Caso de estudio completo: de la modelización de negocios al diseño detallado en Java o C#

Este caso de estudio nos va a permitir partir de la modelización de negocios (oficiosa) hasta llegar al diseño detallado (ya sea implementado en Java o en C#), pasando por la especificación de los requisitos funcionales y el análisis orientado a objetos.

Vamos a ver en particular:

- Qué diagramas de UML utilizar para la modelización de negocios?
- Cómo utilizar esta modelización de negocios para definir mejor las necesidades computacionales?
- Cómo el análisis lingüístico permite ayudar a la modelización del ámbito?
- Cómo describir una arquitectura en capas con UML?
- Cómo utilizar los diagramas de secuencia y de comunicación para describir las interacciones entre objetos computacionales, y distribuir las operaciones?
- Cómo repercuten las decisiones de asignación de las responsabilidades a los objetos en los diagramas de clases?
- Cómo traducir los diagramas UML de diseño detallado en código orientado a objetos?

Etapa 1: Modelización de negocios (business modeling)

En el marco de la mejora que quiere aportar a su sistema de información, una empresa desea modelar, inicialmente, el proceso de formación de sus empleados para que algunas de sus tareas estén automatizadas (desde el punto de vista computacional). *Utilice la palabra formación, porque es más la idea, pero puede ser capacitación.*

1. El proceso de formación es inicializado cuando el responsable de formación recibe una solicitud de formación por parte de un empleado. Esta solicitud se instruye por el responsable quien la califica y transmite su acuerdo o su desacuerdo al interesado.

2. En caso de acuerdo, el responsable busca en el catálogo formaciones autorizadas un período de prácticas que corresponde a la solicitud. Informa al empleado del contenido de la formación y le propone una lista de las próximas sesiones. Cuando el empleado hace su elección, el responsable de formación inscribe al participante a la sesión para el organismo de formación en cuestión.

3. En caso de impedimento, el empleado debe informar al responsable de formación cuanto antes para cancelar la inscripción o la solicitud.

4. Al final de su formación, el empleado debe entregar al responsable de formación una valoración sobre el período de prácticas que efectuó, así como un documento justificando de su presencia.

5. El responsable de formación pide más tarde la factura que el organismo de formación le envió antes de transmitirla al contable de compras.

ESTEREOTIPOS PARA LA MODELIZACIÓN DE NEGOCIOS

En cuanto a modelización de negocioso, Jacobson fue el primero en proponer la utilización de los conceptos UML de actor, caso de uso, clase, paquete, etc, con estereotipos particulares. En la consecución del caso de estudio, utilizaremos los estereotipos siguientes, proporcionados por Rational/Rose, o algún otro software de modelado como Enterprise Architect:

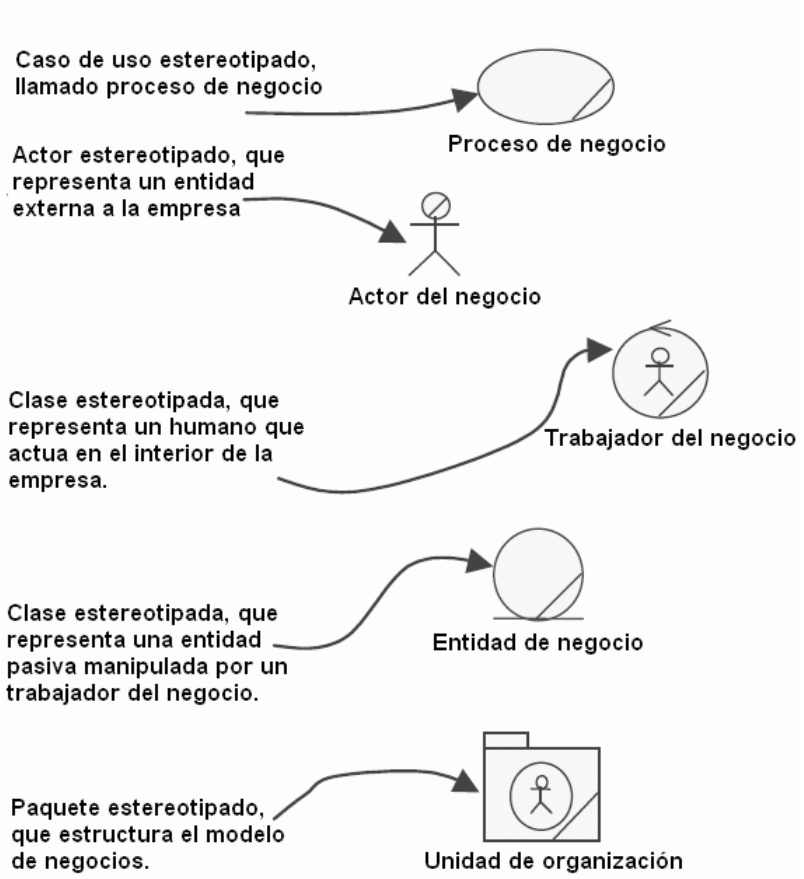


Figura 1: Estereotipos utilizados para la modelización de negocios.

A partir de estos estereotipos, modelaremos el proceso de formación y sus actores. El proceso de formación está representado por un caso de uso estereotipado. Los actores implicados son (en el orden del enunciado):

- El empleado;
- El responsable de formación;
- El organismo de formación;
- El contable de las compras.

Sólo el organismo de formación es una entidad externa a la empresa, lo que da el siguiente esquema:

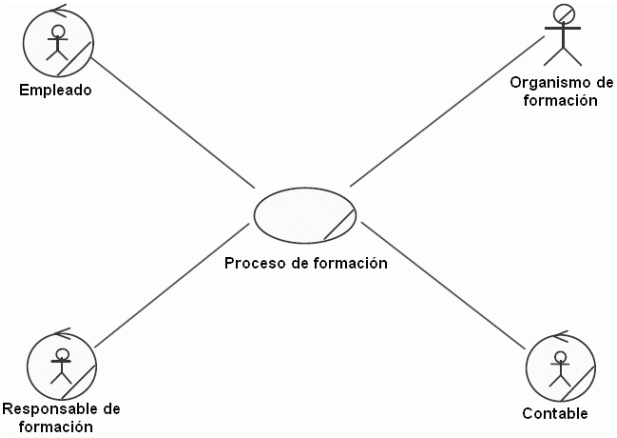


Figura 2: Modelización del proceso de formación con sus actores.

El proceso de formación implica un conjunto de acciones ordenadas en el tiempo y destinadas a uno de los actores identificados anteriormente. Este encadenamiento se representa perfectamente gracias a un diagrama de actividad.

Las particiones permiten arreglar gráficamente las acciones de modo que las que se destinan a un mismo actor se encuentren en la misma banda vertical.

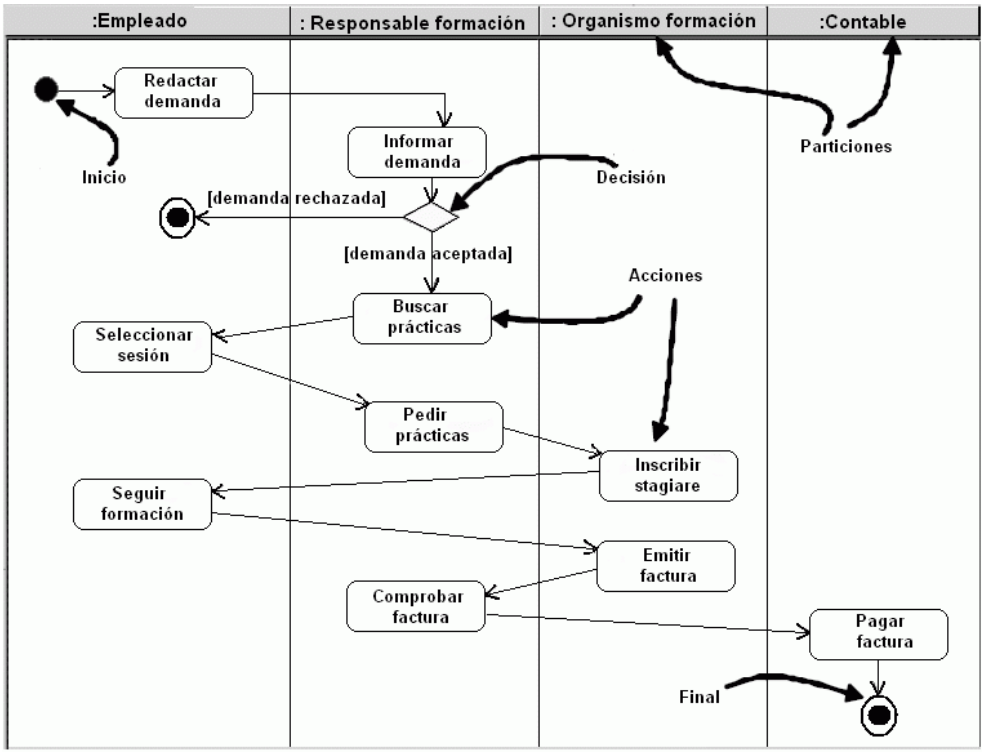


Figura 3: Diagrama de actividad del proceso de formación.

Para completar el diagrama, se pueden también añadir la creación y el cambio de estado de las entidades de negocio, a raíz de la realización de las acciones.

Tomemos en cuenta que no utilizamos el icono específico de la entidad negocio para la SolicitudDeFormación, con el fin de poder más fácilmente indicar sus cambios de estados entre corchetes. El diagrama así obtenido es muy interesante, puesto que hace el puente entre los tres ejes de modelización: funcional (acciones), dinámico (flujos) y estático (entidades y divisiones)!

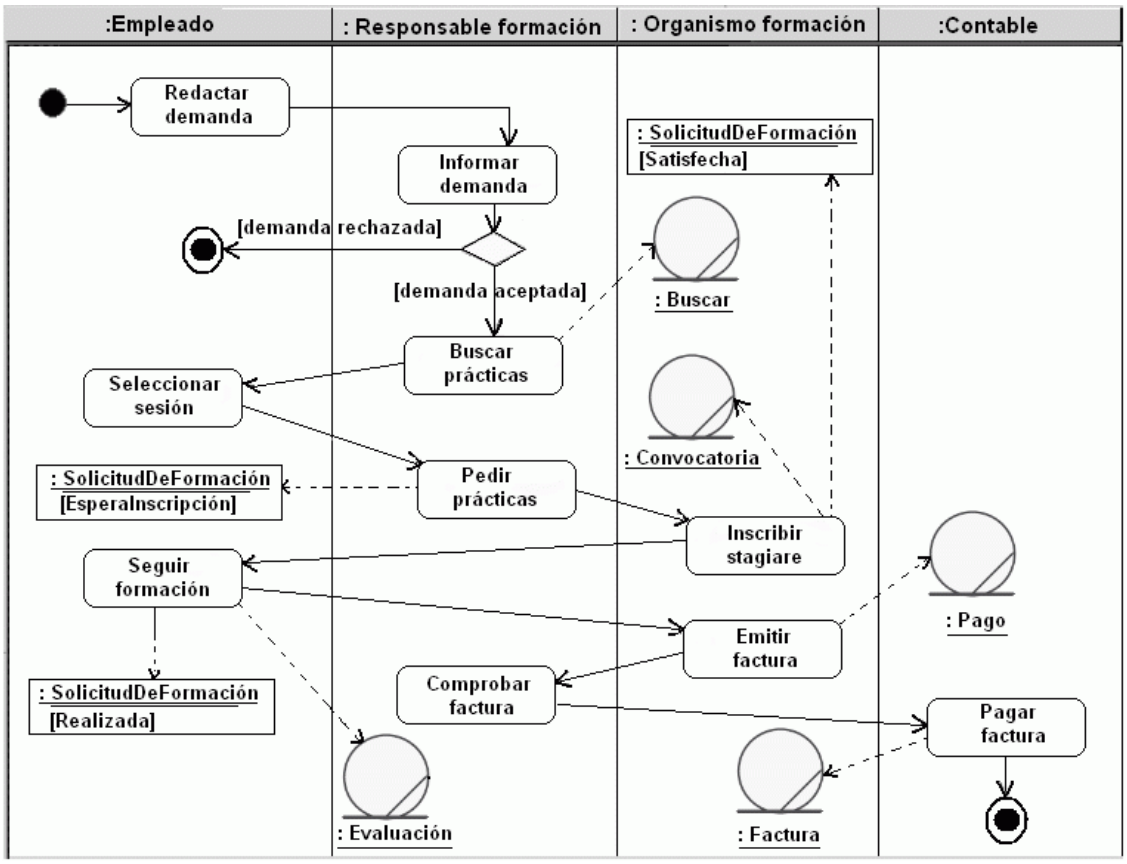


Figura 4: Diagrama de actividad completado del proceso de formación.

Etapa 2: Definición de los requisitos del sistema computacional

Prosiguiendo con nuestro estudio funcional. La definición de las tareas que se automatizarán se realiza por medio de la selección de algunas acciones del modelo de negocio. Vamos así a deducir el pliego de condiciones funcionales del sistema computacional a partir del estudio anterior, y en particular del diagrama de actividad.

El sistema debe permitir inicializar una solicitud de formación y seguir esta solicitud hasta la inscripción efectiva de un empleado.

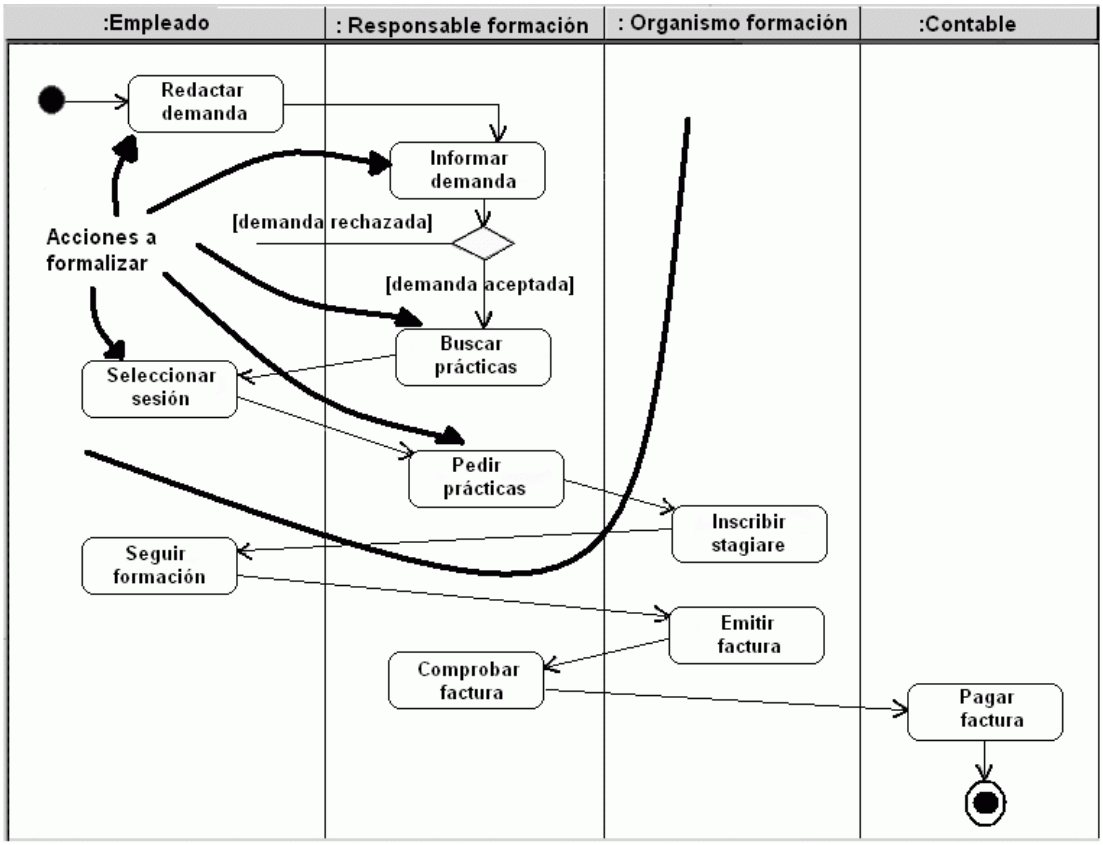


Figura 5: Acciones del proceso de formación que se automatizará.

