2023/2024 2Q

Colección de Soluciones

ECSDI

Javier Béjar

Departament de Ciències de la Computació

Grau en Enginyeria Informàtica - UPC

UPC FIB Facultat d'Informàtica de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA

En l'elaboració d'aquesta col.lecció de problemes han participat els professors:

Javier Béjar Alonso
Javier Vázquez Salceda

FACULTAT D'INFORMÁTICA DE BARCELONA
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Primera edición, Febrero 2013
Esta edición, Febrero 2024

Índice general

CAPÍTULO 1

Diseño del Sistemas Multiagente

1.1. Problema 6

Queremos implementar un sistema distribuido capaz de gestionar el proceso de petición de reparaciones de coche a aseguradoras por parte de sus usuarios. Tenemos que diseñar un sistema que haga de intermediario en el proceso que involucrará tres actores externos, los usuarios, las aseguradoras y los talleres de reparación. El proceso es como sigue:

> El proceso se inicia cuando un usuario informa al sistema de un parte de reparación por accidente indicándole todos los datos necesarios (datos del vehículo, compañía aseguradora, taller preferido). El sistema contacta con la compañía aseguradora para obtener la información sobre las condiciones de la póliza del usuario (a todo riesgo o con franquicia), los talleres que prefiere la compañía para la reparación y el presupuesto máximo que cubrirá.
> Para obtener presupuestos diferentes, el sistema contacta con como máximo tres talleres escogidos de entre los indicados por la compañía y el taller preferido del cliente (si no está entre los talleres de la compañía). Se recopilan los diferentes presupuestos y se contacta con la compañía y el cliente para que indiquen sus preferencias. Si la decisión de aseguradora y cliente no coinciden, se da preferencia a la del cliente. Si este ha elegido el taller que no es de la aseguradora y el presupuesto excede el máximo propuesto por ésta, se registra que la diferencia la pagará el cliente. El sistema contacta con el taller aceptando el presupuesto y con el cliente para que lleve a reparar el coche.
> Cuando este es reparado, el taller lo notifica al sistema, junto con la factura final. El sistema determina como se ha de pagar. Dependiendo del tipo de póliza irá todo a cargo de la compañía o una parte la pagará esta y otra el cliente. Si el taller que ha elegido el cliente es el suyo, le tocará pagar lo que exceda del presupuesto máximo. El sistema enviará los datos de los pagos a aseguradora y cliente y cuando estos confirmen que han hecho el pago, informará del pago al taller.

Asumiremos que los actores externos (usuarios, aseguradoras, talleres) se comunican con el sistema a partir de percepciones y acciones.

a) Especificación del sistema

- Identifica los escenarios del sistema y describe detalladamente al menos dos de ellos (objetivos/ percepciones/acciones/roles).
- Determina cuáles son los objetivos del sistema y como estan organizados y describe brevemente los que no sean autoexplicativos.

- Identifica los roles del sistema, descríbelos brevemente y detalla para dos de ellos qué objetivos, percepciones y acciones tienen asociados.

b) Diseño arquitectónico

- Identifica los agentes del sistema e indica qué roles tienen asignados.
- Identifica y describe brevemente las fuentes de datos del sistema e indica qué roles tienen acceso a ellas.
- Describe los protocolos de interacción entre los agentes, detallando qué agentes interactúan, los mensajes que se intercambian y su secuencia.
- Detalla para uno de los agentes sus percepciones, acciones y protocolos.

c) Diseño detallado

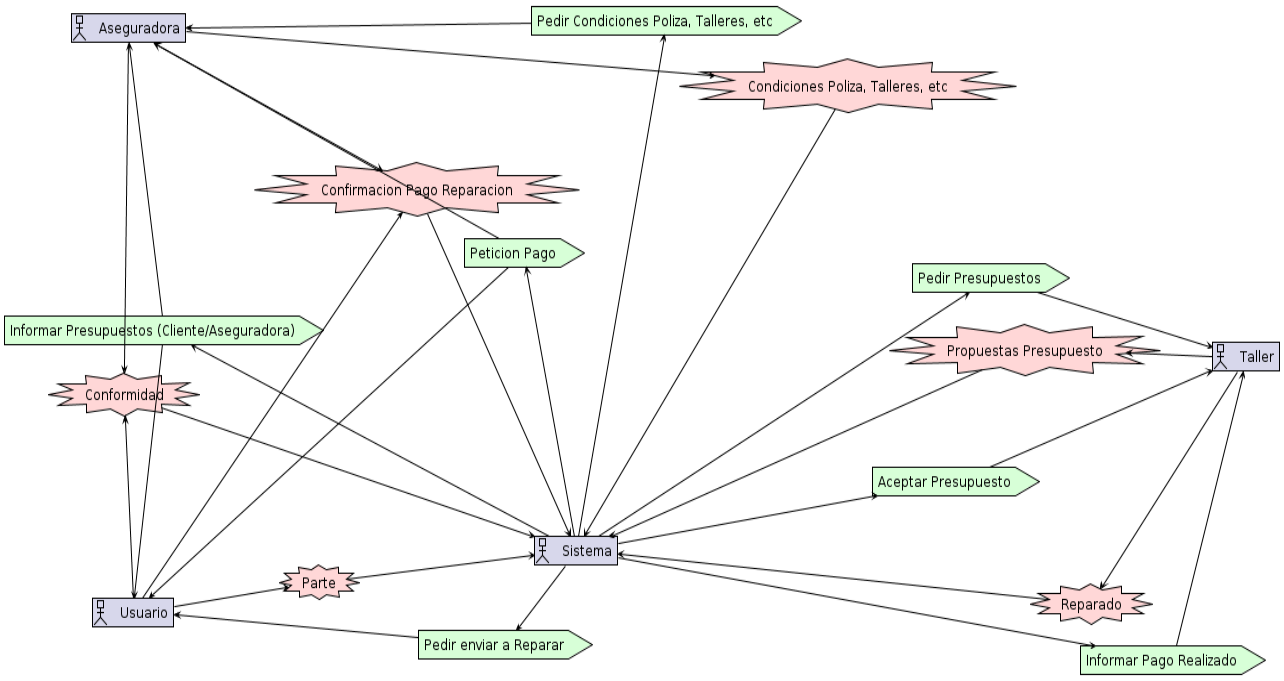
- Describe brevemente como se dividen los agentes en capacidades e indica también brevemente qué planes contienen y qué hacen.
- Detalla para el agente más simple de los que has diseñado qué percepciones, mensajes, acciones y acceso a fuentes de datos tiene cada uno de sus planes.

1.1.1. Solución

Esta es una posible solución, hay decisiones que se pueden hacer de manera diferente, el objetivo principal del ejercicio era mostrar que se había entendido la metodología, se sabía lo que había que hacer en cada fase y que significaba cada abstracción de la metodología.

Especificación del sistema

Este podría ser el *analysis overview*, tenemos cuatro actores implicados, nuestro sistema, el usuario, la compañía aseguradora y los talleres. Solo aparecen las percepciones y acciones que se intercambian.



Hay diferentes maneras de organizar los escenarios, pero se podría organizar el funcionamiento del sistema en tres fases:

1. Escenario en el que el sistema recibe la petición del usuario y pide a la compañía de seguros la información necesaria para poner en marcha el proceso de reparación.

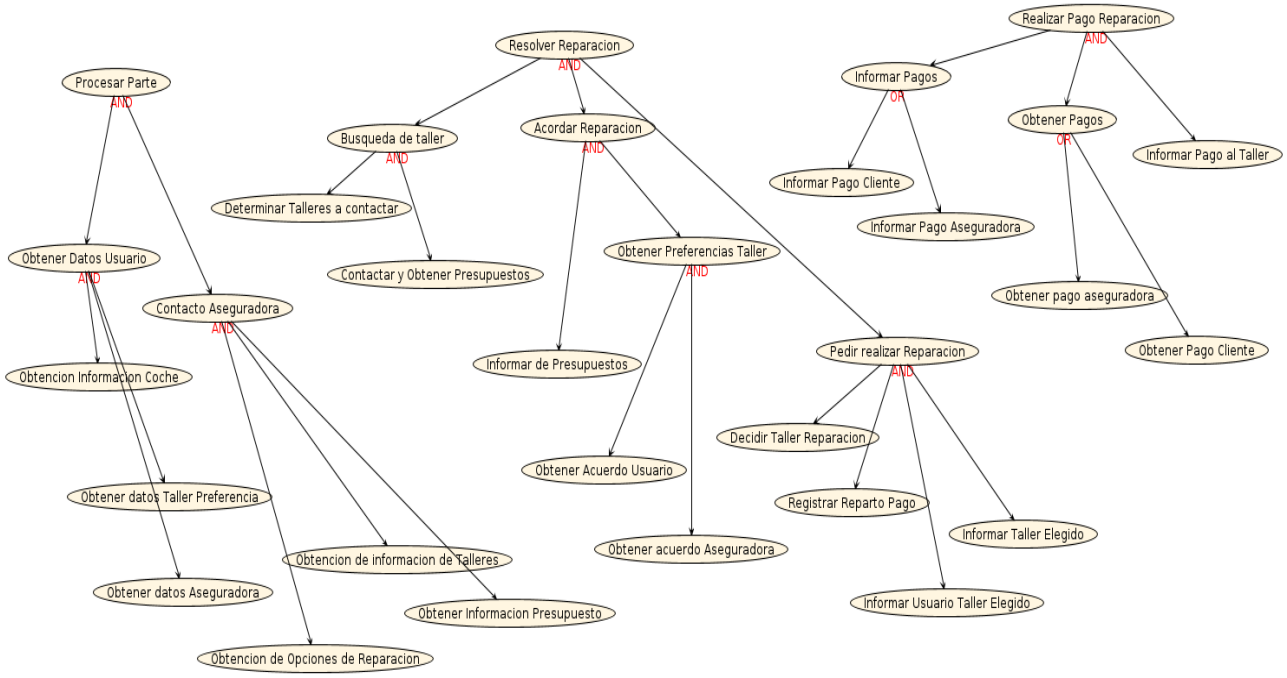
2. Un escenario de gestión de la reparación que se descompondría en dos:

   a) Obtención de los presupuestos de los talleres para poder pedir una decisión a usuario y compañía aseguradora

   b) Paso de la información a usuario y compañía para que tomen su decisión, decisión del sistema sobre el taller, información al usuario y taller de la decisión para que lleve su coche a reparar. También se podría separar el informar de la decisión al taller y al usuario en otro escenario, pero quedaría un poco simple.

3. Un escenario donde se solicite el pago una vez se tiene constancia de que el coche se ha reparado y aviso al taller de que el pago se ha efectuado cuando se reciba la confirmación de los pagos.

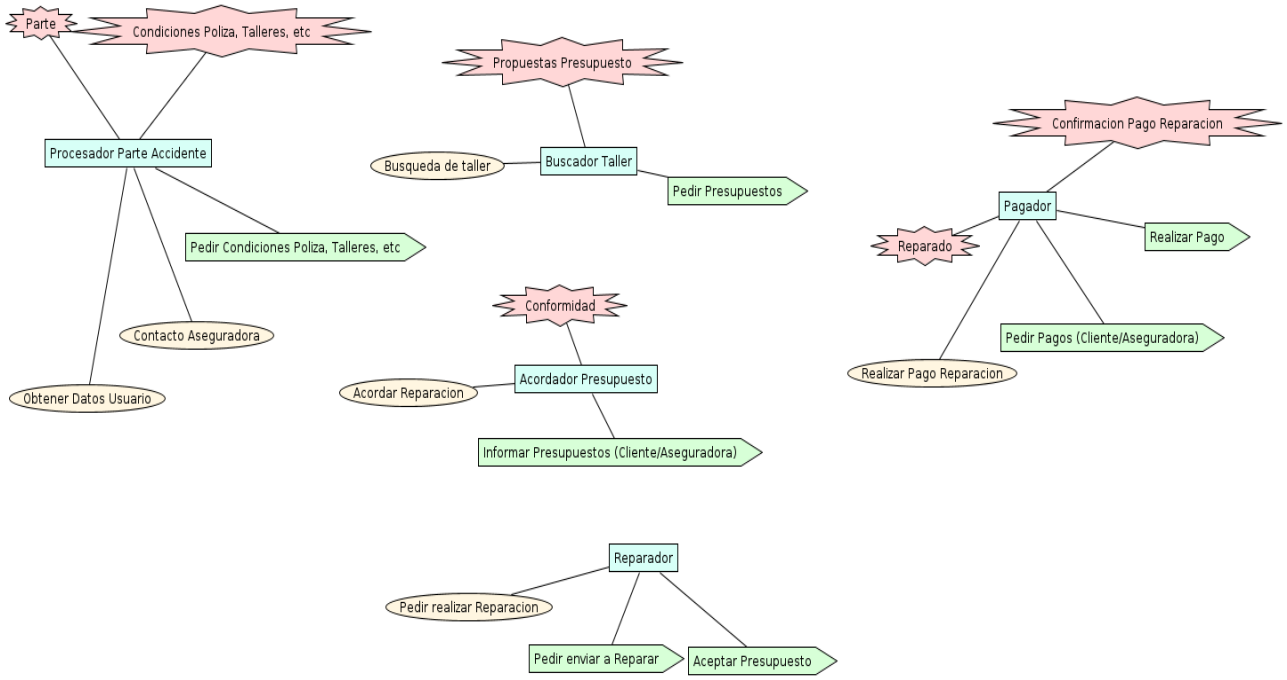
El diagrama de objetivos puede ser más o menos detallado, pero se puede organizar a partir de lo que se resuelve dentro de los tres escenarios principales. Los objetivos más específicos se corresponderían con objetivos que cumplen las acciones hacia el exterior y los planes que después se desarrollarán en el diseño detallado. También se podrían agrupar todos los objetivos principales en un objetivo común que es el que persigue el sistema. Lo que no tendría mucho sentido es tener un montón de objetivos sueltos sin apenas organización.



