Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en Computación

Sistema semántico para la representación de contextos utilizados en aplicaciones de cómputo ubicuo

Alumno:
Antonio Guadarrama Quezada

Matrícula: 208305511

Asesora:
Maricela Claudia Bravo Contreras
Departamento de Sistemas

Julio 2013

Contenido

1. Resumen

En los últimos años se ha incrementado la tendencia hacia el desarrollo de aplicaciones de cómputo ubicuo que usen sistemas que busquen ofrecer a los usuarios altos niveles de comodidad y que sean fáciles de usar, de tal forma que se pone mayor interés en capturar la información del lugar en el que el usuario se desenvuelve. Es por esto que los sistemas semánticos para la representación de contextos han adquirido una enorme importancia en las áreas de investigación y desarrollo vigentes hoy en día.

Al terminar éste proyecto terminal se obtendrá una aplicación que utilice un sistema semántico de recomendación inteligente para la representación de contextos de cómputo ubicuo, que contienen un modelo multidimensional de ontologías con dominios de conocimiento con la información de alumnos, profesores, espacios físicos del departamento de Sistemas de la división de CBI en la UAM-A. Así que estos diferentes tipos de elementos organizacionales modelados en la ontología de contexto multidimensional serán recomendados mediante la aplicación de conjuntos de reglas de inferencia. Estas características son combinadas con las preferencias y los intereses de los usuarios para obtener un conjunto de indicaciones adecuadas para un usuario específico que visita la división de CBI.

2. Introducción

En este proyecto terminal se involucran los temas de diseño y construcción de ontologías, representación de contextos y cómputo ubicuo, los cuales se describen a continuación.

Una ontología se define como una especificación formal de una conceptualización compartida [1]. En otras palabras, define los términos y relaciones básicas que comprenden el vocabulario de un tópico de área, así como las reglas para combinar los términos y relaciones para definir extensiones al vocabulario. Además, puede verse a las ontologías como un sistema para la representación de una base de conocimiento. Las ontologías se han desarrollado dentro de la comunidad de investigación de modelado del conocimiento[1], con el fin de facilitar el intercambio y reutilización de éste [2].

En cuanto al contexto, se define como "cualquier información que puede ser usada para caracterizar la situación de una entidad, siendo ésta una persona, un lugar, un objeto que se considera relevante en la interacción entre un usuario y la aplicación" [3]. Un contexto puede verse como un espacio multidimensional, donde cada dimensión es representada por una ontología específica [4].

Con respecto al cómputo ubicuo, en 1991 Mark Weiser propone este concepto cuyo objetivo es dotar al ambiente físico con dispositivos (sensores, cámaras, termómetros,

[1] Se refiere a los investigadores que estudian los métodos y lenguajes para el modelado de conocimiento, generalmente asociados al campo de la Inteligencia Artificial que estudia la representación de conocimiento.

escáneres, etc.) que cuentan con capacidades computacionales, algoritmos dotados de cierta inteligencia y capacidades de comunicación integrados. Dichos dispositivos se integran de manera natural al ambiente en donde los humanos ejecutan sus actividades diarias [5].

Concretamente en este proyecto terminal se propone diseñar e implementar una aplicación que usa y consulta un sistema semántico con una base de conocimientos, formada por un modelo compuesto por diversas ontologías que se interconectan para modelar un gran número de dimensiones contextuales para la representación de cómputo ubicuo. Para el diseño de éstas ontologías se tomará en cuenta diversos enfoques de diferentes dominios de conocimiento, como la información de alumnos, profesores, espacios físicos y programas académicos del departamento de Sistemas de la división de CBI en la UAM-A.

3. Objetivo general

Diseñar e implementar un sistema semántico basado en un modelo multidimensional de ontologías para representar contextos dinámicos utilizados en aplicaciones de cómputo ubicuo.

3.1 Objetivos específicos

Diseñar e implementar:

- ✓ Un conjunto de ontologías para representar datos de alumnos, profesores, espacios físicos y programas académicos del departamento de sistemas de la división de CBI (Ciencias Básicas e Ingeniería) en la UAM-A (Universidad Autónoma Metropolitana Unidad Azcapotzalco).
- ✓ Un modelo multidimensional basado en las ontologías para la representación de contextos dinámicos utilizados en aplicaciones de cómputo ubicuo.
- ✓ Una aplicación para usar y consultar la información organizada en el modelo multidimensional de ontologías.

4. Desarrollo

4.1. Diseño

4.1.1. Esquema general del sistema

El proyecto terminal consta de las siguientes capas con sus respectivos módulos y componentes para la realización del sistema semántico para la representación de contextos utilizados en aplicaciones de cómputo ubicuo (éstas se muestran en la Figura 1):

❖ Capa de captura:

- Módulo de captura de información.

❖ Capa del modelo semántico:

- Modelo multidimensional.
- Módulo de gestión de ontologías.
- Conjunto de reglas de inferencia.
- Módulo de razonamiento e inferencias lógicas.

❖ Capa de aplicación:

- Módulo de aplicación de usuario para consulta al sistema.

