrones* de un *perfil* para convertirse en sospechosas (Aguas, 2010) (Gonzalez, 2009).

- Prueba de las *micropatrones* iniciales con nuevas secuencias (esta vez con aspectos positivos y negativos). Los *micropatrones* se clasifican según su puntuación F1Score[21] ya que queríamos detectar casos positivos pero sin dejar pasar casos negativos como sea posible (teoría de equilibrio).
- Determinar el porcentaje de inclusión apropiada de un *perfil* en una secuencia. Este proceso consiste en determinar qué porcentaje de inclusión de los patrones de *perfil* en la secuencia de entrada proporciona mejores resultados o mejor caracteriza el comportamiento. Ya que como podemos ver en los resultados de las pruebas en la siguiente sección, es importante mantener un "*equilibrio*", pues si se requiere demasiado alto porcentaje de precisión, el índice de rendimiento es bajo.

Una vez que el sistema está operativo, en tiempo de ejecución podemos examinar el *perfil* con nuevas secuencias de interés, con el objetivo de generar un aviso para el operador humano.

5.2.3 Ejecución del sistema

La etapa de ejecución, comienza con el procesamiento de las imágenes desde *Horus*, a partir de las cuales se obtienen las secuencias de entrada para verificar si alguna de ellas se corresponde con el *perfil* sospechoso, y generar el correspondiente aviso para el operador humano. En la Figura 5-2 se muestra el esquema de funcionamiento de esta etapa:

[21] F1SCORE: http://www.monperrus.net/martin/understanding+the+f1-score, tomado el 13-10-2013.

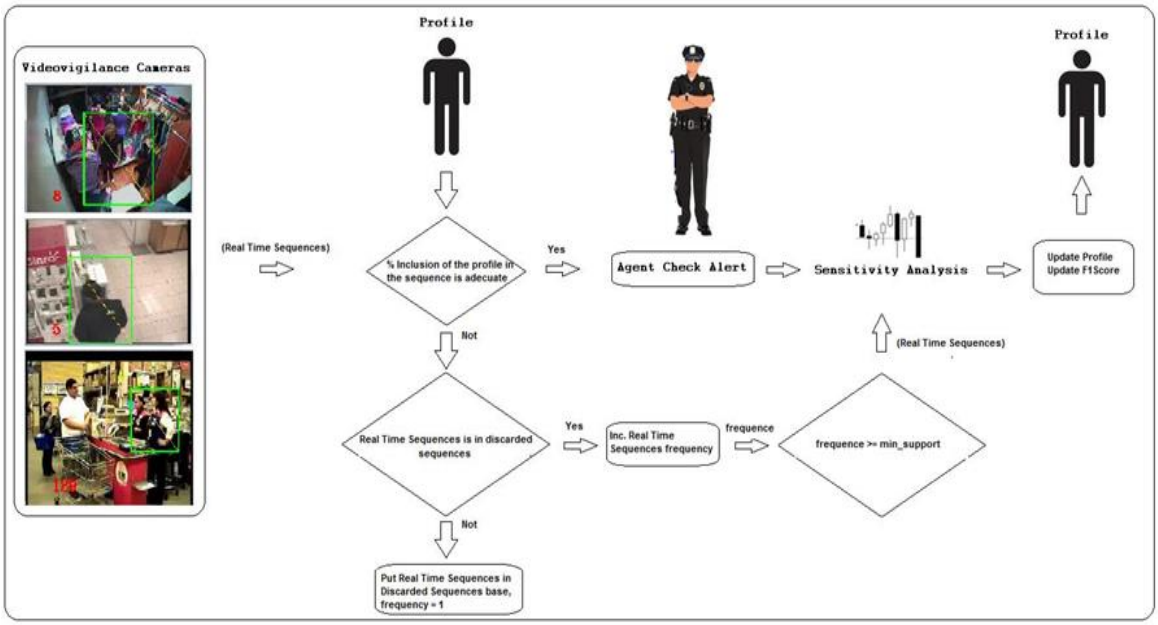


Figura 5-2.Sistema de seguimiento e identificación de situaciones sospechosas o comportamientos humanos sospechosos. El sistema verifica el porcentaje de inclusión del perfil en secuencias de eventos en tiempo real. Cuando coincida con el porcentaje óptimo, se genera una alerta. El perfil y los valores de sensibilidad se obtienen a continuación y se actualizan de acuerdo con la interacción del agente humano con el sistema.

La Figura 5-3 muestra la pantalla de usuario del sistema propuesto. Hay cuatro subventanas que muestran escenas grabadas en las cámaras de vigilancia de 8, 11, 5 y 29 respectivamente. Para cada escena, hay una persona que está siendo *Monitorizada*. Las secuencias obtenidas a través de la transformación de las escenas grabadas por las cámaras 8, 11 y 5 han sido clasificadas como escenas de comportamiento normal. Sin embargo, el operador humano tiene la opción de confirmar un falso negativo con el botón sospechoso o, para confirmar el verdadero negativo si el botón no se presiona dentro de 6 segundos, es decir, con el objetivo de verificar la viabilidad de nuestra propuesta. La cámara 129 detecta escenas cuyas secuencias correspondían a un *perfil* sospechoso. Por esta razón, se muestra una alerta de robo. En este caso, el operador

humano puede confirmar el verdadero positivo pulsando el botón de alerta seleccionada. Alternativamente, si el botón no se presiona dentro de 6 segundos, se genera una señal de falso-positivo.

Figura 5-3. Pantalla de Monitoreo de situaciones normales y sospechosas de comportamiento humano. Un mensaje de alerta es emitido en la zona 129 para el operador humano, el cual debe confirmar si el mensaje es correcto y con ello ejecutar un proceso de actualización del análisis de sensibilidad.

Gracias a la intervención del operador humano, el sistema puede aprender de una manera continua y no solo durante la etapa de entrenamiento. La descripción del aprendizaje dinámico se describe a continuación

5.2.4 Aprendizaje dinámico del sistema

El proceso de aprendizaje dinámico toma en cuenta los siguientes aspectos:

1) Cuando la frecuencia de una secuencia de actividades no alcanzó un nivel mayor que el *min_support* esta es descartada automáticamente, esto ocurre en *GSP* en la fase de entrenamiento. Las secuencias descartadas se guardan en *TH-GSP*.
2) Si las nuevas secuencias que ingresan al sistema contienen alguna de las secuencias descartadas, la frecuencia de la secuencia descartada aumentará. Si

la frecuencia de la secuencia descartada alcanza un nivel más alto que el *min_support*, esta secuencia puede convertirse en un *micropatrón*. Por lo tanto, se tiene que repetir toda la fase de entrenamiento y el análisis de sensibilidad. Por lo tanto, se puede concluir que cada vez que había un nuevo *micropatrón* en el sistema, existe la posibilidad de que aparezcan otros *micropatrones*.

Si se toma en cuenta la precisión a la hora de identificar la situación de interés si se generan demasiadas advertencias hacia el agente humano (advertencias verdaderas de situaciones de sospecha), esto degenera en un sistema de seguridad automático inutilizable. Creamos una alternativa que permite disminuir el número de falsas alertas, la misma que se describe a continuación.

5.2.5 Algoritmo de *Rank* de *micro-patrones*

El Algoritmo 5-1, que permite establecer un *Ranking* de las secuencias, de acuerdo, al número de *micropatrones* que contienen. Esto permite identificar que secuencia es más alertante en relación a todas las que ingresan al sistema.

```
Open "C:\tesis 2012\GSP_M\Programa\videos\videosHurtoYNormal.txt" For Input As #2
contador_de_secuencias = 0
contador = 0
Fila = 0
Do Until EOF(2)
 Line Input #2, secuencia
 contador = contador + 1
 Text1.Text = secuencia
 p = 0
Fila = Fila + 1
 Open "C:\tesis 2012\GSP_M\programa\Patrones\PRank.txt" For Input As #1
 Text2 = ""
 Text3 = ""
 Menor = 1000
 contador_de_patrones = 0
 Do Until EOF(1)
 p = p + 1
 secuenciar = secuencia
Line Input #1, patron
 ' If Len(patron) >= Val(Text4) Then
 For m = 1 To Len(secuencia) - 1
 distancias(m) = LD(Mid(secuencia, 1, Len(patron)), patron)
 secuencia = Mid(secuencia, m + 1, Len(secuencia))
 Next m
 posicion_patron(p, Fila) = 0
 encontrado = 0
For n = 1 To m - 1
 If distancias(n) <= 0 Then
 posicion_patron(p, Fila) = 1
 encontrado = 1
 n = m
 End If
 Next n
 ' End If
 If encontrado = 1 Then
 contador_de_patrones = contador_de_patrones + 1
 End If
 secuencia = secuenciar
 Loop
 Close #1
If contador_de_patrones <> 0 Then
 contador_de_secuencias = contador_de_secuencias + 1
 lista(contador, 1) = secuencia
 lista(contador, 2) = Str(contador_de_patrones)
 lista(contador, 3) = Str(contador)
Else
 lista(contador, 1) = secuencia
 lista(contador, 2) = Str(0)
 lista(contador, 3) = Str(contador)
End If
```

```
List1.AddItem "Secuencia:" + Str(contador) + "------------------------> " + lista(contador, 1) + "<;>"
+ lista(contador, 2)
Loop
 Close #2
 MsgBox "Reconocidas " + Str(contador_de_secuencias) + " secuencias en el test"
 MsgBox "Secuencias totales: " + Str(contador)
 Open "C:\tesis 2012\GSP_M\programa\Patrones\Rank.txt" For Output As #1
 For n = 1 To contador
 Print #1, lista(n, 1) + "," + lista(n, 2)
 Next n
 Close #1
 Call Ordenar(lista, 2)
 Open "C:\tesis 2012\GSP_M\programa\Patrones\RankF.txt" For Output As #1
 For m = 1 To 200
 List2.AddItem "Secuencia:" + lista(m, 3) + "------------------------> " + Mid(lista(m, 1), 1, 25) +
"<;>" + lista(m, 2)
 Next m
 Close #1
```

Algoritmo 5-1.Algoritmo de *Rank* de micropatrones en tiempo ejecución

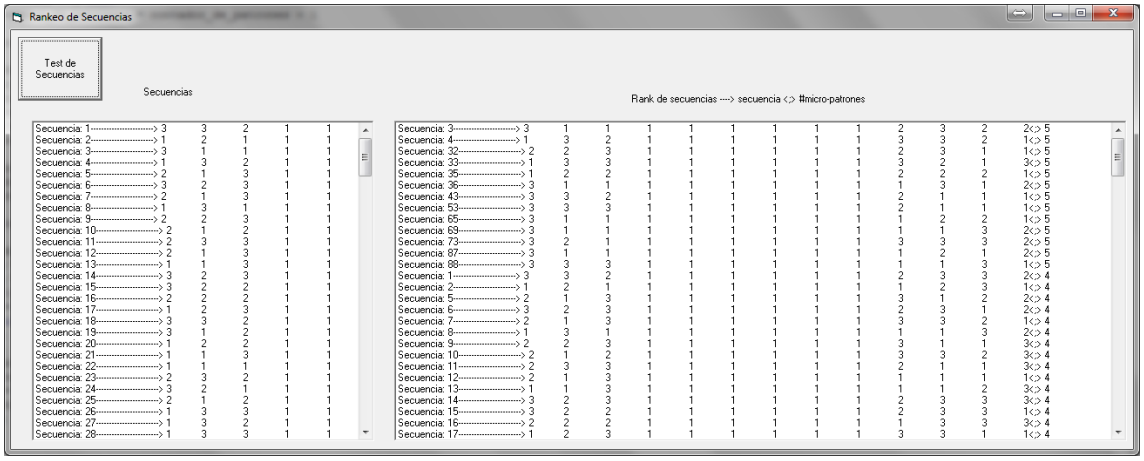


Figura 5-4. Resultado del *Rank* de las secuencias

Un ejemplo de resultados del Algoritmo 5-1, se muestra en la Figura 5-4. Se lista las secuencias, se verifica el número de *micropatrones* que contienen y se ordenan según ese número. De allí que se puede listar las más sospechosas para el operador del sistema.

