

TALLER**RAZONAMIENTO
SEMÁNTICO**

Profesor: Ingrid Durley Torres & Jaime Alberto Guzmán Luna

Contenido del taller:

1. Introducción
2. Componentes de OWL
 - Clases
 - Properties
 - Instancias
3. Probando el Razonador

En términos simples, las ontologías en la web, permiten describir con exactitud la información de algún recurso web y las relaciones entre la información de este recurso con otro u otros. El *Web Ontology Language* OWL, está diseñado para ser utilizado por aplicaciones que necesitan procesar el contenido de la información en lugar de presentar la información a los seres humanos. OWL facilita una mayor interoperabilidad del contenido WEB a las máquinas.

Protégé, es un editor visual de ontologías OWL, amigable con el usuario. Se basa en un software de uso libre, soportado por una interfaz, a la que se le habilitan pestañas (Tabs). se encuentra disponible en: <https://protege.stanford.edu/products.php#desktop-protege>.

El objetivo de esta práctica es, analizar las principales funcionalidades de Protégé, familiarizarse con su entorno de desarrollo. En consecuencia, se abordarán en el mismo orden:

- La visión general de la herramienta
- la descripción de las diferentes ventanas y opciones
- El desarrollo de un ejemplo sencillo.

INTRODUCCION

Como usarlo?

La interfaz de protégé, permite, abrir y crear proyectos. “Abrir (Open)”, significa, cargar una ontología, ya sea recientemente accedida en la interfaz de protégé. Cargarla desde una URL, disponible en la web. O desde, una de las carpetas propias de nuestro computador. Crear un proyecto en Protégé (new), consiste en el desarrollo de una ontología o estructura de conocimiento semántico. Los elementos que se pueden ir creando son fundamentalmente clases, properties, instancias y consultas, aunque la herramienta es modular y permite adicionar más componentes de una forma sencilla. Cada uno de estos elementos dispone de una

etiqueta en la ventana principal de las herramientas, seleccionando cada una de ellas podemos elegir el tipo de elemento concreto sobre el que se va a trabajar.

Una vez iniciado un proyecto (abierto o creado), se puede pasar a modificar elementos del mismo como clases, slots o demás. Una vez realizados los cambios necesarios, o bien, de forma periódica se puede grabar el proyecto abierto seleccionando la opción “Save” del menú “Project”.

Creación de Ontologías

Una vez se ha seleccionado “new”, lo primero que se debe definir es la URI, con la se identificará la ontología. Aunque protégé, provee una por defecto, normalmente esta es modificada para ser mejor adaptada al desarrollo. Para este ejemplo trabajaremos una ontología sobre robots, de manera que nuestra IRI podría ser <http://www.all-robotics.com/>, quedando así.

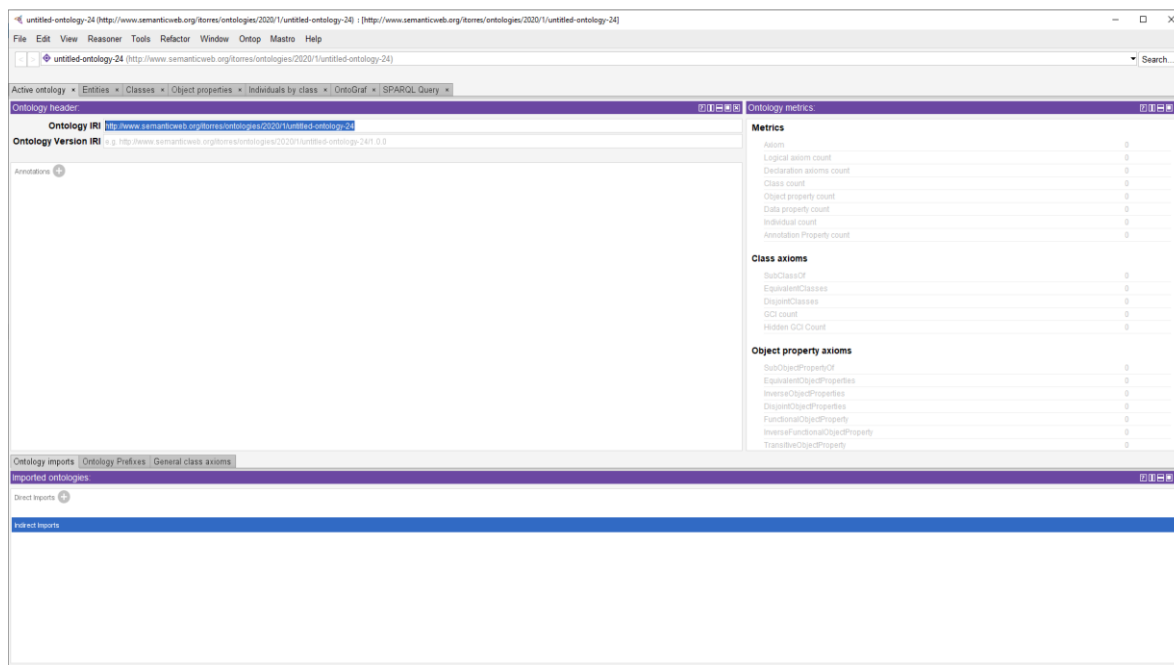


Figura 1. Interfaz protégé

Posteriormente, para actualizar y salvar la ontología, se debe seleccionar la opción “save”, e inmediatamente, aparece una ventana donde nos permitirá identificar el tipo de formato en que se salvará. la recomendación es usar RDF / XML, ya que este es el formato más estable para trabajar en Protégé. Pero siempre se puede optar por exportar el archivo en uno de los otros formatos más adelante. Inmediatamente después, es posible elegir la ruta de la carpeta dónde se debe guardar la ontología (archivo owl). Para ello, una vez se seleccione la ruta, se puede ubicar el nombre “Robotica”.