El sistema de gestión de las solicitudes de formación debe pues permitir automatizar las siguientes acciones de negocios:

- Redactar una solicitud (empleado);
- Instruir una solicitud (responsable formación);
- Buscar prácticas (responsable formación);
- Seleccionar una sesión (empleado);
- Pedir un período de prácticas (responsable formación).

No es necesario olvidar que un empleado tiene la posibilidad de cancelar una solicitud o una inscripción a una sesión.

Para todo eso, es indispensable que el sistema computacional administre un catálogo de formaciones autorizadas a las cuales los empleados pueden acceder parcialmente en lectura, y el responsable de formación globalmente en escritura. Este catálogo contendrá no solamente el contenido técnico, la duración, etc., de las formaciones propuestas por los organismos autorizados, sino que también las fechas y lugares de las próximas sesiones. El responsable de formación podrá también crear reagrupaciones de formaciones llamadas temas.
Los actores implicados son (en orden del enunciado):

- el empleado;
- el responsable de formación;
- el organismo de formación.

En efecto, no se automatizarán las acciones del contable y no se interactuará directamente con el sistema computacional.

Según la lista de las acciones de negocio, se pueden definir los siguientes casos de uso:

Solicitar una formación

El empleado puede consultar el catálogo y seleccionar un tema, o una formación, o incluso una sesión particular. La solicitud está automáticamente registrada por el sistema y transmitida al responsable de formación por correo electrónico.

Tratar las solicitudes

El responsable de formación va a utilizar el sistema para indicar a los empleados su decisión (acuerdo o desacuerdo). En caso de acuerdo sobre una sesión precisa, el sistema va a enviar automáticamente por fax una solicitud de inscripción bajo la forma de orden de pedido al organismo en cuestión. Si el empleado no eligió una sesión, pero simplemente una formación o un tema, el responsable de formación va consultar el catálogo y a seleccionar las sesiones que parecen corresponder mejor a la solicitud. Esta selección será transmitida por correo electrónico al empleado, que podrá así presentar una nueva solicitud más precisa.

Administrar sus solicitudes

El empleado puede consultar el estado de sus solicitudes de formación en curso y cancelarlas eventualmente de forma individual. Puede también precisar una solicitud incompleta. Se informa automáticamente al responsable de formación por correo electrónico.

Actualizar el catálogo

El responsable de formación puede introducir una nueva formación en el catálogo, modificar una formación existente o suprimir una formación que un organismo abandonó. Puede también modificar las reagrupaciones de formaciones que han sido clasificadas por temas. Tiene también la posibilidad de actualizar las fechas y lugares de las sesiones.

El siguiente diagrama preliminar sintetiza todas estas reflexiones.

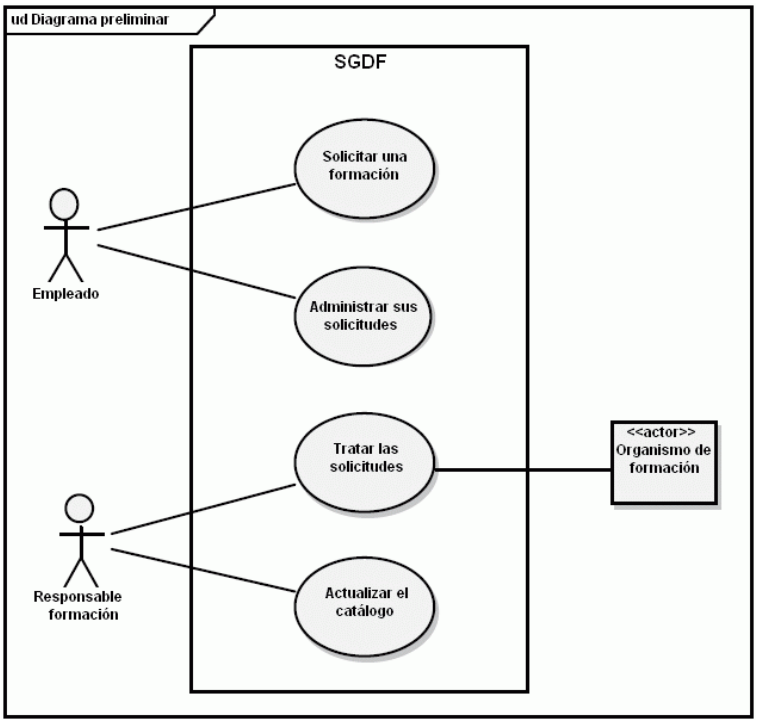


Figura 6: Diagrama de casos de uso preliminares del sistema de gestión de las solicitudes de formación.

Ejemplificaremos la descripción detallada de un caso de uso, actualizar el catálogo.

Titulo: Actualizar el catálogo **Tipo**: esencial detallado

Resumen: el responsable de formación está encargado de la actualización continua de un catálogo que clasifica las formaciones autorizadas disponibles para los empleados. La mayoría de las modificaciones proceden de los organismos de formación

Actores: Responsable formación

Fecha de creación: 03/10/2010 **Fecha de actualización**: 04/10/2010

Versión: 1.1 **Responsable**: Abraham Sánchez

Descripción de los encadenamientos

Precondiciones: el responsable de formación se autentificó en el sistema.

Postcondiciones: una nueva versión del catálogo está disponible en consulta.

Situación nominal:

1. Este caso de uso comienza en general cuando un Organismo de formación informa al	

Responsable de formación de modificaciones por informe a su oferta.	
2. El Responsable de formación puede introducir una nueva formación en el catálogo, modificar una formación existente o eliminar una formación no considerada por el organismo. En el momento de una creación o de una modificación, el Responsable de formación tiene la posibilidad de modificar la agenda de las sesiones previstas para la formación.	3. El Sistema avisa a los usuarios conectados quienes corren el riesgo de trabajar sobre una versión obsoleta. En una supresión, el Sistema indica al Responsable de formación la lista de los participantes que se inscribían a las sesiones canceladas, y se cancelan las inscripciones.
4. El Responsable de formación valida sus modificaciones.	5. El Sistema previene a los empleados conectados que una nueva versión del catálogo está disponible.

Secuencias alternativas:

A1: informaciones incompletas

La secuencia A1 inicia en la etapa 2 de la situación nominal.

2. Cuando las informaciones relativas a una nueva formación son incompletas (por ejemplo, ausencia de fecha de sesión), la formación se pone en el catálogo sin ninguna inscripción. La descripción debe modificarse y completarse más tarde.

La situación nominal sigue en la etapa 2.

A2: gestión de los temas

La secuencia A2 inicia en la etapa 2 de la situación nominal.

2. El responsable de formación puede comenzar por crear un nuevo tema o renombrar un tema existente. Puede también desplazar una formación de un tema a otro.

La situación nominal sigue en la etapa 2.

Especificaciones suplementarias

Concurrencia: este caso de uso no puede ser ejecutado más que por un responsable a la vez.

Disponibilidad: el catálogo es accesible mediante la Intranet de lunes a las 9 h al viernes a las 17 h. Las acciones de mantenimiento deben limitarse al mínimo estricto durante estas horas.

Ejemplo de diagrama de secuencia del sistema del caso de uso: “Actualizar el catálogo”.
Utilizamos un fragmento de tipo «alternativas» (alt) para indicar que las acciones efectuadas por el responsable de formación pueden llegar en cualquier orden. Este fragmento «alternativas» mismo se imbrica en un ciclo (loop).

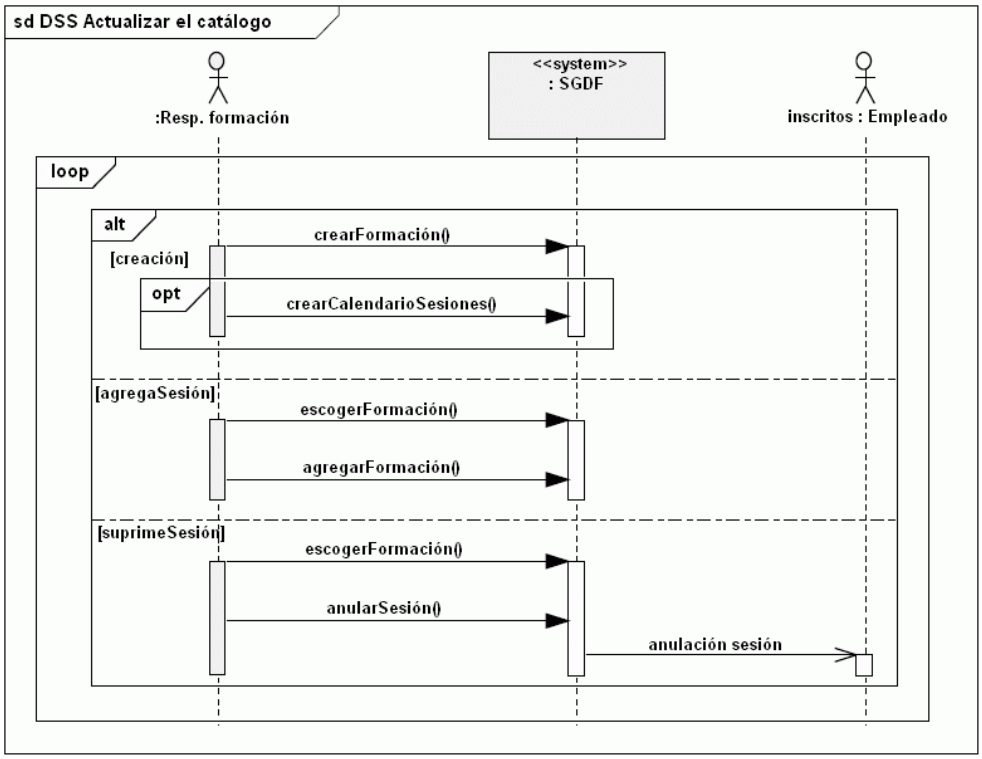


Figura 7: Diagrama de secuencia del sistema del caso de uso: Actualizar el catálogo.

Observemos que los empleados inscritos a una sesión cancelada son informados automáticamente por el sistema (correo electrónico). La flecha del mensaje «anulación sesión» está abierta para indicar que se trata de un mensaje asíncrono.

FLUJOS DE CONTROL DE LOS MENSAJES
Un flujo de control síncrono significa que el objeto emisor se bloquea a la espera de la respuesta del receptor del mensaje. En un flujo de control asíncrono por el contrario, el objeto emisor no espera la respuesta del receptor y prosigue su tarea sin preocuparse de la recepción de su mensaje.

Será necesario pues añadir a un actor secundario a nuestro diagrama de caso de uso preliminar.

Podemos mejorar el diagrama de casos de uso preliminar del sistema computacional de gestión de las solicitudes de formación.
Para pedir una formación, el sistema debe proponer una funcionalidad básica de consulta del catálogo. El empleado puede consultar el catálogo sin presentar solicitud. La creación de una solicitud viene a extender opcionalmente la consulta.

El actor «Organismo de formación» no hace más que recibir mensajes que vienen del sistema, utilizaremos pues a una asociación unidireccional. Del mismo modo, el empleado es actor secundario (receptor) de los casos de uso del responsable de formación y recíprocamente.

Además, un responsable de formación puede también efectuar una demanda de formación, etc. Añadimos entonces una relación de generalización entre actores.

Finalmente, para no sobrecargar el diagrama, no representaremos el proceso de identificación del empleado o del responsable de formación. Pero crearemos más bien un paquete diferente. El modelo de casos de uso se divide así en dos paquetes:

- caso de uso operacionales;
- casos de uso de soporte.

El diagrama de casos de uso del primer paquete sería:

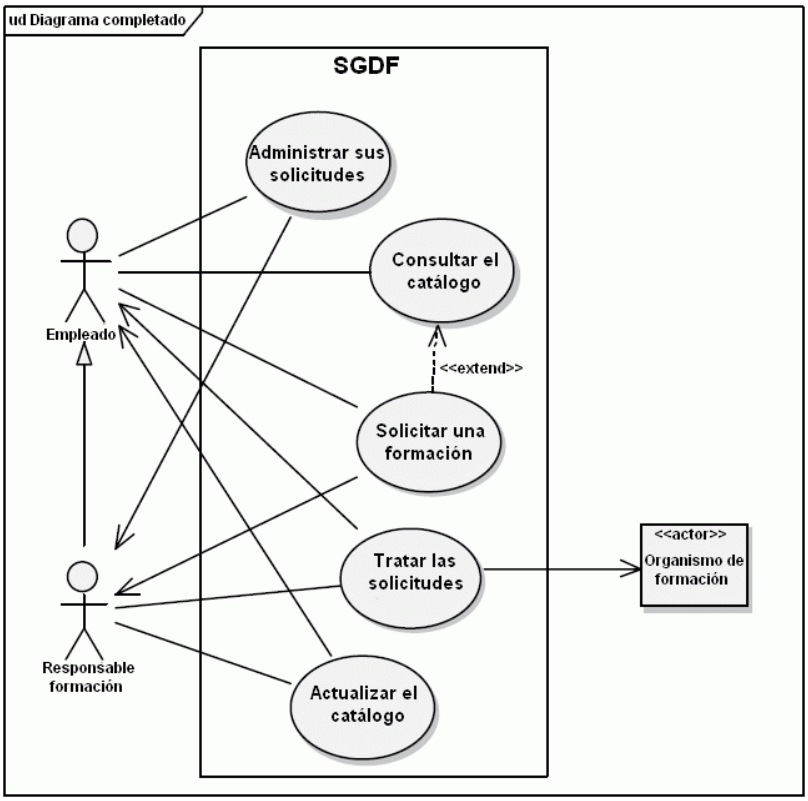


Figura 8: Diagrama de casos de uso del paquete de los casos operacionales

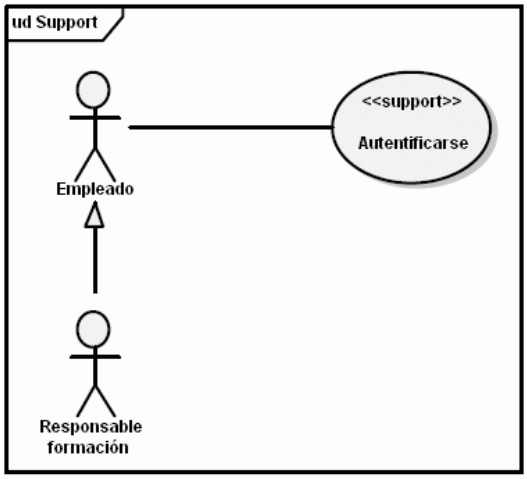


Figura 9: Diagrama de casos de uso del paquete de los casos de soporte.