Para la etapa de experimentación, se trabajó con secuencias etiquetadas manualmente e instanciadas en *CHAO* con el fin de evitar los problemas asociados con sistemas de reconocimiento de visión y para distinguir nuestros resultados de los problemas de reconocimiento de patrones habituales. Solo se consideran el tipo de eventos que eran identificables con algoritmos de visión disponibles (Chun, 2008) (BrotherSoft, 2011). Cada etiqueta se corresponde a un segundo de grabación en video de la persona *Monitorizada*. Los *micropatrones* son automáticamente identificados en dos escenarios diferentes. El primer caso incluye grabaciones en vídeo de las actividades sospechosas, situación de hurto en supermercados. El segundo caso se refiere a grabaciones de video de la situación merodeando en un centro comercial. En ambos casos, el operador humano examina secuencias de vídeo de diferentes cámaras. Entonces, el sistema genera una alerta cuando una persona demuestra un comportamiento que corresponde con el *perfil* sospechoso.

5.3 Hurto en supermercados

La experimentación con *GSP* (entrenamiento y prueba) se compone de lo siguiente:

- Observaciones de vídeo de hurto y el etiquetado manual de las actividades: 200 grabaciones de vídeo de robos en los supermercados (Dinero.com, 2009) fueron analizadas. Un asistente observó y etiquetó cada una de las actividades que se registraron en este tipo de vídeos durante 40 segundos, conformando con ello secuencias de actividades por vídeo. A continuación, cada secuencia de actividades se registró en un en *CHAO* y en un frontal diseñado para esta investigación. Cada

grabación de vídeo genera así una secuencia de entrada de etiquetas para fines de entrenamiento.

- Se etiquetaron 98 secuencias de vídeo (Youtube, 2009), que corresponde situaciones de comportamiento normal.
- Se obtuvieron un total de 298 secuencias mixtas de situaciones de comportamiento normal y sospechoso.
- El *GSP* fue entrenado usando las 200 secuencias positivas. Hemos probado exhaustivamente diversos valores de soporte mínimo y de umbral de inclusión. Mediante el uso de *min_support* = 0.3 (donde el 30 % de las secuencias se incluyen en el *micropatrón*) y $\alpha = 2$ (donde la distancia desde el *micropatrón* es menor que o igual a 2), hemos sido capaces de lograr más resultados fiables y precisos (ver tablas de resultados).
- Para el análisis de sensibilidad de estos *micropatrones*, se utilizaron las 298 secuencias mixtas. Los *micropatrones* con la mejor puntuación de *F1Score* fueron seleccionados para formar el *perfil*, logrando así un equilibrio óptimo entre la *precisión* y *recall*.
- Por último, se trabajó con las mismas secuencias utilizadas para el análisis de sensibilidad (298 secuencias de entrada) para determinar el porcentaje óptimo de *micropatrones* que debe estar presente en una secuencia, es decir, para garantizar que la secuencia contenía el *perfil* de comportamiento sospechoso.

Los resultados de la experimentación se muestran en las Tablas 5-1 y 5-2:

Tabla 5-1. Resultados del análisis de sensibilidad de los *micropatrones*

Micropatrones	***Precision***	***Recall***	***F1Score***
walks,walks,walks,walks,walks,turns-left,walks,walks,walks,turns-	0,67	0,92	0,77
walks, walks, walks, stops, stops, stops, looks around, looks around, looks	0,67	0,92	0,77
walks, walks, walks, stops, stops, looks around, looks around, looks	0,69	0,93	0,79
walks, walks ,walks, stops, stops, stops, looks around, looks around, looks	0,7	0,93	0,79
walks, walks, walks, stops, stops, stops, looks around, looks around, stops,	0,7	0,93	0,79

Tabla 5-2. Análisis de sensibilidad para determinar el porcentaje óptimo de inclusión en el *perfil*

% de inclusión en el *perfil*	***Precision***	***Recall***	***F1Score***
35	0,57	0,84	0,67
40	0,57	0,846	0,68
45	0,64	0,93	0,75
50	0,65	0,94	0,76
55	0,67	0,94	0,78
60	0,72	0,946	0,81
65	0,74	0,952	0,83
70	0,75	0,96	0,84
75	0,76	0,95	0,84
80	**0,81**	**0,95**	**0,87**
85	0,79	0,87	0,82

90	0,84	0,82	0,82
95	0,85	0,84	0,84
100	0,87	0,76	0,81

Interpretación de los resultados

Como puede verse a partir de las Tablas 5.1 y 5.2, se obtienen los resultados óptimos cuando las secuencias positivas se incluyen en el 80% del *perfil*. Los resultados con el *perfil* son válidos, ya que proporcionan un alto valor de memoria sin bajar el valor de precisión. Podemos suponer que es mejor para recuperar los objetos robados, pero, por supuesto, sin sobrecargar el operador humano, con continuas advertencias.

Con el fin de llevar a cabo una simulación de cómo funciona el proceso, llevamos a cabo una prueba final, la aplicación de 100 nuevas secuencias de entrada positivas y 100 nuevas entradas secuencias negativas con el *perfil* seleccionado.

Tabla 5-3. Rendimiento de los micropatrones (nuevas pruebas)[22]

Micropatrones	***Precision***	***Recall***	***F_1Score***
<walks,walks,walks,walks,walks,turns-left,walks,walks,walks,turns-left,stops,stops,stops,looks around, looks around>	0,67	0,92	0,77
<walks, walks, walks, stops, stops, stops, looks around, looks around, looks around, looks around, walks, walks, browses, looks around, looks around>	0,67	0,92	0,77
<walks, walks, walks, stops, stops, looks around, looks around, looks around, looks around, walks, walks, turns-left, walks, walks, walks, looks around>	0,69	0,93	0,79
<walks, walks, walks, stops, stops, stops, looks around, looks around, looks around, looks around, looks around, looks around>	0,7	0,93	0,79
<walks, walks, walks, stops, stops, stops, looks around, looks around, stops, turns-right, walks, walks, walks>	0,7	0,93	0,79
<walks, walks, walks, walks, stops, stops, stops, looks around, looks around, looks around, stops, looks around, turns-right, walks, walks, walks>	0,78	0,9	0,83
<walks,walks,walks,walks,walks,walks,walks,stops,stops,stops,stops,stops,looks around, looks around, looks around, stops, looks around, turns-left, turns-left, turns-left, walks, walks, walks>	0,78	0,9	0,83

Tabla 5-4. Análisis de sensibilidad para determinar el porcentaje correcto de inclusión de un micropatrón en el perfil (nuevo test)

[22]Como se puede observar, se utiliza para etiquetar actividades verbos en tercera persona como *walk*, *stop* y frases como *looking around*. Sin embargo, el sistema también puede trabajar con etiquetas de actividades más descriptivas, tomando en cuenta que estas puedan ser reconocidas por procesos semánticos o por algoritmos de visión artificial –persona nerviosa, formando grupos para distraer al personal de seguridad–.

% de inclusión en el *perfil*	***Precision***	***Recall***	***F1Score***
35	0,66	0,84	0,73
40	0,67	0,86	0,75
45	0,67	0,86	0,75
50	0,67	0,86	0,75
55	0,67	0,87	0,75
60	0,74	0,89	0,80
65	0,74	0,91	0,81
70	**0,81**	**0,91**	**0,85**
75	0,76	0,91	0,82
80	0,81	0,87	0,83
85	0,76	0,83	0,79
90	0,84	0,82	0,82
95	0,86	0,76	0,81
100	0,87	0,77	0,81

Los resultados de las Tablas 5-3 y 5-4 muestran un *perfil* nuevo compuesto de siete micro- patrones (es decir, se ha producido una actualización del *perfil* que inicialmente tenía cinco). Debido a esta nueva actualización, es necesario volver a calcular el porcentaje de inclusión, que se mantuvo en el 70 %, y que demuestra el dinamismo del sistema.

5.4 Merodeo en tiendas

El conjunto de datos de la experimentación, entrenamiento y las pruebas, se compone de lo siguiente:

a) 35 secuencias de *Merodeo* (CAVIAR-Project, 2009), que representan a videos grabados por una cámara situada en uno de los pasillos de un centro comercial

b) Observación de vídeos *Merodeo en tiendas* y etiquetado manual de los hechos: se analizaron 100 grabaciones de vídeo de situaciones de *Merodeo en tiendas*, grabadas por los sistemas de videovigilancia. Al igual que con otros ejemplos anteriores, un ayudante de la seguridad observa cada una de las grabaciones de vídeo durante 40 segundos.

Mediante el uso de las 135 secuencias positivas se entrenó a *GSP*. Los resultados mostraron que con el valor *min_support* = 0,4 (donde el 40 % de las secuencias incluyen la micro - patrón) y $\alpha = 2$ (donde la distancia desde la secuencia al *micropatrón* es menor que o igual a 2), son suficientes para obtener los *micropatrones* buscados.

Para el análisis de sensibilidad de la situación de *Merodeo en tiendas*, se utilizó 135 nuevas secuencias positivas y 100 nuevas secuencias negativas. Este procedimiento nos ha ayudado a obtener *micropatrones* que conforman el *perfil* deseado. Por último, hemos utilizado las mismas secuencias (235 secuencias) para determinar el porcentaje óptimo de inclusión de los *micropatrones* en nuevas secuencias de entrada, cuyos resultados se muestran en la siguiente tabla:

Tabla 5-5. Micropatrones, precisión, recall y F1Score

Micropatrones	***Precision***	***Recall***	***F1Score***
<walks, walks, walks, stops, stops, stops, walks, walks, walks>	0,85	0,94	0,89
<stops,stops,stops,stops,stops,walks,turns-right,walks,walks,turns-right,walks>	0,73	0,96	0,82
<stops,stops,stops,walks,walks,walks,walks,stops,stops,turns-left,walks>	0,71	0,96	0,81
<stops,stops,stops,stops,walks,walks,walks,turns-right,turns-right, browses, browses>	0,81	0,96	0,87

Tabla 5-6. Porcentaje de inclusión en el perfil

% de inclusión en el *perfil*	***Precision***	***Recall***	***F1Score***
35	**0,64**	0,98	0,77
40	0,64	0,976	0,77
45	0,67	0,97	0,79
50	0,71	0,95	0,81
55	0,74	0,943	0,82
60	0,74	0,94	0,82
65	0,74	0,94	0,82
70	0,76	0,94	0,84
75	**0,89**	**0,94**	**0,91**
80	0,9	0,919	0,90

85	0,9	0,87	0,88
90	**0,93**	0,865	0,89
95	**0,93**	0,84	0,88
100	**0,93**	0,77	0,84

Interpretación de los resultados.

Como puede en las Tablas 5-5 y 5-6, los resultados con los valores más altos se observaron cuando las secuencias contienen 75 % de los *micropatrones* del *perfil* (ver fila resaltada en negrita). Estos resultados se consideran válidos (Aguas, 2010) (Gonzalez, 2009) para este estudio, ya que proporcionan un alto valor de *recall* y porqué el nivel de *precisión* no disminuye severamente, sino que aumenta gradualmente basándose en el hecho de que las secuencias de entrada contienen el número óptimo de *micropatrones*. El porcentaje óptimo (75 %) de la inclusión de los *micropatrones* en las secuencias de entrada se determina por el valor más alto de la calificación *F1Score* (0,91).

Cabe destacar la idea, que donde hay un nivel de *precisión* de 0,64; lo que indica que se generan un mayor número de falsas alertas en comparación con los valores de las tres últimas filas de la tabla (0,93). Esta teoría se puede sostener cuando examinamos lo que ocurre con un nivel de *precisión* de 1,00. Cuando exista un nivel de *precisión* óptima de 1,00; habrá un mínimo de falsas alarmas. El valor de *precisión* y *recall* son alternativos, es decir; mientras el uno aumenta el otro disminuye de valor. Los valores de *precisión* y *recall* son mutuamente dependientes y la puntuación de *F1Score* muestra la relación

entre estos valores. Para explicar este punto más lejos de un punto de vista teórico, el sistema identifica el número máximo de secuencias sospechosas (*recall*) en relación con el número mínimo de falsas alarmas (*precisión*). Por lo tanto, el equilibrio entre la *precisión* y la *recall* se puede encontrar en la puntuación de *F1Score* es de 0,91.

La experimentación ha encontrado que si no existe un equilibrio entre estos valores (es decir, cuando el número de secuencias de entrada es demasiado pequeño para generar representativos *micropatrones* de comportamiento sospechoso), hay que aumentar el número de secuencias de entrada, mediante el etiquetado más secuencias de vídeo. Por esta razón, es esencial para alcanzar el equilibrio entre la *precisión* y *recall*. Para descargar nuestro estudio de caso, consulte el siguiente enlace: https://docs.google.com/open?id=0B-C-0w7yDtkIYUM1WTRyYU5UN3c.