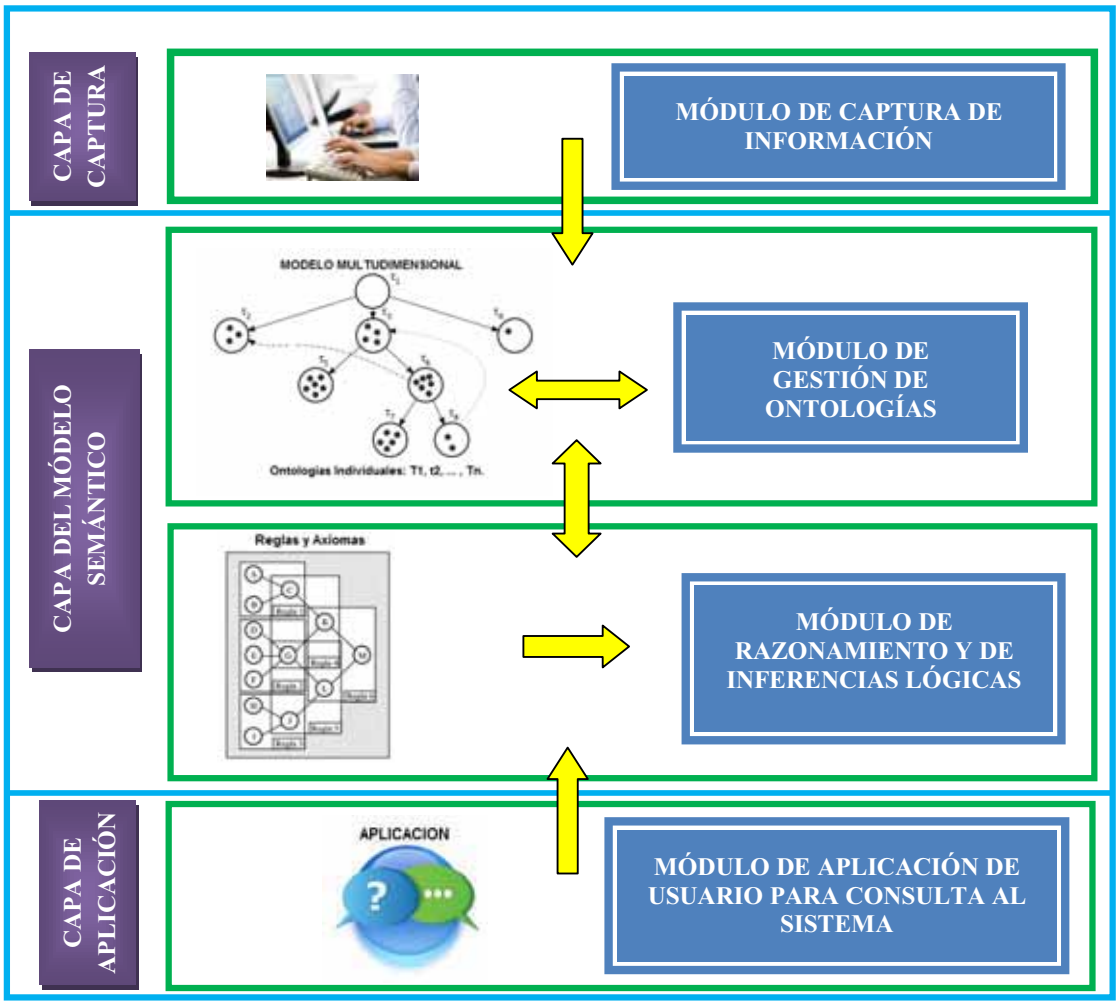

Figura 1. Descripción del sistema.

4.1.2. Capa de captura.

En esta capa se obtienen los datos involucrados en el sistema. Dichos datos nos sirvieron para poder crear el diseño de las ontologías y por lo tanto el modelo multidimensional ontologías que es la representación de una base de conocimiento.

4.1.2.1 Módulo de captura de información.

Una vez obtenidos todos los datos del departamento de Sistemas de la división de CBI de la UAM-A, se diseñó e implemento las siguientes ontologías: Personas, Espacios Físicos, AcadémicoCBI. Éstas contienen distintos datos, los cuales se muestran en la Tabla 1.

Ontología	**Datos**
Persona	Alumnos (licenciatura y posgrado). Empleados: académicos (ayudantes y profesores), administrativos y de servicio. Visitantes.
Espacio físico	Áreas, edificios y salones.
AcadémicoCBI	Coordinación. Departamento. Programas académicos: licenciatura y posgrado (maestria y doctorado). UEA´s del plan de estudios de ingeniería en computación.

Tabla 1. Datos de captura del sistema.

A continuación se muestra el diseño de cada ontología anteriormente mencionadas.

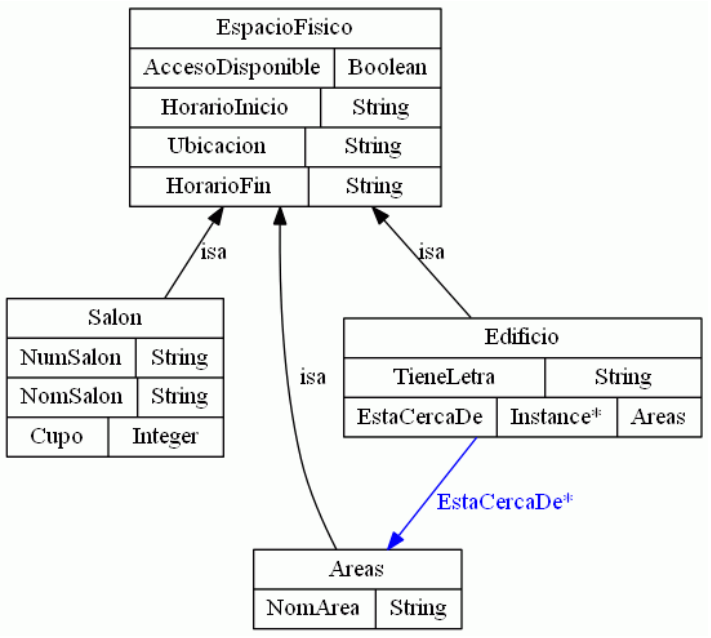

Figura 2. Diseño de la ontología Espacios Físicos.