La restricción de concurrencia sobre este caso de uso nos lleva a una última cuestión. Como elaborar el diagrama de contexto estático del sistema?

El sistema de gestión de las solicitudes de formación es de hecho multiusuarios (típicamente: una Intranet), excepto para el responsable de formación que debe ser el único usuario que modifica en un momento dado.

Los organismos de formación no tienen acceso al sistema: no hacen más que recibir pedidos (uno a la vez), lo que explica la flecha de navegabilidad sobre la asociación entre el sistema y el actor no humano. Podemos notar que hemos esta vez utilizado el símbolo del componente UML para particularizar el sistema.

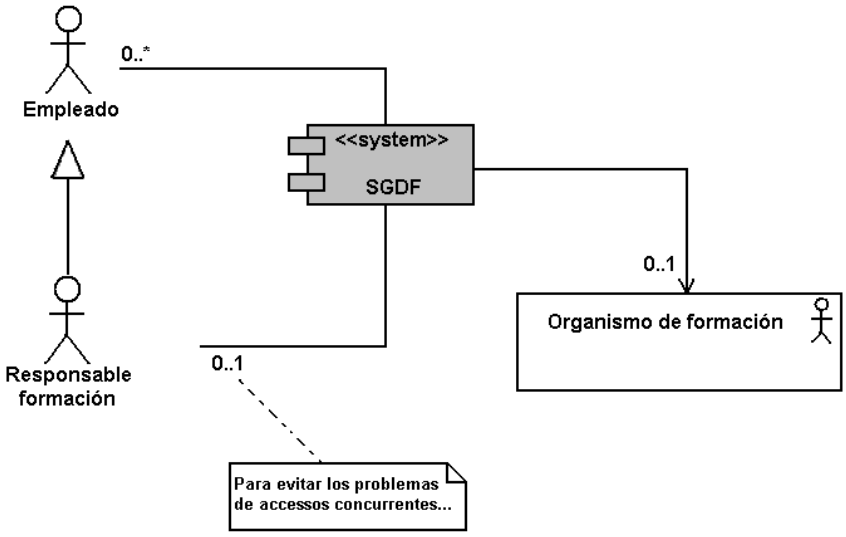


Figura 10: Diagrama de contexto estático del sistema de gestión de las solicitudes de formación.

Etapa 3: Análisis del ámbito (parte estática)

Vamos a reanudar el enunciado del caso de estudio, ya tratado desde el punto de vista funcional en las etapas 1 y 2, reformulándolo y simplificándolo ligeramente.

1. El proceso de formación es inicializado cuando el responsable de formación recibe una solicitud de formación por parte de un empleado.
2. Esta solicitud es informada por el responsable que la califica y transmite su acuerdo o desacuerdo al interesado.
3. En caso de acuerdo, el responsable busca en el catálogo las formaciones autorizadas para prácticas que corresponde a la solicitud.
4. Informa al empleado del contenido de la formación y le propone una lista de próximas sesiones.
5. Cuando el empleado devuelve su elección, el responsable de formación inscribe al participante a la sesión para el organismo de formación en cuestión.
6. El responsable de formación pide más tarde la factura que le envió el organismo de formación antes de transmitirlo al contable de las compras.

Ya identificamos a los trabajadores del negocio implicados en el proceso de formación. Debemos abordar ahora este último bajo el ángulo estático que presenta y descubrir las principales entidades del negocio.

Para esto, se indica totalmente un análisis léxico del texto del enunciado. Esta técnica se sub-utiliza en general, ya que puede parecer aburrida. Es ante todo muy eficaz para descubrir objetos candidatos en los casos difíciles, por ejemplo si el modelizador no conoce bien el ámbito del negocio.

Análisis lingüístico (1/6)

Un análisis simplista de los nombres y grupos nominales proporciona las entidades siguientes: proceso de formación, responsable de formación, solicitud de formación, empleado. Consideremos cada uno de los candidatos por rol.

- El *proceso de formación* ya se definió en la etapa 1 como proceso de negocio: no aparecerá en el diagrama de clases.
- En cambio, *responsable de formación* y *empleado* aparecerán, ya que se definieron como trabajadores de negocio.
- Artículos «uno/una» o «el/la». El artículo indefinido («uno/una») indica que el nombre se utiliza de manera genérica, mientras que el artículo definido («el/la») indica que el nombre es único en el contexto de la frase. Sin embargo: el artículo «uno» significa a menudo «uno en genera», (como en: cuando el responsable de formación recibe una solicitud de formación), pero también a veces «uno y uno solo» para indicar que el plural no sería posible (como en: por parte de un empleado). En ese caso, se obtiene una multiplicidad 1 sobre una asociación.

Deducimos fácilmente el diagrama de clases siguiente.

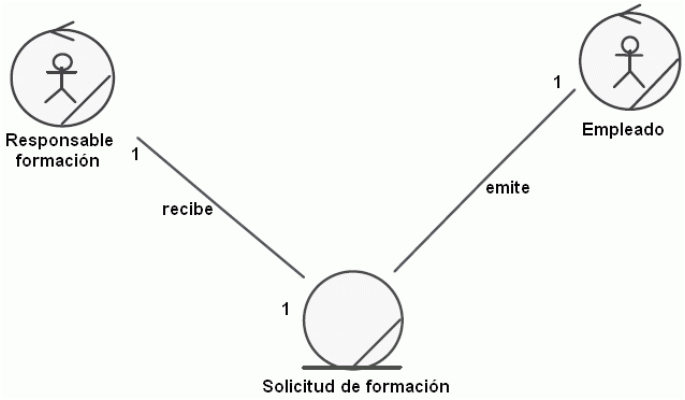


Figura 11: Modelización estática de la frase 1.

Análisis lingüístico (2/6)

Procediendo, como en la primera frase, se tiene un análisis simplista de nombres y grupos nominales, del cual se obtienen las siguientes entidades: solicitud, responsable, acuerdo, desacuerdo, el interesado.

- Referencia indirecta por «este/esta», «estos»: una frase que utiliza la palabra «este» hace casi siempre referencia al sujeto de la frase anterior. Los conceptos solicitud y solicitud de formación son pues idénticos.
- Atención a los sinónimos! Queda claro que responsable no es un nuevo concepto, pero simplemente una forma más corta de responsable de formación. Es algo menos evidente con la palabra *interesado* que hace referencia al empleado que emitió la solicitud.
- Posesivos: «su», «sus». Podemos traducir la posesión de dos maneras: una asociación o un atributo. Elegimos a la asociación si el poseedor y la posesión son ambos de los conceptos. Elegimos el atributo si la posesión es una simple característica del poseedor.
- Conjunción de coordinación «o». Un «o exclusivo» debe hacernos pensar en la relación de generalización/especialización, pero solamente si los conceptos especializados tienen atributos y comportamientos diferentes. En el caso contrario, es mejor introducir un simple tipo enumerado. En nuestro ejemplo, podemos considerar que el acuerdo o el desacuerdo son especializaciones de una entidad *respuesta* relativa a la solicitud. En efecto, el desacuerdo tendrá probablemente un atributo *motivo*, contrariamente al acuerdo.
- Verbos: el responsable recibe la solicitud, luego informa y por fin califica. No hay que dibujar a tres asociaciones para modelar todas las acciones que el responsable puede efectuar con respecto a la solicitud. Al contrario, el diagrama de clases debe representar una vista estática que sea válida en cualquier momento. Reelegimos pues a la asociación entre responsable y solicitud con un verbo más neutro (tratar), y en consecuencia modificamos las multiplicidades.

Para completar el diagrama, supusimos que un empleado no puede no emitir más que una solicitud a la vez. Se tendrán en cuenta también las multiplicidades entre *solicitud* y

respuesta: una respuesta está inevitablemente vinculada a una y solo a una solicitud; una solicitud puede existir sin respuesta (mientras no sea informada).

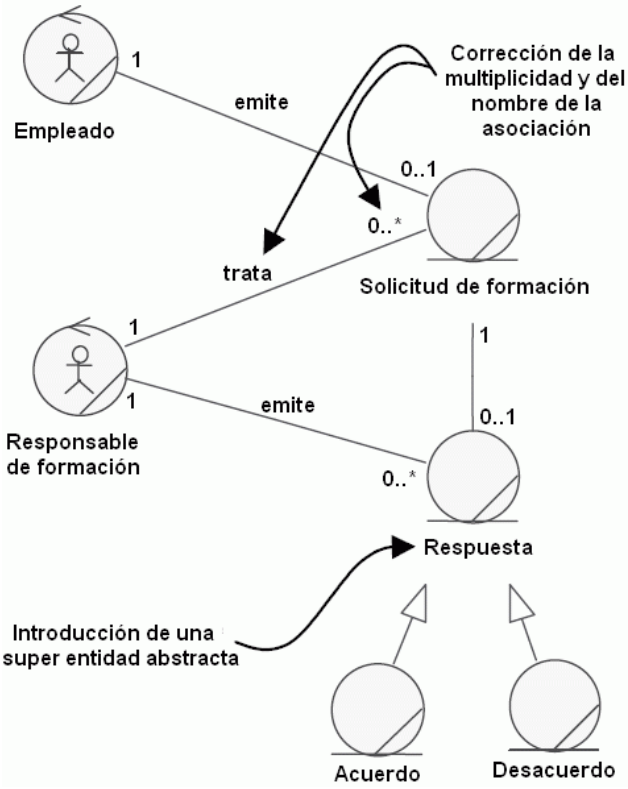


Figura 12: Modelización estática de la frase 2.

Análisis lingüístico (3/6)

Un nuevo análisis rápido de los nombres y grupos nominales proporciona las entidades siguientes: acuerdo, responsable, catálogo, formación, prácticas (stage), solicitud.

- *Acuerdo*, *responsable* y *solicitud* se identificaron anteriormente.
- Contenedor y contenido: *catálogo* es un contenedor compuesto de formaciones; los dos pueden dar lugar a entidades, si llevan atributos y comportamientos. Es el caso en nuestro ejemplo. Se debe entonces estudiar la posibilidad de una agregación o de una composición. Si no, el contenido puede ser un simple atributo del contenedor.
- Plural: el plural sobre un nombre (catálogo de las formaciones) da a menudo lugar a una entidad en singular, pero con una multiplicidad «0..* » sobre una asociación.
- Verbos: atención, los verbos corresponden a menudo a acciones efectuadas sobre las entidades (el responsable *busca*…). Estas acciones no se traducen generalmente en el diagrama de clases de análisis. Dan por lo tanto lugar a las indicaciones sobre la dinámica, y pueden dar lugar a fragmentos del diagrama de secuencia o comunicación.

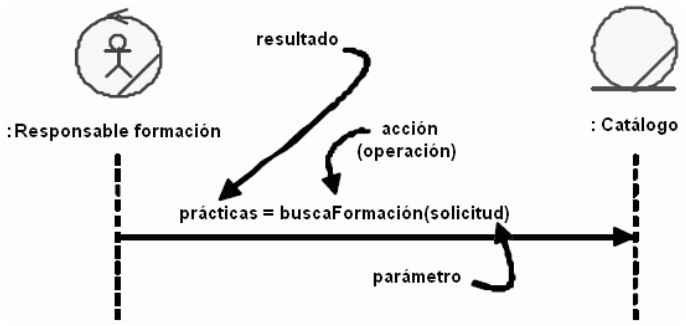


Figura 13: Fragmento del modelo dinámico resultante de la frase 3.

- Adjetivos: representan ya sea los atributos de una entidad ya identificada, o una posibilidad de relación de generalización. Atención: pueden también simplemente añadir «ruido» en el texto, como en nuestro caso donde solamente las *formaciones autorizadas* tienen una existencia notable en el proceso de formación.
- Participios presentes: indican a menudo una asociación entre dos entidades. Por ejemplo, «una práctica corresponde a la solicitud» nos lleva a la creación de una asociación entre las entidades *prácticas* (stage) y solicitud.
- Atención a los sinónimos! Para evitar las repeticiones que sobrecargan el estilo, el uso de los sinónimos es frecuente: *formación* y *prácticas* son una buena ilustración. El modelizador debe desechar estos sinónimos y «reducirlos» eligiendo un nombre principal de entidad. Preferiremos el término *formación* más bien que el de *prácticas*.

Toda esta discusión conduce al siguiente diagrama de clases.

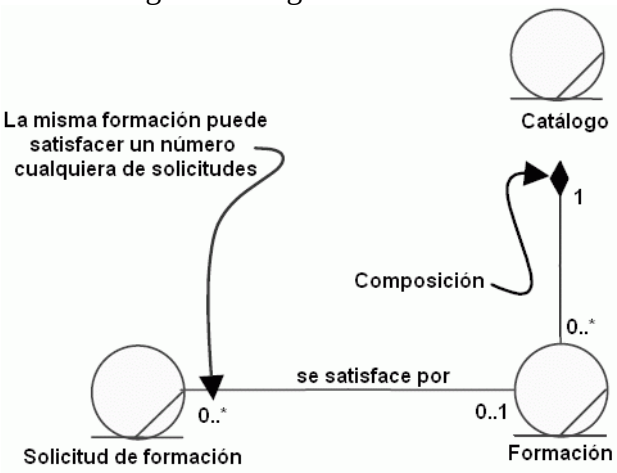


Figura 14: Modelización estática de la frase 3.

Análisis lingüístico (4/6)

Un análisis sencillo de los nombres y grupos nominales permite señalar las siguientes entidades: empleado, contenido, formación, lista, sesión.

- Referencia indirecta por un pronombre: «él/ella», etc. Los pronombres son referencias a otro nombre que es el tema a menudo de la frase anterior. Aquí, «él informa…» se refiere obviamente al responsable.
- *Empleado* y *formación* se han identificado anteriormente.
- Capacidad o posesión: entidad importante o atributo según los casos. Si se considera que una formación tiene un contenido cuya estructura es compleja (requisito previo, objetivos, plan detallado, etc.) y un comportamiento, se justifica totalmente utilizar una entidad. Como lo destacamos anteriormente, se debe estudiar la posibilidad de una agregación o de una composición.
- Contenedor: la palabra *lista* indica simplemente una multiplicidad «*» y aporta a menudo un concepto de orden (restricción UML {ordered}). No es necesario sobre todo definir una entidad *lista* en la fase de análisis: la elección de los tipos de contenedor incumbe de verdad al diseño detallado, o incluso a la programación.
- Atención a los falsos sinónimos! Esta vez, no es necesario creer que *sesión* es sinónima de *formación* o *prácticas*. Es decir, el concepto de *sesión* añade conceptos de fecha y lugar que no forman parte del concepto más genérico de *formación*. Estas entidades tienen comportamientos bien distintos: se puede aplazar o cancelar una *sesión*, sin modificarse de ningún modo la *formación*.
- Verbos: los verbos representan intercambios de mensajes entre instancias, y absolutamente no de las asociaciones.