Para comprobar si nuestro sistema funciona, se realizó una prueba final para comprobar la capacidad de aprendizaje del sistema con 100 nuevas secuencias positivas y 100 secuencias negativas. En este caso, el análisis de sensibilidad no generó nuevos *micropatrones* ya que no había ninguna secuencia descartada que alcanzó el *min_support*. Por lo tanto, no existían suficientes cambios en el análisis de sensibilidad para producir nuevos *micropatrones*. Como se puede ver a continuación, el valor de *F1Score* sigue siendo alta, y al igual que con en la tabla de inclusión de los *micropatrones* descrita anteriormente, los resultados confirmaron que el porcentaje *óptimo* de inclusión del *micropatrón* en la secuencia fue del 75 %.

Tabla 5-7. Micropatrones, precisión, recall y F1Score (nuevas pruebas)

Micropatrones	***Precision***	***Recall***	***F1Score***
<walks, walks, walks, stops, stops, stops, walks, walks, walks,>	0,64	0,94	0,76
<stops,stops,stops,stops,stops,walks,turns-right,walks,walks,turns-rightwalks>	0,67	0,94	0,78
<stops,stops,stops,walks,walks,walks,walks,stops,stops,turns-left,walks>	0,73	0,91	0,81
<stops,stops,stops,stops,walks,walks,walks,turns-right,turns-right> browses, browses>	0,77	0,9	0,82

Tabla 5-8. % de inclusión en el perfil (nuevas pruebas)

% de inclusión en el *perfil*	***Precision***	***Recall***	***F1Score***
35	0,56	0,66	0,60
40	0,62	0,64	0,62
45	0,66	0,67	0,66
50	0,67	0,67	0,67
55	0,67	0,67	0,67
60	0,67	0,72	0,69
65	0,67	0,72	0,69
70	0,7	0,74	0,71
75	**0,7**	**0,77**	**0,73**
80	0,7	0,7	0,7
85	0,7	0,68	0,68

90	0,66	0,61	0,63
95	0,74	0,53	0,61
100	0,87	0,53	0,65

Los resultados de las Tablas 5-7 y 5-8 mostraron que en la fase de prueba del sistema, existieron falsas alarmas. Estas falsas alertas pueden ser reducidos mediante la obtención del porcentaje óptimo de la inclusión de *micropatrones* en las secuencias de entrada y repitiendo el análisis de sensibilidad. También nos pareció que era fundamental en todo momento de preservar la participación del operador humano, ya que su papel es fundamental para actuar sobre los avisos de mensajes generados automáticamente y para actualizar el análisis de sensibilidad. Por esta razón, nuestro sistema no ignora completamente la necesidad de operadores humanos ya que (los operadores humanos) toman la decisión final presionando el botón de alerta de mensaje de seguridad para confirmar que la secuencia de vídeo de entrada se corresponde con una situación sospechosa. El sistema propuesto está diseñado de tal manera que, en teoría, una vez que esté completamente instalado y operativo, que funcionaría en su propio por la generación automática de alertas de mensajes para el operador humano, que, a su vez, tomaría las medidas de seguridad necesarias. Todo este proceso se basa en la identificación y el etiquetado de lo que llamamos las actividades primarias o básicas, es decir; eventos que son reconocibles por algoritmos de visión artificial o técnicas de monitoreo sensoriales inteligentes (segmentación, segmentación, seguimiento y clasificación). Mediante el uso de estas secuencias de actividades etiquetadas podemos

obtener *micropatrones* secuenciales con el algoritmo de *GSP*. Después de hacer un análisis de sensibilidad con secuencias que muestran situaciones normales (negativo) y situaciones sospechosas (positivo), se selecciona el *micropatrón* más característico, conformando así el *perfil* comportamiento sospechoso.

5.5 Conclusiones del modelado del aprendizaje en el marco propuesto

Nuestro trabajo consiste en que es posible caracterizar e identificar automáticamente cualquier comportamiento sospechoso con un *perfil* que se basa en la repetición de secuencias de actividades, que toman el nombre de *micropatrones* sospechosos. El sistema es entrenado con secuencias de una base de datos de vigilancia de vídeo real de comportamiento sospechoso (casos de vigilancia), con el objetivo de definir un *perfil*. Este *perfil* sirve para identificar situaciones sospechosas (sospechoso, porque con el tiempo producirán un delito), o los las que están sin sospechar *a priori* y que se puede confirmar la sospecha (o rechazar) a través del análisis de las nuevas secuencias *in situ*.

En nuestra experimentación se determinó el porcentaje de inclusión apropiada de un *perfil* en una secuencia. Durante esta etapa, se realizó un análisis de sensibilidad, es decir; considerando como aceptable una recuperación superior a 0,9 y una precisión superior a 0,6; según consta en nuestros objetivos (idea de equilibrio). Una vez que el sistema se activa, buscamos el *perfil* del sospechoso dentro de las secuencias de entrada, con el objetivo de generar un aviso para el operador humano. Durante el tiempo de ejecución, cuando una secuencia de entrada contiene el porcentaje óptimo del *perfil*, un mensaje de alerta se muestra al operador humano. El operador humano entonces confirmar los verdaderos positivos y los falsos negativos. Esta interacción humana con

el sistema por lo tanto, ayuda a actualizar el análisis de sensibilidad del *perfil*. Por otra parte, en tiempo real, los resultados que se descartan inicialmente pueden igualmente ser recuperados si la frecuencia de ocurrencias alcanza el nivel mínimo requerido. Los resultados obtenidos después de actualizar el análisis de sensibilidad de las *micropatrones* seleccionados muestran que el *perfil* obtenido se utiliza para distinguir situaciones normales y potenciales de hurto. Por otra parte, con las secuencias que se utilizaron en la etapa de experimentación, el sistema propuesto es capaz de distinguir entre situaciones normales y de *Merodeo*, un problema que no fue resuelto por (Williem, Vamsi, Boles, & Wageeh, 2008). En nuestra investigación, sin embargo, utilizamos las mismas secuencias de entrada y, a diferencia del caso no resuelto, hemos sido capaces de hacer una clara distinción entre estos dos tipos de situaciones sospechosas. De hecho, este hallazgo constituye otra importante contribución a nuestro estudio. Nuestros resultados sugieren fuertemente, que la aplicación de un *perfil* de *micropatrones* en situaciones de videovigilancia ayuda en la predicción y prevención de situaciones sospechosas, sirviendo de este modo como una herramienta fundamental para el operador humano en sistemas de *SV*. Es cierto que el sistema puede funcionar con el análisis de sensibilidad de *micropatrones* como se demostró en la experimentación de este trabajo, de hecho, los resultados son totalmente prometedores. Sin embargo, la crítica puede centrarse en que no se tienen valores predefinidos para la longitud del *micropatrón* en donde hay que tener cuidado, ya que si se selecciona patrones muy cortos, se puede caer en *micropatrones* que también aparecen en las situaciones normales en un supermercado, puesto que no contienen la cantidad suficiente de repeticiones. Por cuya causa hay que hacer distintas pruebas de longitud

del *micropatrón* y lo mismo para el *umbral de inclusión* (α) con el fin de tener la fiabilidad necesaria en la puesta en marcha del sistema.

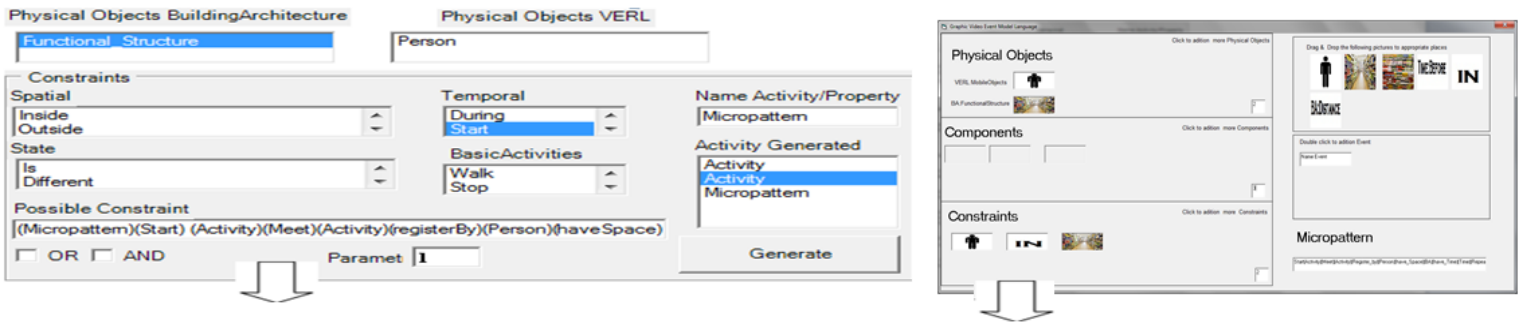
Creemos que este enfoque propuesto es innovador y tiene el potencial de abrir investigación en dominios adyacentes de investigación, como en la asistencia sanitaria y la psicología. Sin embargo, esta propuesta no es un intento de reemplazar al operador humano que controla el comportamiento de las personas, sino que debe ser visto como una alternativa práctica para la prevención de conductas delictivas que apoya a los sistemas de *SV* (*GSP* y análisis de sensibilidad).

Los *micropatrones* tienen son capaces de aportar semántica en el marco. De hecho, las repeticiones de los estados, dan cierta idea de secuencia y por lo tanto de relaciones temporales entre ellos. A continuamos vamos detallar ese aporte semántico, que contiene la correlación de un *micropatrón* con una situación determinada, y las relaciones espaciales y temporales de los eventos.

5.6 Aporte semántico de los micropatrones

Los *micropatrones* que se obtienen con *GSP* están correlacionados con situaciones de sospecha. No podemos afirmar una relación directa, ya que si en verdad son una forma de caracterizar, su cumplimiento no exige que una situación deba darse, mas esta siempre debe estar comprobada por el agente humano. Proponemos que los patrones pueden ser conceptualizados, dado que están conformados por actividades secuenciales. Aprovechamos el lenguaje *VERL* para conceptualizar los patrones y utilizamos el frontal para su exportación a *RDF*.

En la Figura 5-5, se observa la conceptualización de un patrón a partir de *CHAO*. Por ejemplo, un *micropatrón* correlacionado con la situación Persona *Sospechosa* de Hurto es: La persona camina (repite *Walk, Walk,Walk,Walk,Walk,Walk*) inmediatamente se para (repite *Stop,Stop,Stop*).



Conceptualize Micropattern

```
<owl:ObjectProperty rdf:ID="correlationWith">
      <rdfs:domain rdf:resource="#Micro_Patterns"/>
      <rdfs:range rdf:resource="#Situation"/>
</owl:ObjectProperty>
owl:Class rdf:ID="Micro_Patterns">
      <owl:equivalentClass>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#start_Activity"/>
          <owl:cardinality
rdf:datatype="&xsd;int">1</owl:cardinality>
        </owl:Restriction>
      </owl:equivalentClass>
   </owl:Class>
<owl:DatatypeProperty rdf:ID="have_Repetitions">
      <rdfs:domain rdf:resource="#Activity"/>
      <rdfs:range rdf:resource="&xsd;int"/>
  </owl:DatatypeProperty>
  <owl:ObjectProperty rdf:ID="haveSpace">
      <rdfs:domain rdf:resource="#Activity"/>
      <rdfs:range rdf:resource="#BuildingAchitecture"/>
  </owl:ObjectProperty>
  <owl:Class rdf:ID="Instant"/>
  <owl:ObjectProperty rdf:ID="Meet">
      <rdfs:domain rdf:resource="#Activity"/>
      <rdfs:range rdf:resource="#Activity"/>
  </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID=" register_By ">
      <rdfs:domain rdf:resource="#Activity"/>
      <rdfs:range rdf:resource=" Person  "/>
   </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="have_Time">
        <rdfs:domain rdf:resource="#Activity"/>
        <rdfs:range rdf:resource="#Time"/>
    </owl:ObjectProperty>
```

MicroPattern Example

```
<Micro_Patterns rdf:ID="Micro_Patterns_1">
      <correlationWith rdf:resource="#Thieft"/>
      <start_Activity rdf:resource="#Walk"/>
 </Micro_Patterns>
<Activity rdf:ID="Walk">
      <have_Repetitions
rdf:datatype="&xsd;int">6</have_Repetitions>
      <haveSpace rdf:resource="#CASAS"/>
    <have_Time rdf:resource="#Time_51"/>
      <Meet rdf:resource="#Stop"/>
      <registerBy rdf:resource="#Person_41"/>
   </Activity>
   <Activity rdf:ID="Stop">
      <have_Repetitions
rdf:datatype="&xsd;int">3</have_Repetitions>
      <haveSpace rdf:resource="#CASAS"/>
      <have_Time rdf:resource="#Time_52"/>
      <registerBy rdf:resource="#Person_41"/>
   </Activity>
```

Figura 5-5. Aporte de los *micropatrones* a la descripción de las actividades

```
Model:
MicroPattern
        Components:
                (a1: VERL:PrimitiveState)
                (a2: VERL:PrimitiveState)
        constraints:
        (Time:meet(a1,a2))
        (a1 CHAO:repeat n)
        (a2 CHAO:repeat n)
Instance:
e1: Micro-Pattern (a1:Walk,a2:Stop)
```

Código 5-1. Micropatrón en *VERL*

El Código 5-1 permite verificar que los *micropatrones* se pueden conceptualizar en *VERL*. Las actividades se definen como estados primitivos en *VERL*. Una actividad es registrada por una persona (*registerBy*), en un lugar (*haveSpace*) y tiempo (haveTime) determinado. Las propiedades se describen a continuación.