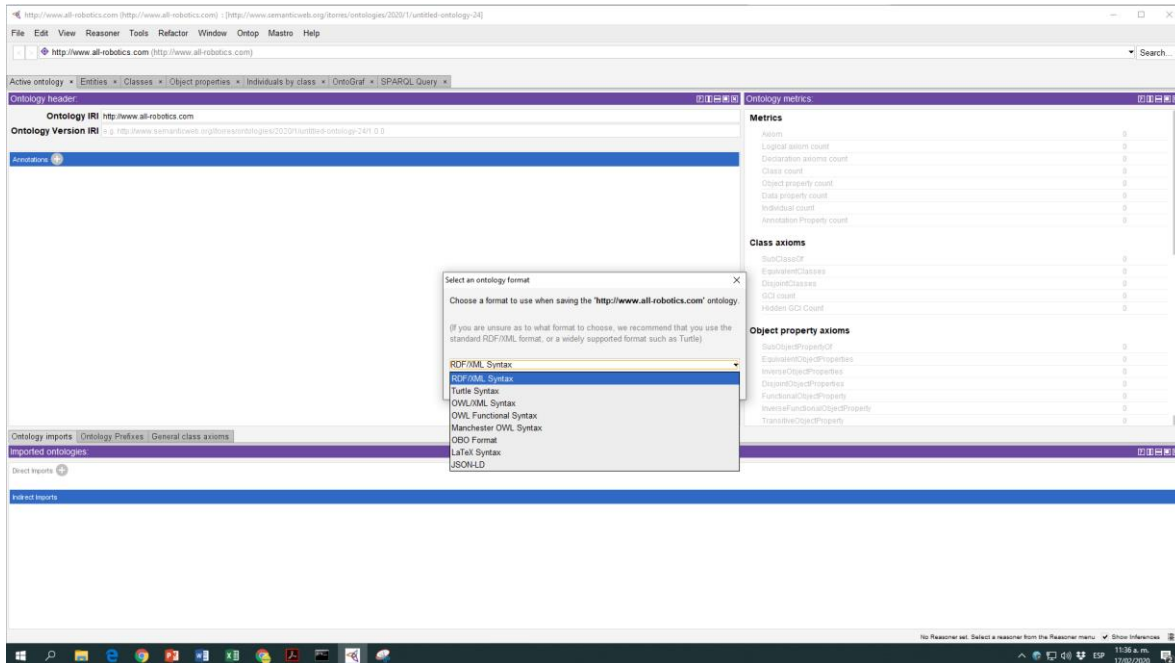
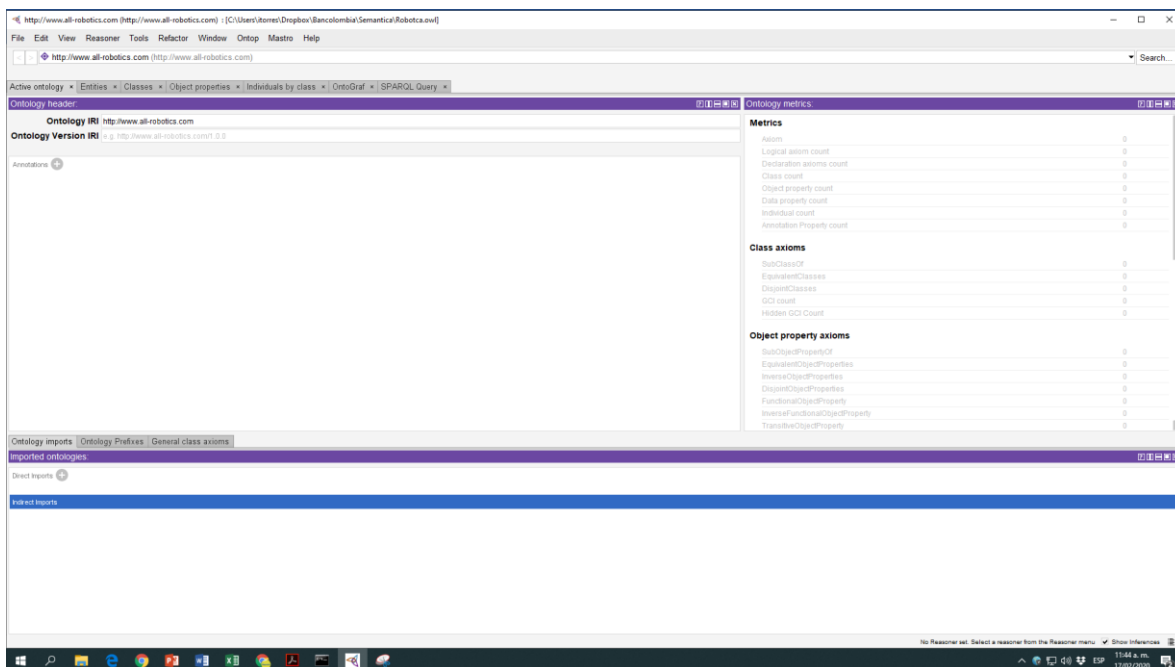


Figura 1. Formato de ontología

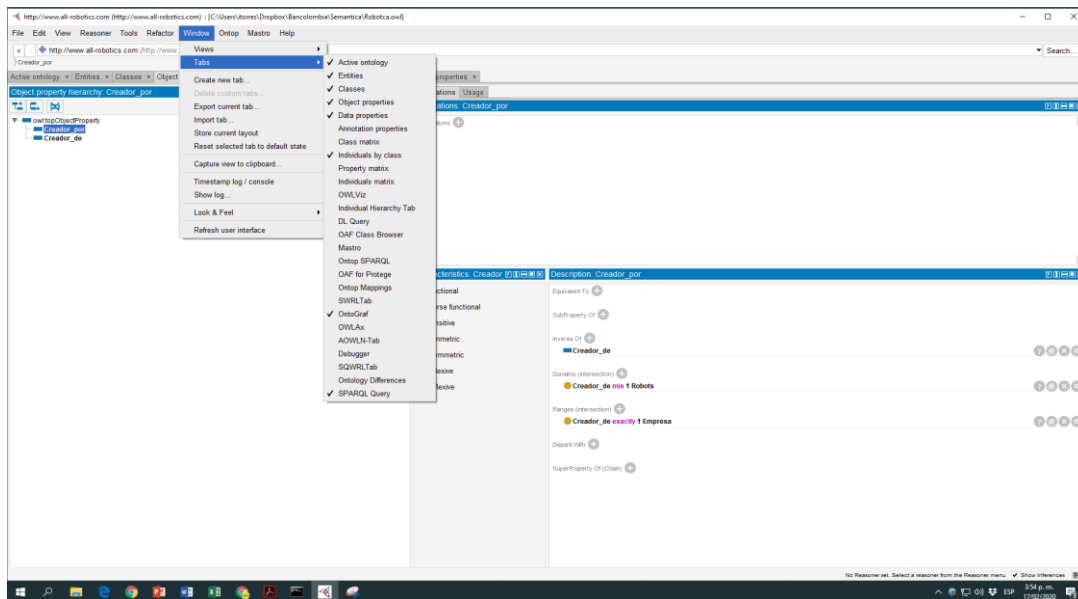
Finalmente tendremos entonces la ventana principal de Protégé, donde podremos crear todo lo que corresponda a nuestra ontología.




COMPONENTES DE UN OWL:

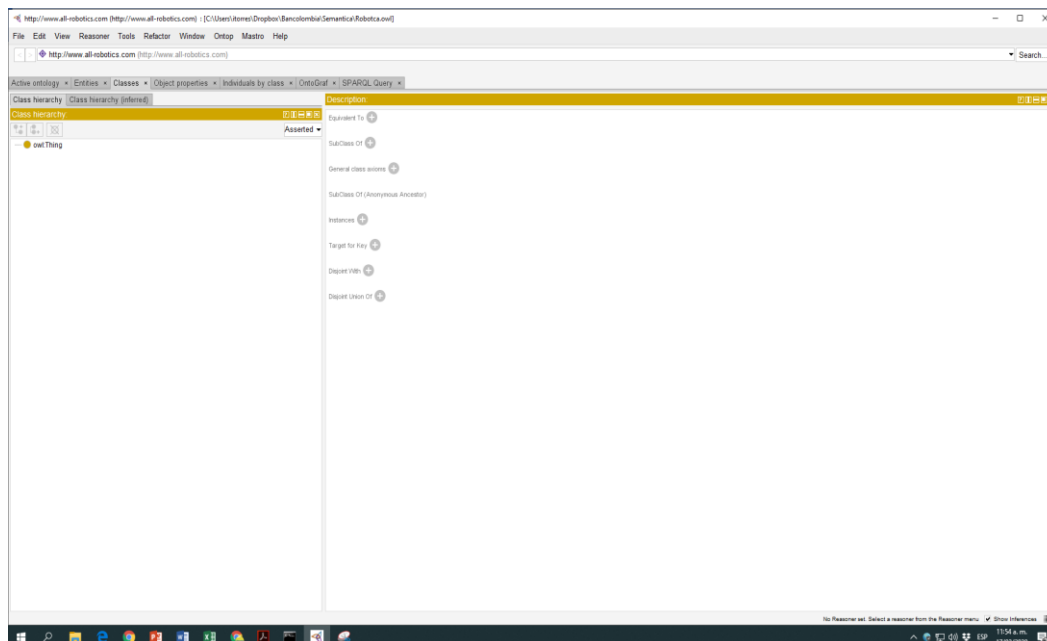
Un archivo OWL tiene varios componentes principales; en la práctica se abordan algunos de los más importantes.