Es un error común el hacer un diagrama de objetivos como una secuencia de acciones una detrás de otra, el objetivo más general debe incluir a los más específicos. Un objetivo es un problema que se puede descomponer en problemas más sencillos y solo tenemos problemas resolubles a partir de acciones cuando llegamos al nivel de descomposición más bajo, en la metodología estos objetivos vendrían ligados a acciones que realizan los planes. Tampoco tiene mucho sentido que un objetivo se descomponga en un solo objetivo.

Los roles se han decidido en función de los diferentes pasos que se realizan en los escenarios. En el primero tenemos simplemente que pedir y gestionar la información necesaria para procesar el parte de accidente (Procesador parte accidente). En el segundo escenario hacemos diferentes roles, primero recolectamos y gestionamos la información de los talleres para poder acordar un taller de reparación (Buscador Taller), después acordamos la reparación intercambiando información con usuario y aseguradora (Acordador Presupuesto) y por último informamos al usuario y taller de la decisión (Reparador). En el último escenario

nos encargamos de que se informe de los pagos a usuario y aseguradora para que los hagan y se informa al taller cuando se recibe confirmación (Pagador). En el diagrama están todas las percepciones y acciones asociada a cada rol. Los objetivos ya están organizados según los roles de manera que solo aparece asignado el objetivo más general que le corresponde al rol.



No tenia mucho sentido hacer roles de los actores externos, ya que no forman parte del sistema, los roles encapsulan una funcionalidad interna del sistema, no replican las entidades externas. Tampoco tenía mucho sentido hacer un rol sistema que lo hiciera todo.

Los roles se pueden hacer más simples, pero tampoco hay que atomizarlos para que solo hagan algo tan simple como lanzar una acción y recibir una percepción asociada o que solo hagan un cálculo. Roles que agrupen una funcionalidad definida luego son más fáciles de organizar en agentes.

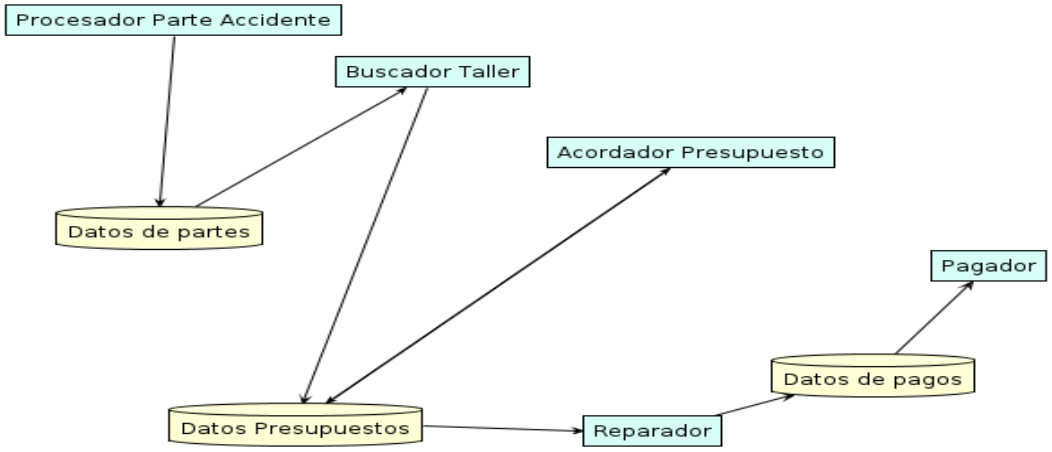
Ahora que tenemos los roles, objetivos, percepciones y acciones podemos detallar los escenarios. Por ejemplo:

- El escenario 1 es iniciado con la percepción de petición de parte que genera el usuario, que pone como objetivo el obtener los datos del cliente, podemos asumir que en esta percepción está toda la información del cliente, si no habría una acción y percepción más para esto. Una vez hemos recolectado la información del usuario pasaríamos al objetivo de contacto con la aseguradora que llevaría a la acción de petición de información a esta seguida de la percepción con la información. Este escenario involucraría el rol de procesador de parte de accidente y los actores usuario y aseguradora.

- El escenario 3 es iniciado por la percepción de que hay un coche que ya se ha reparado que lleva a los diferentes subobjetivos de realizar pago reparación, estos ponen en marcha las acciones que informan a usuario y aseguradora de los pagos a realizar que en algún momento generarán las percepciones de confirmación de pago y que finalizarán el escenario con la acción hacia el taller que informa de que el pago se ha realizado. En este caso está involucrado el rol pagador y todos los actores.

Diseño arquitectónico

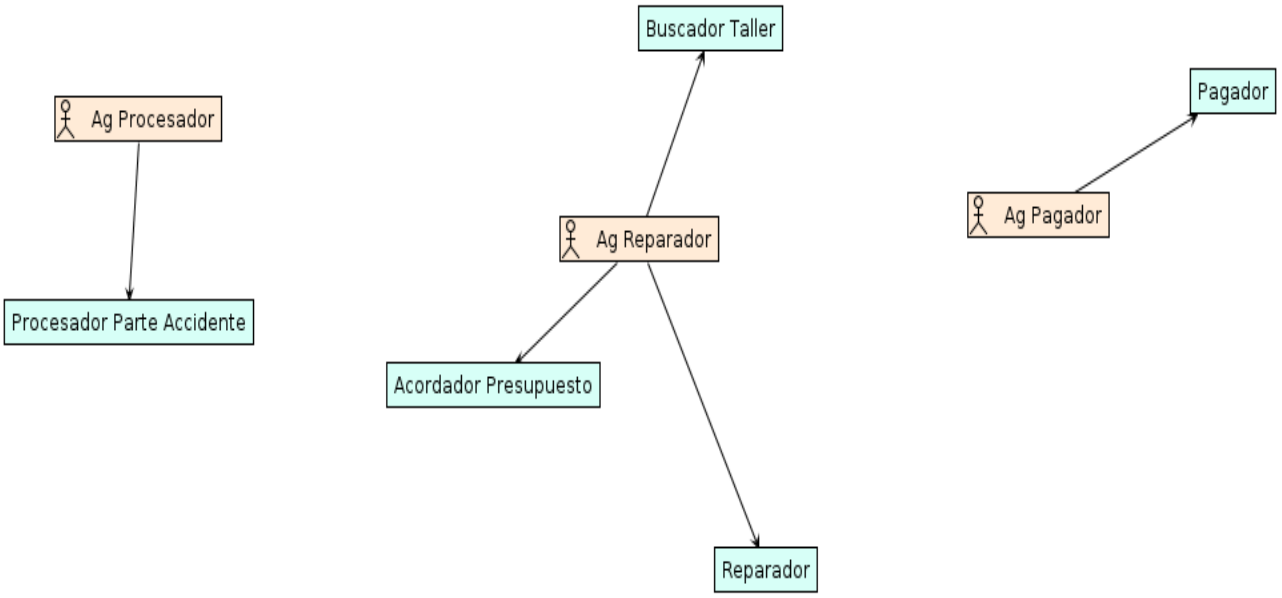
Se pueden escoger diferentes maneras de organizar la información que se ha de tratar. Podemos por ejemplo asumir que trataremos con tres cosas distintas, los partes de accidente y toda la información asociada, los presupuestos de los talleres que usaremos para tomar una decisión sobre la reparación y los datos sobre los pagos. A partir de ahí podemos ser más detallados y guardar otras cosas como fuentes de datos separadas, pero esencialmente toda la información para guardar el estado esta vinculada a esas tres cosas.



La agrupación de roles a agentes en este caso se justifica por lo que se hace en los escenarios, cada agente se encarga de los roles que tiene sentido que vayan agrupados porque se encadenan para cumplir los objetivos de un escenario.

Aquí tampoco tenía sentido hacer un agente sistema que lo haga todo e introducir agentes que correspondan con los actores externos. Aparte de que seria inconsistente con los roles de la especificación del sistema, el diseño arquitectónico sirve para agrupar lo que se hace internamente en entidades coherentes, no es el diagrama del analysis overview.

Si los roles se hubieran atomizado, aquí se podrían organizar de muchas maneras, pero tienen más sentido las que evitan que los agentes tengan que comunicarse información continuamente. Un agente que centralice todas las interacciones con el exterior y que luego distribuya la información a agentes que tomen una decisión simple sería una posibilidad, pero obligaría a tener protocolos con los agentes que toman esas decisiones y constituiría un cuello de botella para el sistema. Tampoco tiene sentido que, si hemos atomizado los roles, cada rol sea un agente, ya que también nos obligaría a una comunicación constante entre ellos para pasarse información que tendría más sentido que estuviera contenida en el estado de un agente que los agrupara.

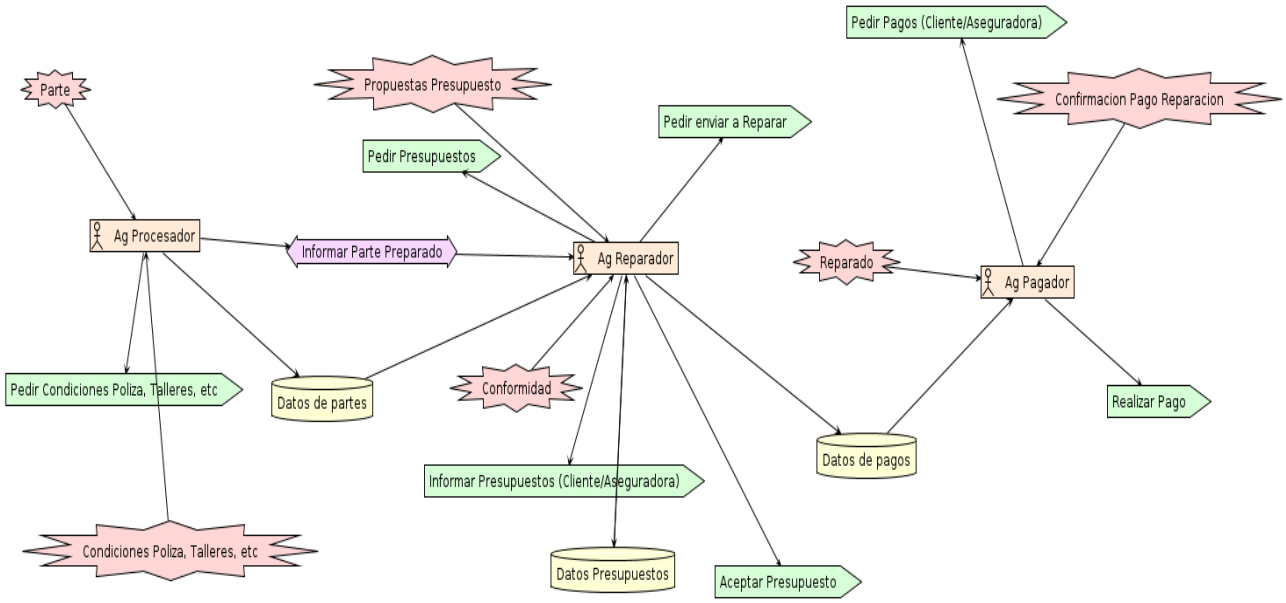


Respecto a los protocolos, no es necesaria mucha comunicación entre los agentes simplemente porque hemos agrupado los roles de cada escenario en un agente. El único sitio donde se justifica la existencia de un protocolo es el momento en el que el escenario que realiza las decisiones sobre la reparación debe iniciarse, eso solo lo sabe el agente procesador, y, por lo tanto, debería informarle al agente reparador. El protocolo sería muy sencillo, simplemente uno le informaría al otro de la referencia del parte sobre el que tiene que trabajar.