Time:meet sirve para relacionar una actividad con actividad de forma temporal, con el fin de formar secuencias de actividades.

CHAO:repeat sirve para relacionar a las actividades con su número de repeticiones en los *micropatrones*. A continuación, describimos los árboles gráficos situación, que son aquellos que nos permiten detallar aún más la relación semántica entre los patrones y una situación de interés. Esto es lo que se persigue con el modelado del conocimiento que no solo quede en el aporte de los resultados, sino que puedan formar parte de la semántica para la interpretación de situaciones de alto nivel, lo cual está descrito en el marco propuesto de esta tesis doctoral.

6 Conclusiones y Trabajos Futuros

En los sistemas de *SV* tradicionales se le pide a un vigilante que esté atento (*monitorizar*) a lo que sucede en la zona a la que se quiere brindar seguridad. El objetivo general de esta *monitorización* es identificar situaciones sospechosas y generar alertas (Aguas, 2010). Esto pone de manifiesto una de las deficiencias de los sistemas de *SV*, la dependencia absoluta del operador humano, en donde factores como la fatiga producida tras varias horas de trabajo, reducen considerablemente la probabilidad de detectar todas las situaciones de interés. El que un sistema de SV proporcione un aviso de la ocurrencia de una actividad o situación, es tan importante como la selección de los elementos tecnológicos que permitieron captarlo.

Para dar solución a este problema, algunos sistemas que procesan señales multisensoriales, identifican un evento y pasan directamente a relacionarlo con la situación de interés (Albanece, Chellapa, Moscato, & Picariello, 2008) (Allen & Ferguson, 1994) (Botia, Villa, & Palma, 2009). A este paso se lo conoce como brecha semántica, ya que allí no se aporta con el detalle semántico de lo que sucede en un

escenario. Algunas investigaciones proponen soluciones para eliminar la brecha semántica, la mayoría de ellas se basan en utilizar estructuras que parten desde el bajo nivel semántico y obtienen un alto nivel que permite descripciones de calidad que ayudan en la búsqueda y recuperación de actividades y situaciones en los sistemas de SV (Ayer & Chellapa, 2000) (Botia, Villa, & Palma, 2012) (Bredmod & Maillot, 2009).

El tener arquitecturas que agrupen sistemas multisensoriales, con el fin de ayudar al operador humano a tomar decisiones según se identifiquen una situación de interés, es con claridad una temática que debe desarrollarse desde la combinación tecnológica. Con esta temática el grupo SIMDA ha llevado a cabo proyectos que proponen la integración de diferentes tecnologías y la conceptualización semántica de las situaciones (SIMDA, 2008): AVANZA, CICYT 2004, CYCYT 2007, INT3. El aporte de INT3 es fundamental para este trabajo, ya que obtuvieron a partir de él a Horus un marco multisensorial para monitorización y detección de actividades. En este trabajo se parte del nivel bajo semántico que aporta el marco *Horus* y a partir de allí se diseña un marco que compone niveles semántico jerárquicos que permiten inferir la situación de interés a este marco lo llamamos *TOPHORUS (TH)*, el cual está compuesto por estructuras semánticas, módulos que permiten la descripción por parte del experto de la situación de interés, herramientas que permiten modelar la descripción del experto, módulos de exportación hacia el lenguaje general semántico *OWL* y un módulo de ejecución en el cual se permite realizar las inferencias de las situaciones de alto nivel semántico. Las conclusiones y trabajos futuros que se obtuvieron luego de estudiar, diseñar y poner a prueba a *TH* se describen a continuación:

- Con *TH* es posible modelar situaciones de alto nivel semántico, por medio de la composición de niveles jerárquicos semánticos. En el proceso de modelado, tanto el experto como el ingeniero del conocimiento intervienen, el uno para describir de forma clara la situación a modelarse y el otro para codificarla en un lenguaje semántico. Nuestro trabajo futuro se centrará en diseñar un método que ayuda a bajar el nivel de abstracción desde lo que describe el experto hacia lo que debe modelarse en el marco (interacción experto- ingeniero del conocimiento), ya que esta tarea a veces suele se tediosa; pues es necesario interpretar el lenguaje natural del experto y convertirlo en un lenguaje que pueda ser interpretado por las herramientas del modelado semántico.
- Comprobamos que *TH* al ser un marco jerárquico-semántico, permite componer niveles semánticos, con el fin de inferir la situación de interés. Para ello, se parte del nivel bajo semántico provisto por *Horus*, se combinan los eventos extraídos de ese nivel para obtener las actividades de nivel medio y por último se combinan ambos niveles para obtener le alto nivelo semántico. Aclaramos aquí que el nivel bajo semántico provisto por *Horus*, trabaja por lo general con señales de vídeo, pero puede trabajar también cualquier señal multisensorial (Castillo, Fernández-Caballero, Serrano-Cuerda, & Sokolova, 2012), y es por eso que en este trabajo hemos realizado experimentaciones que contemplan diferentes tipos de señales.
- Con *TH* creamos un "*bridging the semantic gap"*. Y es que nuestra propuesta trabaja sobre niveles semánticos que tienen el detalle necesario para explicar lo que sucede en un escenario. Nuestro trabajo está listo para acoplarse a

herramientas de visualización de eventos, como las que se trabajaron en nuestro grupo y se explica en la propuesta de (Rivas, Martinez-Tomás, & Fernández-Caballero, 2010). En trabajos futuros realizaremos pruebas en las que se demuestre el acople del marco a este tipo de herramientas, a pesar de que en esta tesis hemos demostrado que los resultados son visibles desde el software Protégé, en dónde; se muestran las instancias de las situaciones de interés.

- Con *TH* se pudo modelar e inferir situaciones de alto nivel semántico. Esto quedó demostrado cuando se pudo modelar las situaciones de *Merodear* y *Merodeo_nocturno* y *Hurto_con_Mochila*. Y es aquí en donde se comprobó la veracidad de lo dicho, las dos últimas situaciones reutilizaron a *Merodear*, pues era su punto de partida, y luego con pocas reglas (facilidad para la inferencia) se pudieron crear prototipos con los cuales se pudo inferir la situación de interés. Aquí es donde queda comprobada nuestra hipótesis, las estructuras semánticas unidas a herramientas a fines facilitan el prototipado y el reuso de los modelos y conceptos semánticos. La hipótesis a quedado demostrada, pero en trabajos futuros realizaremos más pruebas sobre otros dominios, que permitan mejorar las características operativas de *TH.*
- *TH* puede modelar situaciones que el experto puede definir con claridad. Pero, cuando eso no sucede, *TH* trabaja con *GSP* con el fin de esclarecer la descripción de la situación por parte del experto. El resultado de la aplicación de *GSP* fue un *perfil* correlacionado con la situación de interés. Los *micropatrones* que conforman el *perfil*, fueron analizados con el fin de obtener su aporte semántico el mismo que fue conceptualizado en *CHAO*. En trabajos futuros,

realizaremos pruebas con otros algoritmos de aprendizaje que puedan tener en cuenta las características de equilibrio nombradas en esta tesis. Sin embargo, queremos resaltar que nuestro propósito fue el modelado de la situación de interés y que con *TH* eso ha sido posible, inclusive en los casos en los cuales la situación no puede ser descrita con claridad.

- *VERL* facilitó el modelado de actividades de alto nivel semántico, puesto que presta el cual presta las condiciones necesarias para utilizar conceptualizaciones de objetos y componentes del escenario, y además permite diseñar restricciones espacio temporales, que son necesarias para obtener un alto nivel semántico.
- El uso de herramientas que permiten el modelado de situaciones de alto nivel semántico en *VERL*, facilitan la inferencia de alto nivel semántico. Estas herramientas aportan al trabajo al ingeniero del conocimiento, puesto que los modelos diseñados en ellas, pueden ser exportados directamente hacia el lenguaje semántico *OWL* y las restricciones del modelo hacia el lenguaje de reglas *SWRL*. En un trabajo futuro, crearemos interfaces que permitan exportar los modelos hacia otros motores de inferencia como *CLIPS*, con el fin experimentar sobre los tiempos de respuesta, cuando se instancian las ontologías y se infieren las situaciones de interés.
- *CHAO* permitió inyectar mayor detalle semántico a los modelos de *VERL*. En este trabajo diseñamos a *CHAO* una ontología que colecciona las conceptualizaciones de *SSN*, *BuildingArchitecture* y *Time*. *CHAO* tiene además sus propias conceptualizaciones que ayudan a modelar las situaciones de alto nivel semántico. En trabajos futuros seguiremos completando *CHAO* con

mayores conceptualizaciones y actualizaciones de las ontologías que la componen, esto con el fin de tratar de tener los elementos necesarios para poder modelar la mayoría de situaciones posibles.

- Los resultados obtenidos, de las inferencias en los casos de estudio, son buenos en relación al análisis de sensibilidad. En esta tesis, siempre damos validez la disminución de falsos negativos, puesto que en ciertos escenarios de *SV*, se puede dejar pasar un número aceptable de falsos positivos, pero es importante que existan pocos falsos negativos como ocurre en la *monitorización* de pacientes con Alzheimer (Wongpatikaseree, 2012) (Ye, Coyle, Dobson, & Nixon, 2007) (Stevenson, 2007).

En definitiva, el marco propuesto en esta tesis ha sido probado y se ha detallado su funcionamiento, desde las entradas multisensoriales hasta la inferencia de la situación de interés. Con ello, queda comprobada la hipótesis de que las estructuras semánticas permiten el fácil prototipado de sistemas de *SV* y mediante la reutilización de conceptualizaciones y uso de tecnologías semánticas. Es necesario aclarar que nuestro aporte no está terminado, que se encuentra en fase de estudio y experimentación, por lo que hemos propuesto los trabajos futuros que vamos a realizar en durante los próximos años de investigación.

Bibliografía

Aggarwal, J., & Ryoo, M. (2011). Human activity analysis:A review. *Journal ACM Computing Surveys*, 48-56.

Aguas, S. (2010). *Como identificar ladrones de tiendas*. Retrieved 03 20, 2013, from http://retail.about.com/od/lossprevention/qt/spot_shoplifter.htm

Akdemir, U., Turaga, P., & Chellapa, R. (2008). An Ontology based Approach for Activity Recognition from Video. *Proceeding of the 16th ACM international conference on Multimedia*, 709-712.

Albanece, M., Chellapa, R., Moscato, V., & Picariello, A. (2008). Constrained Probabilistic Petri Net Framework for Human Activity Detecion in Video. *Multimedia, IEEE Transactions on, 10*, 982-986.

Albusac, J. (2008). Vigilancia Inteligente:Modelado de Entornos Reales e Interpretación de Conductas para la Seguridad. Castilla-La Mancha, España: Tesis de Master.

Allen, J., & Ferguson, G. (1994). Actions and Events in Interval Temporal Logic. *Journal of Logic and Computation*.

Aranda, G. (2011). *Tratamiento de la semántica emergente mediante sistemas de agentes basados en conocimiento.* Sevilla: Universidad de Sevilla.

Arens, M., & Nagel, H. (2003). Behavioural knowledge representation for the understanding and creation of video sequences. *Proceedings of the 26th German Conference on Artificial Intelligence* (pp. 149-163). Heidelberg: LNAI Springer-Verlag.

Ayer, D., & Chellapa, R. (2000). *Scenario recognition from video using a hierarchy of dynamic belief networks.* Barcelona: IEEE.