Si por alguna razón, no aparece el correspondiente Tab, ingrese a Window y en la opción Tab, active ✓ el que se requiera.

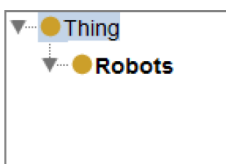


CLASES: Las clases son el componente principal de las ontologías, son las que definen de algún modo los tipos de elementos de los que vamos a describir y a utilizar sus relaciones en nuestra ontología. Para crear una clase al interior de la ontología en Protégé es muy sencillo. En la pestaña llamada *Classes*, nos ubicamos sobre la clase *thing*. De la clase *thing* descenden todas las demás clases, la clase *thing* en OWL sería algo así como la clase *object* en Java. Puestos allí, damos clic

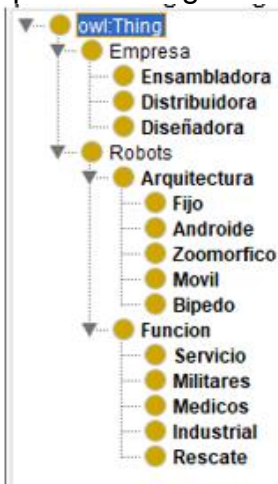
en el botón , el cual creará una clase hijo de *thing*.



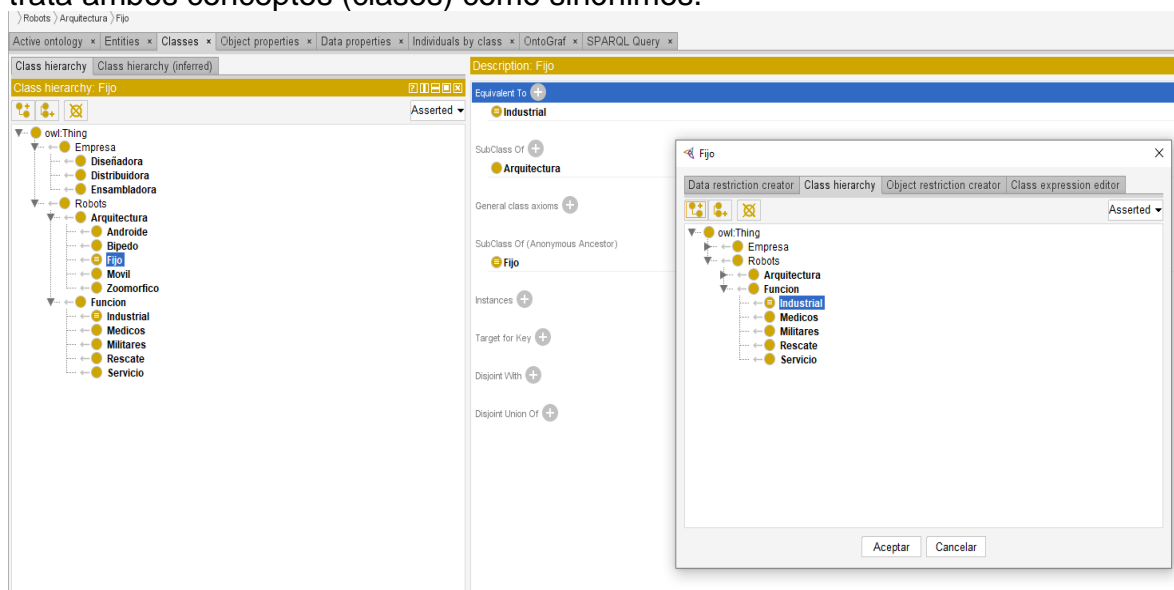
Una vez hecho esto, se nos pedirá el nombre de esta clase, le pondremos Robots: Al hacer clic en aceptar, tendremos una estructura como la siguiente:



Actividad 1: Crear las clases necesarias al interior del documento OWL, de manera que se consiga la siguiente estructura:



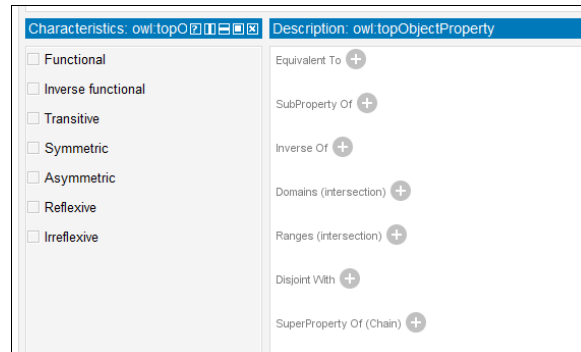
Ahora, comúnmente se considera que un robot industrial es igual a un robot fijo; para expresar tal relación, se usa la expresión “*Equivalent To*”. para ello, por ejemplo, nos ubicamos en la subclase Fijo, y en la ventana *Description*, en la opción “*Equivalent To*”, seleccionamos Industrial. En ese caso el modelo de conocimiento trata ambos conceptos (clases) como sinónimos.




The screenshot shows the Protégé ontology editor. The left pane displays the class hierarchy, with 'Fijo' selected under 'Arquitectura'. The right pane shows the 'Description: Fijo' window, where 'Equivalent To' is set to 'Industrial'. A 'Fijo' dialog box is open, showing the class hierarchy and 'Industrial' selected.

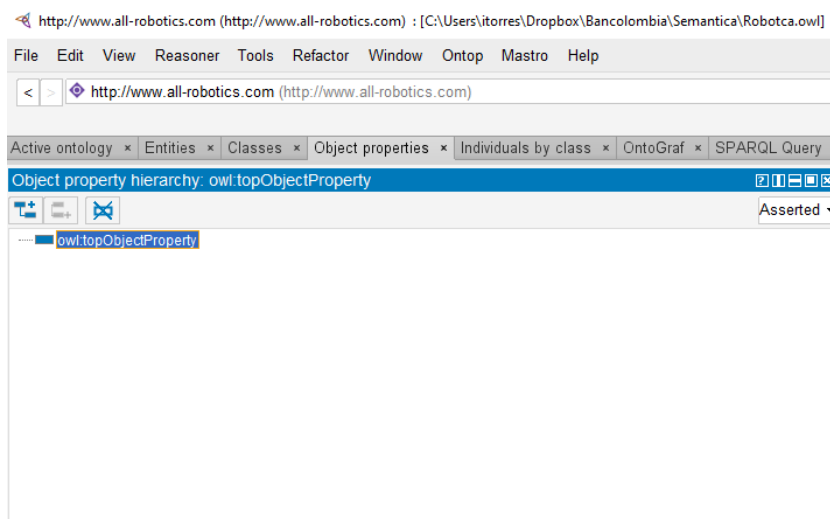
PROPERTIES: Existen dos tipos de *properties*, las *data properties* y las *object properties*. Las primeras son para determinar las propiedades básicas de cada clase (atributos simples - literales). La segunda define las relaciones que existen entre las clases, es decir, atributos, pero entre tipos de datos complejos.