Podríamos tener también un protocolo entre el agente reparador y el pagador, por ejemplo para informarle de que se ha tomado la decisión sobre una reparación, pero en este caso es la percepción de que un coche esta reparado la que pone en marcha las acciones del pago y la información del pago esta registrada en una fuente de datos, así que no sería necesario en realidad ese protocolo.

Las interacciones entre los agentes del sistema y los actores externos no se ponen como protocolos para eso tenemos las acciones y percepciones.

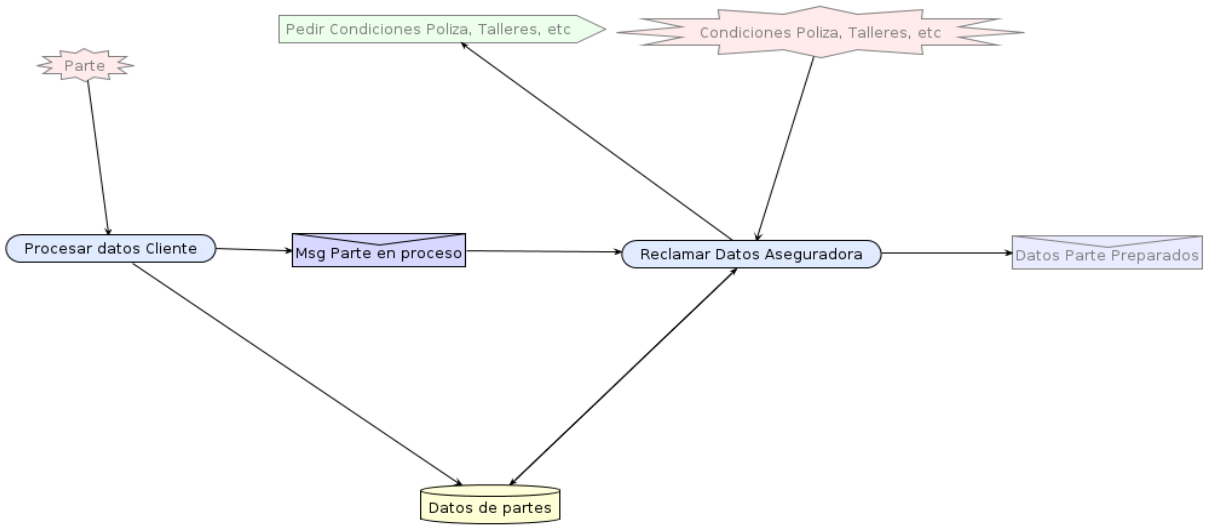
Este sería el *system overview* asignando cada elemento a su agente correspondiente.



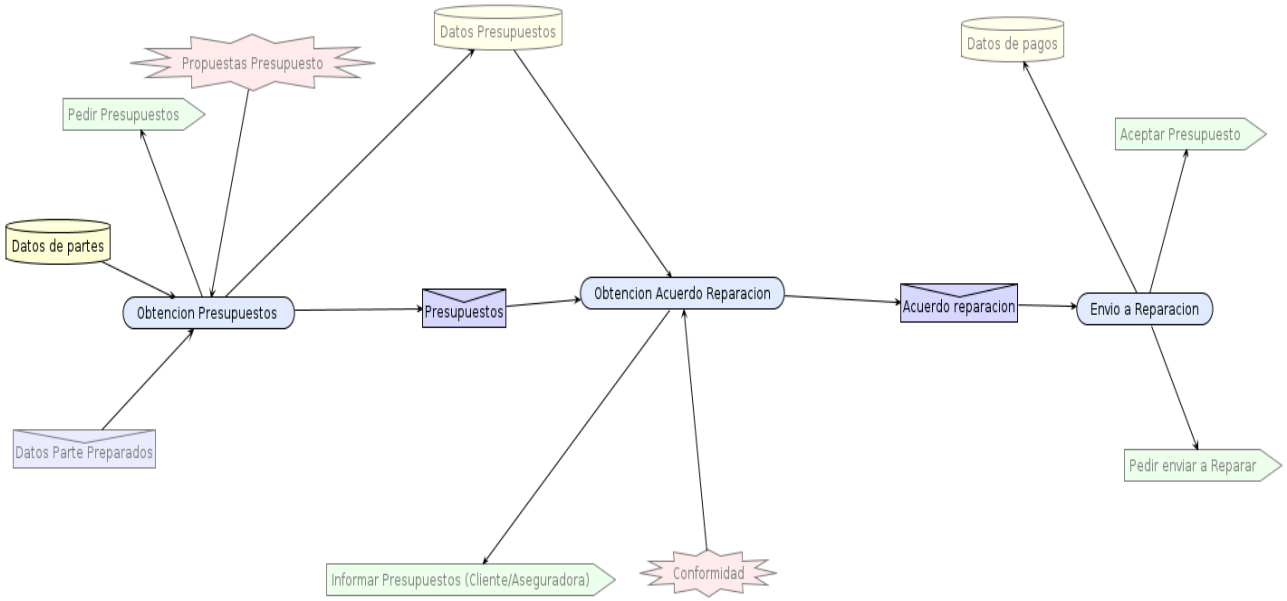
Diseño detallado

En este caso no tiene mucho sentido hacer una división en capacidades, ya que lo que hacen los agentes es suficientemente simple para que sean directamente planes.

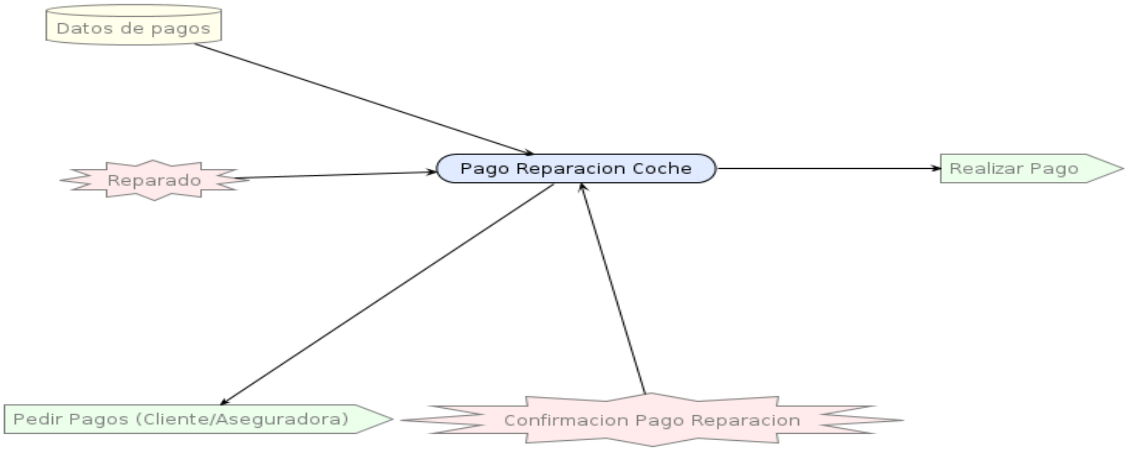
El agente Procesador podría tener dos planes uno que se encargara de obtener y guardar la información del usuario y otro que se encargara de los datos de la aseguradora, un mensaje haría que se comunicaran. También podríamos tener un único plan que lo hiciera todo dado que lo que se hace es suficientemente sencillo, pero de esta manera cada plan cubre uno de los objetivos en los que se divide el objetivo procesar parte.



El agente reparador tiene un plan por cada rol y un mensaje que conecta la cadena de planes. También se justifica porque cada rol tiene uno de los objetivos en los que se divide resolver reparación, cada plan haría las acciones que cubren sus subobjetivos.



El agente pagador puede hacerse con un solo plan que lo haga todo, ya que es suficientemente sencillo o hacer dos planes uno que gestione las peticiones de pago que inicia la percepción *reparado* y otro que confirme el pago que se ponga en marcha cuando lleguen las percepciones de confirmación de los pagos.



1.2. Problema 7

El Institut Català de la Salut (CatSalut) quiere transformar su sistema informático para hacer más flexible y eficiente su relación con los diferentes actores que intervienen en el sistema de salud. Estos actores son los pacientes, los médicos, los centros de diagnóstico y tratamiento y las farmacias. Los diferentes procesos que integrará el sistema son los siguientes:

Los pacientes solicitan al sistema una cita con su médico de cabecera, dando sus restricciones horarias y este busca el horario de visitas del médico y su ocupación, e informa a los dos de la visita concertada.
En la visita del paciente al médico, este puede tomar cuatro posibles decisiones:

a) Recetar al paciente un medicamento

b) Derivar al paciente a un especialista

c) Solicitar una prueba o un tratamiento para el paciente en un centro de diagnóstico y/o tratamiento

d) Considerar al paciente curado y no realizar ninguna nueva acción

De cara a racionalizar el gasto farmacéutico, en el caso de que se recete al paciente un medicamento, el sistema solicita a las farmacias cercanas al domicilio de éste cuáles tienen el fármaco, si lo tienen como genérico y su precio. El sistema decide qué farmacia es la más conveniente, le informa de que se le ha asignado la receta e informa al paciente de donde tiene que ir a recoger su medicamento.
En el caso de que el médico derive al paciente a un especialista, el sistema solicita al paciente sus restricciones, se encarga de concertar la cita e informar a médico y paciente. El especialista puede tomar las mismas decisiones que el médico de cabecera.
En el caso de que se solicite una prueba o tratamiento, el sistema primero solicita sus restricciones al paciente y se encarga de solicitar a diferentes centros adecuados (públicos o privados) su disponibilidad. Con la respuesta, el sistema decide el centro teniendo en cuenta las restricciones del paciente y lo pronto que se pueda realizar la cita, dando preferencia a los centros públicos. Finalmente, informa al centro elegido y al paciente de la cita.
Cuando la prueba o tratamiento se ha realizado, el centro de diagnóstico y/o tratamiento informa al sistema entregando los resultados para que sean informados al médico que lo solicitó, pide sus restricciones al paciente y concierta una nueva cita entre este y el médico.
En el caso de que se considere que el paciente esta curado el sistema no realiza ninguna acción.

Asumiremos, que los actores externos (pacientes, médicos, farmacias, centros de diagnóstico/tratamiento) se comunican con el sistema a partir de percepciones y acciones, mediante agentes propios ya implementados.

A la hora de hacer la distribución del sistema consideraremos que la gestión de las recetas y el concertar citas se puede hacer localmente desde los centros de atención primaria o centros de especialistas, pero la gestión de las peticiones de pruebas y tratamientos es más conveniente que sea centralizada para poder tomar mejor las decisiones de asignación de pacientes a esos centros.

(A) Especificación del sistema

1. Supón que hemos considerado que hay tres escenarios básicos:

   - Concertar una cita para un paciente con un médico
   - Realizar todo el proceso de una prueba/tratamiento
   - Asignar una receta a una farmacia

   a) Asumiendo que cada escenario corresponde a un objetivo general, define los subobjetivos en que se descompondrían (no es un problema si algunos objetivos están compartidos).

> Para el escenario de concertar una cita del paciente con un médico tenemos los siguientes objetivos, obtener las restricciones del paciente, decidir la cita a partir de las restricciones de su médico, e informar al paciente y al médico de la cita.

> Para el escenario del proceso de prueba y tratamiento tenemos, el pedir las restricciones al paciente, obtener la disponibilidad de los centros adecuados, decidir la cita con el centro, informar a paciente y centro de la cita y enviar los resultados al médico.
>
> Para el escenario de asignar una receta tenemos, obtener la disponibilidad del fármaco en las farmacias cercanas, decidir la mejor opción, asignar la receta a una farmacia e informar al paciente de la asignación.

b) Para el escenario que concierta una cita de un paciente con un médico podemos distinguir tres variantes distintas, cuando el paciente decide ir su médico, cuando es derivado a un especialista y cuando tiene que visitar al médico cuando llegan los resultados de su tratamiento/prueba ¿que es lo que distingue cada una de estas variantes?