Baader, F., Calvanese, D., McGuiness, D., Nardi, D., & Patel-Schneider, P. (2003). *The Description Logic Handbook: Theory, Implementation, Applications.* Cambridge: Cabridge University Press.

Bai, L., Dick, R., & Dinda, P. (2009). Archetype-based design: Sensor Network programming for application experts, not just programming experts. *Information Processing in Sensor Network*, 85-96.

Balanzinska, M., Deshpande, A., Franklin, M., Gibbons, P., Gray, J., Hansen, M., et al. (2007). Data management in the wordwide sensor web. *IEEE Pervasive Computing*, 30-40.

Barake, B., & Saddik, A. (2008). Towards and intelligent tele-surveillance system for public transport areas: Fuzzy Logic based camera control. *IEEE Xplore*, 87-90.

Bechhofer, S., Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., et al. (2004). *OWL Web Ontology Language Reference.* W3C .

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web: A New form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 34-43.

Berning Prieto, A. D. (2008, 05). *Artículos Doctrinales:Derecho Administrativo.* Retrieved 10 29, 2013, from http://noticias.juridicas.com/articulos/15-Derecho%20Administrativo/200805-25836974123658.html

Berrier, M. (2008). *Estrategias contra delitos en transportes públicos. El caso británico: una solución aplicable a la realidad Argentina*. Retrieved 2011, from http://www.reeditor-project.com/columna/666/14/nacional/seguridad/publica/estrategias/contra/delitos/transportes/publicos/caso/britanico/solucion/aplicable/la/realidad/argentina

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., et al. (2010). A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput.*, 161-180.

Biaget, J., Orozco, X., Roca, X., & González, J. (2007). *Situation Graph Trees for Human Behaviour Modeling*. Retrieved 11 06, 2013, from http://www.academia.edu/3155525/Situation_Graph_Trees_for_Human_Behavior_Modeling

Bickmore, T., Schulman, D., & Sidner, D. (2011). A reusable framework for health counseling dialogue systems based on a behavioural medicine ontology. *Journal of Biomed Inform*, 183-187.

Botia, J., Villa, A., & Palma, J. (2009). Ambient Assisted Living system for in-home monitoring of healthy independent elders. *Expert system with applications*, 8136-8148.

Botia, J., Villa, A., & Palma, J. (2012). Ambient Assisted Living system for in-home monitoring of healthy. *Expert Systems with Applications*, 8136-8148.

Botts, M., Percival, G., Reed, C., & Davidson, J. (2008). OGC Sensor Web Enablement: Overview and High Level Architecture. *IEEE Workshop on Policies for Distributed Systems and Networks* (pp. 241-242). Washington: IEEE Computer Society.

Bredmod, F., & Maillot, N. (2009). *Ontologies for Video Events*. Retrieved 12 07, 2010, from http://www-sop.inria.fr/orion/personnel/Van

Bredmond, F., Corvee, E., Patiño, J., & Thonnat, M. (2008). *CARETAKER Project*. Retrieved 07 08, 2011, from http://www-sop.inria.fr/members/Francois.Bremond/topicsText/caretakerProject.html

Bremond, F., Maillot, N., Thonnat, M., & Vu, V.-T. (2004). *Ontologies For Video Events.* INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE.

Brickely, D., & Miller, L. (2003). *FOAF Vocabylary Specification.* 1.47.

BrotherSoft. (2011, 04 17). *Matlab: Simulation of fixed-mass rigid body 6DOF*. Retrieved 03 20, 2013, from http://www.brothersoft.com/matlab-simulation-of-fixed-mass-rigid-body-6dpf-379573.html

Cáceres, A. (2010, 05 20). *La métrica de Levenshtein.* Retrieved 03 20, 2013, from http://www.publicaciones.ujat.mx/publicaciones/revista_dacb/Acervo/v7n2OL/v7n2a4-ol/v7n2a4-ol.html, the 2012/06/19

Calero, D., & Wis, M. (2010). *Desarrollo de un sistema de video sincronizado abierto.* Barcelona: Instituto de Geomática.

Capra, L., Emmerich, W., & Mascolo, C. (2001). Reflective middleware solutions for context aware applications. *Lecture Notes in Computer Science*, 2192.

Carmona, E., Rincon, M., Bachiller, M., Martínez-Cantos, J., Martínez-Tomás, R., & Mira, J. (2009). On the effect of feedback in multilevel representation spaces for visual surveillance task. *Neurocomputing*, 916-927.

Castillo, J., Fernández, A., & López, M. (2011). A Framework for Multisensory Intelligent Monitoring and Interpretation of Behaviors through Information Fusion. *2011 Seventh International Conference on Intelligent Environments*, (pp. 1-4).

Castillo, J., Fernández-Caballero, A., Serrano-Cuerda, J., & Sokolova, M. V. (2012). Intelligent monitoring and activity interpretation framework - INT3 Horus general description. *16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems. Lecture Notes in Artificial Intelligence.* San Sebastian.

CAVIAR-Project. (2009, 05 20). *CAVIAR Project.* Retrieved 03 20, 2013, from http://home-pages.inf.ed.ac.uk/rbf/CAVIAR

Cervantes, J. (2005). *Representacio y aprendizaje de conocimiento con redes de Petri Difusas.* Mexico: Tesis, CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS.

Charkraborty, B., Bagdanov, A., Gonzalez, J., & Roca, X. (2011). Human action recognition using an ensemble of body-part detectores. *Expert Systems*(doi: 10.1111/j.1468-0394.2011.00610.x.).

Chen, H., Finin, T., & Joshi, A. (2004). An ontology for context-aware pervasive computing environments. *Ontologies for Distributed Systems, Knowledfe engineering Review*, 197-207.

Cheng, F., & Chen, F. (2007). An Efficient Method for Human Behaviour Identification. *Conference on Machine Vision Applications*.

Chikhaoui, B., Wang, S., & Pigot, H. (2010). A new Algorithm Based on Sequential Pattern for person identification in ubiquitous environments. *Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data*, 20-28.

Chun, L. (2008). Definition, detection and evaluation of meeting events in airport surveillance videos. *TRECVID Workshop*.

Clark & Parsia. (n.d.). *Pellet: OWL 2 Reasoner for Java.* Retrieved Abril 5, 2013, from clark & parsia: http://clarkparsia.com/pellet

Coen, M. (1998). Design principles for intelligent environments. *AAAI/IAAI*, (pp. 545-554).

Colitti, W., Steenhaut, K., Descouvemont, N., & Dunkels, A. (2008). Satellite based wireless sensor networks: global scale sensing with nano- and pico-satellites. *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems* (pp. 445-446). New York: ACM.

Collins, R., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., et al. (2000). A System for Video Surveillance and Monitoring. *CiteSeeer*, 50-62.

Council, N. R. (2000). *Expanding the vision of sensor materials.* Washington: National Academy Press.

Descamps-Vila, L., Casas, J., Conesa, J., & Pérez-Navarro, A. (2008). Cómo introducir semántica en las aplicaciones SIG móviles: expectativas, teoría y realidad. *Jornadas de SIG libre.* Catalunya: SIGTE.

Dinero.com. (2009, 05 20). *Dinero.* Retrieved 03 20, 2013, from http://www.youtube.com/watch?v=ayIYsb9ADkQ&feature=relmfu

Douglas, B., Guha, B., & Guha, R. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project.* New York: Addison-Wesley.

Dumbil, E. (2002). *Finding friends with xml and rdf.* IBM developer Works: XML Watch.

Fern, X., Komireddy, C., & Burnnet, M. (2007). Mining Interpretable Human Strategies: A Casae Study. *7 IEEE International Conference on Data Mining*, 475-480.

Fernández, C., & González, J. (2007). Ontology for Semantic Integration in a Cognitive Surveillance System. (Springer, Ed.) *Semantic Multimedia, 4816*, 260-263.

Gangemi, A., Guarino, N., Masolo, C., Oltramani, A., & Shneider, L. (2002). Sweetening Ontologies with DOLCE. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, 166-181.

Ghanem, N. (2007). *Petri Net Models for Event Recognition in Surveillance Videos.* Maryland: University of Maryland.

Gibbons, P., Karp, B., Ke, Y., & Seshan, S. (2003). Iris-Net: An architecture for a wordwide sensors web. *IEEE Pervasive Computing*, 22-33.

Gómez-Romero, J., García, J., Patricio, M., Serrano, M., & Molina, J. (2012). Lógica Difusa en Sistemas de Fusión de Información Visual: Aplicaciones, Extensiones y Propuestas. *ESTYLF2012* (pp. 19-25). Valladolid: ESTYLF.

Gonzalez, S. (2009, 05 17). *La inteligencia artificial aplicada a la videovigilancia revoluciona los sistemas de cctv.* Retrieved 03 20, 2013, from http://www.borrmart.es/articulo_seguritecnia.php?id=1226&numero=324

Grütter, R., Scharrenbach, T., & Bauer-Messmer, B. (2008). Improving an RCC-derived geospatial approximation by OWL axioms. (S. Berlin/Heidelberg, Ed.) *The Semantic Web-ISWC 2008*, 293-306.

Guzmán, A., & Cabello, E. (2013, 10 13). *Video-Sensor distribuido basado en CORBA para el reconocimiento de caras.* Retrieved from http://132.248.182.159/LuCAS/Presentaciones/200103hispalinux/cabello/pdf/articulo.pdf

Haigh, K., Kiff, L., Myers, J., Guralnik, V., Geib, C., Phelps, J., et al. (2004). The independent lifestyle assistant (I.L.S.A): Al lessons learned. *Proceeding IAAI'04 Proceedings of the 16th conference on Innovative applications of artifical intelligence* (pp. 852-857). ACM Digital Library.

Hampapur, A. (2007). Searching surveillance video. *Advanced Video and Signal Based Surveillance*, 75-80.

Haritaoglu, I., Harwood, D., & Davis, L. (2000). Real-time surveillance of people and their activities. *Image and Vision Computing*, 137-153.

Held, A., Buchholz, S., & Schill, A. (2002). Modelling of context information for pervasive computing applications. *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics*, (pp. 34-57).

Hobbs, J. (2002). *A daml ontology of time*. Retrieved from http://www.cs.rochester.edu/ferguson/daml/daml-time-20020830.txt

Hois, J., Bhatt, M., & Kutz, O. (2009). Modular Ontologies for Architectural Desing. *IOS Press*, 66-77.

Hong, T.-P., Lin, K.-Y., & Wang, S.-L. (2006). Mining fuzzy sequential patterns from quantitative transactions. *Soft Computing*, 925-932.

Hongeng, S., & Nevatia, R. (2001). Multi-Agent Event Recognition, in Proc. of the 8th Intl. Conf. on Computer Vision (ICCV '01). *Eighth International Conference on Computer Vision* (pp. 84-93). Vancouver: IEEE Computer Society.

Honovich, Jhon. (2009). *Security Manager's Guide to Video Surveillance. Version 3.0.* Honolulu: IPVideoMarket.info.

Honovich, Jhon. (2010, 11 21). *IPSurveillance*. Retrieved 11 21, 2010, from http://ipvideomarket.info/

Horrocks, I., & Patel, P. (2009). *A Semantic Web Rule Language Combining OWL and RuleML*. Retrieved 12 07, 2010, from http://www.w3.org/Submission/SWRL/, el 2010-12-07

Hu, W., W, T., Wang, L., & Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors",IEEE Transactions on Systems, Man and Cybernetics Part C,. *IEEE Transactions on Systems, Man and Cybernetics Part C, 34*(3), 334-354.

IBM. (2013, 10 10). *IBM Smart Surveillance System*. Retrieved from http://researcher.watson.ibm.com/researcher/view_project.php?id=1394

Iglesias, A. (2010). Modelado Automático del Comportamiento de Agentes Inteligentes. *PHD dissertation Universidad Carlos III Madrid, Spain*.

Intille, S., Larson, J., Beaudin, J., Mungia, E., Kaushik, P., Nawyn, J., et al. (2005). The PlaceLab: a live-in laboratory for pervasive computing research (video). *Proceedings of Pervasive 2005 Video Program.*

Isard, M., & Blake, A. (2008). Condensation- conditional density propagatin for visual tracking. *International Journal of Computer Vision*, 5-28.

Jhonson, M., Shotton, J., & Cipolla, R. (2013). Semantic Texton Forest for Image Categorization and Segmentation. *Decision Forest for Computer Vision and Medical Image Analysis, Advances in Computer Vision and Pattern Recognition*, 211-227.