OBJECT PROPERTIES: Para crear un *object property*, el procedimiento es bastante similar, a clases, pero adicionalmente es necesario añadir un dominio y un rango para dicha propiedad, también es necesario especificar las características de esta propiedad (funcional, simétrica, etc.)



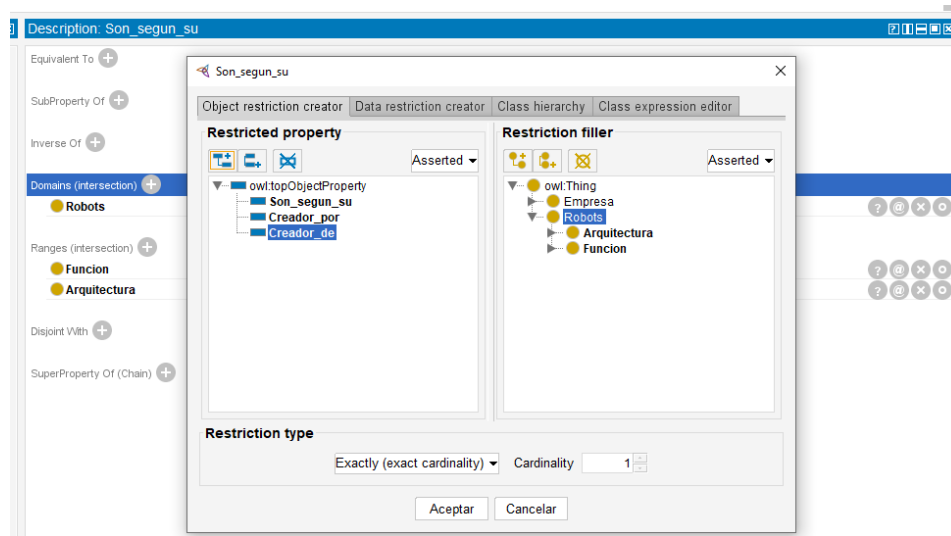
Ahora se crearán, dos *object properties* llamadas *Creador_De* y *Creado_Por*, como se supone, debemos decir que una Empresa es el creador de un robot y a su vez que un robot es creado por una empresa.

Veamos cómo quedaría la primer property *Creador_De*: Hacemos clic en el botón  , el cual nos pedirá el nombre de la property así:

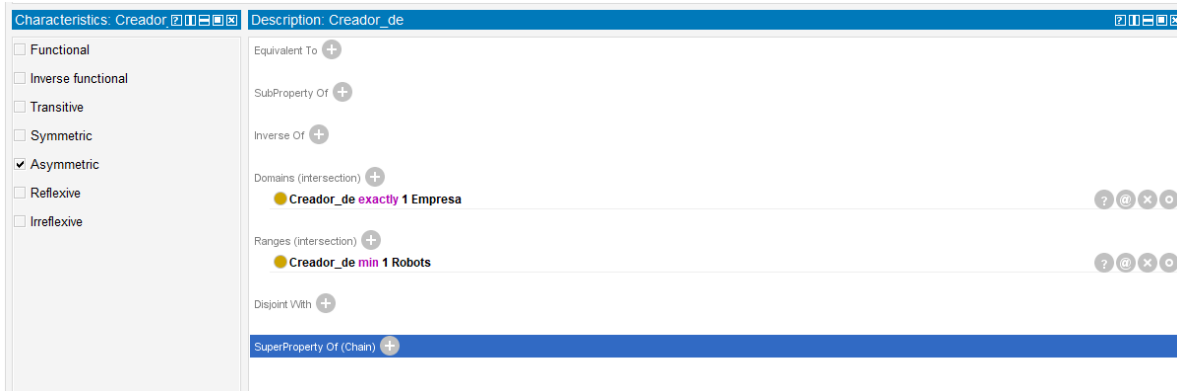


Adicionalmente, a ubicar el nombre de la *property*, si se desea ubicar una restricción, es necesario seleccionar la pestaña *Object restriction creator*, para

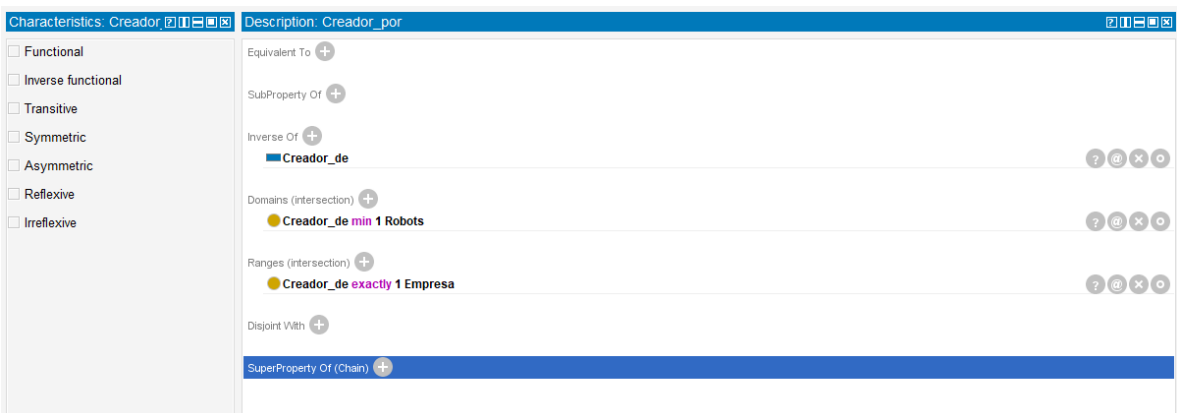
definir la cardinalidad, en este caso, una empresa (exactly 1), debe crear mínimo un robot.




Esta *property* además es asimétrica, dado que la empresa crea el robot, pero el robot no debería poder ser creador de una empresa, quedando así:




Actividad 2: Definir una nueva *Object Property*, llamada Creado_Por, definiendo correctamente su dominio y rango y sus características. Adicionalmente, una vez creada esta *property*, definirle que la *property* Creador_De, es inversa a ésta