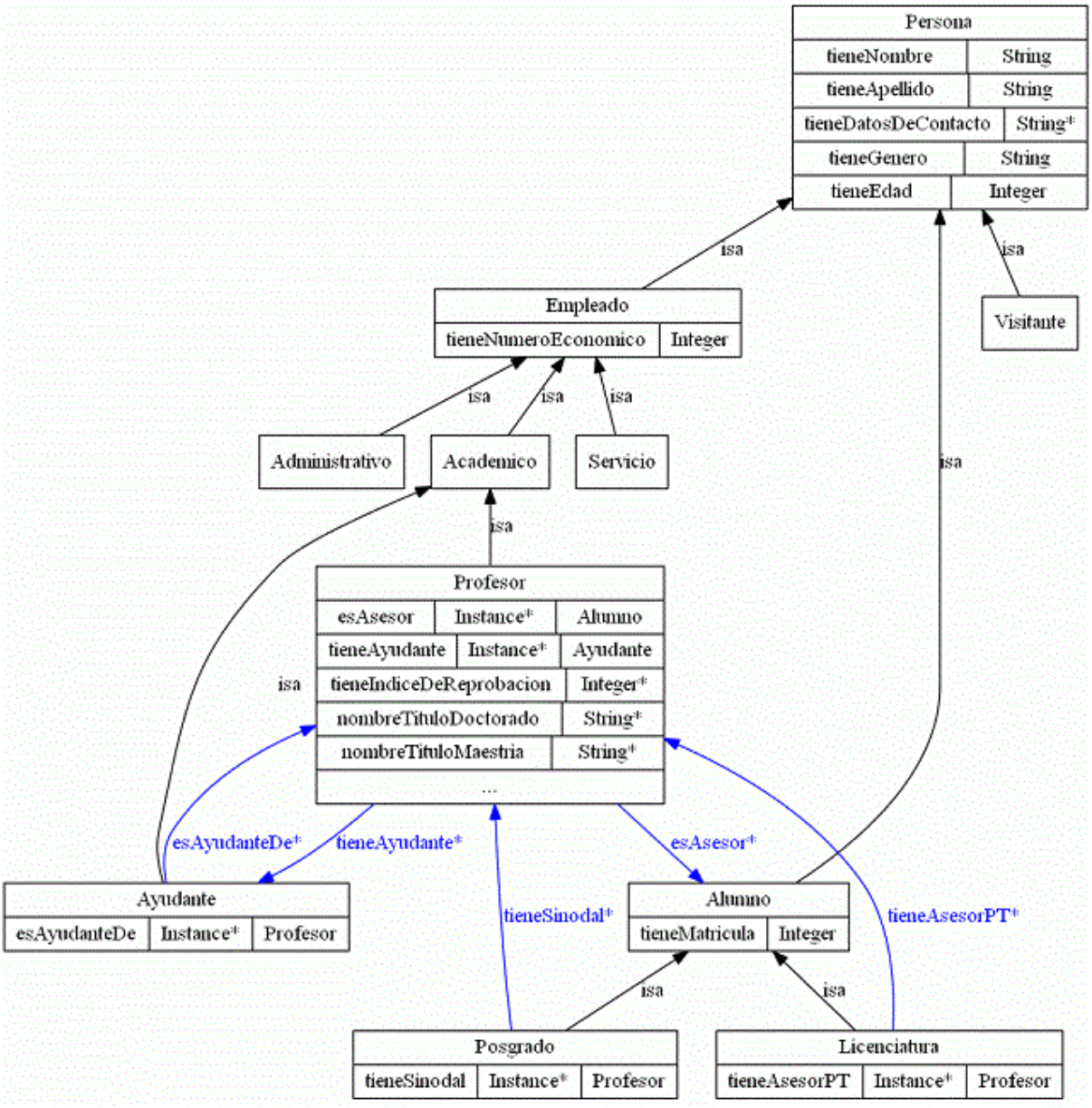

Figura 3. Diseño de la ontología Persona.

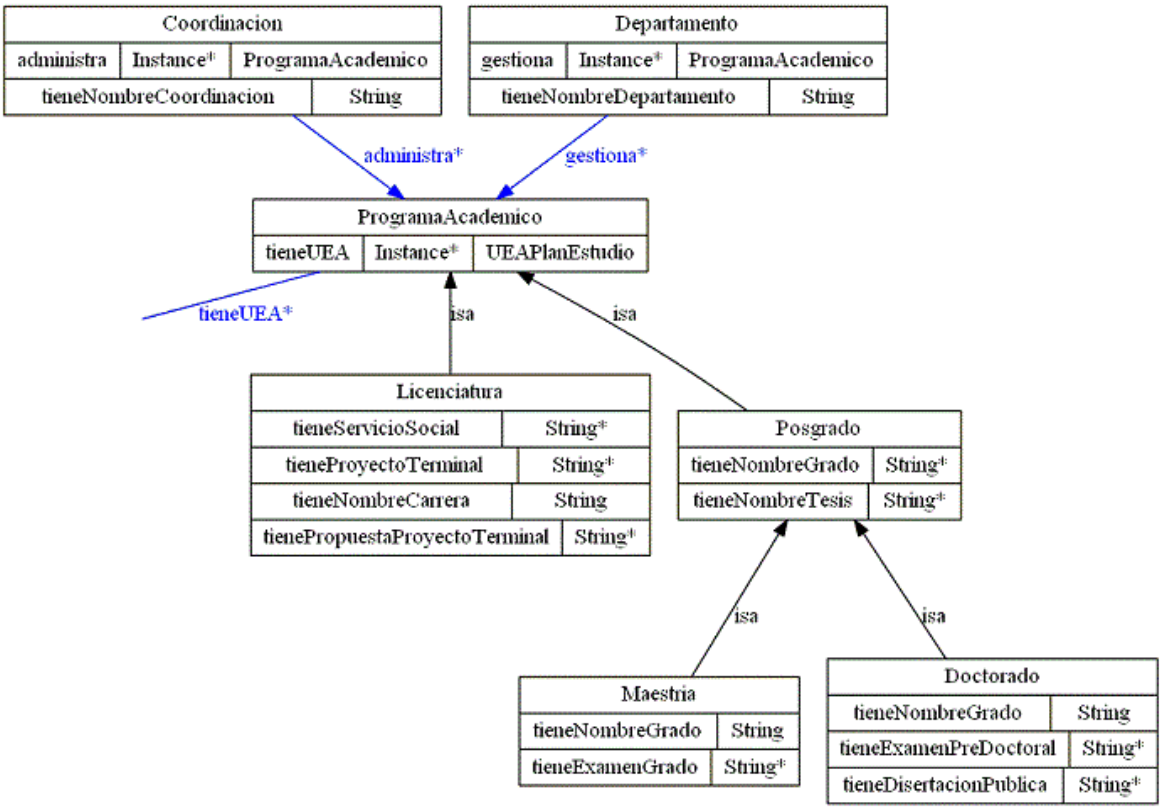

Figura 4. Diseño de la ontología AcadémicoCBI (el diagrama continua en la Figura 5.)