Kang, H., & Lee, H. (1998). An intelligent automatic surveillance system via fuzzy rule base system and genetic algorithms. *Journal of Advanced Computational Intelligence*, 37-42.

Karikrishna, N. (2011). Temporal Classification of events in cricket videos. *National Conference on Communications*, 1-5.

Kemke, C. (2001). *About the Ontology of Actions.* New Mexico: Computing Research Laboratory.

Konstantinou, N., Solidakis, E., Zoi, S., Zafeiropoulos , A., Stahopoulos, P., & Mitrou, N. (2007). Priamos: A middleware architecture for real-time semantic annotation of context features. *IE*. Ulm.

Kwon, J., & Murphy, K. (2000). *Modeling Freeway Traffic with Coupled HMMs.* Citesser.

Lewis, M., Cameron, D., Xie, S., & Arpinar, B. (2006). ES3N: A semantic approach to data management in sensor networks. *Semantic Sensor Networks.* Athens.

Los Santos Aransay, A. (2013, 10 12). Retrieved from http://www.albertolsa.com/wp-content/uploads/2010/04/rsi-aplicacion-de-las-redes-de-sensores-en-el-entorno-vehicular-alberto-los-santos.pdf

Maditskos, G., Dasiopoulou, S., Efstathiou, V., & Kompatsiaris, I. (2013). SP-ACT: A hybrid framework for complex activity recognition combining OWL and SPARL rules. *Pervasive Computing and Communications Workshops* (pp. 25-30). San Diego, CA: IEEXplore.

Maillot, N., Thonnat, M., & Boucher, A. (2004). Towards Ontology Based Cognitive Vision. *Machine Vision and Applications*, 33-40.

Martinez-Tomas, R., & Rivas-Casado, A. (2009). Knowledge and Event-Based System for Video-Surveillance Task. *Springer*, 396-384.

Martínez-Tomas, R., Rincón, M., Bachiller, M., & Mira, J. (2008). On the correspondence between objects and events for the diagnosis of situations in visual surveillance task. *Pattern Recognition Letters, 29*(8), 1117-1135.

Masoud, O., & Papanikolopoulos, N. (2001). A novel method for tracking and counting pedestrians in realtime using a single camera. *IEEE Transactions on Vehicular Technology*, 1267-1278.

Masseglia, F., Poncelet, P., & Tesseire, M. (2009). Efficiente mining of sequential patterns with time constraints: Reducing the combinations . *Expert Systems with Applications, 35*(2), 2677-2690.

Mechsner, F. (2012). Anticipation and Ontology. *International Journal of General Systems*, 23-42.

Moodley, D., & Simonis, I. (2005). A new architecture for the sensor web: The SWAP framework . *Sensor Networks Workshop.* Athens.

Morillo, H., Maciá, F., & Jorquera, D. (2013, 10 12). *Redes Inalámbricas de Sensores Inteligentes. Aplicación a la Monitorización de Variables Fisológicas.* Retrieved from http://www.dtic.ua.es/grupoM/recursos/articulos/JDARE-06-H.pdf

Nathaniel, B., Masoud, O., Papanikopoulos, P., & Issacs, A. (2005). Detection of Loitering Individuals in Public Transportation Areas. *IEEE Transactions on Intelligent Transportation Systems*, 167-172.

NCYT. (2012, 03 16). *Un proyecto internacional desarrolla un sistema de videovigilancia inteligente para grandes ciudades.* Retrieved 10 29, 2013, from http://noticiasdelaciencia.com/not/3755/

Nemanja, S. (2007). *Anomaly Detection and Prediction of Human Actions in a Video Surveillance Environment.* Western Cape: University of CapeTown.

Nevatia, R., & Bredmond, F. (2001). Multi-Agent Event Recognition.

Nevatia, R., & Hobbs, J. (2004). An Ontology for video event representation. *Computer Vision and Patterns Recognition Workshop*, 119-129.

Oliver, N., Rosario, B., & Pentland, P. (2000). A Bayesian Computer Vision System for Modelling Human Interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 831-843.

Oracle. (n.d.). *NetBeans IDE - The Smarter and Faster Way to Code*. Retrieved Abril 5, 2013, from Oracle Corporation and/or its affiliates: https://netbeans.org/features/index.html

Orten, B. (2005). Moving object identification and event recognition in video surveillance systems. *Phd. Dissertation, Middle East Technical University, Turkey*.

Pan, E., & Hobbs, J. (2004). Time in owl-s. *Proceedings of AAI-04 Spring Symposium on Semantic Web Services* (pp. 25-42). California: Stanford University.

Park, K., Lin, Y., Metsis, V., Le, Z., & Makedon, F. (2010). Abnormal human behavioral pattern detection in assisted living environmets. *3rd International Conference on Pervasive Technologies Related to Assistive Environments*.

Park, S; Agrawal, J. (2004). Semantic-level Understanding of Human Actions and Interactions using event Hierarchy. *Conference on Computer Vision and Pattern Recognition*, (pp. 12-22).

Parry, D. (2008). *Evaluation of a Fuzzy Ontology-Based Medical Information.* New Zealand: Auckland University of Technology.

Pastor Sánchez, J. A., & Díaz Ortuño, P. M. (2008, Enero 15). *SPARQL Lenguaje de consulta para RDF*. Retrieved Abril 5, 2013, from Recomendación del W3C de 15 de enero de 2008: http://skos.um.es/TR/rdf-sparql-query/

Perianu, R., & Lombriser, C. (2008). Towards activity recognition in Service-Oriented Body Area Networks. *Sensei Project*, 50-72.

Perse, M., Kristan, M., Pers, J., & Kovacic, S. (2008). Recognition of Multi-Agent Activities with Petri Nets. *17th International Electrotechnical and Computer Science Conference* (p. 24). Protoroz: IEEE.

Perzold, J., & Pietzowski. (2005). Prediction of indoor movements using Bayesian Networks. *Lecture Notes in Computer Science*, 2111-2222.

Powers, S. (2003). *Practical RDF*. O'Really Associates.

Ramirez, J., Royo, F., Olivares, T., & Roncero, J. (2010). Estableciendo las bases para redes de sensores. *5ª Conferencia Ibérica de Sistemas y Tecnologías de Información.* Santiago de Compostela: AISTI.

Rantring. (2008). *Soluciones Tecnológicas e Integrales*. Retrieved 10 28, 2013, from http://www.rantring.com/index.htm

Real Academia de la Lengua Española. (2010, 11 10). *Diccionario de la Real Academia de la Lengua Española*. Retrieved 11 10, 2010, from http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=seguridad

Reed, C., Botts, M., Davidson, J., Percivall, G., & Collins, F. (2007). Ogc sensor web enablement:overview and high level architecture. *Autotestcon*.

Riboni, D., Pareschi, L., Radaelli, L., & Bettini, C. (2011). Is ontology-based activity recognition really effective? *Pervasive Computing and Communications Workshops* (pp. 427-431). Seattle: IEEEXplore.

Rivas, A., Martinez-Tomás, R., & Fernández-Caballero, A. (2010). Multi-agent system for knowledge-based event recognition and composition. *Expert Systems, 28*(5), 488-501.

Rivas-Casado, A., & Martínez-Tomás, R. (2011). Event detection and fusion model for semantic interpretation of monitored scenarios within ASIMS architecture.

International Work-Conference on the Interplay between Natural and Artificial Computation (pp. 521-530). La Palma: Springer.

Rodriguez, S., & Holgado, J. (2008). Servicios Sensibles al Contexto en Sistemas de Computación Ubicua. *II Simposio en Desarrollo de Software* (pp. 235-248). Granada: Dpto. de Lenguajes y Sistemas Informáticos.

SIMDA. (2008). *Projects*. Retrieved 10 28, 2013, from http://simda.uned.es/proyects.html

Simon, F., & Zhuang, Y. (2007). A Security Model for Detection Suspicious Patterns in Physical Environment. *Third International Symposium of Information Assurance and Security*, 221-226.

SINC. (2012, 03 15). *Desarrollan un sistema de videovigilancia inteligenet para grandes ciudades.* Retrieved 10 29, 2013, from http://www.agenciasinc.es/Noticias/Desarrollan-un-sistema-de-videovigilancia-inteligente-para-grandes-ciudades

Sokolava, M., Castillo, J. C., Fernández-Caballero, A., & Serrano-Cuerda, J. (2012). Intelligent Monitoring and Activity Interpretation Framework - INT3Horus Ontological Model. *IOS Press*, 980-988.

Solana-Ciprés, C.J; Albuscac, J; Castro-Sanchez, J.J; Moreno-García, J; Rodriguez-Benítez, L. (2010). Construcción de Conjuntos de Reglas Difusas para la clasificación de objetos. *ESTYLF*, (pp. 20-30). Huelva.

Somboon, H., Bremond, F., & Nevatia, R. (2000). *Representation and optimal recognition of human activities.* Hilton Head Island: IEEE Computer Vision.

Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. *5th International Conference on Extending Database Technology: Advances in Database Technology*, 3-17.

Stevenson, G. (2007). Managing Information Privacy & Security in HealthCare. *Healthcare Information and Management Systems Society*, 1-7.

Sunico, J. (2008). *Reconocimiento de imágenes: usuarios, segmentos de usuarios, gestos, emociones y empatía.* Retrieved 09 21, 2010, from http://jm.sunico.org/4007/06/48/reconocimiento-de-imagenes-usuarios-segmentos-de-usuarios-gestos-emociones-y-empatia/

Tapia, D. (2004). *Arquitectura Multiagente para entornos de inteligencia artificial.* Salamanca: Universidad de Salamanca.

The Apache Software Foundation. (n.d.). *Jena Ontology API.* Retrieved Abril 5, 2013, from The Apache Software Foundation: http://jena.apache.org/documentation/ontology/#general-concepts

Tian, Y., Brown, L., Hampapur, A., & Lu, M. (2008). IBM Smart Surveillance System (S3): Event based video surveillance system with and open and extensible framework. *Machine Vision and Aplications, 18*(5-6), 315-327.

Town, C. (2006). Ontological Inference for Image and video Analysis. *Machine Vision and Applications*, 94-115.

Vain, J., & Ramazan, S. (2009). Spatio-Temporal Querying of Video Content Using SQL for Quantizable Video Databases. *Journal of Multimedia, 4*(4), 215-227.

W3C. (2006, September 27). *Time Ontology in OWL* . Retrieved from http://www.w3.org/TR/owl-time/

W3C. (2011). *Semantic Sensor Network Ontology*. Retrieved from http://www.w3.org/2005/Incubator/ssn/ssnx/ssn

Wang, J. (2010). *Computer and Information Science.* Beijin: Ciza Thomas.

Wang, L., Hu, W., & Tan, T. (2003). Recent developments in human motion analysis. *Patter Recognition*, 585-601.

Williem, A., Vamsi, M., Boles, K., & Wageeh, W. (2008). Detecting Uncommon Trajectories. *Digital Image Computing: Techniques and Applications*, 398-404.

Wongpatikaseree, K. (2012). *Activity Recognition using Context-Aware Infrastructure Ontology in Smart Home Domain.* Iowa: Tan Laboratory.

Ye, J., Coyle, L., Dobson, S., & Nixon, P. (2007). Ontology-based models in pervasive computing systems. *The Knowledge Engineering Review*, 315-347.

Youtube. (2009, 05 20). *Videos de Hurto.* Retrieved 03 20, 2013, from http://www.youtube.com/watch?v=LWZ_vo1pCTQ

Zhao, J., Wang, P., & Chong-Qing, L. (2002). An object tracking algorithm based on occlusion mesh model. *Machine Learning and Cybernetics* (pp. 288-292). Pekin: IEEEXploe.

Zhu, H., Yang, M., Yu, K., & Gong, Y. (2009). Detecting video events based on action recognition in complex scenes using spatio-temporal descriptor. *17th ACM International Conference on Multimedia*, 165-174.

ANEXOS

Anexo I. Estudio de tecnologías de videovigilancia

En el estudio de tecnologías de videovigilancia participaron los siguientes productos:

- AISight de BRSLabs
- Axis M1031-W Cube Cámara con IR integrado
- Axis Q1755 HD/Cámara Mega pixel
- Axis M7001 Mini-Encoder
- Cernium CheckVideo
- CMS300 Central de manejo de *software*, Avermedia
- Digital Window Cámara Panorámica Mega píxel, Scallop
- DV-IP Hybrid NVR, Dedicated Micros
- HD Day/Night IR Cámara Megapixel Line, Avigilon
- Intransa StarterBlock Small Form Factor Storage Array
- Ocularis Investigate – Video Management User Interface
- Panasonic 2 MP CCD Cámara
- Pivot3 Serveless Computing – Run VMS Software inside of Storage Array
- PoE PTZ from JVC
- Sarix Megapixel Camara, Pelco
- TimeSigth Systems – NVR Optimized for Storage Minimization
- XProtect Analytics 2.0 from Milestone

Para elaborar el reporte se tomó en cuenta las siguientes preguntas:

- ¿Cuál es la diferencia del nuevo producto con las ofertas actuales?