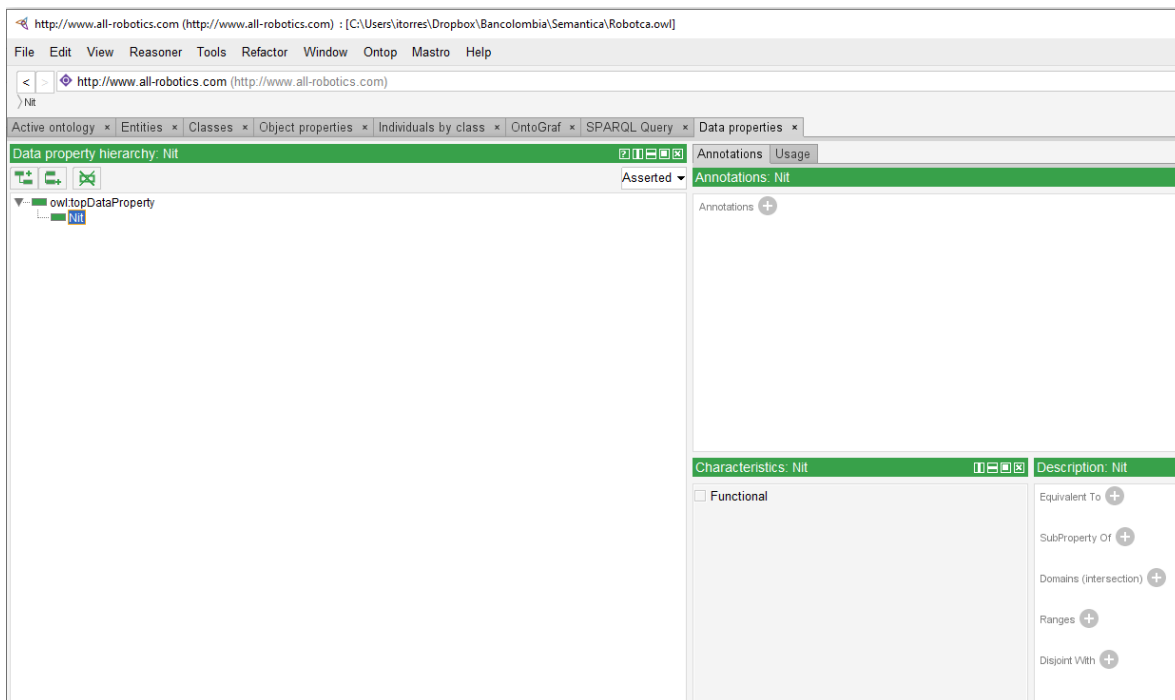


Nota: Para definir una property inversa, se usa el botón 

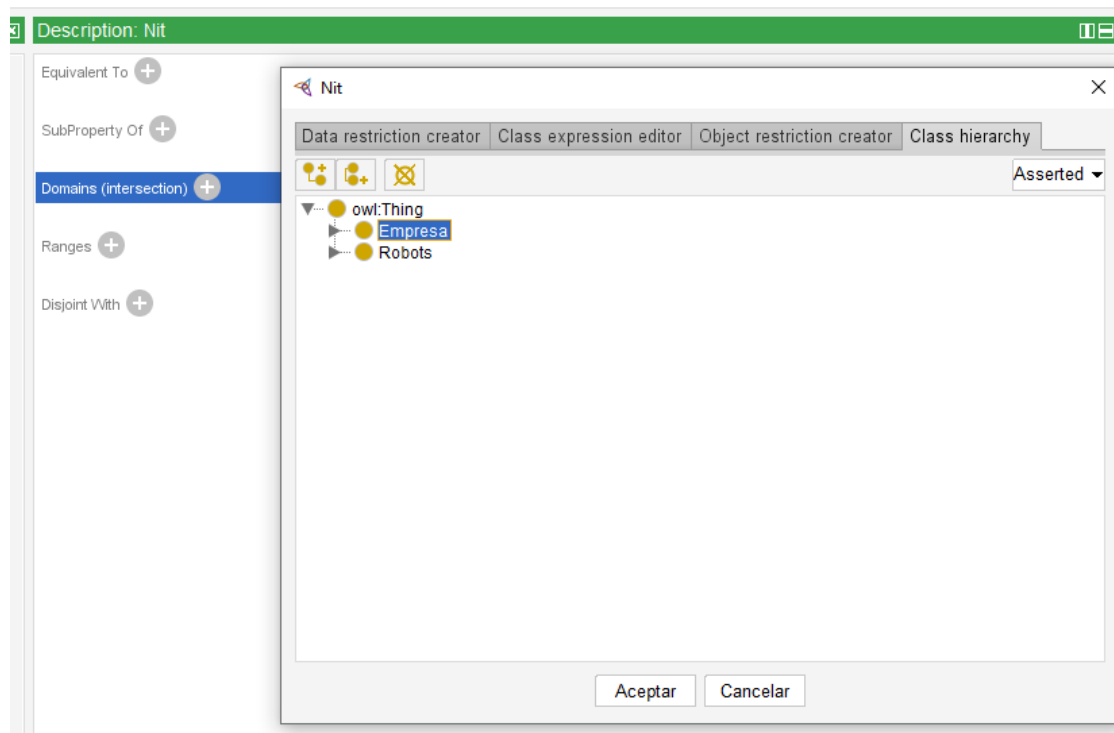
DATA PROPERTIES Para crear una data property, es bastante simple y similar al object properties, con la diferencia de que el dominio es una clase y el rango un tipo de datos básico.

Vamos a crear una data property, llamado Nit, la cual corresponde a las empresas y es de tipo int; para hacerlo, vamos a la pestaña *Data Properties* y damos clic en

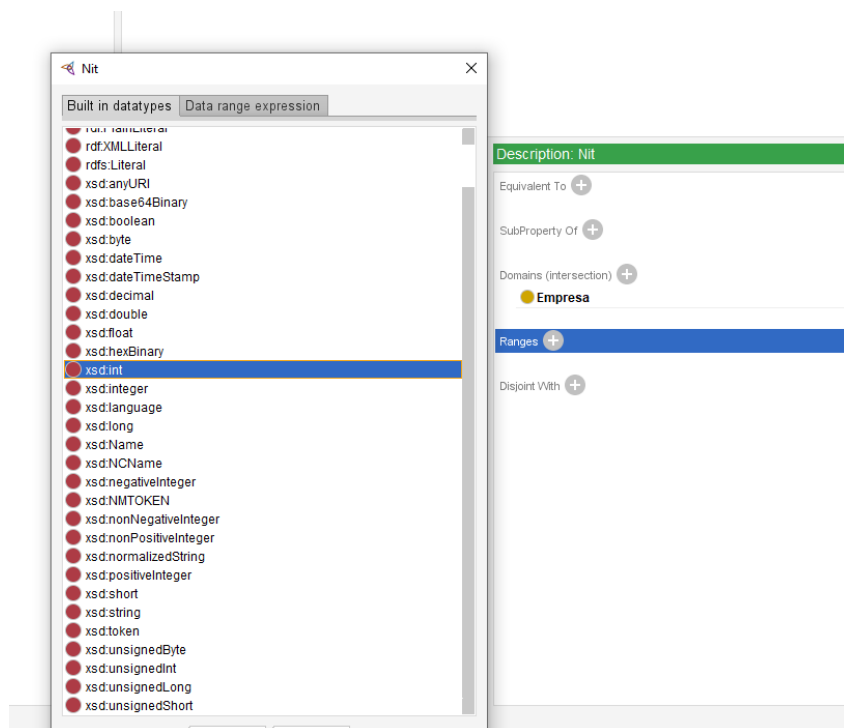
el botón , allí se nos pedirá el nombre:



Con el Nit que se acaba de crear, seleccionado, en la ventana (de esa misma interfaz) *Description: Nit*, se añade el dominio y el rango, de manera equivalente al *Object property*.




Nota: El dominio se selecciona desde la pestaña *Class Hierarchy* y el rango desde la pestaña *Build in DataTypes*.