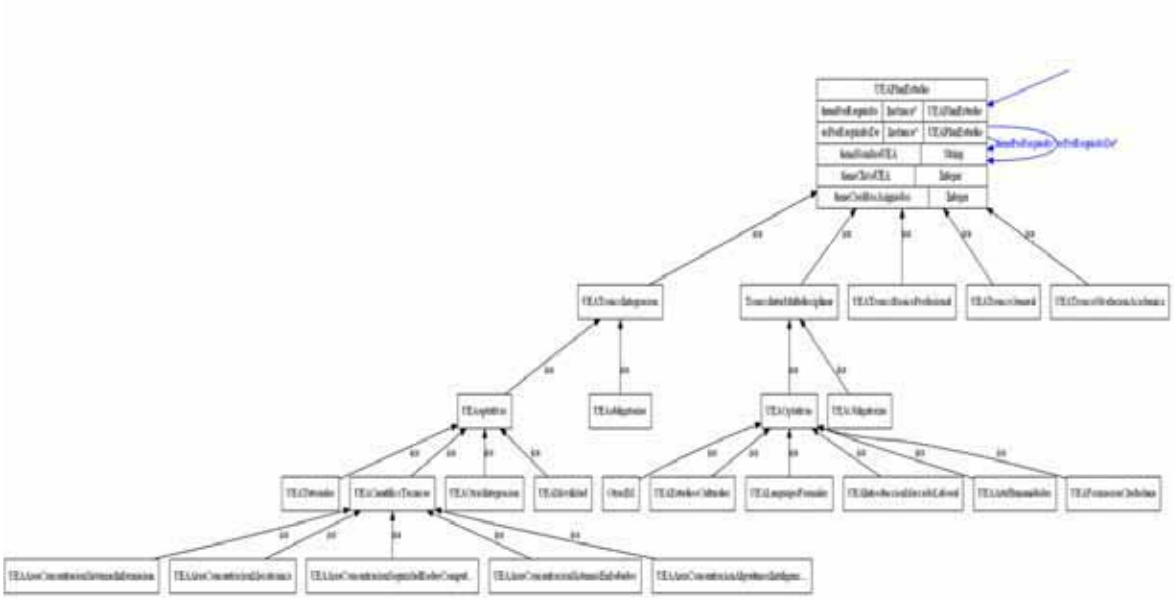

Figura 5. Diseño de la ontología AcadémicoCBI.

4.1.3.1 Modelo multidimensional.

Para modelar el contexto como un espacio multidimensional, se implementa con la red de estas ontologías individuales (descritas anteriormente), en la que se utilizan ontologías específicas relacionadas entre sí de acuerdo a las dimensiones requeridas, las cuales, en conjunto contienen la información que describe el contexto.

Se creó una ontología llamada OntologíaRed la cual contiene las tres ontologías individuales (Personas, Espacios Físicos y AcadémicoCBI). Una vez importadas estas ontologías se precedió a crear relaciones entre estas, para esto se crearon dos clases llamadas "horario" y "clase", las cuales tienen relaciones con las tres ontologías individuales, las características de esta clase se muestran en la Figura 6.

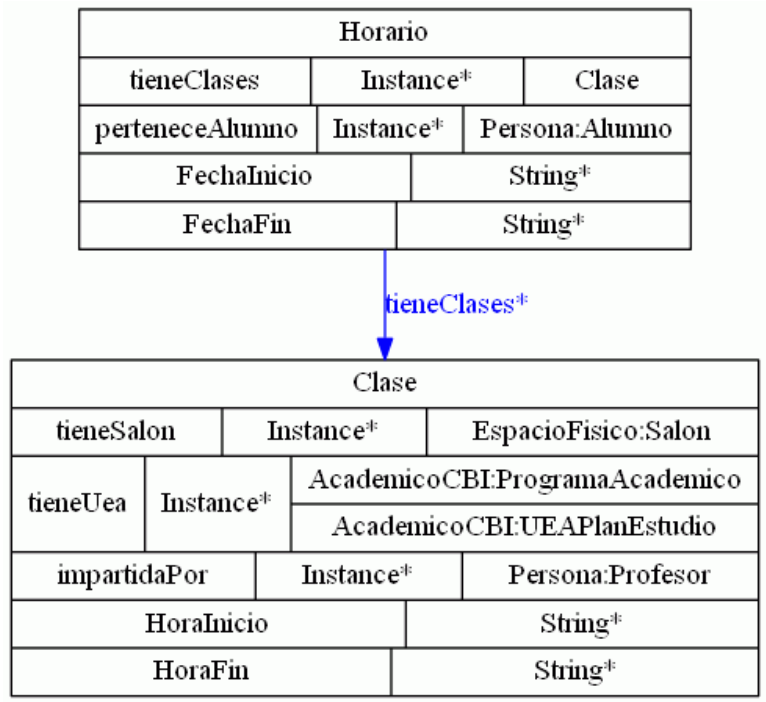

Figura 6. Diseño de las clases "horario" y "clase" de la OntologíaRed.