- ¿Cuál es el valor potencial de este producto?
- ¿Cuál es la facilidad de uso del producto?

Tabla de comparación de tecnologías y su aporte a los sistemas de *SV*

Producto	**Manufactura**	**Lo que lo hace diferente**	**Valor agregado**	**¿Quién puede usar este producto?**
AISigth	Diseñado para generar alertas cuando ha ocurrido un comportamiento anormal en un entorno, basado en aprendizaje Adaptativo.	La mayoría de empresas trabajan con alertas ya definidas, este producto genera alertas cuando el comportamiento no corresponde al escenario en estudio	Sostiene un análisis detallado del vídeo y por ende la *Monitorización* de la zona de estudio.	Empresas con capacidad tecnológica y capacidad económica para invertir en seguridad.
Axis M1031-W Cube Cámara con IR integrado	Sensor de movimiento infrarrojo, con iluminación, iluminación LED y dos vías de audio con micrófonos y parlantes integrados. Video inteligencia, detección de movimiento y audio.	Bajo precio en las cámaras	Reducción del ancho de banda y consumo de almacenamiento.	Dedicada para pequeños negocios y es soportada por gran cantidad de *software*.
Axis Q1755 HD/Cámara Mega píxel	Contiene HDTV (High Definition TV) que se complementa con SMPTE (Society of Motion Picture and Television Engineers) estándar.	Existe pocos productos en el mercado que sostengan una resolución de H.264 píxeles y una tasa de 1080 (full frame) para la resolución del vídeo.	Usa menor ancho de banda y mayor resolución en los frames transmitidos. Tiene autofocus con el fin de obtener una imagen óptima.	La principal barrera es el precio de la cámara la cual es 60% más cara que otras ofertas que también sostienen potencialidad en el envío de frames y capacidad de resolución.
Axis M7001 Mini_Encoder	Es una solución efectiva para conectar cámaras análogas en un sistema de videovigilancia IP.	Existen pocos productos en el mercado.	Puede ser ubicado junto a la cámara y no necesita de conexiones especiales.	Puede ser usado en sistemas de videovigilancia ya existentes, al sostener un bajo costo y una capacidad de transmisión de datos importante con comprensión de 4CIF H.264.
Cernium	Contiene la verificación inteligente del vídeo que se	Bajo precio en la conectividad de los	Al permitir la conexión con una central de alarmas	Solamente puede ser usado con una

CheckVideo	integra sin problema con una central de alarma.	sistemas ya existentes IP con cámaras análogas	el tratamiento de las alertas es más manejable, que el sistema tradicional de envío de alertas a móvil o PDA	Central de Alarmas.
CMS3000 Central Management Software, AveMedia	Permite el manejo de 320 canales y configuraciones de canales, vídeo y presentación de alarmas	Una solución integral barata	Los elementos adicionales para su funcionamiento tienen bajo precio	Solo se puede usar con productos AverMedia.
Digital Window Panoramic Cámara Mega píxel, Scallop	Cámara de vídeo de 7 Mega pixel con 180 grados de visibilidad, muy pequeña y permite reducir la distorsión mediante cinco módulos micro cámaras.	7 Mega píxel y 180 grados de visibilidad	Bajo precio y alta resolución	Usada en bancos, minoristas, oficinas corporativas.
DV-IP Hybrid Dedicated Micros	Servidor de video que permite un almacenamiento seguro y acceso a todas las partes del sistema.	Almacenamiento híbrido digital y analógico	Bajo precio y permite la conexión de usuarios IP con analógicos.	Lo pueden usar usuarios generales.
HD Day/Nigth IR Megapixel Camera, Avigilon	Primer sistema auto/iris día/noche, disponible en 1 (720 p) a 50 megapíxeles.	Integración del sensor infrarrojo con las cámaras multi-mega-píxel	Muchos usuarios la prefieren por estética y durabilidad, además de los pocos imprevistos que genera un sensor infrarrojo.	Solamente es soportada por el *software* de Avigilon.
Intransa StarterBlock Small, Factor Storage Array	Extiende el tiempo de vida de un circuito cerrado de televisión, con un costo más bajo que un nuevo grabador digital.	Ofrece una solución que empieza con 2TB y que se puede escalar a 10 TB.	Capacidad de redundancia el almacenamiento.	Ha sido testeado con algunos sistemas de video IP y grabadores de vídeo (DVR), sin ningún inconveniente en las pruebas
Ocularis Investigate – Video Management User Interface	Facilita la investigación de actividades. El sistema permite la revisión de múltiples cámaras en un corto período de tiempo.	No existe en el mercado otro producto que facilite la detección de actividades ni las funcionalidades que Ocularis ofrece	Reduce el tiempo de investigación al tener *software* de detección de actividades integrados en el *software* y ayuda a resolver el problema de la detección de actividades.	Destinada a investigadores
Panasonic 2MP CCD Camara	2.6 Mega pixel, escaneo progresivo mediante Mega Super Dynamic image	Tiende a reducir los problemas de	Provee un nuevo nivel en la calidad de la imagen	Es soportada por los sistemas VMS (vídeo

	Technology (MegaSD).	luminosidad.		monitoring services).
PoE PTZ, JVC	Resistente a lluvias, vandalismo, etc. Opera bajo temperaturas desde los -10 a 40 grados, tiene una rotación de 360 grados, zoom óptico de 36x.	Es el único con la directiva IP PTZ (Pan, Tilt, Zoom) en el mercado, es durable y usa poca cantidad de energía para su funcionamiento.	No existe otro producto en el mercado con estas características	Requiere *software* JVC
Pivot3 Serverless Computing – Run VMS *software* inside of Storage Array	Primera y única solución que ofrece elementos cluster para su almacenamiento y ejecución del *software* de seguridad	La combinación de almacenamiento y *software* combinados	Usa 36TB para los cluster de almacenamiento los mismos que pueden soportar hasta 250 cámaras.	Cualquier sistema VMS (video monitoring services)
Sarix Megapixel Camera Line, Pelco	Compresión y optimización del ancho de banda sin bajar la calidad de la imagen	Autofocus	Bajo costo y fácil de instalar	Es un opción muy usada cuando se intenta migrar de video análogo a digital
TimeSight Systems – NVR Optimized for Storage Minimization	La primera red inteligente de grabación de vídeo, que permite automatizar el ciclo de vida del manejo del vídeo (VLM), al aplicar reglas que determinan el almacenamiento o no del vídeo en dependencia de ciertas actividades que suceden en el entorno.	Se encuentra centralizado en la disminución del costo de almacenamiento.	Permite la reducción del almacenamiento por lo general en un 90% sin sacrificar la resolución.	Cualquier sistema VMS.
XProtect Analytics 2.0, Milestone	Integra distintas herramientas de análisis de vídeo de muchos proveedores	Ofrece un estándar para el manejo de vídeo de distintos proveedor	Al ser un integrador de herramientas permite la revisión analítica del vídeo	Solamente para sistemas Milestone

Según la información que se presenta en la tabla anterior, se puede concluir que los productos: Alsight, Ocularis Investigate – Video Management User Interface; TimeSight Systems – NVR Optimized for Storage Minimization y XProtect Analytics 2.0 Milestone permiten la detección de actividades, almacenamiento en dependencia del tipo de actividades y la revisión analítica de los vídeos para la obtención de situaciones, siendo esto muy conveniente en la identificación de situaciones de alerta.

Anexo II: Clases de *SSN*

Estimulo Fuente: (W3C, Semantic Sensor Network Ontology, 2011)

URI:	http://purl.oclc.org/NET/ssnx/ssn#Stimulus
Label:	Stimulus
Source:	[SSN XG]
Subclass of	DUL:Event
Paraphrase (experimental)	ssn:Stimulus proviene de DUL:Event

Clase Sensor Fuente: (W3C, Semantic Sensor Network Ontology, 2011)

URI:	http://purl.oclc.org/NET/ssnx/ssn#Sensor
Label:	Sensor
Source:	skos:exactMatch 'sensor'
Subclass of	DUL:PhysicalObject
Paraphrase	ssn:implements

Clase Observación Fuente: (W3C, Semantic Sensor Network Ontology, 2011)

URI:	http://purl.oclc.org/NET/ssnx/ssn#Observation
Label:	Observation
Source:	skos:closeMatch 'observation' skos:closeMatch 'measurement result'
Subclass of:	DUL:Situation

Paraphrase (experimental)	ssn:observedProperty ssn:Property ssn:sensingMethodUsed ssn:featureOfInterest ssn:FeatureOfInterestna ssn:sensingMethodUsed ssn:Sensing DUL:includesEvent ssn:observationResult ssn:SensorOutput ssn:SensorInput ssn:observedB ssn:Sensor ssn:observedBy ssn:observationSamplingTime ssn:observationResultTime ssn:qualityOfObservation

En la tabla anterior se presentan las propiedades de la clase Observación. Las más importantes se describen a continuación. Dadas las múltiples instancias que pueden darse en un proceso de detección; se denomina características de interés a los objetivos específicos del proceso en una realidad física. En la ontología *SSN* se definen mediante la propiedad *FeatureOfInterest*. Cuando un sensor es accionado por un estímulo, en muchos casos es necesaria información sobre cómo funciona el sensor. Pueden existir diversas maneras de procesar la detección, algunos sensores realizan representaciones digitalizadas de un estímulo, por tanto, la información de cómo fue hecha la

observación o que método de análisis es implementado por el sensor aporta datos adicionales sobre la observación; todos los datos relacionados con el proceso de detección de un sensor están conceptualizados en la propiedad *Property*. El proceso de detección genera datos que pueden ser el resultado como tal del proceso de detección o puede calcularse a través de algoritmos. Para indicar aquello se hace uso de la propiedad *Sensing*. Los datos específicos de un sensor en relación con sus entradas y salidas se expresan en valores simples o relacionados con objetos de tipo *xsd* en la ontología *SSN* estos valores pueden ser descritos mediante las propiedades *SensorInput* y *SensorOutput*.

Clase Sistema. Fuente: (W3C, Semantic Sensor Network Ontology, 2011)

URI:	http://purl.oclc.org/NET/ssnx/ssn#System
Label:	System
Source:	[SSN XG]

Subclass of	DUL:PhysicalObject ssn:OperatingRange ssn:System ssn:Deployment, ssn:SurvivalRange ssn:Platform
Paraphrase (experimental)	ssn:hasOperatingRange ssn:hasSubSystem ssn:hasDeployment ssn:hasSurvivalRange ssn:onPlatform

Clase Process Fuente: (W3C, Semantic Sensor Network Ontology, 2011)

URI:	http://purl.oclc.org/NET/ssnx/ssn#Process
Label:	Process
Source:	[SSN XG]
Subclass of	DUL:Method.
Paraphrase (experimental)	ssn:hasMeasurementCapability ssn:hasAccuracy ssn:hasFrequency ssn:hasLatency ssn:hasPrecition ssn:hasResolution

La forma de expresar la medida de la observación de un sensor viene dado por instancias de las clases *MeasurementCapability* a través de la propiedad *hasMeasurementCapability*. Un valor de una medida observada puede no ser la medida real de un fenómeno, la proximidad entre el valor medido y el valor real de un fenómeno observado viene dado por la propiedad *hasAccuracy*. Se puede hacer observaciones cada cierto intervalo de tiempo; por ejemplo adquirir datos de otro

sistema puede estar determinado por las restricciones de los servicios expuestos por el sistema externo y se podría por ejemplo hacer n consultas por día, de esta manera funcionan las APIs sociales; en *SSN* la frecuencia de observación de un fenómeno está dado por la propiedad *hasFrequency*. En proceso de adquisición de datos de un sensor existen muchos otros valores susceptibles a parametrización y que modifican la forma en que el proceso como tal se ejecuta; para ello en *SSN* se puede usar las propiedades *hasLatency*, *hasPrecition* y *hasResolution*.