> Las diferentes variantes se distinguen por sus percepciones, en la primera la percepción que pone en marcha el escenario es que el paciente indica sus restricciones, la segunda es iniciada por la percepción que genera el médico de que se deriva al paciente a un especialista y la tercera se inicia cuando el centro de prueba/tratamiento genera la percepción de que han llegado los resultados.

2. Hemos decidido que hay un rol *recetador* que es el encargado del proceso de buscar la farmacia adecuada para una receta y asignársela. Indica cuáles son las percepciones, acciones y objetivos que tendría.

> Este rol tendría como objetivos todos los del escenario de asignar receta.
>
> Las percepciones corresponderían a la que genera el médico cuando receta un medicamento a un paciente y las respuestas de las farmacias indicando la disponibilidad del medicamento.
>
> Las acciones corresponderían a la consulta del sistema a las farmacias sobre la disponibilidad del medicamento, la asignación del medicamento a una farmacia y la comunicación al paciente de la farmacia elegida.

3. Hemos decidido hacer un rol *concertador de citas con un médico* encargado de las tres variantes del escenario de cita con el médico y otro rol separado *concertador de citas de pruebas/tratamientos* que se encarga de citar al paciente para hacer una prueba/tratamiento. Este último rol colaborará con un rol *gestor de centros* que se encarga de buscar un centro de centro/tratamiento y enviar los resultados al médico. Indica cuáles son las percepciones, acciones y objetivos que tendría este último rol.

> Los objetivos de este rol corresponderían a todos los del escenario que realiza la prueba/tratamiento, salvo el que corresponde a informar al paciente de la cita.
>
> Este rol recibiría la percepción del médico indicando que ha de hacer una prueba/tratamiento, las percepciones de los centros consultados con su disponibilidad y la información de la prueba.
>
> Las acciones serían la de preguntar a los centros su disponibilidad, informar al centro al que se le asigna la prueba/tratamiento y dar los resultados del diagnóstico/tratamiento al médico.

(B) Diseño arquitectónico

1. Para agrupar los roles en agentes hemos decidido unir los dos roles relacionados con las citas en un agente *Citas*, hacer un agente *Farmacia* para el rol *Recetador* y un agente *Centros* para el rol *gestor de centros*. Define lo necesario para que el agente *Citas* y *Centros* puedan coordinarse para obtener una cita para un paciente en un centro de diagnóstico/tratamiento.

> Dado que el agente Citas y Centros están separados, pero el rol que concierta la cita con el paciente debe colaborar con el rol de gestor de centros debería haber un protocolo de interacción entre ellos.
>
> Esta colaboración necesitará que se comunique al agente Centro la disponibilidad del paciente y este comunique los datos de la cita una vez decidida por el agente Citas. Por lo tanto, el protocolo se compondrá de dos mensajes con esos contenidos.

2. Si decidiéramos unir los roles *concertador de citas de pruebas/tratamientos* y *gestor de centros* en un mismo agente. ¿Cuáles serían las ventajas e inconvenientes respecto a la comunicación entre agentes, la compartición de fuentes de datos y la distribución de tareas en el sistema?

> La ventaja obvia es que se evitaría tener un protocolo de comunicación entre el agente Citas y Centros.
>
> La información de las citas de los pacientes ahora o debería ser accedida por los agentes Citas y Centros como una fuente compartida o se debería separar en dos fuentes de datos, una para las citas de los pacientes con los médicos, que llevaría el agente Citas, y otra para las citas de los pacientes con los centros.
>
> Respecto a las tareas, esta nueva distribución requiere que todas las citas con los centros se establezcan de manera centralizada, ya que como dice el enunciado, el obtener y decidir los centros para pruebas y tratamientos se quiere hacer centralizado, lo que reduciría la distribución de las tareas del sistema.

3. Imagina que ahora decidimos cambiar nuestra especificación del sistema añadiendo un rol *diagnosticador* que es el que recibe las percepciones que genera el médico (deriva especialista, petición de pruebas/tratamientos, recetas) y la percepción de llegada de resultados de una prueba/tratamiento y lo asignamos a un agente *Médico*. Asumiremos que este agente estará asociado a los centros de atención primaria y de especialistas ¿Cómo cambia esto el diseño arquitectónico? ¿Comenta las ventajas/inconvenientes de esta modificación en el diseño?

> Obviamente, el primer cambio es que todas esas percepciones llegarán a ese agente y las acciones que informan de la cita de un médico con un paciente o de los resultados de las pruebas/tratamientos saldrán de este nuevo agente.
>
> El otro cambio es que al separar estas percepciones, este rol deberá colaborar con los roles complementarios para comunicarles su información.
>
> - Habrá un mensaje entre el agente Médico y Citas para informar de que se deriva al paciente a un especialista.
> - Habrá un mensaje entre Citas y Médico para informar de una cita concertada con un paciente
> - Habrá un mensaje entre Médico y Farmacia cuando se recete un medicamento
> - Habrá un mensaje entre Médico y Centros para informar de que se solicita una prueba/tratamiento para un paciente
>
> La principal desventaja del diseño es que aumentamos la comunicación entre los agentes, ya que lo que antes eran solo percepciones ahora tiene asociado un mensaje.
>
> Una ventaja es que ahora distribuimos una tarea que antes estaba centralizada en el agente Centros, que es la gestión de los resultados de las pruebas y su información a los médicos.

(C) Diseño detallado

1. Manteniendo el diseño con tres tipos de agentes con los dos roles de citas agrupados en el agente *Citas*. Estamos diseñando las capacidades/planes del agente *Centros* y decidimos que tenga dos, una que se encargue de buscar un centro adecuado y citar al paciente y otra que se encargue de informar al médico cuando llegan resultados de los centros. Indica qué acciones, percepciones y mensajes están conectados con cada una de estas capacidades.

> La capacidad que busca un centro adecuado para una prueba/tratamiento tendrá como percepciones la petición de un médico para hacer una prueba/tratamiento y las percepciones de los centros consultados con su disponibilidad. Como acciones el solicitar la disponibilidad a los centros y el asignar una cita a un centro. Finalmente como mensajes, los que corresponden al protocolo que recibe la información de disponibilidad del paciente y el que informa de los datos de las citas concertadas.
>
> La capacidad que informa de la llegada de las pruebas tiene como percepción la llegada de una prueba y como acción el informar al médico de los resultados.

2. Para el agente *Farmacia* definimos que tenga un único plan, indica cuál sería la secuencia de percepciones y acciones que realizaría su implementación.

> *a*) Percepción: receta un medicamento
>
> *b*) Acción: Consultar a farmacias cercanas al paciente
>
> *c*) Percepción: Respuesta de farmacias
>
> *d*) Acción: Asignar receta a una farmacia
>
> *e*) Acción: Informar al paciente de la farmacia elegida

CAPÍTULO 2

Ontologías

2.1. Problema 5

Después de observar los problemas de descoordinación de la ayuda humanitaria en desastres como los terremotos de Haiti o Chile, la Oficina Para la Coordinación de Asuntos Humanitarios de la ONU ha decidido crear un sistema que, según el tipo de catástrofe, ayude a tomar decisiones sobre el tipo de ayuda a enviar en cada caso.

El sistema se limita a la resolución puntual de las emergencias ocurridas por desastres naturales. Para poder decidir el tipo de ayuda que se necesita en cada caso, el sistema ha de tener en cuenta:

- el tipo de desastre (huracán, terremoto, inundación, erupción...);
- la magnitud del desastre (cada desastre tiene su propia escala de magnitudes.);
- el número estimado de personas afectadas por el desastre que necesiten ayuda;
- el número estimado de heridos;
- el lugar donde ha ocurrido el desastre (isla, costa, interior);
- las características orográficas relevantes (si hay ríos o lagos cerca, si es zona montañosa, altiplano, desierto, estepa...);
- el clima de la zona (tropical húmedo, desértico...);
- los servicios públicos que aun funcionan en la zona (luz, agua potable, alcantarillado, gas, radio, telefonía);
- el estado de las vías de comunicación hasta el área del desastre (tipo de vía y su estado actual);
- la existencia de reservas de alimentos y medicinas en la zona y cuantos días durarán;
- si en la zona del desastre existe algún conflicto militar o terrorista que pueda dificultar el suministro.

Con toda esa información el sistema ha de poder identificar el tipo de catástrofe humanitaria a resolver, y decidir el tipo de ayuda y la cantidad que se ha de mandar a la zona.

La ayuda puede ser de muchos tipos, y puede incluir: alimentos, ropa, tiendas de campaña, potabilizadores (si existe una fuente de agua utilizable) o cisternas de agua potable (si no hay fuentes de agua en la zona),

medicinas, hospitales de campaña (si hay un número muy elevado de heridos), combustible, generadores de electricidad, equipos de comunicaciones (cuando no funcionan los servicios de comunicación de la zona o son insuficientes), maquinaria pesada (en terremotos, erupciones y desprendimientos de tierra, para desenterrar a los heridos), camiones (para transportar gente o mercancías, si existen carreteras en buen estado), barcos (si es zona costera o esta en la ribera de un río navegable) aviones y helicópteros (si no hay carreteras adecuadas). Además, la ayuda también puede incluir personal humano: médicos, psicólogos, traductores, bomberos, ingenieros civiles, expertos en logística, pilotos, policías y efectivos militares (para proteger al resto del personal humano y/o en zonas de guerra).

a) Diseña la ontología del dominio descrito, incluyendo todos los conceptos que aparecen en la descripción e identificando los atributos más relevantes. Lista que conceptos forman parte de los datos de entrada del problema y que conceptos forman parte de la solución.

2.1.1. Solución

En este problema la ontología ha de incorporar, como mínimo, todos los conceptos que forman parte de la entrada del problema, así como conceptos que formen parte de la solución.

Los conceptos que forman parte de la entrada del problema son todos referidos a características del desastre sobre el que se ha de actuar:

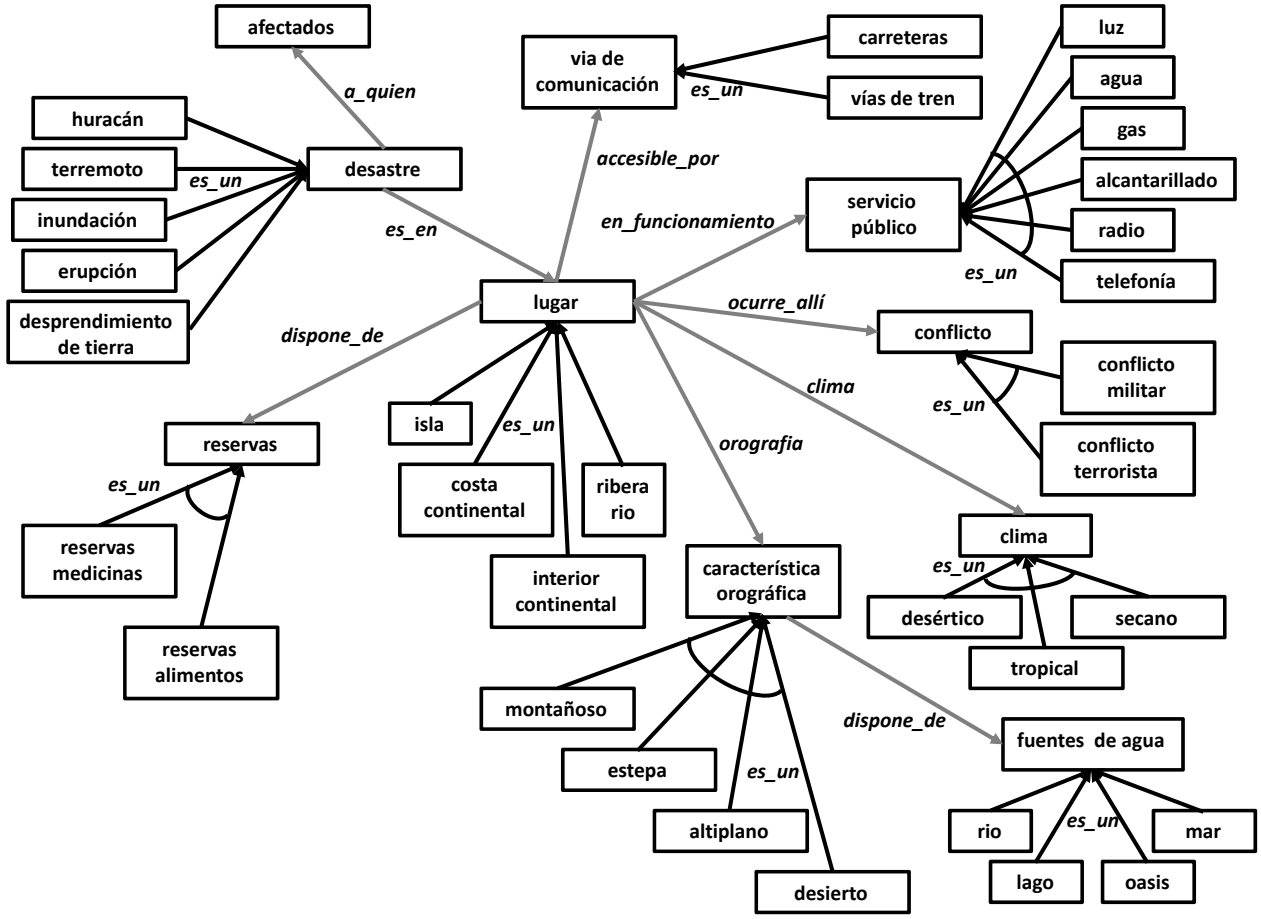
- la clase del desastre
- quien se ha visto afectado
- las características del lugar donde ha ocurrido
- las reservas disponibles de alimentos y medicinas
- los servicios públicos que aún están en funcionamiento
- las vías de comunicación disponibles

No hay mucho que remarcar en este caso, ya que el enunciado es muy detallado.