Figura 15: Fragmento del modelo dinámico resultante de la frase 4.

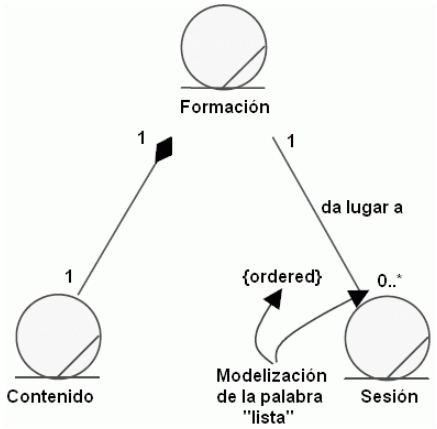


Figura 16: Modelización estática de la frase 4.

Análisis lingüístico (5/6)

Una vez más, el análisis lingüístico nos proporciona las entidades candidatas: empleado, elección, responsable de formación, participante, organismo de formación.

- Empleado, responsable y organismo de formación se han identificado anteriormente.
- Nuevamente, es necesario tener cuidando en no modelar un comportamiento dinámico en el diagrama de clases! La frase 5 se traduciría directamente en el fragmento de diagrama de secuencia siguiente.

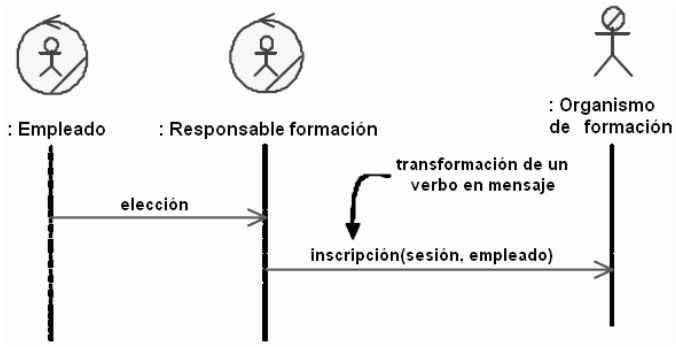


Figura 17: Modelización dinámica de la frase 5.

- Verbos: a menudo el verbo oculta un nombre! En el ejemplo anterior, donde «el responsable de formación *inscribe* al participante», el diagrama de secuencia incluye un mensaje *inscripción* que lleva parámetros. En realidad, necesitamos una entidad *inscripción* que representa una clase de contrato entre el responsable y el organismo externo. Esta entidad lleva atributos (fecha, precio, etc.) y comportamientos (reportar, cancelar, etc.). Las entidades de tipo *contrato* se modelan muy frecuentemente como clases de asociación.
- Términos vagos: la palabra *elección* es delicada a modelar. Es decir, se trata de una palabra imprecisa, de un término vago. Es necesario pues situarlo en el contexto al cual se produce algún beneficio. Según la frase 4, el empleado elige una de las sesiones propuestas por el responsable. La palabra *elección* en este contexto no sirve más que para definir una *sesión* particular para la cual el responsable va a presentar una solicitud de inscripción para el organismo de formación. No se trata pues de una nueva entidad, sino más bien de un rol desempeñado por una sesión en una relación con una inscripción.
- Roles: es necesario cuidar de no crear sistemáticamente nuevas entidades. En efecto, algunos nombres representan simplemente roles desempeñados por entidades ya definidas. Es el caso para *participante*, que sólo describe un papel desempeñado por un empleado en el marco de una *sesión*.
- Actores. Es necesario conectar *organismo de formación* a *sesión*? Es lo que parece indicar la frase 5. No obstante, como se vio en la frase 4 que las sesiones se refieren todas a una formación. Es pues más juicioso conectar directamente *organismo de formación* a *formación*.

La modelización estática de la frase 5 se ilustra en la figura siguiente.

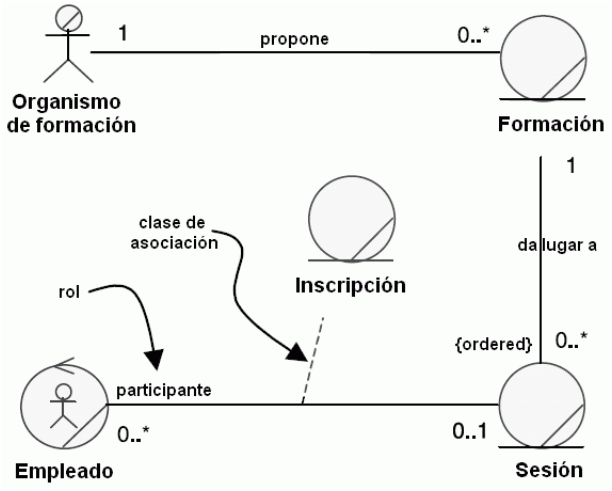


Figura 18: Modelización estática de la frase 5.

Análisis lingüístico (6/6)

Para esta última frase, también el análisis lingüístico nos proporciona las entidades candidatas: responsable de formación, consecuencia, factura, organismo de formación, contable de las compras.

- *Responsable de formación* y *organismo de formación* se han identificado anteriormente. *Contable de las compras* es un trabajador de negocios como nosotros lo habíamos indicado en la etapa 1.
- Proposiciones temporales: sólo sirven para la modelización dinámica. En nuestro caso, «pedir más tarde…» no hace más que indicar una sucesión temporal de mensajes. Permite implícitamente conectar la factura con la inscripción (ver frase 5).

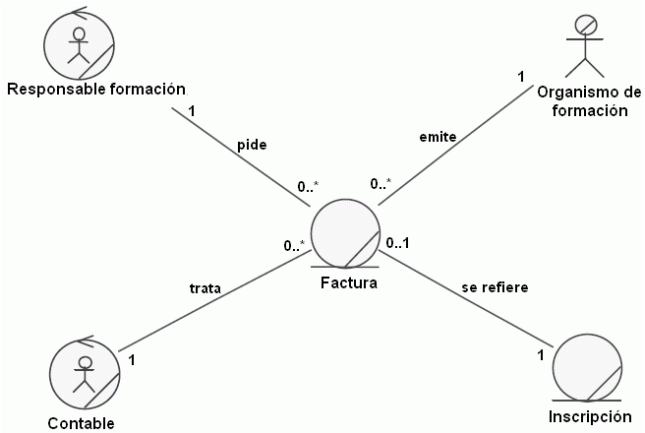


Figura 19: Modelización estática de la frase 6.

Modelo de negocios – recortado en paquetes

Para reunir todos los fragmentos anteriores en un mismo diagrama de clases. Nos lleva a proponer un recorte del modelo en paquetes que representan unidades de organización de negocios.

El modelo estático preliminar de nuestro caso de estudio procede de la reunión de todos los diagramas anteriores.

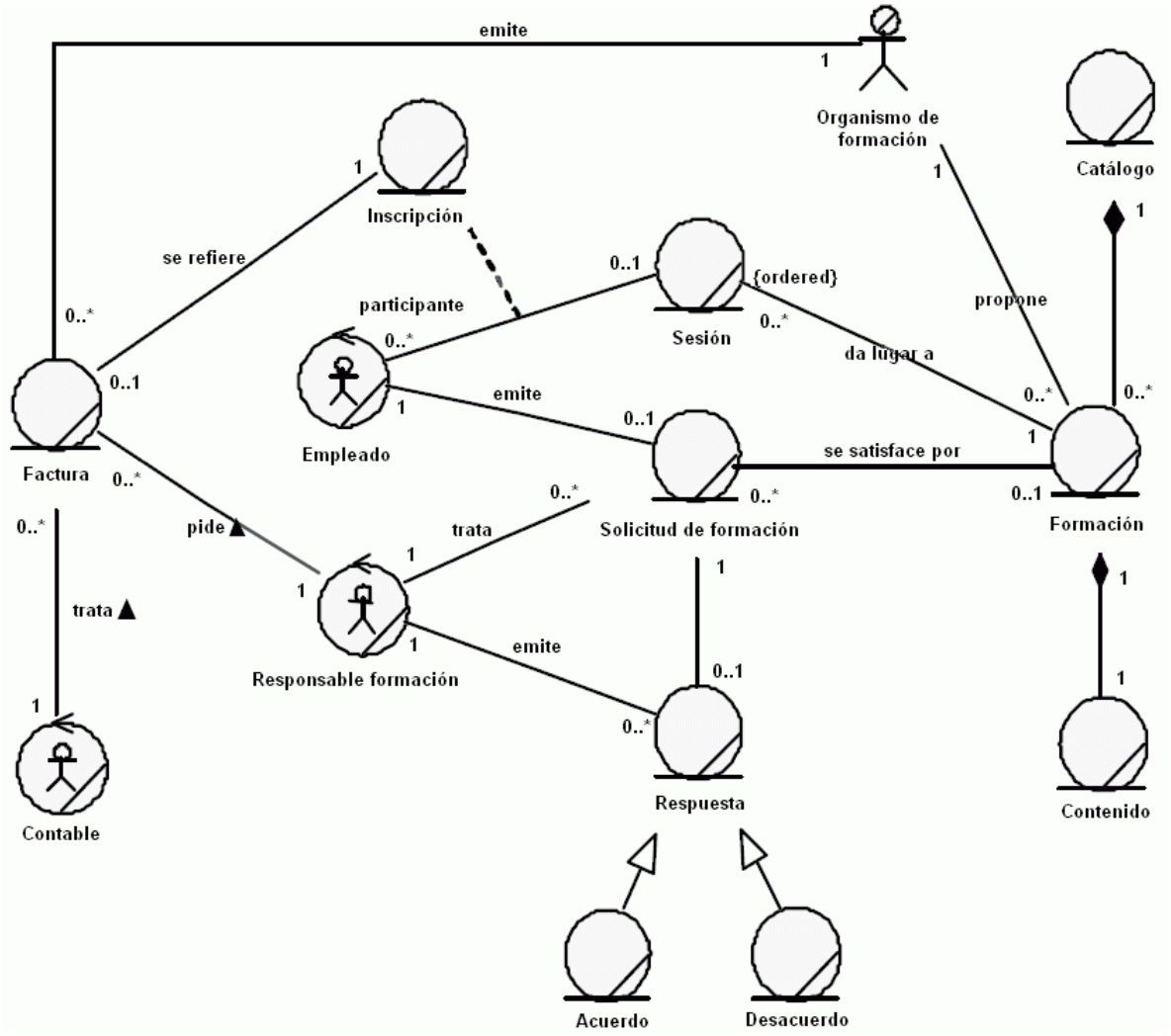


Figura 20: Modelización estática preliminar del caso de estudio.

Cómo proceder para recortar este modelo en unidades de organización de negocios?

- Queda claro que toda la parte derecha del modelo (incluida la entidad *sesión*) se refiere al catálogo de formación y constituye una unidad coherente, relativamente estable en el tiempo.

- El par factura-contable es también relativamente independiente del resto, y corresponde por otra parte a un servicio bien identificado de la empresa.
- Todo el resto es de la responsabilidad del responsable de formación y constituye un conjunto coherente, centrado en la solicitud de formación.

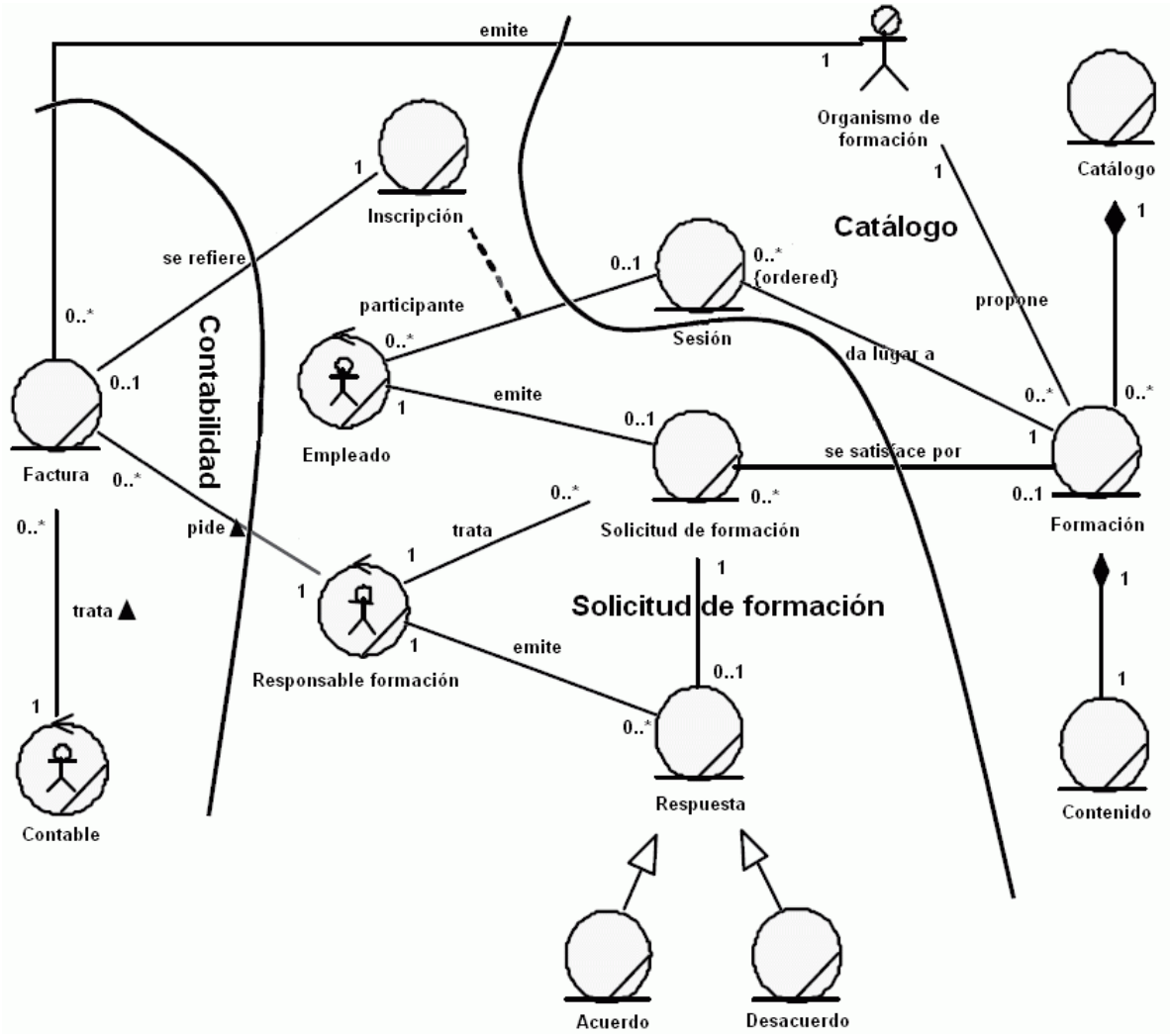


Figura 21: Recorte del modelo estático del caso de estudio.