Deployment. Fuente: (W3C, Semantic Sensor Network Ontology, 2011)

URI:	http://purl.oclc.org/NET/ssnx/ssn#DeploymentRelatedProcess
Label:	Deployment
Source:	[SSN XG]
Subclass of	DUL:Process
Paraphrase (experimental)	ssn:deploymentProcessPart

DeploymentRelatedProcess. Fuente: (W3C, Semantic Sensor Network Ontology, 2011)

URI:	http://purl.oclc.org/NET/ssnx/ssn#DeploymentRelatedProcess
Label:	Deployment-related Process
Source:	[SSN XG]
Subclass of	DUL:Process
Paraphrase	ssn:deploymentProcessPart

En escenarios de *SV* que operan con redes de sensores, existen condiciones muy específicas para determinados comportamientos de los sensores que en algunos casos son necesarias especificarlas como restricciones más que como producto de inferencias. Este tipo de axiomas están en general relacionados con los requerimientos de sistema. Para conceptualizar las restricciones se utiliza la propiedad *deploymentProcessPart*.

Dado que existen diversos ambientes que soportan sistemas de redes de sensores, se requiere una propiedad que permita hacer referencia a determinado sistema para este propósito se utiliza la propiedad *deployedOnPlatafrom*. También existen algunos sensores que requieren un manteniendo riguroso para su correcto funcionamiento. La conceptualización del mantenimiento se realiza instanciando ese proceso en la propiedad *MaintennaceSchedule*. Para referirse al tiempo en que un sensor debe estar operativo se conceptualiza la propiedad *OperatingProperty* y para indicar su rango estándar (*Survival Range*) de valores se conceptualiza la propiedad *OperantigRange*.

A continuación se describe el módulo semántico que verifica si han existido errores en el diseño. Este módulo es muy importante ya que funciona como un razonador que verifica la taxonomía de las clases utilizadas.

ANEXO III

Módulo de consistencia

En términos de diseño arquitectónico se puede verificar la consistencia de un modelo mediante razonadores al verificar el cumplimiento de los axiomas y características definidos en lógica descriptiva, *TBox*(caja terminológica) *y Abox*(caja de aserciones). "En *TBox* se describe a los conceptos jerárquicos y *ABox* describe la pertenencia de las instancias a la jerarquía. Por ejemplo, la frase: Cada empleado es una persona se describe en *TBox*, mientras que la frase *Bob* es un empleado se describe en *ABox*. Los axiomas necesarios para conceptualizar eventos se detallan en *SBox*, este tipo de axiomas son de obligatorio cumplimiento*"* (Baader, Calvanese, McGuiness, Nardi, & Patel-Schneider, 2003). En ese sentido, los requerimientos de cualquier dominio pueden ser axiomatizados, esto significa que cualquier que sean las condiciones bajo las cuales se desarrolla un escenario, pueden ser puestas en términos de lógica descriptiva. –El razonamiento espacial es de vital interés para el diseño arquitectónico prueba de ello son las capacidades específicas de motores de razonamiento como *RacerPro*[23] que soporta razonamiento espacial directamente trabajado sobre *SBox*, el mismo que se basa en algoritmos de cálculo de conexión entre regiones como *RCC.5* y *RCC.8* (Grütter, Scharrenbach, & Bauer-Messmer, 2008). Un esquema *SBox* se ocupa de la descripción de la similitud de objetos basado en las definiciones formales de una clase y las instancias generadas. En *SBox*, un atributo cualquiera de una clase puede determinar

[23] Racerpro: http://www.racer-systems.com/products/racerpro/, tomado el 22-10-2013

formas de asociar una instancia y las diferentes regiones que se pueden dar en un espacio determinado pueden ser usadas como criterio de separación de dominios durante el diseño arquitectónico. Rara vez se llega a este nivel de análisis en los procesos de diseño actuales; pero con *SBox* se logra aquello con el cálculo de regiones de conexión basados en sus atributos cualitativos. De allí que con *SBox,* se puede conceptualizar tipos de relaciones entre regiones:

- ***Desconexión (DC)***: Se refiere a conjuntos de instancias que no comparten ninguna región.
- ***Conexión externa (EC)***: Se refiere a conjuntos de instancias que poseen un solo punto de conexión común en el límite de cada conjunto.
- ***Igualdad (EQ)***: Se refiere a los conjuntos de instancias de *X* y *Y* que comparten el mismo espacio de conexión; toda *X* está en *Y* y toda *Y* está en *X*.
- ***Sobre posición parcial (PO)***: Se refiere a la región que contiene a todos los elementos comunes a *X* y a *Y*; también conocida como intersección.
- ***Rol tangencialmente correspondiente (TPP)***: Se refiere a la región de todos los elementos de *X* que también pertenecen a *Y*, donde *X* es un subconjunto tangencial a *Y*.
- ***Rol tangencial inverso (TPPi)***: Se refiere a la región de todos los elementos de *X* que no pertenecen a *Y*, donde *Y* es un subconjunto tangencial a *X*.
- ***Rol no tangencialmente correspondiente (NTTP)***: Se refiere a las regiones de todos los elementos de *X* también perecen a *Y*, donde *X* es un subconjunto no tangencial a *Y*.

- ***Rol no tangencial inverso (NTTPi)***: Se refiere a la región de todos los elementos de *X* que no pertenecen a *Y*, donde *Y* es un subconjunto no tangencial a *X*.

Este tipo de relaciones permiten obtener la consistencia de un modelo arquitectónico conceptualizado de forma semántica. En nuestra propuesta, este tipo de modelos nos van a servir para conceptualizar actividades y situaciones de alto nivel semántico como cambio de zona, seguimiento (*tracking*), *Merodeo*.

ANEXO IV: Modelo Temporal

```
:Instant
a owl:Class ;
rdfs:subClassOf :TemporalEntity .
:Interval
a owl:Class ;
 rdfs:subClassOf :TemporalEntity .
:TemporalEntity
 a owl:Class ;
rdfs:subClassOf :TemporalThing ;
 owl:equivalentClass
[ a owl:Class ;
 owl:unionOf (:Instant :Interval)
]
```

Principales entidades *Time* Ontology Fuente: (W3C, Time Ontology in OWL , 2006)

```
:hasBeginning

a

owl:ObjectProperty ;

rdfs:domain

:TemporalEntity ;

rdfs:range :Instant .
```

hasBeginning Fuente: (W3C, Time Ontology in OWL , 2006)

```
:inDateTime
        a owl:ObjectProperty ;
        rdfs:domain :Instant ;
        rdfs:range :DateTimeDescription.
```

inDateTime Fuente: (W3C, Time Ontology in OWL , 2006)

```
:DurationDescription
 a owl:Class ;
 rdfs:subClassOf
        [ a owl:Restriction ;
        owl:maxCardinality 1 ;
        owl:onProperty :seconds] ;
rdfs:subClassOf
 [ a owl:Restriction ;
        owl:maxCardinality 1 ;
        owl:onProperty :minutes] ;
rdfs:subClassOf
        [ a owl:Restriction ;
        owl:maxCardinality 1 ;
```

```
        owl:onProperty :hours] ;

rdfs:subClassOf

        [ a owl:Restriction ;

        owl:maxCardinality 1 ;

        owl:onProperty :days] ;

rdfs:subClassOf

        [ a owl:Restriction ;

        owl:maxCardinality 1 ;

        owl:onProperty :weeks] ;

        rdfs:subClassOf

        [ a owl:Restriction ;

        owl:maxCardinality 1 ;

        owl:onProperty :months] ;

        rdfs:subClassOf

        [ a owl:Restriction ;

        owl:maxCardinality 1 ;

        owl:onProperty :years]
```

DurationDescription Fuente: (W3C, Time Ontology in OWL , 2006)

```
:TemporalUnit

        a owl:Class ;

        owl:equivalentClass

        [ a owl:Class ;

 owl:oneOf (:unitSecond :unitMinute :unitHour :unitDay :unitWeek :unitMonth :unitYear)

        ] .
```

Unidad temporal Fuente: (W3C, Time Ontology in OWL , 2006)

```
:DayOfWeek
 a owl:Class ;
 owl:equivalentClass
 [ a owl:Class ;
 owl:oneOf (:Sunday :Monday :Tuesday :Wednesday :Thursday :Friday :Saturday)] .
```

DayOfWeek Fuente: (W3C, Time Ontology in OWL , 2006)

Anexo V.

***Time*:Zonas Horarias**

Es la conceptualización para las distintas zonas horarias del planeta. Algunos de los componentes de un objeto temporal son sensibles a la zona horaria. La implicación de la relatividad horaria en función de su espacio geográfico y su estandarización, implican la interacción con una ontología de geo-localidades. Tradicionalmente lo que se hace en sistemas es trabajar con estandarizaciones propias de cada zona horaria. La complejidad de manejar equivalencias temporales de zonas horarios no es de interés de la ontología propuesta por (W3C, Time Ontology in OWL , 2006); que más bien trata de introducir una manera genérica de definir el tiempo independientemente de la localización. La tarea de llevar un esquema a otro es trabajo de personalizaciones ontológicas o del nivel de aplicación.

Para el manejo de valores temporales relativos a la hora; se definen unidades de tiempo. Esto se traduce en la ontología como una restricción que implica que la hora viene dada en una unidad y solo una. La restricción referente a la unidad de la hora se representa en la propiedad *TemporalUnit* –un tipo de unidad para una descripción temporal que haga referencia a las 11:00 am puede venir dada en la unidad minutos, mientras que; un tipo de unidad para la fecha 14 de abril del 2013 puede venir expresada en términos de días como unidad. El tipo de unidad temporal es una propiedad obligatoria. Por ejemplo, la fecha viene con un tipo de unidad día, en donde las propiedades hora, minuto y segundo son ignoradas. Esto no supone perdida de datos por qué debemos recordar que un objeto temporal puede estar asociado a un *DurationDescription* como también a un

DateTimeDescription a través de las propiedades *hasDurationDescription* y *hasDateTimeDescription*, de la ontología *Time*. Para los días de la semana existe la propiedad *dayOfWeek* y el dominio para esta propiedad es *DayOfWeek*.

DateTimeDescription puede servir para representar constantes temporales. Por ejemplo, enero tiene como constante ser el mes 1. Su propiedad *unitType* proviene de objetos tipo *unitMonth* y su valor es 1:

```
:January
a owl:Class ;
 rdfs:subClassOf :DateTimeDescription ;
        rdfs:subClassOf
        [ a owl:Restriction ;
        owl:onProperty :unitType
        owl:hasValue :unitMonth
        ] ;
        rdfs:subClassOf
        [ a owl:Restriction ;
        owl:onProperty :month
        owl:hasValue --01 ;
        ]
```

Restricción valor constante Enero como mes 1 Fuente: (W3C, Time Ontology in OWL , 2006)

En *OWL* existen dos propiedades referentes al tiempo *xsdDateTime* e *inXSDDateTime*, cuya diferencia con las propiedades propuestas *hasDateTimeDescription* e *inDateTime* radica principalmente en el rango. Para las primeras se usa un rango de datos definidos

en un esquema simple de datos *XML* mientras que para el rango de las propiedades de *Time* son objetos de tipo *DateTimeDescription*.

Las ontologías que se utilizan en nuestra propuesta para inferir actividades y situaciones de alto nivel semántico han sido descritas. La necesidad de ontologías en el dominio del análisis e interpretación automática de video surge del aumento de sistemas orientados a esta tarea alrededor del mundo. Una ontología agrupa conceptos, propiedades y restricciones de un dominio bajo un determinado nivel de convenciones de diseño, siendo los recursos ontológicos la principal herramienta de los expertos en la creación de sistemas que respondan de manera autónoma ante la detección de eventos en fuentes de video. Una ontología permite hacer entendible el concepto de video en sistemas enfocados en la experiencia del usuario final; y le permite al experto describir modelos muy complejos en términos entendibles por las computadoras, así como también abstraer modelos ya existentes para un dominio de interés. La ontología es crucial al momento de describir un video ya que nos permite tener una mejor comprensión de los eventos específicos que deben ser reconocidos en un video.

La consolidación de una ontología para el reconocimiento de eventos resulta muy complicado debido a que desarrolladores y expertos relacionados al dominio de aplicaciones en esta área tienen diferentes percepciones de como es el comportamiento humano. Esto lleva a generar múltiples representaciones ontológicas del mismo, además existen otras variables que dificultan las tareas y que no están relacionados con la descripción; cada región del mundo posee conductas propias de la realidad del país al que pertenecen. La terminología seleccionada para la descripción de las relaciones y

conceptos en el contexto de análisis de video, son tomados de la vida diaria y esto genera toda clase de problemas en términos de ambigüedad.