Atributos:

- `desastre`: nombre (string), fecha
- `afectados` : num_total (entero positivo), num_heridos (entero positivo)
- `huracán`: velocidad_viento (km/h)
- `terremoto`: grados_richter (real positivo)
- `inundación`: nivel_agua (centímetros), m2_inundados (metros cuadrados)
- `erupción`: radio_de_afectación (km)
- `desprendimiento de tierra`: m3_tierra (metros cúbicos)
- `reservas`: estimación_días (entero positivo)
- `ribera rio`: navegable (booleano)
- `fuente de agua`: distancia (km), potable (booleano)
- `servicio público`: grado_funcionamiento (%)
- `agua`: potable (booleano)
- `via de comunicación`: estado_actual (totalmente funcional, parcialmente funcional, inoperativa)

En el caso de los atributos solo cabe destacar que, aunque todos los desastres tienen algún tipo de magnitud, no es buena idea poner la magnitud como atributo de `desastre`, ya que el enunciado nos dice que cada tipo de desastre tiene diferentes formas de medir su magnitud. Por ello en la lista de atributos se puede ver que se han puesto atributos distintos para los diferentes tipos de desastre, que se ajustan mejor a cada uno de ellos.

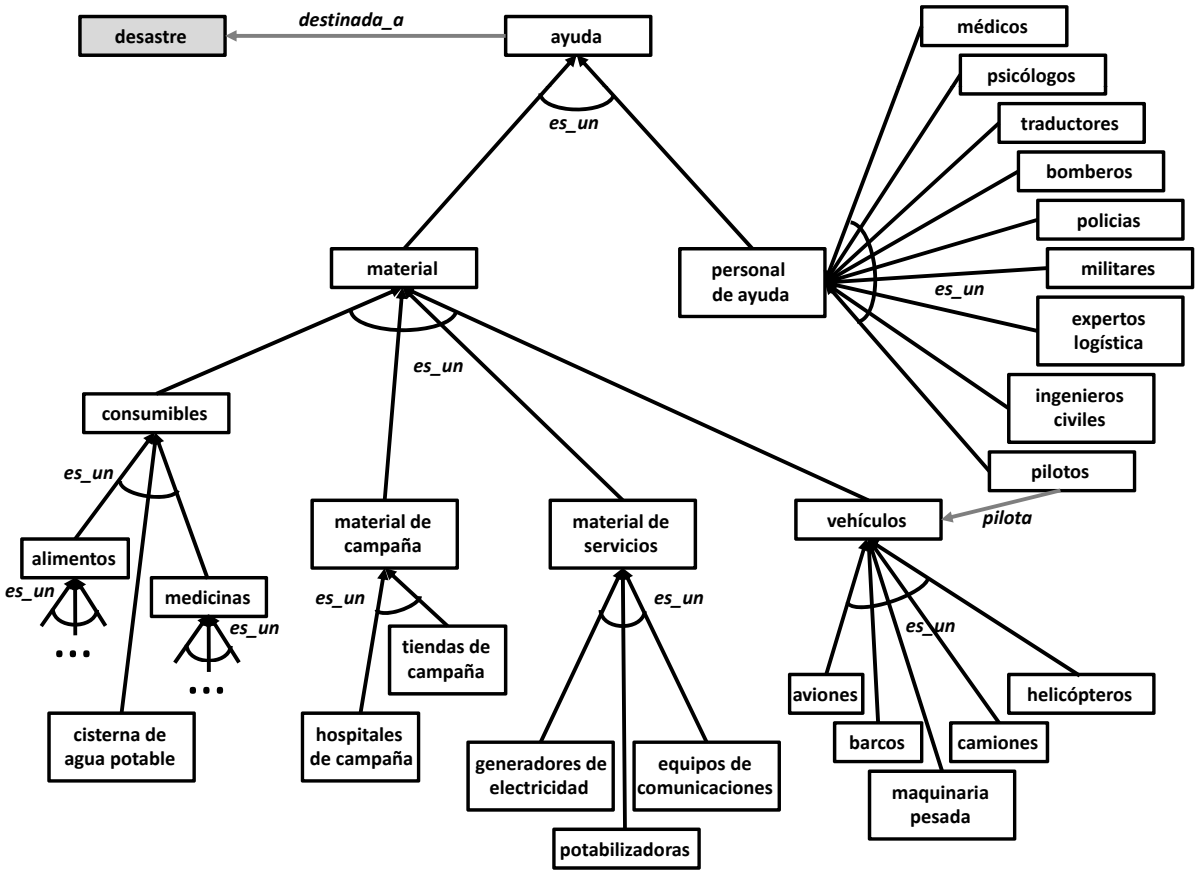


La ontología asociada a la solución esta compuesta por los diferentes tipos de ayuda y las cantidades a enviar. Esta parte de la ontología esta dominada por relaciones taxonómicas, a las que se han añadido algunas relaciones no taxonómicas, como `pilota(piloto, vehículo)`. Como en este caso tenemos más libertad, la ontología se podría hacer con más o menos detalle, más o menos niveles intermedios (en el enunciado solo aparecen los nodos hoja). Se ha optado por tener el concepto `ayuda` con dos subclases (`material` y `personal de ayuda`), pero una alternativa válida sería que ayuda fuera la composición de una lista de materiales y de una lista de personal. También se ha representado como se conecta esta parte de la ontología con la anterior, a través del una relación `destinada_a` hacia el concepto `desastre`. Además, se ha optado por indicar en el diagrama que de los conceptos `alimentos` y `medicinas` seria necesario especificar sus subclases, que en el diagrama solo indicamos con los puntos suspensivos.

Atributos:

- `ayuda`: volumen (muchas, pocas, ninguna)
- `personal de ayuda`: num_personas (entero positivo)
- `material`: num_unidades (entero positivo)
- `material de campaña`: capacidad (entero positivo)

Es importante destacar también que en este caso se han puesto los conceptos de la ontología en plural, y no en singular, ya que en este caso no pretenden representar conceptos ontológicos generales sino los predicados que describen la solución (ver los apartados b y c).



2.2. Problema 6

La cadena de supermercados *HIPERFUR* nos pide que diseñemos un SBC orientado a la creación de folletos de ofertas de productos personalizados para cada cliente. Cada folleto consta de 8 páginas y en cada página se pueden colocar 9 ofertas diferentes, que provienen de una base de ofertas a la que tendrá acceso el sistema. Toda oferta tiene una fecha de inicio y de fin, y un producto o productos asociados. Hay dos tipos de ofertas:

- *descuentos*, que tienen entre otros datos el porcentaje de descuento, e información sobre si son descuentos directos (el cliente paga el precio rebajado) o indirectos (el cliente recibe un vale de compra).
- *promociones*, que pueden ser packs de productos diferentes (por ejemplo, un queso y un jamón) u ofertas al llevarse varias unidades del mismo producto (2x1, 3x2, 4x3).

Los productos pueden ser de diferente tipología: alimentación (leche, huevos, pescado, carne, bebidas…), belleza (cremas, maquillaje, after-shave…), menaje (sartenes, ollas, vasos, cubertería…), limpieza del hogar (jabón de ropa, lavavajillas, quitamanchas, mocho, escoba…). Además, los productos pueden pertenecer a una marca blanca (precio muy bajo), a una marca generalista (precio medio) o a una marca premium (precios muy altos).

Este sistema obtendrá la información de los hábitos de compra de un cliente (una lista de las ofertas y los productos individuales que ha comprado en los últimos 6 meses donde, para cada oferta y para cada producto individual, se dice cuantas unidades ha comprado. Además los clientes pueden (a través de la página web) añadir una lista de productos preferidos , y/o una lista de restricciones que se pueden aplicar a un producto concreto (ej: no quiero el jabón "Pixan plus") o a un subtipo de producto (ej: no me gustan las leches de soja).

Se quiere construir este sistema de manera que, en base a toda la información disponible sobre el cliente, seleccione un conjunto de ofertas adecuadas para ese tipo de cliente y las asigne a alguna de las 8 páginas del folleto. El director de marketing de *HIPERFUR* nos recomienda que incluyamos las siguientes características abstractas como base de la selección:

- el nivel de **gasto mensual** (alto, medio, bajo) que servirá para calcular el importe total de los productos que se coloquen en el folleto
- la **calidad** de los productos (mayoría premium, mayoría normal, mayoría marca blanca) que servirá para seleccionar, dentro de un (sub)tipo de producto, el producto adecuado al patrón de compra del cliente
- la **cantidad_alimentación** (exagerada, mucha, normal, poca, limitada), que servirá para decidir que porcentaje del folleto se dedica a productos alimenticios (y en los casos extremos -exagerada, limitada- se colocarán preferentemente productos de alto valor nutricional)
- la **cantidad_belleza** (narcisista, normal, dejado/a) que servirá para decidir que porcentaje del folleto se dedica a productos de belleza
- la **cantidad_menaje** (exagerado/a, normal, minimalista) que servirá para decidir que porcentaje del folleto se dedica a productos de menaje
- y la **cantidad_limpieza** (obseso/a, normal, sucio/a) que servirá para decidir que porcentaje del folleto se dedica a productos de limpieza.

a) Diseña la ontología del dominio descrito, incluyendo todos los conceptos que aparecen en la descripción e identificando los atributos más relevantes. Lista que conceptos forman parte de los datos de entrada del problema y que conceptos forman parte de la solución.

2.2.1. Solucion

En este problema la ontología ha de incorporar, como mínimo, todos los conceptos que forman parte de la entrada del problema, así como conceptos que formen parte de la solución.