Podemos representar esta estructuración recortando el esquema anterior gracias a los paquetes estereotipados, como se indicó en la etapa 1.

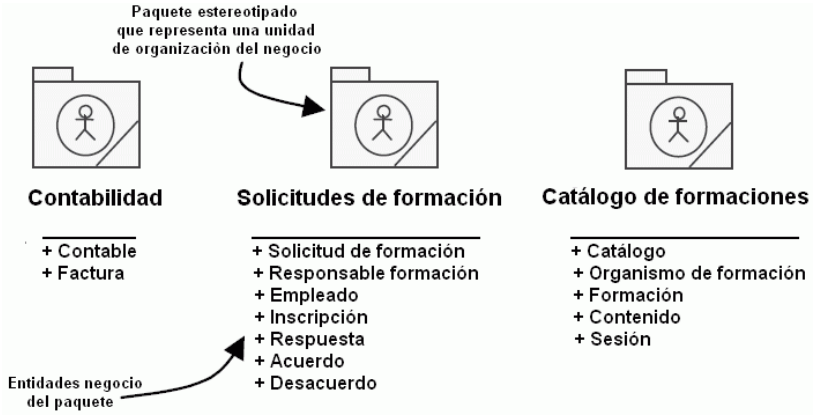


Figura 22: Paquetes estereotipados que representan el recorte de modelo de negocios.

Modelo de negocios – diagrama de clases por paquete

Debemos diseñar un diagrama de clases por unidad de organización, intentando minimizar las dependencias entre paquetes. Añadimos algunos atributos de negocios pertinentes para completar el modelo de negocios estático.

Queda claro que el paquete *Catálogo de formación* puede ser autónomo y que puede de hecho constituir un componente de negocios reutilizable. Es también lógico considerar que la factura de la solicitud depende de la formación más bien que el caso contrario. El esquema de dependencias entre unidades de organización de negocios que se obtiene es pues el que se presenta a continuación. Se trata de un diagrama de paquetes que respeta los sagrados principios de las dependencias entre paquetes:

- no hay dependencias mutuas;
- no hay dependencias circulares.

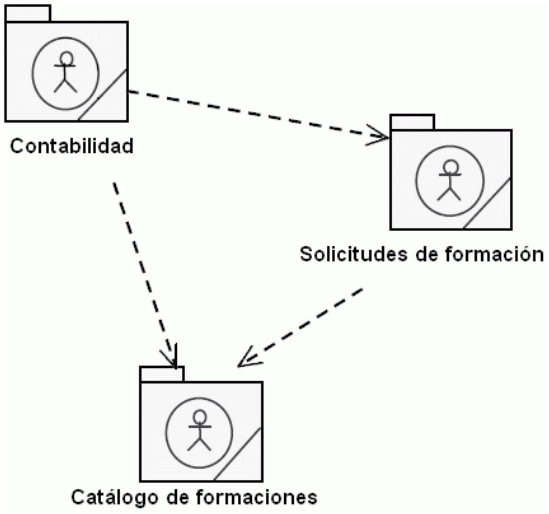


Figura 23: Los diagrama de paquetes muestran las dependencias deseadas entre paquetes.

Este objetivo de dependencias entre paquetes impone una dificultad sobre la navegabilidad de las asociaciones que cruzan dos unidades de organización, de la siguiente forma.

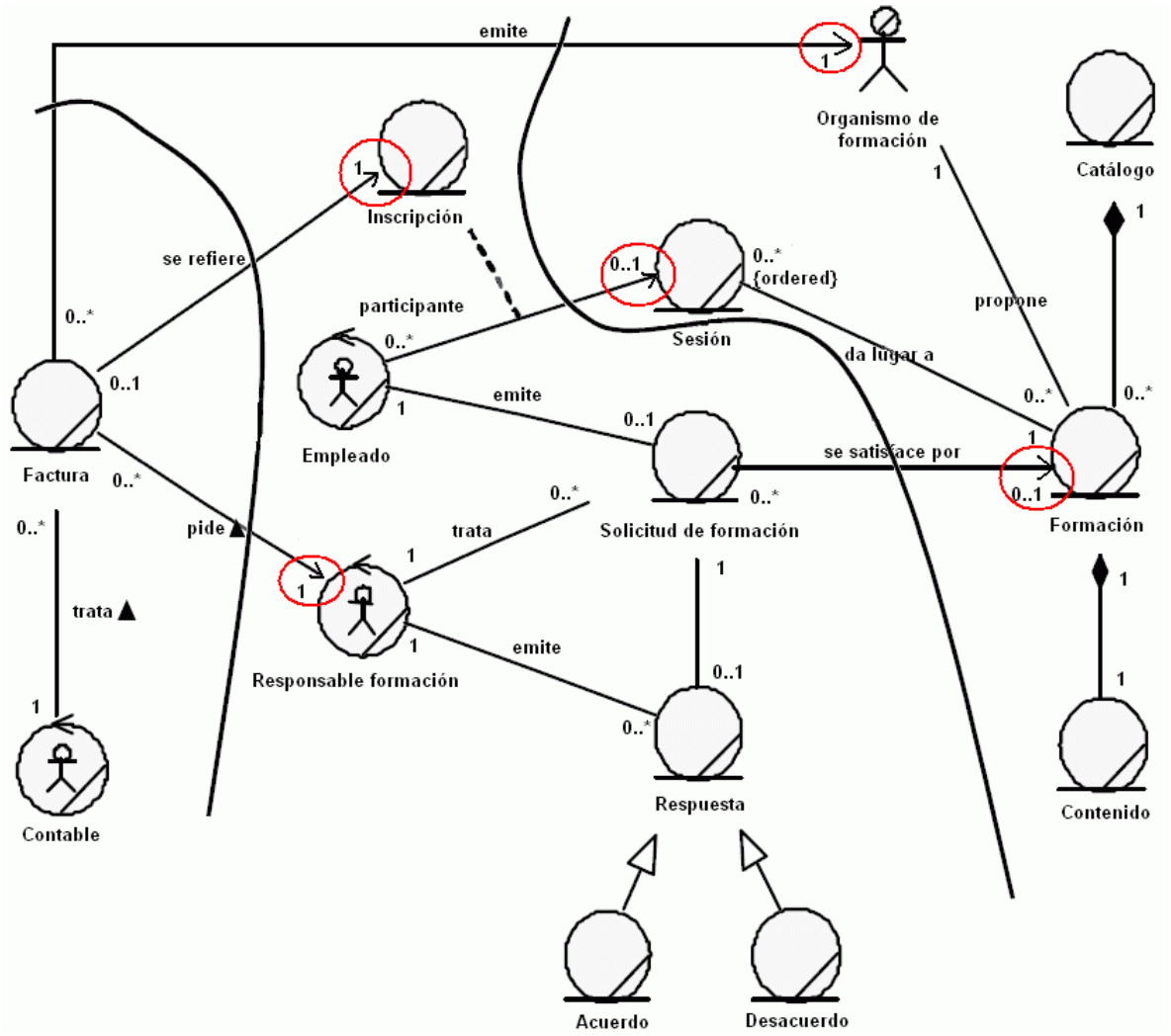


Figura 24: Adición de las navegabilidades sobre las asociaciones que cruzan dos paquetes.

Podemos ahora dibujar un diagrama de clases por paquete añadiendo algunos atributos de negocio pertinentes.

Tengamos en cuenta la presencia en los diagramas de las clases conectadas que pertenecen a los otros paquetes (con la indicación «(from xxx)»).

Esta convención gráfica eficaz no es estándar de UML, se ha más bien popularizado por el software Rational/Rose e incluso el software Enterprise Architect la incluye.

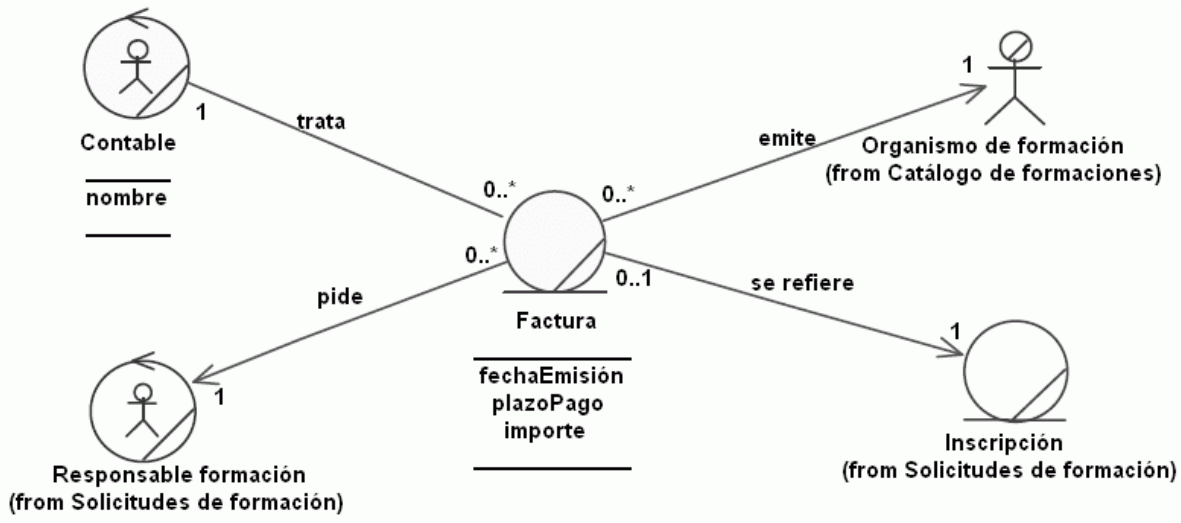


Figura 25: Diagrama de clases del paquete Contabilidad.

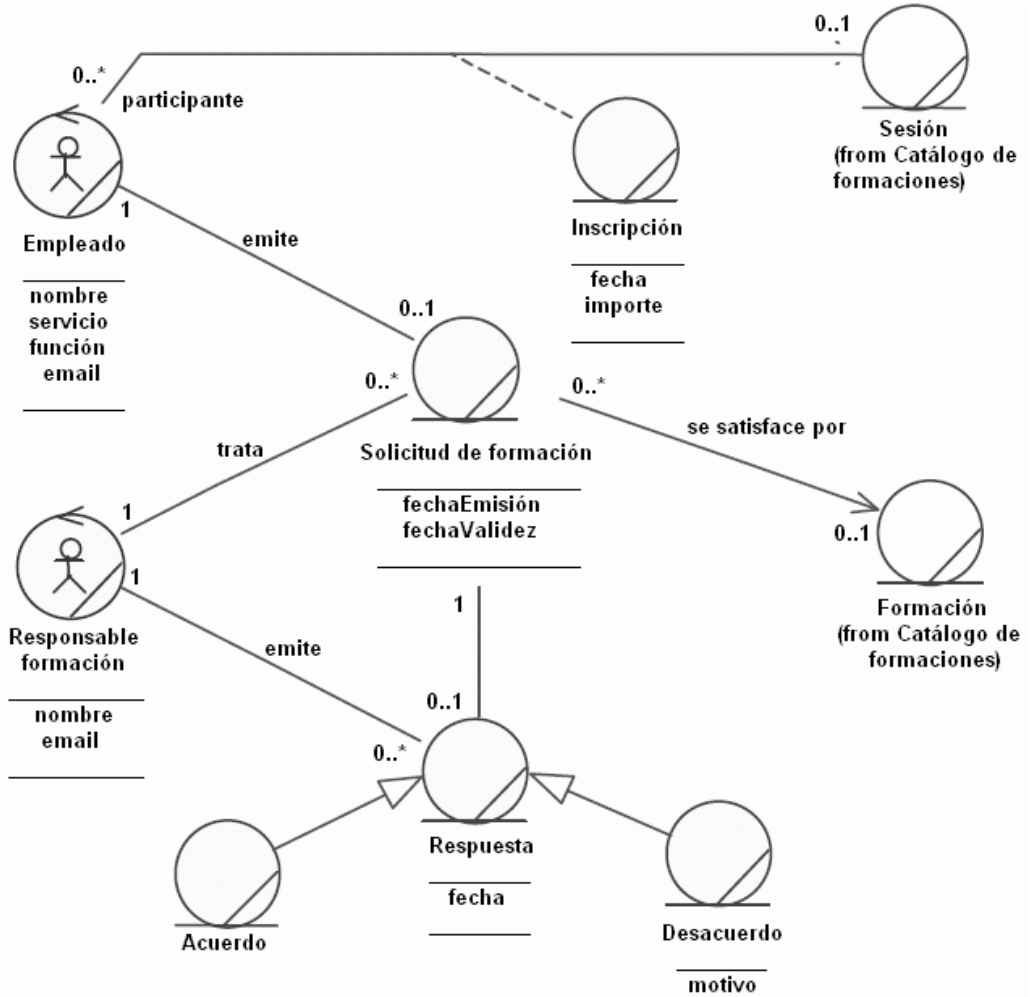


Figura 26: Diagrama de clases del paquete Solicitudes de formación.

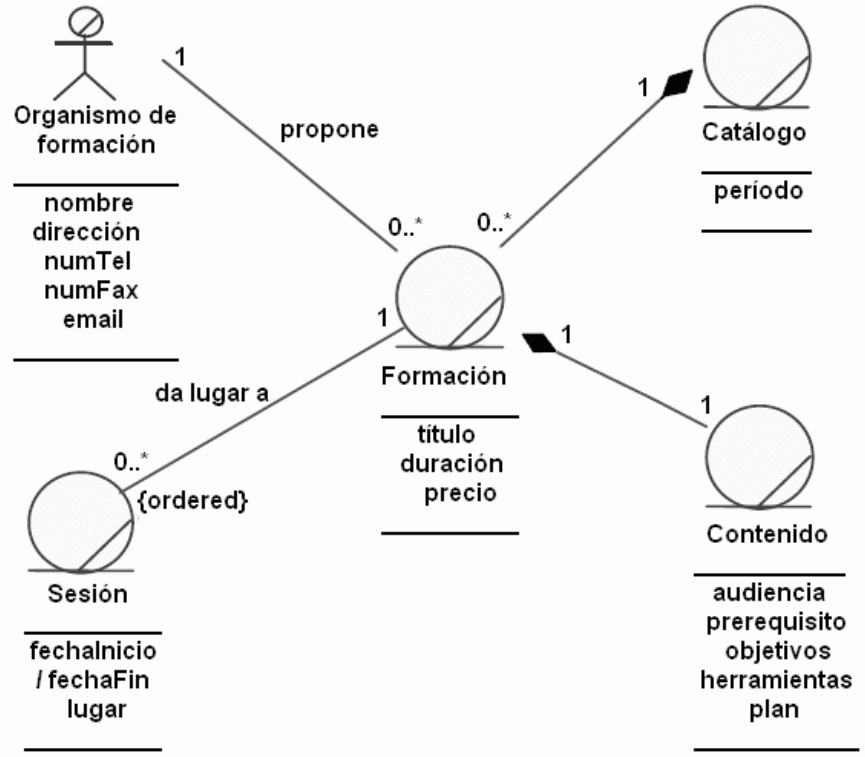


Figura 27: Diagrama de clases del paquete Catálogo de formaciones.