El diseño de esta ontología permitió la realización de diversas reglas de inferencia ya que relaciona de una buena manera cada ontología individual.

4.1.3.2. Módulo de gestión de ontologías.

En éste se realizan las consultas, actualizaciones y eliminaciones (individuos y relaciones) de las ontologías. Este módulo se realizó directamente en el editor de ontologías Protégé una vez que se obtuvo el diseño de las ontologías individuales así como la ontología que contiene el conjunto de todas éstas.

4.1.3.3. Conjunto de reglas de inferencia.

Se definieron diversas reglas de inferencia necesarias. Las reglas que se incluyeron permitieron descubrir y generar nuevas relaciones (conocimiento) entre los elementos incluidos en las ontologías. En general, todo el modelo semántico ayudará a realizar consultas de nuestras ontologías diseñadas. En la Tabla 2 se muestran tres tipos distintos de consultas (preguntas) realizadas en nuestro sistema así como los parámetros necesarios para su ejecución.

Pregunta / Parámetros	**Regla de inferencia**
¿En qué salón se encuentra el profesor "X" a la hora "Y"? / apellido, hora.	Persona:Profesor(?x) ∧ Persona:tieneApellido(?x, "+apellido+") ∧ Clase(?y) ∧ impartidaPor(?y, ?x) ∧ HoraInicio(?y, "+hora+") ∧ tieneSalon(?y, ?z) ∧ EspacioFisico:NumSalon(?z, ?t) → sqwrl:selectDistinct(?t)
¿Qué clases de imparten en el salón "X"? / salón.	Clase(?x) ∧ EspacioFisico:Salon(?y) ∧ EspacioFisico:NumSalon(?y, "+salon+") ∧ tieneSalon(?x, ?y) ∧ EspacioFisico:NumSalon(?y, ?t) ∧ tieneUea(?x, ?w) ∧ AcademicoCBI:tieneNombreUEA(?w, ?z) → sqwrl:selectDistinct(?z)
¿En qué horarios y salón se imparte la UEA "X"? / nomUea	Clase(?x) ∧ HoraInicio(?x, ?a) ∧ HoraFin(?x, ?b) ∧ tieneUea(?x, ?c) ∧ AcademicoCBI:tieneNombreUEA(?c, "nomUea") ∧ tieneSalon(?x, ?t) ∧ EspacioFisico:NumSalon(?t, ?w) → sqwrl:selectDistinct(?a, ?b, ?w)

Tabla 2. Algunas reglas de inferencia empleadas.

4.1.3.4. Módulo de razonamiento e inferencias lógicas.

Éste incorpora motores de razonamiento y de inferencias; utiliza como base de conocimiento el modelo multidimensional para poder generar nuevas relaciones semánticas y nuevos conceptos. En este módulo se ejecutaron cada una de las reglas de inferencia las cuales arrojaron resultados de acuerdo a los parámetros insertados. En la Tabla 3 se muestra las respuestas a las consultas de la tabla anterior con parámetros correctos que no generaron errores.

Pregunta / Parámetros	**Resultado de la regla ejecutada**
¿En qué salón se encuentra el profesor "X" a la hora "Y"? / apellido: Aguilar Vázquez, hora: 11:30:00	En el salón: E004
¿Qué clases de imparten en el salón "X"? / salón: E102	UeaIntroduccionAlgebraLineal UeaProgramacionSistemas
¿En qué horarios y salón se imparte la UEA "X"? / nomUea: UeaCompiladores	En el salón: E001 Hora de inicio: 16:30:00 Hora de fin: 18:00:00

Tabla 3. Resultados de consultas.

Cabe mencionar que se procedió a diseñar y comprobar si las reglas de inferencia funcionaban primeramente en Protégé de tal manera que cuando se llegara a la capa de

aplicación se tuviera el mínimo de errores por parte de la lógica de las reglas de inferencia.

4.1.4. Capa de aplicación.

4.1.4.1. Módulo de aplicación de usuario para consulta al sistema.

La aplicación usa y consulta la información organizada en el modelo multidimensional de ontologías con los diferentes tipos de elementos organizacionales modelados en ésta, mediante la aplicación de los conjuntos de reglas de inferencia. Estas características son combinadas con las preferencias y los intereses de los usuarios para obtener un conjunto de recomendaciones adecuadas para un usuario específico que visita la UAM-A (las cuáles son preguntas prediseñadas para obtener conocimiento sobre el departamento de Sistemas de CBI de la UAM-A).

Figura 7. Vista de los componentes la interfaz gráfica desde NetBeans.

Para la ejecución de las reglas de inferencia desde código se usaron las API´s: Protégé-API y Jena-API. Gracias a estas API´s se pudieron ejecutar las reglas de inferencia así como se hacía desde Protégé de tal manera que esto nos permitió la creación de una aplicación que cumple con los objetivos de este proyecto.

La interfaz se realizó con un conjunto de componentes gráficos que posibilitan la interacción entre el usuario y la aplicación la cual es llamada Interfaz Gráfica GUI (Graphical User Interface) , el funcionamiento de esta se puede ver en la sección de pruebas del sistema de este documento.

Nuestra interfaz gráfica se compone de una clase principal “InterfazUsuario” en la cual está contenido todo el código de los componentes gráficos así como los métodos de las clases que contienen las API´s utilizadas que permitieron la ejecución de las reglas de inferencia. El proyecto de NetBeans también contiene las imágenes utilizadas así como las ontologías del sistema (en la Figura 7 se muestran los componentes). En los apéndices de este documento se mostrara como se realizó esta clase principal.

4.2. Implementación.

La implementación de este proyecto se realizó en una computadora portátil marca hp® con las siguientes características:

- 500 GB de disco duro.
- 4 GB de RAM.
- Procesador Intel® Core™ i3
- Sistema operativo Windows® 7.