Los conceptos que forman parte de la entrada del problema son todos referidos a características del cliente al que queremos generarle el folleto personalizado:

- las compras que ha realizado
- el tipo de los productos que ha comprado
- la marca y calidad de los productos que ha comprado
- si ha sacado provecho de descuentos u ofertas anteriores

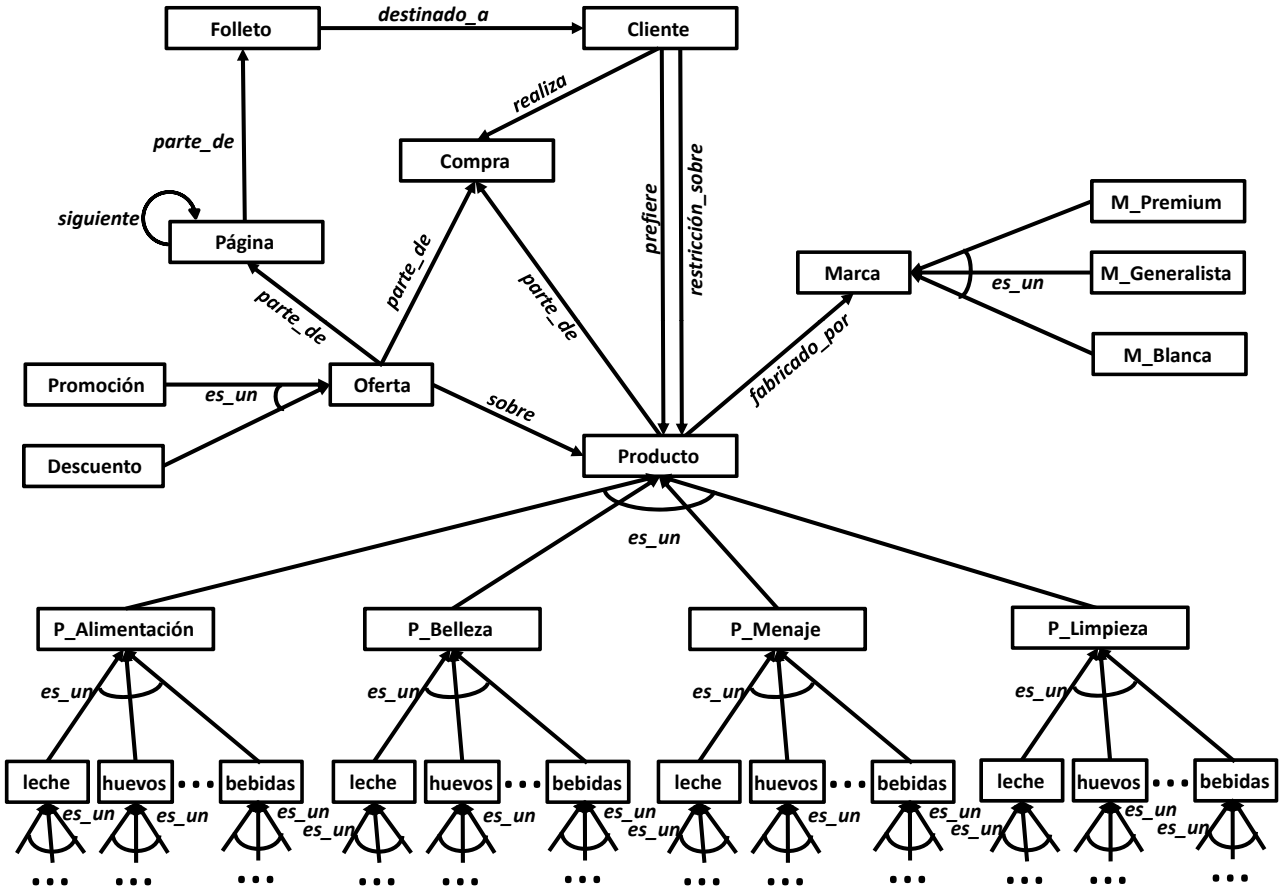
Los conceptos que forman parte de la salida del problema son el folleto, las páginas del folleto y toda la información relevante sobre los descuentos y promociones que se incluirán en el folleto. En este caso, como se ha de tener la información de los productos que se incluyen en el folleto, todos los conceptos excepto `Compra` pertenecen a la solución.

No hay mucho que remarcar sobre la ontología resultante, ya que el enunciado daba bastante detalle. Se ha optado por indicar en el diagrama que de algunos conceptos (como `P_Alimentación`, `P_Belleza`, `P_Limpieza` o `P_Menaje`) seria necesario especificar sus subclases, que en el diagrama solo indicamos con los puntos suspensivos. También hemos reducido el valor nutritivo de un producto de alimentación a un número, pero para hacerlo más realista deberíamos tener la tabla nutricional completa.

Atributos:

- `Cliente`: nombre (string), dirección (string)
- `Producto` : nombre (string), referencia (string), importe (real positivo),
- `Oferta`: fecha_inicio (fecha), fecha_fin (fecha)

- `Descuento`: porcentaje (entero positivo), directo (booleano)
- `P_Alimentación`: valor_nutritivo (real positivo)



2.3. Problema 7

La compañía de subastas on-line eBay quiere permitir que su plataforma sea accedida como un conjunto de servicios web, de manera que se puedan implementar servicios cliente/agentes que permitan a los usuarios usar sus capacidades. Como paso inicial, se quiere definir una ontología que describa toda la información que se maneja y las diferentes acciones que se pueden realizar.

Tenemos el concepto *usuario*, que puede especializarse en vendedor y comprador (un usuario puede pertenecer a ambas categorías). El concepto *producto*, que puede especializarse en electrónica (ordenadores, cámaras, consolas, móviles, ...), coleccionables (monedas, sellos, cómics, ...), entretenimiento (cine, música, ...), motor (coches, motos, barcos, ...), juguetes (radio control, trenes, infantiles, ...), ...El concepto *subasta*, que puede especializarse en subastas rápidas (menos de 24 horas) y subastas largas.

Aparte tenemos el concepto *acción*, que representa toda una serie de diferentes acciones que se pueden realizar en e-Bay. Estas se pueden clasificar en acciones sobre productos, como registrar un producto o buscar un producto de ciertas características y acciones sobre subastas, como abrir una subasta para un producto, cancelar una subasta, buscar subastas y pujar en una subasta con un precio de puja. Estos conceptos de acciones son los que se envían al servicio de e-Bay para que se ejecuten.

Para toda relación/atributo que definas indica su dominio, su rango y su cardinalidad (en ambos sentidos si es una relación). Por ejemplo, si definiéramos la relación *realiza* entre usuario y acción, el dominio sería

usuario, el rango acción y la cardinalidad 1 a N en el sentido dominio a rango (cardinalidad 1 a 1 en el sentido inverso).

a) Añade lo necesario a la ontología (relaciones/conceptos/atributos) para que se pueda representar la búsqueda de un producto/subasta por parte de un comprador. La ontología debería poder responder como mínimo a las preguntas ¿qué productos hay registrados antes/después de una fecha? ¿qué subastas comienzan/acaban antes/después de una fecha? ¿qué subastas con un precio de inicio menor/mayor de x euros hay?

> La solución más sencilla es añadir a las acciones de buscar producto y buscar subasta los atributos que permiten restringirlos.
>
> De esta manera, la acción buscar producto tendría los atributos de fecha registro máxima/mínima, y la acción buscar subasta tendría los atributos: fecha inicio máxima/mínima, fecha fin máxima/mínima y precio máximo y mínimo
>
> Correspondientemente, el concepto producto debería tener el atributo fecha de registro y el concepto de subasta el precio inicial del producto y su fecha de inicio y fin.
>
> Una alternativa sería crear un concepto restricción que se especializara por fechas y precios y una o varias relaciones entre las acciones de búsqueda y las restricciones (dominio acción, rango restricción, cardinalidad 1 a N de dominio a rango)

b) Añade lo necesario a la ontología (relaciones/conceptos/atributos) para que se pueda representar el proceso de puesta a la venta de un producto en una subasta. La ontología debería poder responder como mínimo a las preguntas ¿qué vendedor ha registrado un producto? ¿Qué vendedor ha abierto una subasta? ¿Qué producto se esta subastando en una subasta?

> Definimos una relación entre vendedor y producto, la relación podría llamarse *registra*, con dominio vendedor y rango producto y cardinalidad 1 a N en el sentido dominio a rango (cardinalidad 1 a 1 en el sentido inverso).
>
> Definimos una relación entre vendedor y subasta, la relación podría llamarse *abre*, con dominio vendedor y rango subasta y cardinalidad 1 a N en el sentido dominio a rango (cardinalidad 1 a 1 en el sentido inverso).
>
> Definimos una relación entre subasta y producto, la relación podría llamarse *vende*, con dominio subasta y rango producto y cardinalidad 1 a 1 en el sentido dominio a rango (cardinalidad 1 a 1 o 1 a N en el sentido inverso dependiendo de si consideramos que un producto puede resubastarse si no se vende).

c) Añade lo necesario a la ontología (relaciones/conceptos/atributos) para que se pueda representar el proceso de puja en una subasta. La ontología debería poder responder como mínimo a las preguntas ¿qué pujas se han hecho en una subasta? ¿Cuánto ha pujado un comprador en una subasta? ¿Quién ha ganado una subasta? ¿Cuándo se ha realizado una puja?

> Primero haría falta un nuevo concepto puja, que debería tener como atributos cantidad (cuanto se ha pujado) y fecha de puja (cuando se ha pujado). Este concepto será diferente de la acción de pujar (es el efecto de esa acción)
>
> Necesitamos una relación entre subasta y puja, que podría llamarse *ofertas*, con dominio subasta y rango puja y cardinalidad 1 a N en el sentido dominio a rango (cardinalidad 1 a 1 en el sentido inverso).
>
> Necesitamos una relación entre comprador y puja, que podría llamarse *ofrece*, con dominio comprador y rango puja y cardinalidad 1 a N en el sentido dominio a rango (cardinalidad 1 a 1 en el sentido inverso).

> Necesitamos una relación entre subasta y comprador, que podría llamarse *ganador*, con dominio subasta y rango comprador y cardinalidad 1 a 1 en el sentido dominio a rango (cardinalidad 1 a 1 en el sentido inverso).

d) Supongamos que e-Bay quiere introducir un sistema de reputación para los usuarios que pueda ser consultado. Añade lo necesario a la ontología (relaciones/conceptos/atributos) para que esto se pueda representar. La ontología debería poder representar como mínimo las acciones de pedir la reputación de un usuario como vendedor/comprador y buscar qué usuarios con una reputación mayor/menor que *r* existen.

> Deberíamos añadir un atributo de reputación como vendedor y como comprador en los conceptos correspondientes. Añadirlos en el concepto usuario tiene el problema de tener usuarios que sean solo compradores o vendedores, deberíamos darle un valor por defecto al atributo para evitarlo.
>
> Deberíamos añadir en el concepto acción una especialización más, que sean acciones de reputación, que se especializara en consulta de reputación y búsqueda por reputación.
>
> El primero debería tener algún atributo que indicara el usuario del que queremos consultar la reputación y el segundo debería tener atributos para representar las restricciones sobre las reputaciones. También podríamos usar la segunda opción del primer apartado y añadir estas restricciones como conceptos.

e) Si e-Bay quisiera guardar información histórica de quién ha comprado/vendido un producto ¿qué es lo mínimo que habría que añadir a la ontología para que esta información pudiera representarse explícitamente? ¿Podría deducirse esta información de lo que ya tenemos en la representación? Suponiendo que cuando se asigna un producto de una subasta, el comprador ha de realizar un pago a e-Bay y e-Bay después realiza un pago al vendedor ¿que habría que añadir/modificar en la ontología para representar esta información suponiendo que además se incluyeran las fechas de esos pagos?

> Lo mínimo necesario para representar explícitamente la venta/compra de un producto por parte de un vendedor/comprador es añadir relaciones de compra/venta entre estos conceptos.
>
> Esta información ya está implícitamente en la representación, ya que el comprador de un producto debería ser el que ha ganado la subasta del producto y el vendedor el que ha abierto la subasta o el que ha registrado el producto.
>
> De todas formas, si tal como se dice después en el enunciado, se han de hacer pagos en los que e-Bay es el intermediario, estas relaciones implícitas no tienen por que representar los actos de compra y venta, ya que estos son efectivos cuando se ha realizado el pago.
>
> Para guardar la información con los pagos sería necesario el añadir el concepto pago que tuviera como atributo la fecha en el que se ha realizado y una relación entre comprador/vendedor y pago. Tener también otra entre pago y producto permitiría formalizar explícitamente las relaciones descritas al principio.

f) De todos los conceptos que han ido apareciendo ¿Cuál crees que es en conjunto mínimo de ellos que deberían compartir los servicios de e-Bay con los clientes/agentes para que la comunicación entre ellos se pueda realizar?

> Los clientes/agentes solo necesitan la información mínima para comunicar las acciones, toda información que representan conceptos internos a e-Bay no la necesitan, por lo que habría suficiente con los conceptos descritos inicialmente en el enunciado y las acciones añadidas.
>
> Los conceptos de puja y pago serían internas a e-bay (guardan su estado) y no necesitan ser compartidas.

2.4. Problema 9

Tenemos un sistema de gestión de proyectos informáticos basado en agentes que permite a los diferentes roles del proyecto y entidades involucradas poder realizar su labor. Una versión inicial de la ontología identifica tres conceptos principales con una descomposición posiblemente incompleta:

- **Roles**: que son las entidades que hacen tareas y se comunican en el sistema, estos se dividen en *personas* (jefe de proyecto, diseñador, programador y tester) y *servicios* (repositorio, calendario).
- **Tareas**: que representan lo que los roles pueden hacer, estas se dividen en *diseño*, *programación* (codificación, resolución de errores) y *testeo*.
- **Objetos**: que son entidades sobre las que los roles pueden hablar o realizar tareas, estas se dividen en *proyecto*, *paquete*, *ejecutable*, *error*.