Se tendrá en cuenta el atributo derivado /fechaFin de *Sesión*. La restricción podría escribirse simplemente en OCL de la siguiente manera:

{context Session inv: self.dateFin = self.dateInicio + self.formacion.duración}.

OCL (Object Constraint Language) es un pequeño lenguaje de expresiones, que permite expresar restricciones sobre los diagramas de clases UML por medio de expresiones booleanas que deben ser verificadas por el modelo. OCL forma parte integral de UML desde la versión UML 1.1.

Etapa 4: Análisis del ámbito (parte dinámica)

Con qué información contamos para modelar la dinámica de una solicitud de formación? Retomando las tres primeras frases del enunciado:

1. El proceso de formación es inicializado cuando el responsable de formación recibe una solicitud de formación por parte de un empleado. Esta solicitud se instruye por el responsable que califica dicha solicitud y transmite su acuerdo o su desacuerdo al interesado.
2. En caso de acuerdo, el responsable busca en el catálogo las formaciones autorizadas para un período de prácticas que corresponde a la solicitud. Informa al empleado del contenido de la formación y le propone una lista de las próximas sesiones. Cuando el empleado devuelve su elección, el responsable de formación inscribe al participante a la sesión con el organismo de formación correspondiente.
3. En caso de impedimento, el empleado debe informar al responsable de formación cuanto antes para cancelar la inscripción o la solicitud.

Habíamos realizado también un diagrama de actividad del proceso de formación mostrando los objetos de negocio y sus cambios de estados.

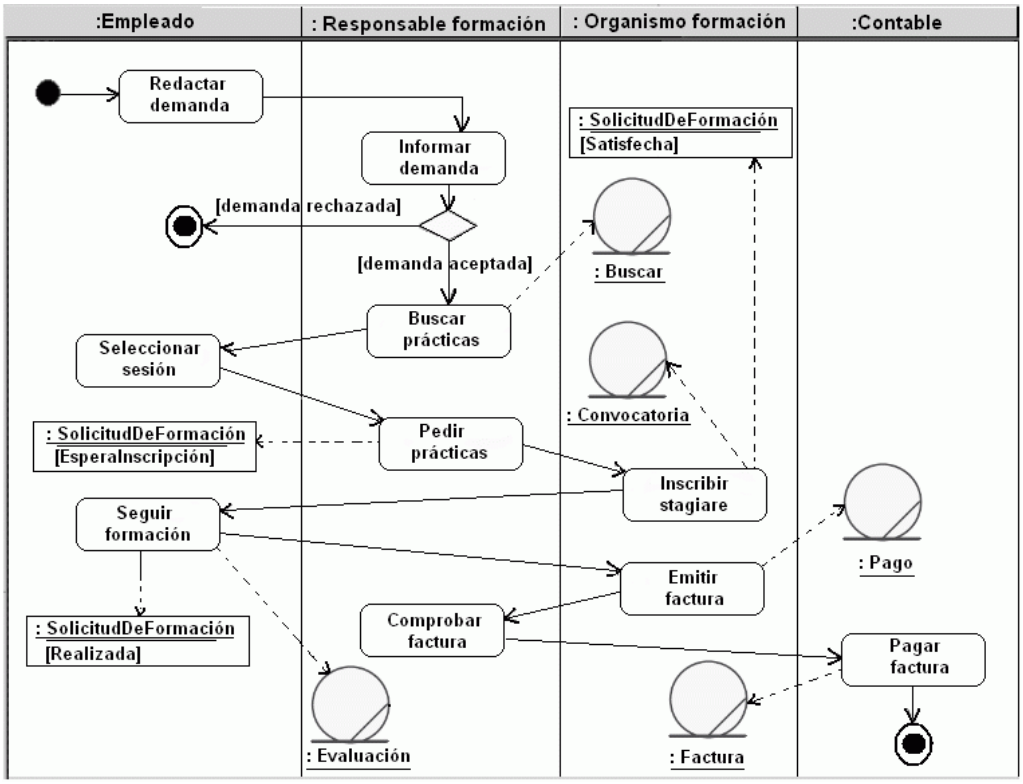


Figura 28: Diagrama de actividad del proceso de formación.

A partir de este diagrama de actividad, podemos en primer lugar identificar a cuatro estados principales de la solicitud de formación, como se muestra en la siguiente figura.

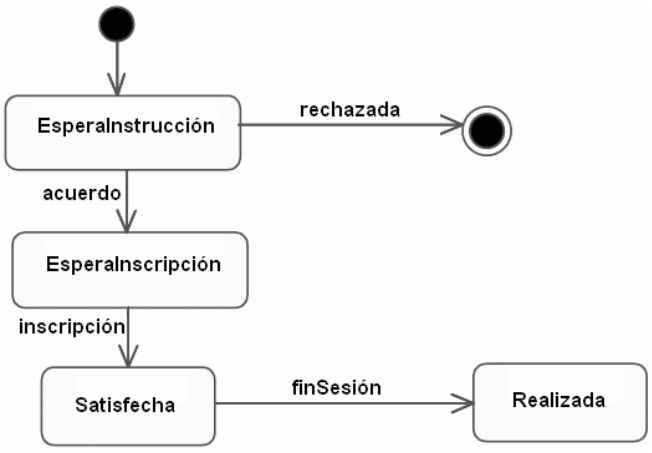


Figura 29: Diagrama de estados inicial de la solicitud de formación.

Al releer atentamente la primera frase, nos damos cuenta que la solicitud es iniciada por el empleado y enviada al responsable de formación, luego se instruye por este último quien transmite su acuerdo o desacuerdo al interesado. Para poder completar el diagrama de estados, vamos a reflexionar que pasa después de la vida de una solicitud. Esto nos lleva a añadir un estado antes de EsperaInstrucción, puesto que es la validación de la solicitud que desencadena la transmisión al responsable de formación.

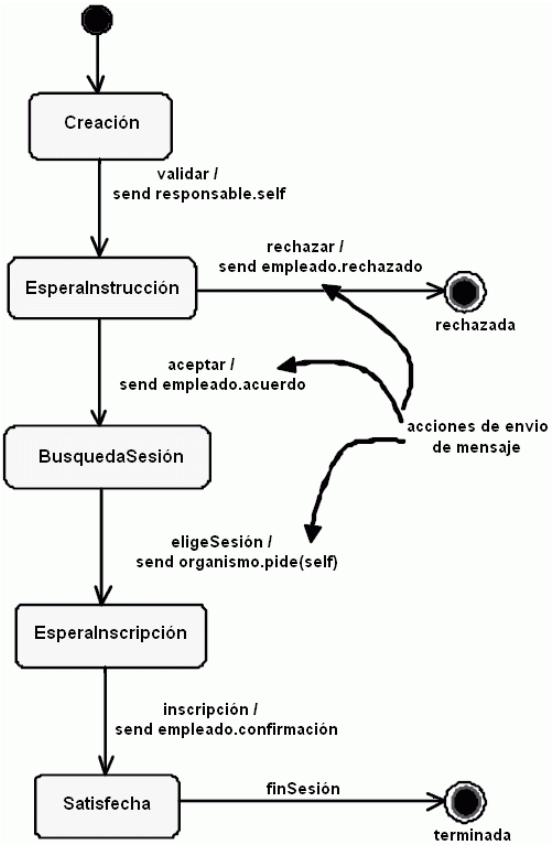


Figura 30: Segunda versión del diagrama de estados de la solicitud de formación.

La creación propiamente dicha de esta solicitud no es atómica, ya que el empleado debe efectuar varias selecciones (tema, período, etc.) antes de proceder a la validación. Definimos también las acciones de envío de mensaje que se materializarán por la palabra clave «send» sobre las transiciones del diagrama de estados.

Una nueva versión más completa del diagrama de estados se muestra en la figura 30.

Qué puede faltarnos para terminar nuestro diagrama de estados? En realidad, todas las transiciones de anulación o de error. El empleado puede así cancelar su solicitud en cualquier momento, el organismo de formación puede indicar una anulación de sesión, etc.

El diagrama de estados completo está representado en la figura 31.

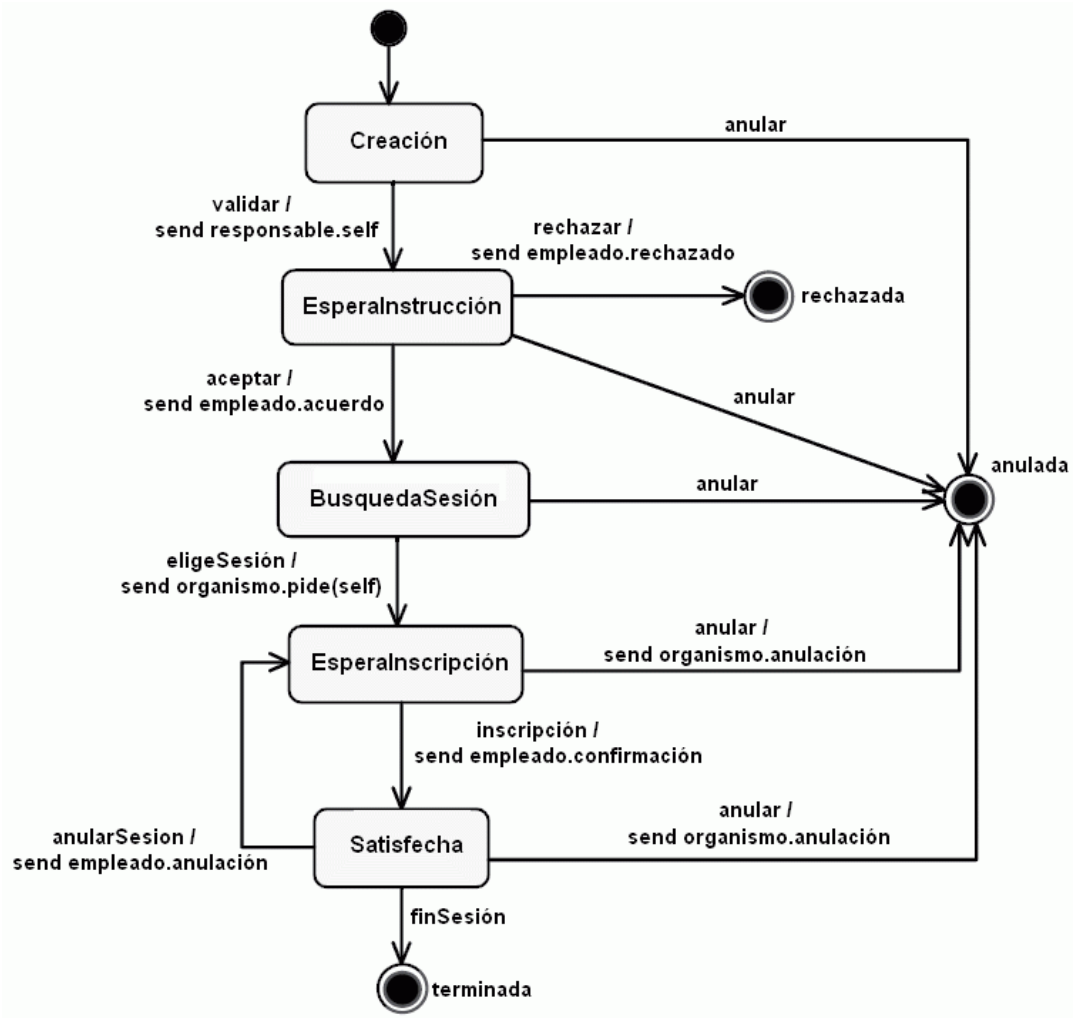


Figura 31: Diagrama de estados completo de la solicitud de formación.

Etapa 5: Definición de iteraciones

Vamos ahora a definir iteraciones a partir del trabajo ya efectuado y a fijarnos como objetivo el diseño de la primera de estas iteraciones con el lenguaje Java como objetivo principal.

Planificación de iteraciones

Propondremos un recorte del proyecto en iteraciones a partir del trabajo de análisis anterior (casos de uso y modelo de negocios estático). Tomaremos en cuenta el espíritu de uno de los principios principales del proceso unificado: el guiado por los casos de uso…

En vista de las dependencias entre los paquetes de negocios, así como entre los casos de uso, parece natural comenzar por la gestión del catálogo. Es decir, los dos paquetes de negocio dependen del Catálogo de formaciones y el caso de uso Solicitar una formación es conectado por extensión al caso Consultar el catálogo. Elegimos pues realizar los dos casos de uso que se refieren al catálogo en la primera iteración.

Para la segunda iteración, es indispensable ocuparse del caso de uso principal del sistema, es decir Solicitar una formación. En una tercera iteración, trataremos los aspectos más administrativos (inscripción, etc.) con Tratar las solicitudes. Finalmente, la última iteración se dedicará a Administrar sus solicitudes, algo menos importante funcionalmente hablando.

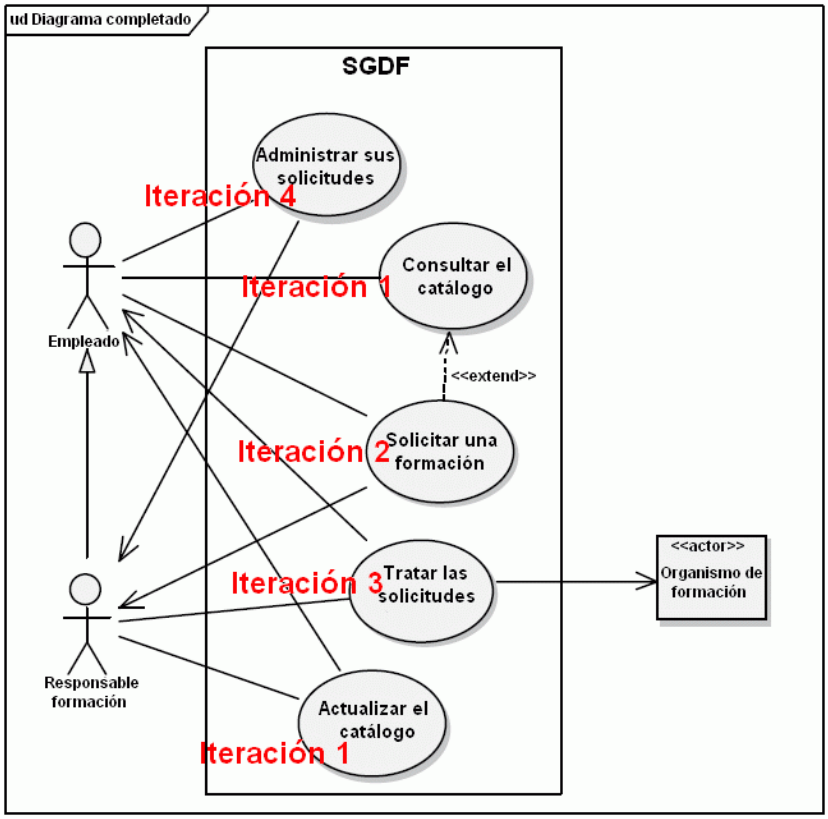


Figura 32: Proposición de distribución de los casos de uso en iteraciones.