4.2.1. Herramientas del sistema.

El Software que se utilizó para el diseño e implementación del proyecto es el siguiente:

- **Lenguaje de programación Java:** Este lenguaje de programación es de alto nivel y orientado a objetos. Es por esto que se eligió ya que permitió la creación de la aplicación del proyecto terminal.

- **IDE NetBeans 7.1.2:** Se usó NetBeans [6] como entorno de desarrollo integrado (IDE). El uso de este IDE permite al programador un desarrollo más ágil y una mejor estructuración del proyecto. Esta herramienta permitió una mejor documentación del código fuente de la aplicación así como la realización de las pruebas unitarias. En este entorno de desarrollo se utilizo La Interfaz Gráfica de Usuario (GUI, por sus siglas en inglés que significan Graphical User Interface) que es un conjunto de formas y métodos que posibilitan la interacción de un sistema con los usuarios utilizando formas gráficas e imágenes. Con formas gráficas se refiere a botones, íconos, ventanas, fuentes, etc. los cuales representan funciones, acciones e información.

- **Protégé versión 3.4.8:** Se utilizó para la realización de las ontologías y los modelos multidimensional de estas. Protégé 3.4.8 [7] es un editor de ontologías de libre distribución y de código abierto. Está basado en Java, es extensible, y proporciona un entorno que hace que sea una base flexible para la creación rápida de prototipos y desarrollo de aplicaciones.

- **Razonadores:** Se utilizó el razonador Jess para la interpretación y ejecución de las reglas de inferencia y consultas. Jess [8] es un motor de reglas y entorno de programación escrito completamente en java que tiene la capacidad de "razonar" con el conocimiento que usted suministra en forma de reglas declarativas. Jess es pequeño, ligero, y uno de los motores de reglas más rápidas

disponibles. Su potente lenguaje de script le permite acceder a todas las API de Java.

- **Interfaces de programación de aplicaciones (API´s) utilizadas:** Éstas se utilizara para desarrollar el módulo de aplicación.

    - Protégé-API: Se utilizo OWL API [9] ya que proporciona clases y métodos para el trabajo con ontologías, permitiendo la creación, manipulación y lectura. Particularmente esta API es muy importante en el proyecto ya que nos permite realizar el manejo de la ontología (cargar y guardar, consultar ontologías y para llevar a cabo el razonamiento). Esta API esta implementada en código abierto java lo que permitió una fácil integración con el proyecto y las demás API´s utilizadas, de tal manera que la integración de éstas resulto sencilla y no se produjo ningún conflicto al trabajar con ella.

    - Jena-API: Se utilizaron las clases y métodos de Jena-API [10] para el manejo de las ontologías pero principalmente para el uso de su motor de inferencia que nos sirvió poder ejecutar las reglas de inferencias desde java ya que esta API esta implementada en código abierto de Java.

4.2.2. Pruebas del sistema.

Para verificar el correcto funcionamiento del sistema, se realizara una prueba de ejecución de éste y se observa el resultado obtenido. En la Figura 8, observamos la ventana de bienvenida, que es la interfaz que nos permite interactuar con el sistema.

Figura 8. Ventana de bienvenida.

Una vez iniciada la aplicación, lo que sigue es hacer clic en el botón “Iniciar”, al hacer esto se abrirá una ventana principal la cual contiene tres botones: “Persona”, “Espacio Físico” y “Plan de Estudio” que dan acceso a los tres diferentes tipos de consultas. Esta ventana la observamos en la Figura 9.

Figura 9. Ventana principal de consultas.

Cada consulta necesita de parámetros para poder obtener un resultado. Una vez insertados los parámetros se debe de dar clic en el botón “Ejecutar” y con esto se mostrara el resultado en un cuadro de texto en la sección de cada consulta. Si se insertan parámetros para los cuales no se obtiene ningún resultado o si estos no son insertados correctamente nos aparecerá una ventana de error la cual se muestra en la Figura 10. Para poder a insertar nuevos parámetros solo se tiene que dar clic en el botón “Aceptar” o el botón de cerrar y la ventana se desaparecerá.

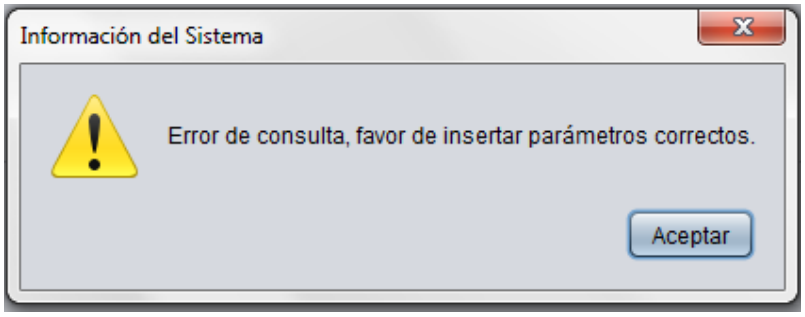

Figura 10. Ventana de error.

Un ejemplo de error se muestra en la Figura 11, ya que como no inserto ningún parámetro por lo tanto la consulta no se pudo realizar y por lo tanto generó un error.

Figura 11. Ejemplo de error.

Otro tipo de erro nos resulta cuando deacuerdo a los parámetros insetados, estos no generan ningun resultado. Por ejemplo en la Figura 12 se muestra que los parámetros insertados en la consulta número 3 no se obtiene ningun resultado ya que no existe ningun profesor con ese apellido.

Figura 12. Ejemplo de error de parámetros.

Ahora se mostrara la ejecución de las consultas de los tres distintos tipos. Al dar clic al botón “Personas” tendremos acceso a la ventana de las consultas referentes a las personas del Departamento de Sistemas de CBI de la UAM-A (Como lo son profesores, alumnos, visitantes, etc.). Una vez dentro de esta ventana podemos realizar 20 tipos diferentes de consultas, en la Figura 13 se muestran algunas de las consultas ejecutándose correctamente.