El objetivo es completar esta ontología con los atributos, relaciones y conceptos necesarios para que se puedan responder las cuestiones que se plantean a continuación. En el caso de que encuentres diferentes opciones justifica tus decisiones.

Para toda relación/atributo que definas indica su dominio, su rango y su cardinalidad (en ambos sentidos si es una relación). Por ejemplo, si definiéramos la relación *participa* entre persona y proyecto, el dominio sería persona, el rango proyecto y la cardinalidad 1 a N en el sentido dominio a rango (cardinalidad 1 a N en el sentido inverso), si definieramos el atributo *horas dedicadas* en tarea el dominio sería tarea y el rango los reales.

a) Añade lo necesario a la ontología (relaciones/conceptos/atributos) para que se pueda responder a las siguentes preguntas: ¿Quién tiene asignada una tarea? ¿Cuándo se ha de finalizar una tarea? ¿Quién ha asignado una tarea? ¿Sobre que objeto se realiza una tarea?

> Solo tenemos que añadir los atributos y relaciones que permiten representar esta información en la ontología
>
> Relación: Asignada a, D: Tarea, R: Persona, CardDR: 1 a N, CardRD: 1 a N (se podría discutir si una tarea concreta se puede asignar a una sola o a más personas, podría depender del tipo de tarea y el rol que la realiza, por ejemplo un paquete podría ser diseñado por un solo diseñador, pero codificado por varios programadores)
>
> Relación: Asignada por, D: Tarea, R: Persona, CardDR: 1 a 1, CardRD: 1 a N
>
> Relación: Objetivo, D: Tarea, R: Objeto, CardDR: 1 a 1, CardRD: 1 a N
>
> Atributo: Fecha Finalización, D: Tarea, R: Fecha
>
> El rango de asignada a y asignada por se ha puesto como Persona, pero se podría concretar más, ya que no todos los roles pueden asignar tareas, ni todos recibir una asignación de tarea.

b) Si quisiéramos que los roles persona comunicaran al rol calendario la asignación de las tareas y los días de antelación con los que se ha de avisar de que una tarea aún no se ha finalizado ¿qué sería necesario añadir a la ontología (relaciones/conceptos/atributos)? ¿Y si quisiéramos también que las personas comunicaran la asignación de tareas a otras personas? Añade lo que consideres necesario a la ontología.

> En estos casos necesitaríamos nuevos conceptos que permitieran representar la comunicación entre los diferentes roles. Podemos llamar al concepto general por ejemplo *comunicación*, que tendría dos especializaciones, una para informar al calendario, que podríamos llamar *registro* y otra entre roles persona que podríamos llamar *asignación*

El primer concepto tendría una relación (que podríamos llamar *asignación de*) con un objeto tarea (este objeto tarea ya nos indica toda la información necesaria sobre quién ha de realizar la tarea, quién se la ha asignado y qué hay que hacer), el dominio seria registro y el rango tarea, con cardinalidad uno a uno en los dos sentidos y un atributo *días de antelación* en el concepto registro con rango entero.

El segundo concepto sería idéntico, pero solo tendría la relación con la tarea, con las mismas características y que también podríamos llamar igual.

La justificación de usar dos conceptos es el tener un atributo diferente, que le confiere también una semántica diferente. El primer concepto es un registro en un calendario para poder saber qué tareas se han asignado, el segundo es simplemente informar a alguien de qué ha de hacer.

c) Los diseñadores reciben la tarea de dividir el desarrollo del proyecto en paquetes de manera jerárquica (diseño). Añade lo necesario a la ontología para que se pueda representar e informar al repositorio de la descomposición hecha por un diseñador de paquetes en otros paquetes.

Ahora necesitaríamos una relación entre paquetes que representará la descomposición. Podemos llamar a esta relación *compuesto por* con dominio y rango paquete con cardinalidad 1 a N de dominio a rango y de 1 a 1 de rango a dominio.

Para comunicar al repositorio la descomposición y quién la ha hecho necesitamos un nuevo concepto en la clase comunicación que se podría llamar *descomposición*.

Tenemos dos opciones para representar la comunicación. La primera es que este objeto ligue el diseñador con la descomposición completa, de manera que tendríamos dos relaciones en el objeto descomposición una *descompuesto por* con dominio descomposición y rango diseñador (cardinalidad 1 a 1) y otra *descompuesto en* con dominio descomposición y rango paquete (cardinalidad 1 a 1), asumiendo que en la comunicación se incluye todo el árbol de descomposición y esta relación apunta al primer paquete.

La segunda es que cada comunicación solo informara de una relación *compuesto por* (o todas las relaciones de un paquere a su descomposición inmediata) y en lugar de la relación *descompuesto en* hubiera dos relaciones, una llamada *sobre* (paquete que se descompone) y otra *se descompone en* (descomposición del paquete), si esta fuera 1 a 1 (en las dos direcciones) informaríamos de cada descomposición de manera individual y si fuera 1 a N del dominio al rango, informaríamos de una descomposición completa de un paquete.

d) ¿Le es posible al repositorio, con la representación obtenida hasta ahora, deducir quién es el diseñador responsable de cualquier paquete? Si es posible, explica como, si no, añade lo necesario a la representación para que se pueda.

Se puede añadir explícitamente en la representación para cada paquete quién es su responsable a partir de la información que recibe el repositorio de la descomposición con una relación *responsable de* entre diseñador y paquete.

También aparece esta información explícitamente en los objetos que comunican la descomposición, así que simplemente podríamos deducirla guardando estos objetos y recorriendo las relaciones que tienen entre los paquetes y los diseñadores para deducir la responsabilidad.

e) Los diseñadores también se informan entre ellos de la descomposición en paquetes y determinan sus dependencias cruzadas. Añade lo necesario a la ontología para que los diseñadores puedan informar al repositorio de las dependencias que han encontrado entre paquetes que han diseñado y los de otro diseñador y para que el repositorio pueda representar estas dependencias.

> Para las dependencias necesitamos otra relación entre paquetes que podría llamarse *depende de* con dominio y rango paquete con cardinalidad 1 a N en los dos sentidos.
>
> Para la comunicación necesitaríamos otro concepto llamado *dependencia* con dos relaciones una *es dependiente* entre dependencia y paquete (indicando que el paquete tiene una dependencia) y *depende de* entre dependencia y paquete (indicando el paquete del que es dependiente), para ser completos podríamos añadir una relación entre dependencia y el diseñador que ha informado de la dependencia.

f) Los diseñadores asignan a los programadores la tarea de codificar paquetes y estos cada vez que acaban la implementación de una nueva versión la informan al repositorio. Añade lo necesario a la ontología para que el repositorio pueda deducir qué se debe compilar y cómo se ha de hacer para obtener un ejecutable del estado actual del proyecto.

> Un ejecutable se obtiene compilando la última versión de las implementaciones de los paquetes.
>
> En primer lugar, necesitamos un nuevo concepto en el concepto objetos que represente la *implementación* de un paquete. Necesitaremos también una relación entre paquete e implementación que podemos llamar *implementado por* con cardinalidad 1 a N de dominio a rango y 1 a 1 de rango a dominio (tenemos diferentes implementaciones de un paquete, una por cada versión). Nos hará falta también algo que nos indique la versión de la implementación para que podamos escoger la última. Para esto nos bastaría un atributo en implementación que permitiera ordenarlas, ya sea un número de versión o una fecha.
>
> Esto completa el grafo de dependencias que ya han informado los diseñadores. Con esto, el saber como hay que compilar las implementaciones de los paquetes para obtener el ejecutable solo es necesario usar la descomposición y las dependencias y compilar la última versión de la implementación de cada paquete.

g) Los programadores asignan a los tester tareas de test cuando informan al repositorio de una nueva versión de las implementaciones de los paquetes que tienen asignados. Estos últimos reportan los errores encontrados al repositorio y al programador. Añade a la ontología lo necesario para que el repositorio pueda representar que un error ha sido reportado para una implementación y también la descripción asociada al error.

> En este caso simplemente nos hace falta una relación entre objetos implementación y objetos error. Podríamos llamarla *tiene el fallo* con cardinalidad 1 a N de dominio a rango y 1 a 1 de rango a dominio.
>
> La descripción del error puede ser simplemente un atributo del objeto error.
>
> También se podría hacer más complicado añadiendo un objeto nuevo que ligara a estos dos y guardara la descripción.

h) Hemos representado una parte de las clases de la ontología del problema anterior en RDFS/OWL usando el espacio de nombres `p` de la siguiente manera:

```
p:Proyecto rdf:type owl:Ontology.

p:Rol rdf:type owl:Class.

p:Persona rdfs:SubClassOf p:Rol.
p:JefeProyecto rdfs:SubClassOf p:Persona.
p:Diseñador rdfs:SubClassOf p:Persona.
```

```
p:Programador rdfs:SubClassOf p:Persona.
p:Tester rdfs:SubClassOf p:Persona.
p:Servicio rdfs:SubClassOf p:Rol.
p:Repositorio rdfs:SubClassOf p:Servicio.
p:Calendario rdfs:SubClassOf p:Servicio.

p:Tarea rdf:type owl:Class.

p:Diseño rdfs:SubClassOf p:Tarea.
p:Programacion rdfs:SubClassOf p:Tarea.
p:Testeo rdfs:SubClassOf p:Tarea.
```

Sobre estas clases hemos definido las siguientes propiedades y relaciones:

```
p:Nombre rdf:type rdfs:Property;
         rdfs:domain p:Persona;
         rdfs:range xsd:String.

p:IdTarea rdf:type rdfs:Property;
         rdfs:domain p:Tarea;
         rdfs:range xsd:String.

p:HorasDedicadas rdf:type rdfs:Property;
         rdfs:domain p:Tarea;
         rdfs:range xsd:Integer.

p:AsignadaA rdf:type rdfs:Property;
         rdfs:domain p:Tarea;
         rdfs:range p:Persona.

p:AsignadaPor rdf:type rdfs:Property;
         rdfs:domain p:Tarea;
         rdfs:range p:Persona.

p:JefeDe rdf:type rdfs:Property;
         rdfs:domain p:Persona;
         rdfs:range p:Persona.
```

1) ¿Que aserciones en RDF representarían que un programador llamado “Pedro”, tiene asignada la tarea con identificador “T1234” y se han dedicado 7 horas a la tarea?

Necesitaremos definir URIs para los objetos del programador y la tarea, por ejemplo `p:Pedro` y `p:T1234`, las aserciones que representan lo que se pide serían:

```
p:Pedro rdf:type  p:Programador;
        p:Nombre "Pedro".

p:T1234 rdf:type  p:Tarea;
        p:IdTarea "T1234";
```

```
          p:HorasDedicadas 7;
          p:AsignadaA  p:Pedro.
```

2) ¿Que aserciones en RDF añadiríamos a las anteriores para representa que un diseñador llamado "Juan", ha asignado la tarea con identificador "T4321" al programador llamado "Pedro"?

Necesitaremos definir URIs para los objetos del diseñador y la tarea, por ejemplo `p:Juan` y `p:T4321`, las aserciones que representan lo que se pide serían:

```
p:Juan rdf:type  p:Diseñador;
       p:Nombre "Juan".

p:T4321 rdf:type  p:Tarea;
        p:IdTarea "T4321";
        p:AsignadaA  p:Pedro;
        p:AsignadaPor  p:Juan.
```

3) Asumiendo que tenemos un conjunto de aserciones sobre este problema usando la ontología que hemos definido en este apartado ¿qué nos retornaría la siguiente consulta SPARQL?

```
SELECT ?n ?c ?id
WHERE {
       ?p1 p:Nombre "Pedro".
       ?t p:AsignadaA ?p1.
       ?t p:IdTarea ?id.
       ?t p:AsignadaPor ?p2.
       ?p2 p:JefeDe ?p1.
       ?p2 rdf:type ?c.
       ?p2 p:Nombre ?n.
      }
```

Esta consulta SPARQL nos retornaría los nombres de la personas que han asignado una tarea a Pedro y que son su jefe, a que clase pertenecen y el identificador de esa tarea

2.5. Problema 10

Tenemos un sistema de gestión de un hospital basado en agentes, que permite a los diferentes roles y entidades del sistema involucrados poder realizar su labor. Una versión inicial de la ontología identifica cuatro conceptos principales con una descomposición posiblemente incompleta:

- **Servicio**: que son las unidades que gestionan las diferentes tareas que se realizan en el hospital. Estas incluyen servicios médicos (cirugía, UCI, farmacia, pruebas diagnósticas, enfermería...) y de gestión (admisiones, alojamiento, programación, mantenimiento...)
- **Personal**: que son las personas designadas por los servicios para realizar algunas de las tareas que se hacen en el hospital. Este incluye personal médico (cirujanos, cardiólogos, anestesistas...), de enfermería, camilleros, celadores y administrativos.
- **Sala**: que son las ubicaciones donde se realizan las tareas del hospital. Estas incluyen salas quirófano, salas de cuidados intensivos, salas de recuperación, salas de ingreso, salas de diagnóstico...