Hay que tener en cuenta que el servicio más técnico de autenticación del empleado o del responsable de formación sobre la intranet puede ser realizado en paralelo en los casos de uso funcionales.

Etapa 6: Definición de la arquitectura del sistema

Los sistemas computacionales modernos se organizan en capas horizontales, estas mismas capas recortadas en divisiones verticales. Este recorte es en primer lugar lógico, luego eventualmente físico en términos de máquinas.

El problema general de la arquitectura de los sistemas computacionales no es el tema de este caso de estudio. Sin embargo, aprovecharemos este espacio para recordar algunas ideas fundamentales sobre las arquitecturas en capas denominadas «n-tiers», así como sobre los diagramas UML que son útiles para esta actividad.

ARQUITECTURA EN TRES NIVELES

La arquitectura a tres niveles, se ha convertido ahora en clásica, era buena, al principio, una división lógica, pero se interpretó erróneamente pensando que podía implicar nudos de ejecución físicamente separados.

El principal objetivo de esta separación en tres capas (3-tiers) es de aislar la lógica de negocios de las clases de presentación (IHC), así como de de prohibir un acceso directo a los datos almacenados por estas clases de presentación. La primera preocupación es responder al criterio de carácter evolutivo: poder modificar la interfaz de la aplicación sin deber modificar las reglas de negocio, y poder cambiar de mecanismo de almacenamiento sin tener que retocar la interfaz, ni las reglas de negocios.

A manera de ejemplo, podemos considerar el caso de una caja registradora de un supermercado.

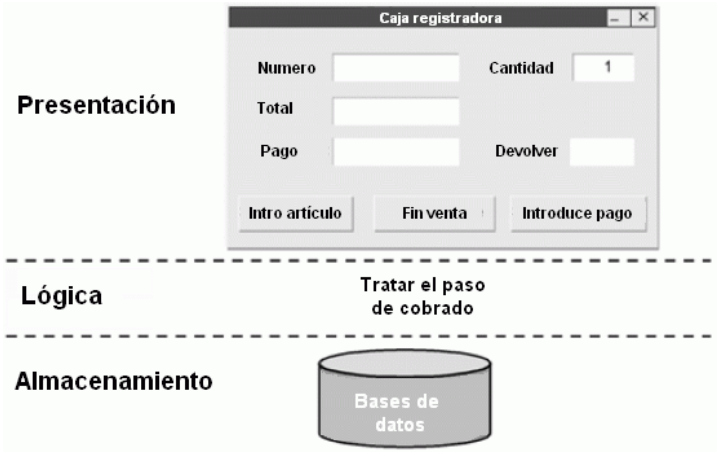


Figura 33: Arquitectura de tres capas de una caja registradora.

No se considera en la actualidad que esta descomposición en tres capas sea suficiente, si se tienen objetivos importantes de modularidad y reutilización. Esto conduce a los objetos gráficos de presentación a conocer el detalle de la capa lógica, lo que daña su capacidad de mantenimiento y su reusabilidad.

Para mejorar este estado, la idea consiste en introducir un objeto artificial, a menudo llamado «controlador» (se trata del segundo patrón GRASP propuesto por Larman. El nombre de controlador también hace referencia al patrón muy conocido MVC (Model - View - Controller), así como a las clases de «control» de Jacobson), entre los objetos gráficos y los objetos de negocios. Es el objeto de diseño controlador quien conoce ahora la interfaz de los objetos de la capa de negocios y que desempeña el rol de «fachada» frente a la capa de presentación, como eso se ilustra en la figura siguiente.

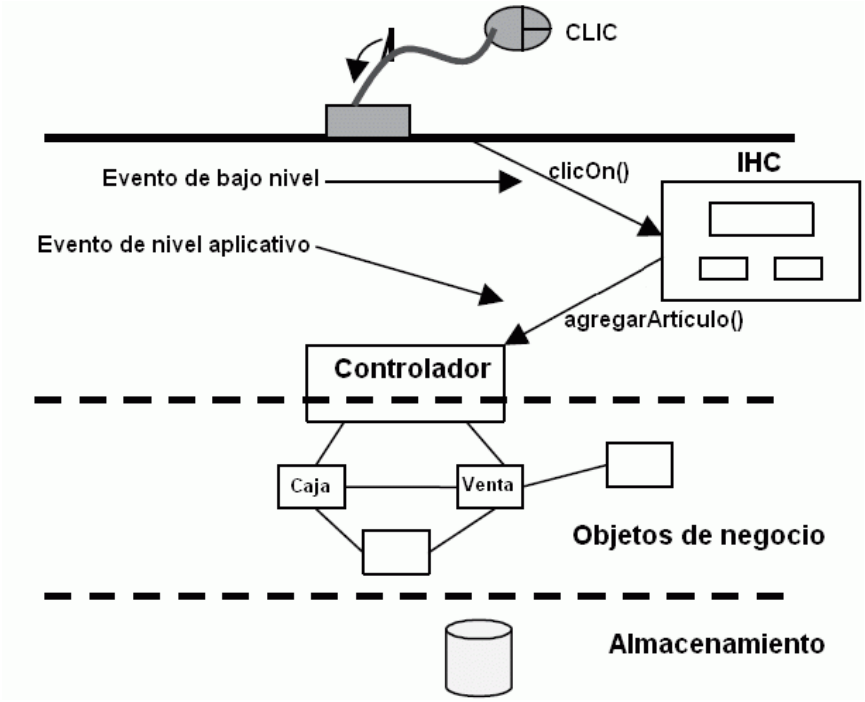


Figura 34: Diagrama que ilustra la adición del objeto controlador.

Todos estos nuevos objetos controladores, introducidos en diseño, van a ser ensamblados en una nueva capa llamada «lógica aplicativa», que contribuye a realizar los casos de uso del sistema, y a aislar la capa de presentación de los objetos de negocio, a menudo persistentes y muy reutilizables.

La arquitectura en tres capas puede así completarse con dos capas suplementarias que representan para la primera estos objetos controladores que desacoplan la presentación del negocio y para la segunda los servicios técnicos generales como el acceso a las bases de datos, la generación de informes, etc.

Vamos a utilizar estos principios de arquitectura con varias capas en lo que sigue de nuestro caso de estudio, en el marco del sistema de gestión de las solicitudes de formación.

PAQUETES, CAPAS Y PARTICIONES

En UML, el único mecanismo de reagrupación de clases disponible es el paquete. Por lo tanto, las capas horizontales y las particiones verticales se traducen también en paquetes.

Así pues, una arquitectura en capas se describe por un diagrama estático que sólo muestra paquetes y sus dependencias. UML 2 reconoció la importancia de este tipo de diagrama de alto nivel oficializando el diagrama de paquetes como un tipo de diagrama UML. Puede utilizarse la palabra clave «capa» para distinguir los paquetes que representan las capas.

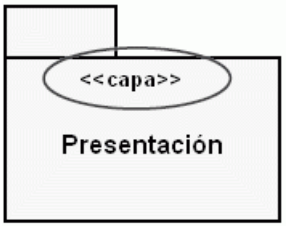


Figura 35: Representación de UML de una capa de software.

Arquitectura en capas preliminar

Para esta parte, se propone un diagrama de arquitectura preliminar del proyecto sobre la base de las recomendaciones anteriores.

Describimos pues un paquete estereotipado «capa» por capa de software. Dentro de cada capa, damos una estructura preliminar en particiones.

La capa de negocio incluye a priori los tres paquetes identificados en la etapa 3: Contabilidad (la parte relativa a la contabilidad no forma con todo rigor parte del sistema computacional, tal como se indica en el estudio de los casos de uso. La conservamos sin embargo en la consecuencia del ejercicio para trabajar sobre una arquitectura lógica de tamaño consecuente), Solicitudes y Catálogo. Este recorte podrá ser precisado por la consecuencia en nuestro estudio; sólo se trata de una estructuración preliminar.

La capa aplicativa no está detallada sobre el esquema. Puede estructurarse ya sea idéntica de la capa de negocios, o por el contrario de un punto de vista funcional copiando los paquetes de casos de uso.

La capa de presentación agrupa más bien las clases gráficas de las interfaces respectivas del responsable y el empleado.

La capa de los servicios técnicos incluye al menos un paquete para administrar el servicio técnico de autenticación, identificado a partir de la etapa 1.

De hecho, no es necesario olvidar las clases básicas Java proporcionadas por el JDK y que son utilizadas por todas las capas. La capa de presentación por ejemplo va a utilizar las clases gráficas. La capa de los servicios técnicos por su parte va en particular a utilizar las

clases JDBC de acceso a las bases de datos relacionales. Todas las capas van a servirse de las clases básicas como los contenedores, las fechas, etc.

Es necesario no obstante considerar bien que esta arquitectura preliminar podrá ser precisada o modificada (principalmente en las particiones dentro de cada capa) por el trabajo de diseño que va a seguir más adelante. No olvidar que el proceso de análisis/diseño es básicamente iterativo.

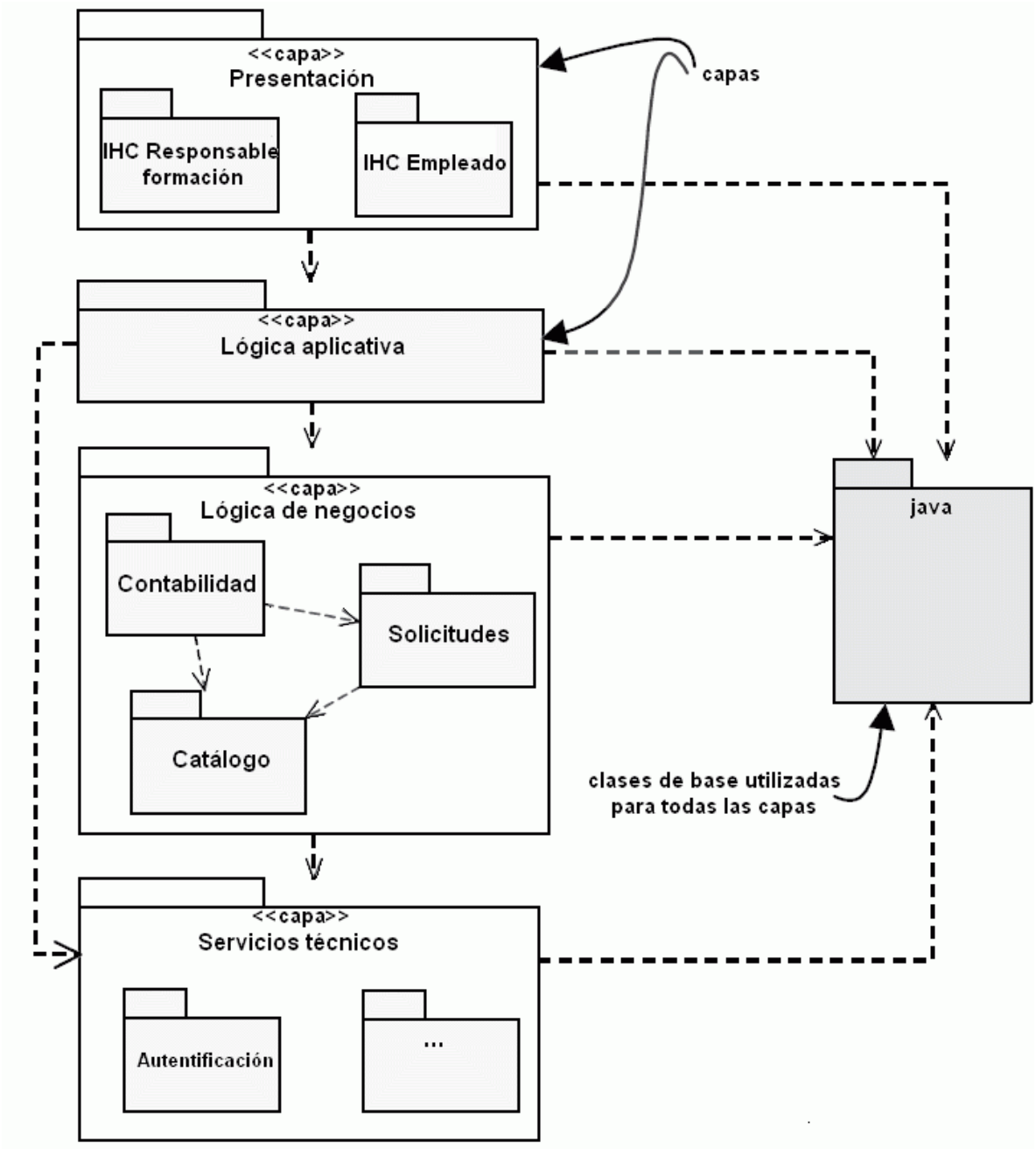


Figura 36: Arquitectura en capas del sistema de gestión de las solicitudes de formación.

Etapa 7: Definición de las operaciones del sistema (iteración #1)

La primera iteración corresponde a los casos de uso Consultar el catálogo y Actualizar el catálogo. Se procedió a una descripción de alto nivel en la etapa 1. Recordemos: El responsable de formación puede introducir una nueva formación en el catálogo, modificar una formación existente o suprimir una formación eliminada por un organismo. Puede también modificar las reagrupaciones de formaciones llamadas temas. Tiene por eso la posibilidad de actualizar las fechas y lugares de las sesiones.

Para pedir una formación y mantener el catálogo, el sistema debe proponer una funcionalidad básica de consulta del catálogo.

Las operaciones del sistema para el caso de uso Actualizar el catálogo se deducen fácilmente de su descripción de alto nivel. Es necesario sin embargo pensar en la creación y el mantenimiento de los organismos de formación, lo que no aparece claramente en el texto.

Las operaciones del sistema se reúnen en el esquema siguiente, donde una clase simboliza el sistema visto como una caja negra, con sus operaciones.

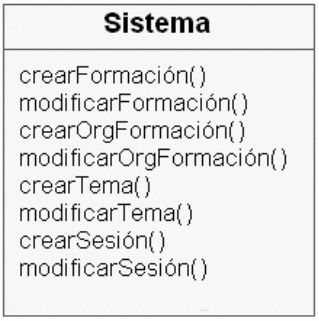


Figura 37: Operaciones del sistema para el caso de uso Actualizar el catálogo.

Para simplificar, consideramos que la acción de modificación incluye siempre la supresión y omitimos las operaciones de consulta pura.

Definimos las operaciones del sistema en la etapa anterior. Pero como podemos especificar el resultado de la ejecución de una operación del sistema?

CONTRATO DE OPERACIONES

Larman propuso establecer un «contrato» para cada operación del sistema.

Un contrato de operación describe los cambios de estado del sistema cuando se efectúa una operación del sistema. Se expresan estas modificaciones en términos de «postcondiciones» que enumeran el nuevo estado del sistema después de la ejecución de la operación.

Las principales postcondiciones se refieren a la creación (o la destrucción) de objetos y de los vínculos resultantes del modelo estático de análisis, así como la modificación de valores

de atributos. Los contratos de operaciones permiten así efectuar el vínculo entre el punto de vista funcional/dinámico de los casos de uso y el punto de vista estático del análisis.