Figura 13. Ventana de consultas de persona.

Procederemos con los siguientes tipos de consultas. Al dar clic al botón "Espacios Físicos" tendremos acceso a la ventana de las consultas referentes a los espacios físicos de la UAM-A (Como lo son áreas, edificios y salones). Una vez dentro de esta ventana podemos realizar 10 tipos diferentes de consultas, en la Figura 14 se muestran algunas de las consultas ejecutándose correctamente.

Figura 14. Ventana de consultas de espacios físicos.

Los últimos tipos de consultas se muestran al dar clic al botón "Plan de Estudios" tendremos acceso a la ventana de las consultas referentes a él plan de estudios de la carrera ingeniería en computación de CBI de la UAM-A (Incluyen la informacion del plan de estudios, programa académico, coordinación y departamento de Sistemas de

CBI de la UAM-A). Una vez dentro de esta ventana podemos realizar 10 tipos diferentes de consultas, en la Figura 15 se muestran algunas de las consultas ejecutándose correctamente.

Figura 15. Ventana de consultas de plan de estudio.

5. Conclusiones.

El proyecto terminal se concluyó de una manera satisfactoria ya que se cumplieron los objetivos de éste. El diseño e implantación de este proyecto cumplió con lo esperado ya que se realizó un buen diseño de las ontologías, las cuales permitieron la creación de gran cantidad de consultas sobre estas ontologías. Es importante decir que las consultas funcionan gracias a la lógica empleada de la creación de las reglas de inferencia.

Al realizar este proyecto terminal se obtuvo conocimientos concretos de ontologías como lo son su diseño y manipulación. Se llegó a la conclusión que para poder diseñar una ontología primero se debe de tener bien delimitado el área del conocimiento que se quiere modelar y que el modelado se convierte en un proceso iterativo porque el diseño puede ir cambiando conforme se encuentren otros requerimientos que se necesiten representar.

Una de las dificultades encontradas en la elaboración de este proyecto se presentó en la construcción de las reglas de inferencia ya que la lógica de creación de estas no es nada trivial. Otra dificultad fue el uso de las distintas API´s de este proyecto ya que sus métodos y clases me resultaron un poco difíciles de comprender. A pesar de estas dificultades, se investigó de manera exhaustiva para poder comprender mejor de tal manera que no fue un impedimento en la realización de este proyecto.

Personalmente, realizar este proyecto me resulto muy interesante y muy útil porque me di cuenta de el gran uso que puede tener el uso de ontologías actualmente. Antes de realizar el proyecto desconocía totalmente esta área de la informática y ahora que ya conozco el proceso y las ventajas de trabajar con ontologías y las distintas API´s podré realizar proyectos a futuro con distintas aplicaciones.

Finamente se puede decir que una aplicación con un sistema semántico de recomendación inteligente que se desarrolló en este proyecto es una herramienta de gran apoyo para un usuario específico de la UAM-A para poder obtener de una manera más fácil y eficiente información que involucra este contexto que servirá de gran ayuda mientras permanezca en esta estancia.

6. Bibliografía.

[1] T. R. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", *International Journal of Human an Computer Studies*, Vol. 43, November, 1995, pp. 907-928.

[2] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition*, Vol. 5, 1993, pp. 199-220.

[3] A. K. Dey, "Providing Architectural Support for Building Context-Aware Applications," Ph.D. Thesis, Georgia Institute of Technology, November, 2000.

[4] A. Bouzeghoub, K. Ngoc Do, L. Krug Wives, "Situation-Aware Adaptive Recommendation to Assist Mobile Users in a Campus Environment." AINA, Bradford, *International Conference on Advanced Information Networking and Applications*, 2009, pp. 503-509.

[5] K. D. Wong. *Wireless Internet Telecommunications*. Artech House, 2005.

[6] NetBeans.org (Julio 2013). NetBeans IDE [En línea]. Disponible en: http://netbeans.org/

[7] Protege.stanford.edu (Julio 2013). Protégé [En línea]. Disponible en: http://protege.stanford.edu/.

[8] Jessrules.com (Julio 2013). the Rule Engine for the Java [En línea]. Disponible en: http://www.jessrules.com/jess/index.shtml

[9] Protege.stanford.edu (Julio 2013 protégé-owl api [En línea]. Disponible en: http://protege.stanford.edu/plugins/owl/api/

[10] Jena.apache.org (Julio 2013). Apache Jena [En línea]. Disponible en: http://jena.apache.org/

7. Apéndice.

7.1. Código fuente en JAVA.

En esta sección solo se mostrara código específico para lograr entender cómo se realizó la aplicación.

7.1.1 Método main de la clase InterfazUsuario

A continuación se muestra nuestro main de nuestra clase principal InterfazUsuario:

```
public static void main(String args[]) throws OntologyLoadException,
SQWRLException, URISyntaxException  {


        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(InterfazUsuario.class.getName()).lo
g(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(InterfazUsuario.class.getName()).lo
g(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(InterfazUsuario.class.getName()).lo
g(java.util.logging.Level.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(InterfazUsuario.class.getName()).lo
g(java.util.logging.Level.SEVERE, null, ex);
        }

        //Cargar la ontología desde la ubicación del archivo
        //String name ="ontologias/OntologiaRed.owl";

        URI uri = new File(name).toURI();

      //Se crea un modelo para poder realizar las reglas de inferencia
        OWLModel owlModel =
ProtegeOWL.createJenaOWLModelFromURI(uri.toString());


//Se importan las ontologías individuales
        owlModel.getNamespaceManager().setPrefix(new
URI("http://www.w3.org/2003/11/swrl#"),"swrl");
```

```
        ImportHelper importhelper = new
ImportHelper((JenaOWLModel)owlModel);



        String name1 = "ontologias/Persona.owl";
        String name2 = "ontologias/Espacio_Fisico.owl";
        String name3 = "ontologias/AcademicoCBI.owl";


        URI uri1 = new File(name1).toURI();
        URI uri2 = new File(name2).toURI();
        URI uri3 = new File(name3).toURI();

//Se importan las ontologías individuales a la OntologíaRed
        importhelper.addImport(uri1);
        importhelper.addImport(uri2);
        importhelper.addImport(uri3);

        importhelper.importOntologies(false);

//Se arranca el motor de inferencia para ejecutar las reglas de
inferencia
        queryEngine = SQWRLQueryEngineFactory.create(owlModel);

        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {

//Se ejecuta las ventanas para que se muestren
                new InterfazUsuario().setVisible(true);


                BienvenidaFrame.setVisible(true);
                Principal.setVisible(false);
                PersonaPanel.setVisible(false);
                EspacioFisicoPanel.setVisible(false);
                PlanEstudioPanel.setVisible(false);
                ScrollPersona.setVisible(false);
                ScrollEspacioFisico.setVisible(false);
                ScrollPlanEstudio.setVisible(false);
            }
        });
    }
```