- **Paciente**: que son las personas que son atendidas por los servicios y personal del hospital. Se clasifican en pacientes ingresados y externos.

Asumiremos que los servicios y el personal están representados por agentes y entre ellos colaboran para realizar las tareas del hospital.

El objetivo es completar esta ontología con los atributos, relaciones y conceptos necesarios para que se puedan responder las cuestiones que se plantean a continuación. En el caso de que encuentres diferentes opciones justifica tus decisiones.

Para toda relación/atributo que definas indica su dominio, su rango y su cardinalidad (en ambos sentidos, si es una relación). Por ejemplo, si definiéramos la relación *asignado a* entre personal y servicio, el dominio sería personal, el rango servicio y la cardinalidad 1 a 1 en el sentido dominio a rango (cardinalidad 1 a N en el sentido inverso), si definiéramos el atributo *nombre* en personal el dominio sería personal y el rango string.

a) Añade lo necesario a la ontología (relaciones/conceptos/atributos) para que el servicio de admisiones pueda registrar que ha realizado la tarea de admitir un paciente en una fecha concreta.

> Necesitaremos un concepto *tarea* en la ontología que represente las tareas que pueden realizar los diferentes servicios/personal del hospital. Como especialización de tarea tendemos el concepto de *admisión* de un paciente. La admisión necesitará tener un atributo fecha de admisión para registrar la fecha y una relación con un paciente, esta sería 1 a 1 en el sentido dominio a rango y 1 a N en el sentido rango a dominio.

b) El servicio de admisiones ha de obtener una cama para alojar a los pacientes admitidos. Para realizar esto, este servicio debe utilizar un mensaje para pedir al servicio de alojamiento que reserve una cama en una sala de ingreso para un número de días específico a partir de una fecha. El servicio de alojamiento registra internamente que una cama de una sala de ingreso está ocupada. Añade lo necesario a la ontología para que esta comunicación se pueda realizar y que se registre la ocupación de la cama.

> Los servicios se comunican una acción que se ha de realizar, por lo que nos hará falta un concepto *acción* que los incluya. Podemos por ejemplo llamar a esta acción *reserva* que tenga un atributo *fecha inicio* y otro *días de ocupación*.
>
> El servicio de alojamiento debe registrar la ocupación de una cama. En la ontología solo aparece sala de ingreso, podemos añadir un concepto más general que incluya los componentes de las salas (camas de una sala de ingreso o UCI o cubículos de las salas de pruebas o de recuperación) como por ejemplo *unidades*, del que podría colgar *cama*.
>
> La ocupación podríamos considerarla una tarea (o un concepto aparte) que debería tener un atributo *fecha inicio*, otro *días de ocupación* y una relación con la cama ocupada, esta sería 1 a 1 de dominio a rango y 1 a N de rango a dominio.
>
> También debemos añadir una relación entre *cama* y *sala de ingreso* que se puede llamar *pertenece* que sea 1 a 1 de dominio a rango y 1 a N de rango a dominio.

c) El servicio de programación es el encargado de comunicar a los servicios con tareas médicas (operaciones, pruebas diagnósticas, ...) que cierta tarea ha de realizarse sobre un paciente entre dos fechas. Supongamos que el servicio de programación necesita que se realice una operación a un paciente. Éste utiliza un mensaje pidiendo a la unidad de cirugía la realización de cierto tipo de operación a un paciente entre dos fechas y esta responde con un día y hora para la operación y la registra. Añade lo necesario a la ontología para que la comunicación de servicio de programación a servicio de cirugía se pueda realizar (solo en ese sentido) y que este último registre que una sala quirófano está reservada cierto día a cierta hora para esa cirugía.

> El mensaje que se intercambian el servicio de programación y el de cirugía corresponde a una acción, podemos añadir un nuevo concepto a acción que corresponda a la petición de una operación, la podemos llamar *operar*.
>
> Este concepto tendría como atributos el tipo de operación, que podría ser un tipo enumerado o un string (o si queremos hacerlo mas formal debería ser un concepto de la ontología que registrara las operaciones que se pueden realizar en el hospital), necesitamos un atributo que identifique al paciente (o una relación con este) y dos atributos que indiquen el rango de fechas en los que se ha de realizar la operación.
>
> El servicio de cirugía ha de registrar una tarea que podemos llamar *operación*. Una operación tendrá los atributos que correspondan al tipo de operación, el paciente al que se le hace la operación, la fecha de la operación y una relación con la sala quirófano donde se va a hacer, que sea 1 a 1 de dominio a rango y 1 a N de rango a dominio.

d) El servicio de cirugía ha de obtener un equipo médico para poder hacer las operaciones que tiene programadas. Para ello se comunica con los agentes que corresponden a los cirujanos, anestesistas y enfermeras de quirófano para obtener su disponibilidad. Con las respuestas elige un equipo médico que realice una operación y registra que esa operación será realizara por ese equipo médico. Añade lo necesario a la ontología para que se pueda representar esta información en el servicio de cirugía.

> Podemos partir de que tenemos registrada la tarea operación y, o bien creamos una relación entre operación y cada una de las personas del equipo médico con la operación (cardinalidad 1 a N en los dos sentidos), o alternativamente, creamos un concepto nuevo que podemos llamar *equipo* (podría ser un concepto independiente o incluirlo como parte del concepto personal) que tenga esa relación con las personas del equipo médico (1 a N en los dos sentidos) y además incluir una relación en operación que se llame por ejemplo *realizada por*, que ligue el equipo con la operación, que sea 1 a 1 en ambos sentidos.

e) Los anestesistas que participan en una operación deben consultar los análisis realizados a un paciente para determinar el tipo de anestesia que necesitarán. Para ello han de pedir al servicio de pruebas diagnósticas las analíticas de sangre más recientes del paciente. Añade lo necesario para que el servicio de pruebas diagnósticas pueda responder a esta consulta representando los diferentes tipos de pruebas de un paciente y en particular las analíticas de sangre.

> Nos va a hacer falta añadir un nuevo concepto que corresponda a *prueba diagnóstica*. Esta podrá especializarse en diferentes pruebas, en particular las analíticas de sangre. Deberá haber un atributo *fecha de la prueba* que debe estar en prueba diagnóstica y una relación de esta con el paciente, esta se puede llamar *realizada a* que sea de 1 a 1 del dominio al rango y 1 a N del rango a dominio, los subconceptos obtendrán por herencia estas características. El resto de atributos deberán estar ligados a las pruebas diagnósticas particulares.

f) Una vez realizada una operación a un paciente, el servicio de cirugía envía al paciente a una cama en una sala de cuidados intensivos o a su cama en la sala de ingreso. Dependiendo del tipo de pacientes ingresados en el servicio UCI, éste encarga a diferentes servicios médicos enviar a una hora determinada del día al médico de guardia en el servicio. Añade a la ontología lo necesario para que un servicio médico pueda saber como responder a esta petición.

> Cada uno de los servicios necesitará tener un concepto que represente una *guardia*, puede ser un concepto aparte o puede ser una tarea que realice el servicio.

> Este concepto debería tener atributos que permitieran saber cuando se esta realizando esa guardia, así sabremos cual coincide con la hora en la que hay que enviar al médico (por ejemplo, hora inicio y hora fin de la guardia). Por último nos hará falta una relación que podemos llamar *realizada por*, que relacione la guardia con el médico, cardinalidad 1 a 1 de dominio a rango y 1 a N de rango a dominio.

g) Hemos representado una parte de las clases de la ontología del problema anterior en RDFS/OWL, usando el espacio de nombres `h`, de la siguiente manera:

```
h:Hospital rdf:type owl:Ontology.

h:Servicio rdf:type owl:Class.
h:Personal rdf:type owl:Class.
h:Turno rdf:type owl:Class.

h:ServicioMedico rdf:SubClassOf h:Servicio.
h:ServicioGestion rdf:SubClassOf h:Servicio.

h:Cirugia rdf:SubClassOf h:ServicioMedico.
h:Programacion rdf:SubClassOf h:ServicioGestion.

h:PersonalMedico rdf:SubClassOf h:Personal.
h:Cirujano rdf:SubClassOf h:PersonalMedico.
h:Anestesista rdf:SubClassOf h:PersonalMedico.
h:Enfermera rdf:SubClassOf h:Personal.
```

Sobre estas clases hemos definido las siguientes propiedades y relaciones:

```
h:Nombre rdf:type rdfs:Property;
         rdfs:domain h:Personal;
         rdfs:range xsd:String.

h:CodigoServicio rdf:type rdfs:Property;
         rdfs:domain h:Servicio;
         rdfs:range xsd:String.

h:CodigoTurno rdf:type rdfs:Property;
         rdfs:domain h:Turno;
         rdfs:range xsd:String.

h:FechaInicio rdf:type rdfs:Property;
         rdfs:domain h:Turno;
         rdfs:range xsd:DateTime.

h:FechaFin rdf:type rdfs:Property;
         rdfs:domain h:Turno;
         rdfs:range xsd:DateTime.

h:AsignadoA rdf:type rdfs:Property;
         rdfs:domain h:Personal;
```

```
        rdfs:range h:Servicio.

h:TieneResponsable rdf:type rdfs:Property;
        rdfs:domain h:Servicio;
        rdfs:range h:Personal.

h:Realiza rdf:type rdfs:Property;
        rdfs:domain h:Personal;
        rdfs:range h:Turno.
```

1) ¿Qué aserciones en RDF representarían que un cirujano llamado "Tomás" está asignado al servicio de cirugía con código de servicio "SCIR01", es su responsable y además realiza el turno "SCR0120160608-01" que empieza en "2016-06-08 10:00" y acaba en "2016-06-08 18:00"?

Necesitaremos definir URIs para los objetos del cirujano, servicio y turno, por ejemplo `h:Tomas` `h:SCIR01` y `h:SCR0120160608-01`, las aserciones que representan lo que se pide serían:

```
h:Tomas rdf:type  h:Cirujano;
        h:Nombre "Tomás";
        h:AsignadoA  h:SCIR01;
        h:Realiza  h:SCR0120160608-01.

h:SCIR01 rdf:type  h:ServicioMedico;
        h:CodigoServicio "SCIR01";
        h:TieneResponsable h:Tomas.

h:SCR0120160608-01 rdf:type h:Turno;
        h:CodigoTurno "SCR0120160608-01";
        h:FechaInicio "2016-06-08 10:00";
        h:FechaFin "2016-06-08 18:00".
```

2) Asumiendo que tenemos un conjunto de aserciones sobre este problema usando la ontología que hemos definido en este apartado ¿qué nos retornaría la siguiente consulta SPARQL?

```
SELECT ?n1 ?n2 ?dh1
WHERE {
       ?p1 rdf:type h:PersonalMedico.
       ?p1 h:Nombre ?n1.
       ?s1 h:TieneResponsable ?p1.
       ?p1 h:Realiza ?t1.
       ?t1 rdf:type h:Turno.
       ?p2 rdf:type h:Enfermera.
       ?p2 h:Nombre ?n2.
       ?p2 h:Realiza ?t2.
       ?t2 rdf:type h:Turno.
       ?t1 h:FechaInicio ?dh1.
       ?t2 h:FechaInicio ?dh1.
      }
```

> Esta consulta SPARQL nos retornaría ternas (nombre personal médico, nombre enfermera, fecha) para el personal médico y enfermera que hiciera un turno que comenzara en la misma fecha y hora, siempre que el personal médico fuera responsable de algo, con esta ontología solo lo podrá ser un servicio.