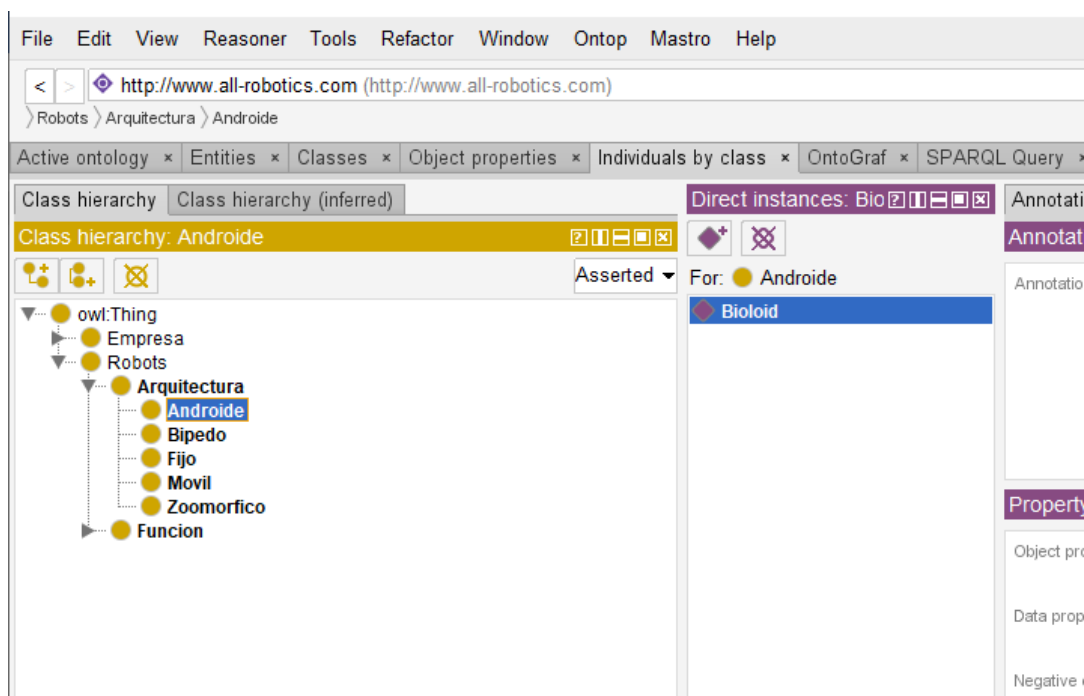
Actividad 3: Se desea crear los *Data Types* restantes que sean necesarios para que una empresa tenga Nit, Telefono, Direccion y Nombre y otros datos para identificar los Robots, como: Nombre, Peso e Identificación.

Nota: Tenga en cuenta que un property puede tener varios dominios.

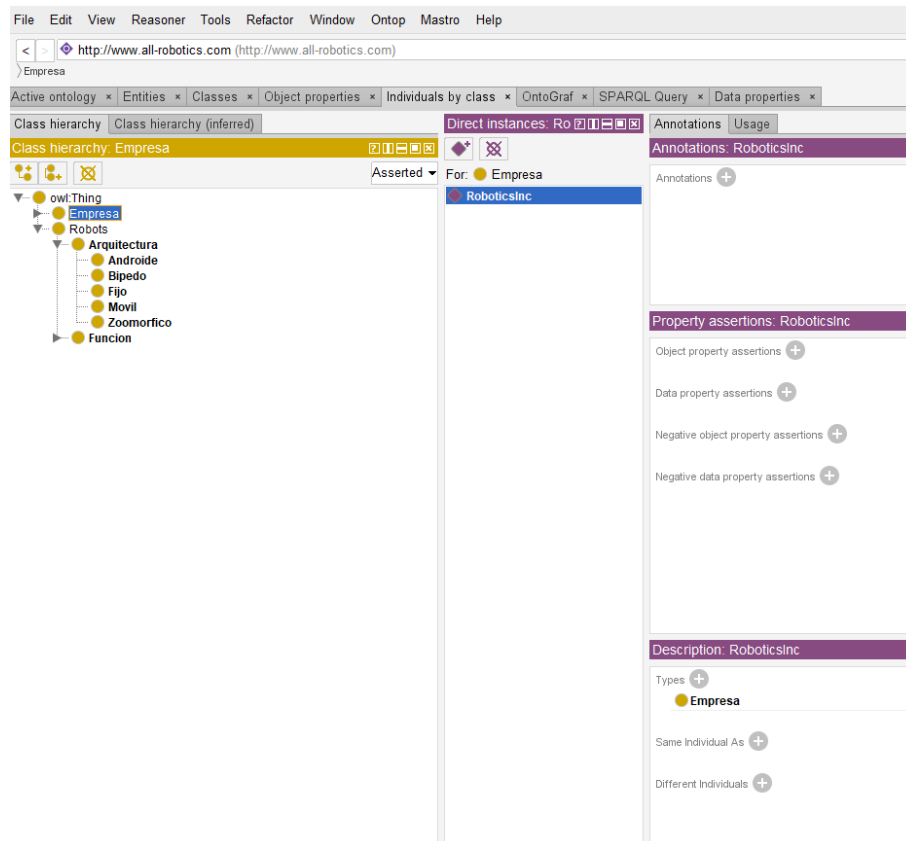
CREACIÓN DE INSTANCIAS. Los *individuals* se pueden entender como las instancias de las clases, su creación es bastante simple:

Primero, es seleccionar la pestaña *Individuals by class*, luego seleccionar la correspondiente clase, e inmediatamente seleccionar el botón  en crear *Direct instances*.

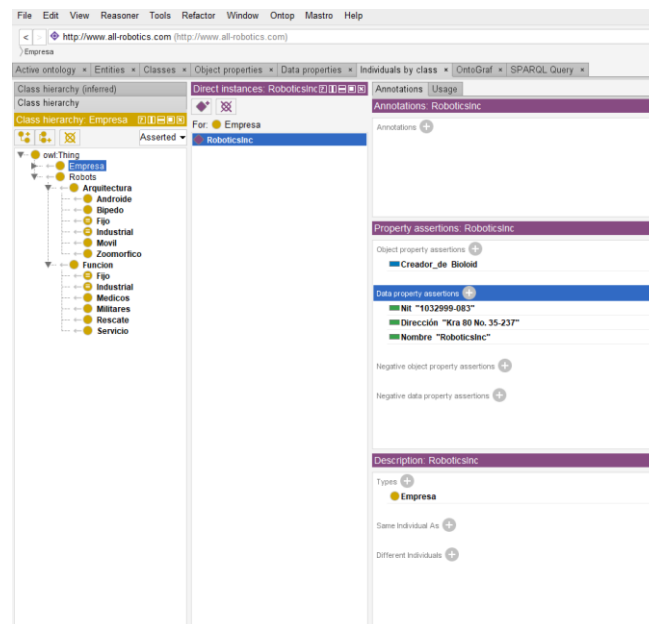
Para crear un Individual, llamado Bioloid el cual va a corresponder a una instancia de la clase Bípedo y otro llamado RobotsInc, del tipo empresa. Para esto, hacemos clic sobre la clase Bípedo y posteriormente sobre el botón . Allí se nos pedirá el nombre del individual.



Procedemos de manera equivalente, pero ahora desde la clase empresa para crear RobotsInc:



Una vez creado, podremos definir las instancias de las *Data Properties* y *Object Properties* definidas anteriormente quedando básicamente así:

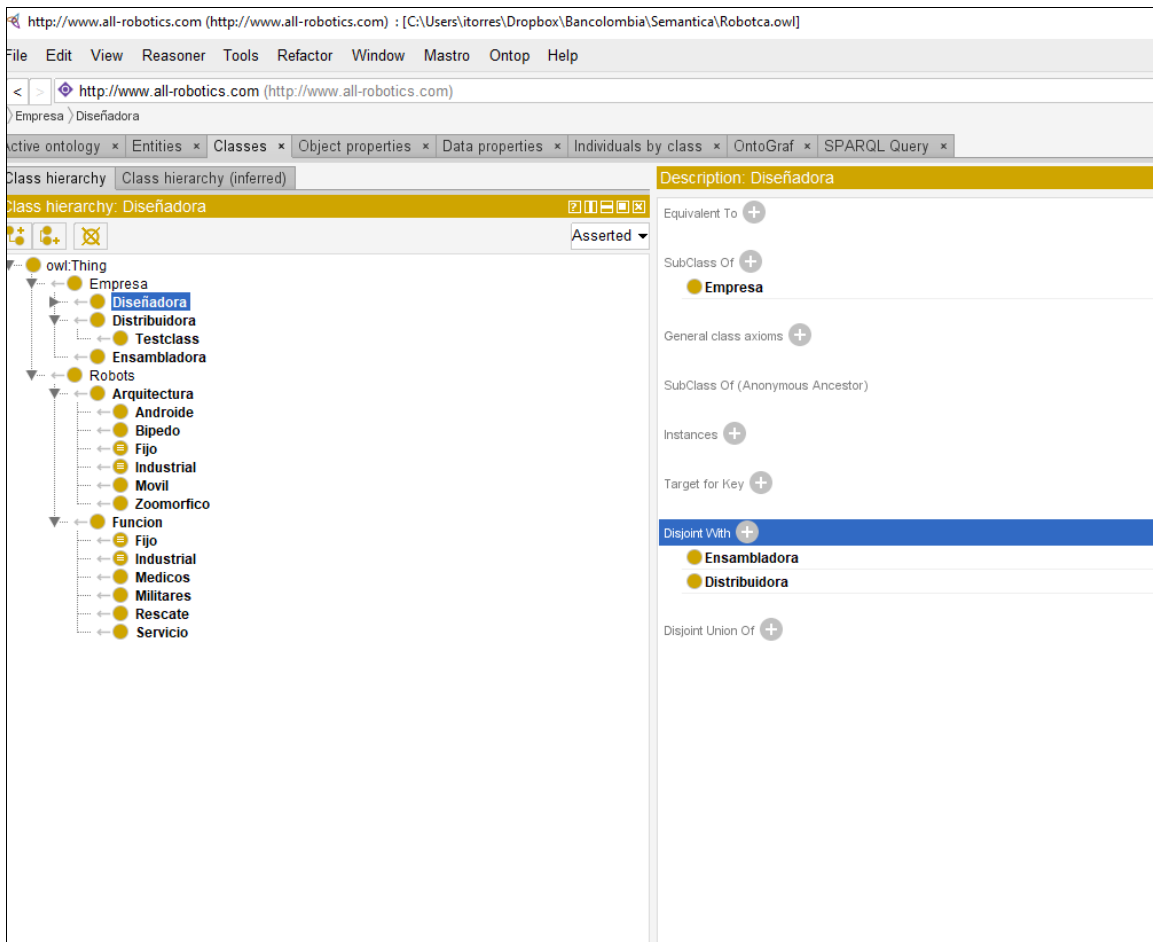


Actividad 4: Crear las siguientes Object_Properties faltantes de manera que cumplan con las siguientes tripletas:

- Empresa Diseñadora es Diseñadora de Robots
- Empresa Ensambladora es Ensambladora de Robots
- Empresa Distribuidora es Distribuidora de Robots
- Robots Diseñados por empresa Diseñadora
- Robots Ensamblados por empresa Ensambladora
- Robots Distribuidos por empresa Distribuidora

Actividad 5: PROBANDO EL RAZONADOR

Sobre la clase empresa, se selecciona la subclase Diseñadora, y desde la vista *Diseñadora:Description*, en la opción *Disjoin with*, se seleccionaran Ensambladora y Distribuidora, de tal manera que se advierte que un elemento (subclase o instancia) de la empresa Diseñadora, no puede ser Distribuidora y ensambladora a la vez. es decir, solo puede ser de un único tipo.

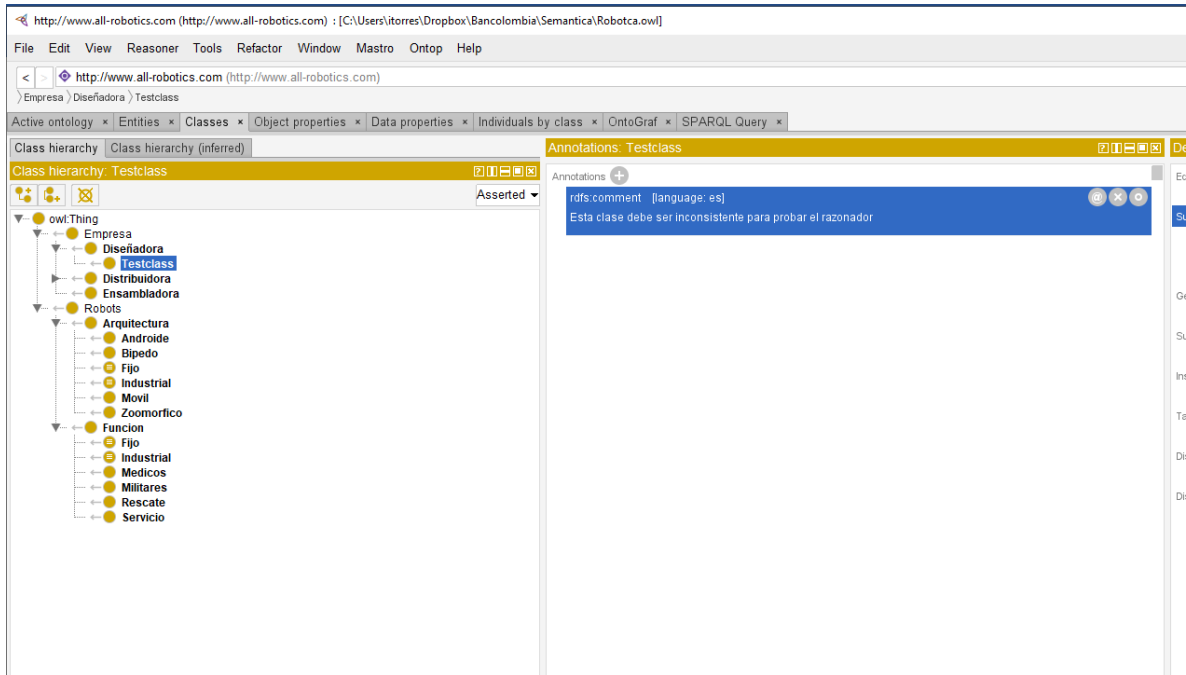


The screenshot shows the Protégé ontology editor interface. The left pane displays the class hierarchy, with 'Diseñadora' selected under the 'Empresa' class. The right pane shows the 'Description: Diseñadora' tab, where the 'Disjoint With' section is active, listing 'Ensambladora' and 'Distribuidora' as disjoint classes from 'Diseñadora'.

Lo mismo se hará desde las otras clases, se adiciona la correspondiente disyunción. Ahora seleccionando la clase Diseñadora, se creará una subclase que denominaremos *Testclass*, y nuevamente en la vista *Description:Diseñadora* en la opción *subClass Of*, se adicionará *Distribuidora*, con ello, estaremos afirmando que

esta clase, actúa con los atributos de ambas superclases. Es decir, es *Diseñadora*, a la vez que es *Distribuidora*.