7.1.2. Código de las reglas de inferencia.

Solo se muestran el código de las tres reglas que se mostraron en el módulo de razonamiento e inferencias lógicas.

Código del botón de la consulta 1 (¿En qué salón se encuentra el profesor "X" a la hora "Y"?):

```
private void jButton8ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Boolean  info=false;
```

```
        contador++;
        String nomregla="regla";
        String nam= '"'+Text1.getText()+'"';
        String hor='"'+Text2.getText()+'"';
        DefaultListModel modelo= new DefaultListModel();

        try {
            result =
queryEngine.runSQWRLQuery(nomregla+String.valueOf(contador),"Persona:P
rofesor(?x) ∧ Persona:tieneApellido(?x, "+nam+") ∧ Clase(?y) ∧
impartidaPor(?y, ?x) ∧  HoraInicio(?y, "+hor+") ∧ tieneSalon(?y, ?z) ∧
EspacioFisico:NumSalon(?z, ?t) →  sqwrl:select(?t)");
            while(result.hasNext ()) {
            modelo.addElement("En el salón:
"+result.getDataValue("?t").getString());
            result.next();
            info=true;
            }
            List1.setModel(modelo);
        }
        catch (SQWRLException ex) {

Logger.getLogger(InterfazUsuario.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (SWRLParseException ex) {

Logger.getLogger(InterfazUsuario.class.getName()).log(Level.SEVERE,
null, ex);
        }
        if(info==false){
            JOptionPane.showMessageDialog(this,"Error de consulta,
favor de insertar parámetros correctos.","Información del Sistema",2);
        }
    }
```

Código del botón de la consulta 2 (¿Qué clases de imparten en el salón "X"? / salón: E102):

```
private void jButton10ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        contador++;
        String nomregla="regla";
        String salon= '"'+jText1.getText()+'"';
        DefaultListModel modelo= new DefaultListModel();
        Boolean  info=false;

        try {
            result =
queryEngine.runSQWRLQuery(nomregla+String.valueOf(contador),"Clase(?x)
∧ EspacioFisico:Salon(?y) ∧ EspacioFisico:NumSalon(?y, "+salon+") ∧
tieneSalon(?x, ?y) ∧  EspacioFisico:NumSalon(?y, ?t) ∧ tieneUea(?x,
?w) ∧ AcademicoCBI:tieneNombreUEA(?w, ?z) →
sqwrl:selectDistinct(?z)");
            while(result.hasNext ()) {

modelo.addElement(result.getDataValue("?z").getString());
                result.next();
                info=true;
            }
```

```
            jList1.setModel(modelo);
        } catch (SQWRLException ex) {

Logger.getLogger(InterfazUsuario.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (SWRLParseException ex) {

Logger.getLogger(InterfazUsuario.class.getName()).log(Level.SEVERE,
null, ex);
        }
        if(info==false){
            JOptionPane.showMessageDialog(this,"Error de consulta,
favor de insertar parámetros correctos.","Información del Sistema",2);
        }
    }
```

Código del botón de la consulta 3 (¿En qué horarios y salón se imparte la UEA "X"?):

```
private void jButton41ActionPerformed(java.awt.event.ActionEvent evt)
{
        // TODO add your handling code here:
        contador++;
        String nomregla="regla";
        String clase= '"'+kText2.getText()+'"';
        DefaultListModel modelo= new DefaultListModel();
        Boolean  info=false;

        try {
            result =
queryEngine.runSQWRLQuery(nomregla+String.valueOf(contador),"Clase(?x)
∧ HoraInicio(?x, ?b) ∧ HoraFin(?x, ?c) ∧
AcademicoCBI:UEAPlanEstudio(?y) ∧ AcademicoCBI:tieneNombreUEA(?y,
"+clase+") ∧  tieneUea(?x, ?y) ∧  tieneSalon(?x, ?t)  ∧
EspacioFisico:NumSalon(?t,?w)→  sqwrl:selectDistinct(?b,?c,?w)");
            while(result.hasNext ()) {
                modelo.addElement("En el salón:
"+result.getDataValue("?w").getString());
                modelo.addElement("Hora de inicio:
"+result.getDataValue("?b").getString());
                modelo.addElement("Hora de fin:
"+result.getDataValue("?c").getString());
                result.next();
                info=true;
            }
            kList2.setModel(modelo);
        } catch (SQWRLException ex) {

Logger.getLogger(InterfazUsuario.class.getName()).log(Level.SEVERE,
null, ex);
        } catch (SWRLParseException ex) {

Logger.getLogger(InterfazUsuario.class.getName()).log(Level.SEVERE,
null, ex);
        }
        if(info==false){
            JOptionPane.showMessageDialog(this,"Error de consulta,
favor de insertar parámetros correctos.","Información del Sistema",2);
        }
    }
```