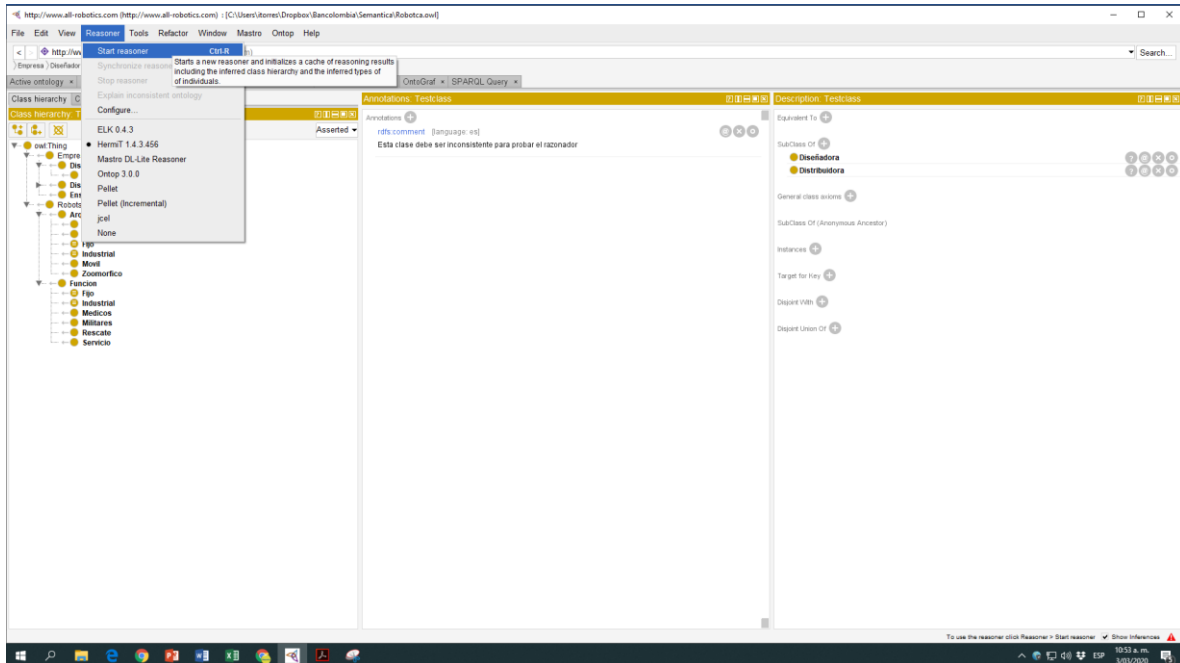
Ahora en la ventana de *Anotations:Testclass*, se agrega el siguiente comentario “Esta clase debe ser inconsistente para probar el razonador”.



Ahora en la pestaña *Rasoner*, se activará uno de los razonadores, puede ser:

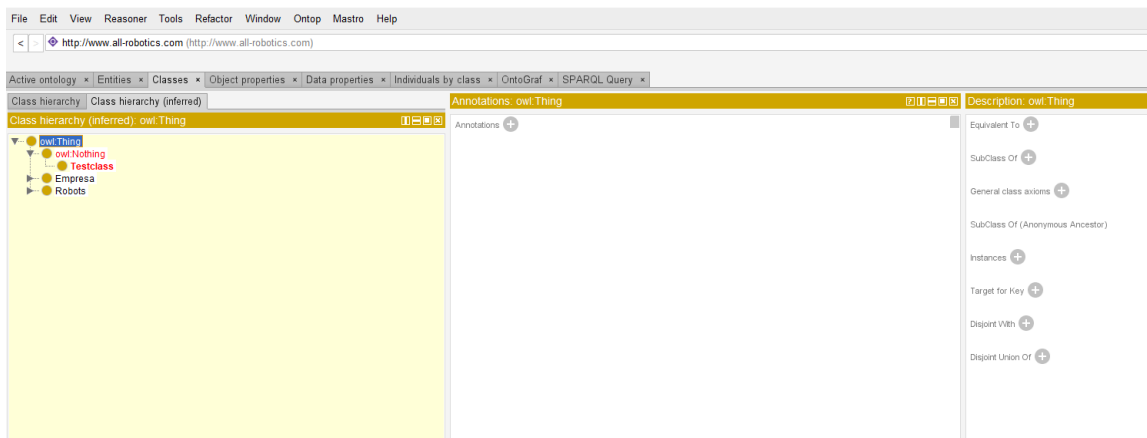
- HermiT 1.4.3.456 ó
- Pellet

Una vez seleccionados, daremos click en *start rasoner*.



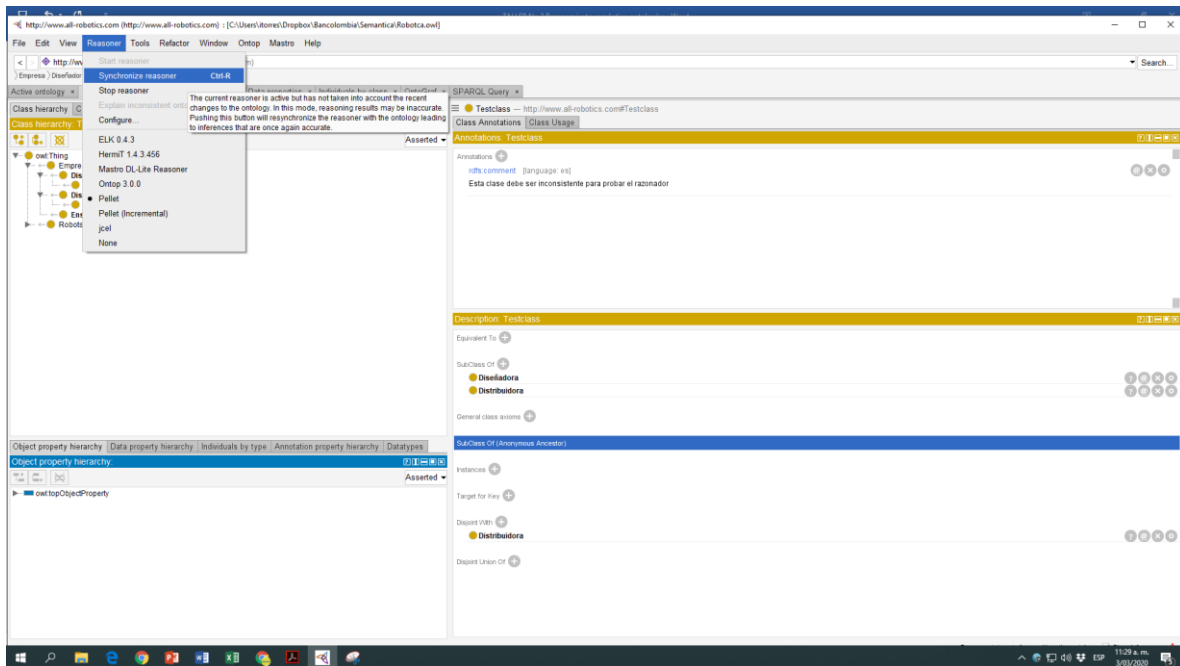
Inmediatamente, se verá como la subclase *Testclass*, que habíamos creado para comprobar la eficacia del razonador, es señalada en color rojo, advirtiendo la inconsistencia, originada por la restricción de Disyuntas definida en las super clases.

Para revisar su funcionamiento se selecciona la ventana *Class heranrchy (inferred)*, donde se identificará las inferencias generadas por el razonador y el color rojo que identifica la advertencia. ¿Por qué pasó esto? Intuitivamente y explícitamente sabemos que algo no puede al mismo tiempo Empresa distribuidora y Diseñadora y/o Ensambladora.



Ahora, elimine la declaración disyunta entre *Distribuidora* y *Ensambladora* para ver lo que pasa.

Una vez hecho esto, se selecciona en la ventana de *Rasoner* y se hace click en “*Synchronize rasoner*”.



Material de apoyo:

- Características de OWL [<http://www.w3.org/TR/owl-features/>]
- Tutorial Protégé [<http://owl.cs.manchester.ac.uk/publications/talks-and-tutorials/protg-owl-tutorial/>]