Un plan tipo de descripción textual de contrato de operación se detalla a continuación:

- nombre
- responsabilidades
- referencias
- precondiciones
- postcondiciones
- excepciones (opcional)
- notas (opcional)

En primer lugar, vamos a extraer del diagrama de clases del paquetes Catálogo de formaciones (ver figura 27) la parte afectada por nuestra cuestión. En efecto, las operaciones de sistema crearFormacion y crearTema van a actuar sobre objetos y vínculos provenientes del siguiente diagrama:

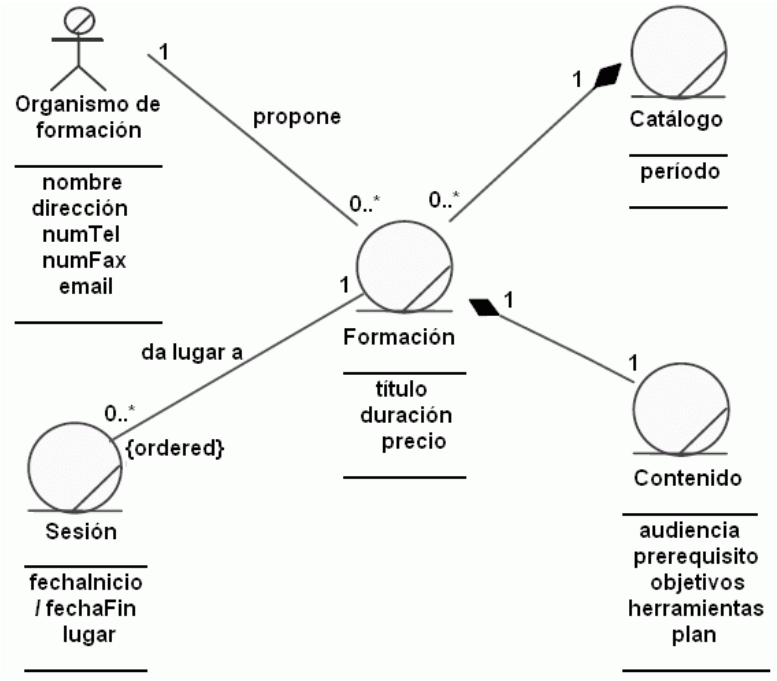


Figura 38: Diagrama de clases del paquete Catálogo de formaciones.

Sin embargo, faltaba el concepto de tema en nuestro modelo de negocios. Este concepto de tema es puramente aplicativo: facilita el trabajo del empleado en una solicitud de formación permitiéndole seguir siendo impreciso voluntariamente y no elegir una formación particular, pero más bien un conjunto de formaciones sobre un tema dado.

Supondremos que los temas estructuran el catálogo, pero que lo no particionan una formación pertenece al menos a un tema. La siguiente figura muestra las modificaciones aportadas por la introducción del concepto de tema.

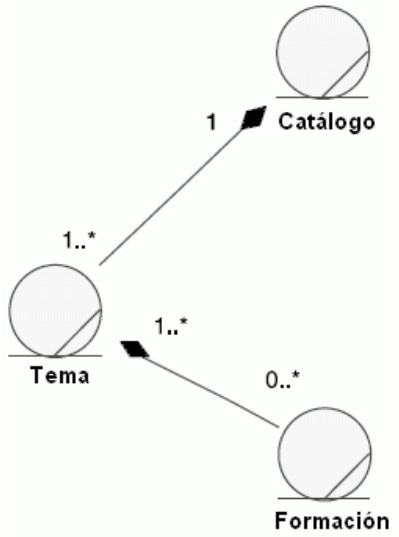


Figura 39: Introducción del concepto de tema.

Podemos ahora describir el contrato de la operación crearFormación:

- Nombre
  crearFormación.
- Responsabilidades
  Crear una nueva formación según la descripción proporcionada por el organismo de formación en cuestión y clasificarlo en al menos uno de los temas existentes.
- Referencias
  Caso de uso *Actualizar el catálogo*.
- Precondiciones
  - el catálogo de formaciones existe;
  - hay al menos un tema en el catálogo;
  - el organismo proveedor de la formación ya existe en el catálogo;
  - el responsable se conecta a la intranet.
- Postcondiciones
  - se creó una formación f con sus atributos;
  - se creó un objeto contenido c con sus atributos;
  - c se ha vinculado con f;
  - f se vinculó con el organismo proveedor;
  - se crearon posibles objetos sesiones con sus atributos;
  - se vincularon estos objetos sesiones a f;
  - f se vinculó con al menos un tema.

Etapa 8: Diagramas de interacción (iteración #1)

Los contratos de operaciones constituyen el último disponible (entregable) en cuanto al análisis. Es decir, si describen lo que hace una operación en términos de cambios de estado, no deben aún describir cómo procede esta operación.

Es precisamente el trabajo del diseñador elegir cómo los objetos computacionales van a interactuar entre ellos para realizar tal o cual operación. Jacobson propuso el primero (aún él!) de los estereotipos de clases para describir la realización de un caso de uso. Vamos a inspirarnos en sus trabajos para sustituir al sistema visto como una caja negra (desde el punto de vista del análisis) por objetos computacionales (desde el punto vista del diseño), como es ilustrado por los diagramas de secuencia presentados a continuación.

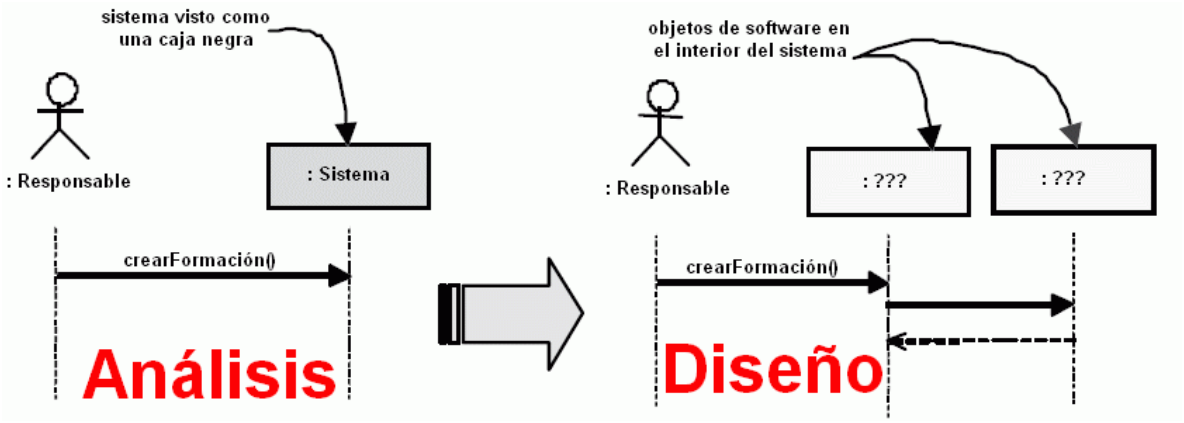


Figura 40: Paso del análisis al diseño.

ESTEREOTIPOS DE JACOBSON

Dentro del sistema, Jacobson distingue los tres estereotipos siguientes:

- «diálogo» clases que sirven para modelar las interacciones entre el sistema y sus actores;
- «control» clases utilizadas para representar la coordinación, el encadenamiento y el control de otros objetos - se conectan en general a un caso de uso particular;
- «entidad» clases que sirven para modelar información duradera y a menudo persistente.

Utilizaremos estos tres estereotipos (con sus símbolos gráficos asociados, en los diagramas de secuencia) para mostrar gráficamente cómo un mensaje emitido por un actor cruza las capas de presentación, aplicación y negocio.

Tengamos en cuenta la representación «foco de control» - bandas blancas que representan los períodos de actividad en las líneas de vida de los objetos -, así como las flechas punteadas de retorno de invocación de una operación.

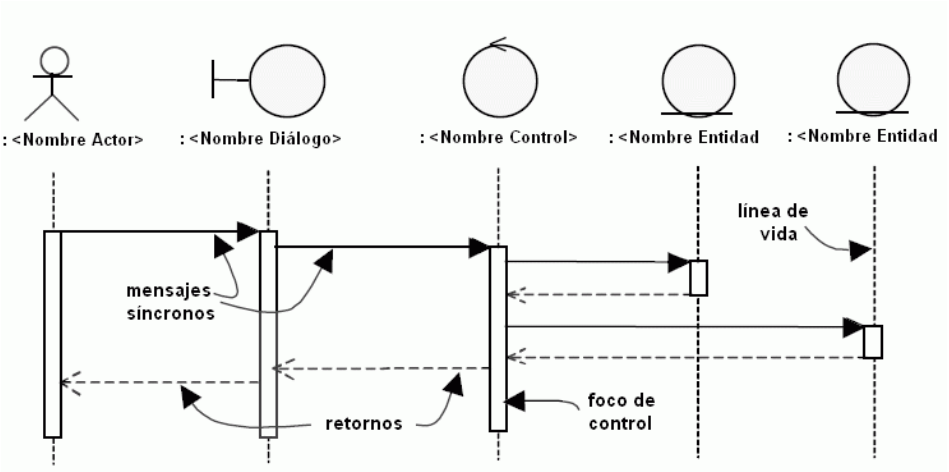


Figura 41: Ilustración de los tres estereotipos de Jacobson en un diagrama de secuencia.

Tomemos igualmente en cuenta la numeración decimal que permite mostrar la imbricación de mensajes, de una manera comparable a la representación de los «focos de control» sobre el diagrama de secuencia anterior.

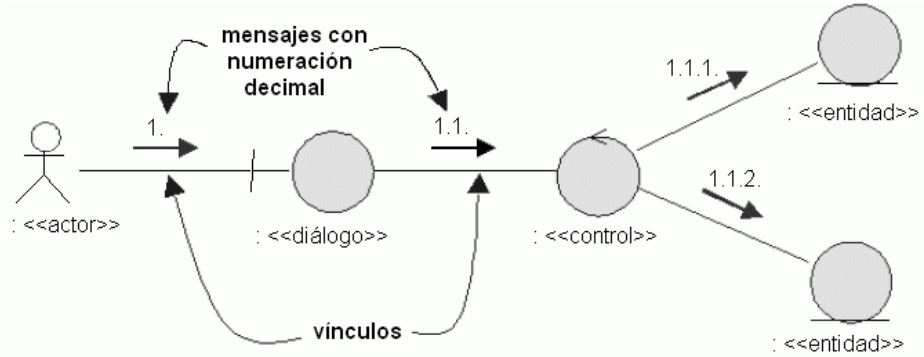


Figura 42: Ilustración de los tres estereotipos de Jacobson en un diagrama de comunicación.

Diagramas de interacción de diseño

Necesitamos elaborar un diagrama de interacción para crearFormación.

Qué se debe hacer? Para saberlo, debemos retomar todas las postcondiciones comentadas anteriormente:

- se creó una formación f con sus atributos;
- se creó un objeto contenido c con sus atributos;
- c se ha vinculado con f;
- f se vinculó con el organismo proveedor;
- se crearon posibles objetos sesiones con sus atributos;
- se vincularon estos objetos sesiones a f;
- f se vinculó con al menos un tema.

No olvidemos que las postcondiciones sólo representan el nuevo estado del sistema al final de la ejecución de la operación sistema. No están de hecho ordenadas: es el rol del diseñador elegir ahora cuál objeto debe realizar cada acción y en qué orden. La postcondición fundamental se refiere más bien a la creación del objeto formación, con su contenido y sus sesiones, luego la definición de sus vínculos con el otros objetos del catálogo como los temas y los organismos. Es razonable pensar que la creación del objeto formación f se va a hacerse en cuatro etapas:

1. inicialización del objeto f y de sus atributos;
2. creación de su contenido;
3. creación de las sesiones;
4. validación de f.

Veamos con detalle una solución posible para la primera etapa, poniendo en juego dos objetos «diálogo», un «control» y una «entidad».

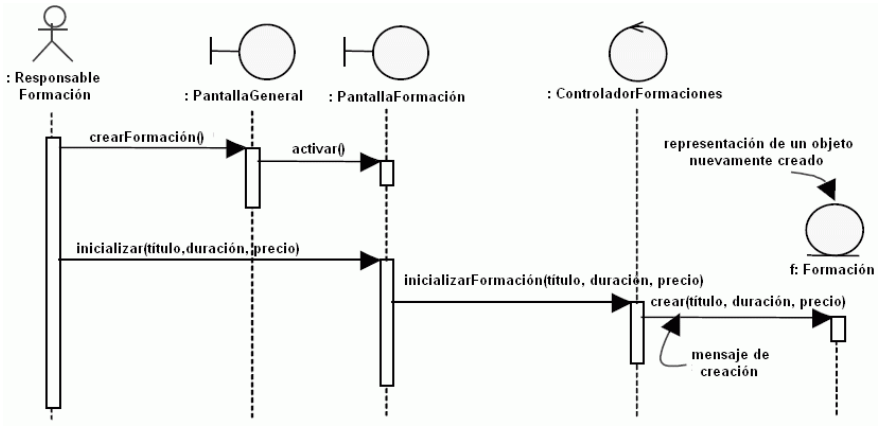


Figura 43: Diagrama de secuencia de la inicialización de f.

La misma situación se puede representar por medio de un diagrama de comunicación, como el que se muestra en la siguiente figura.

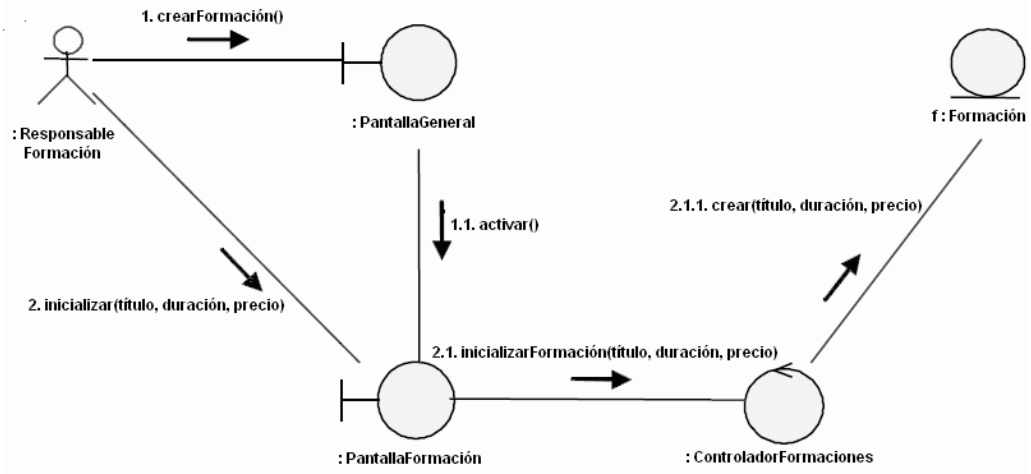


Figura 44: Diagrama de comunicación de la inicialización f.

Prosigamos por la creación del contenido. El diagrama de secuencia completado pasa a ser entonces:

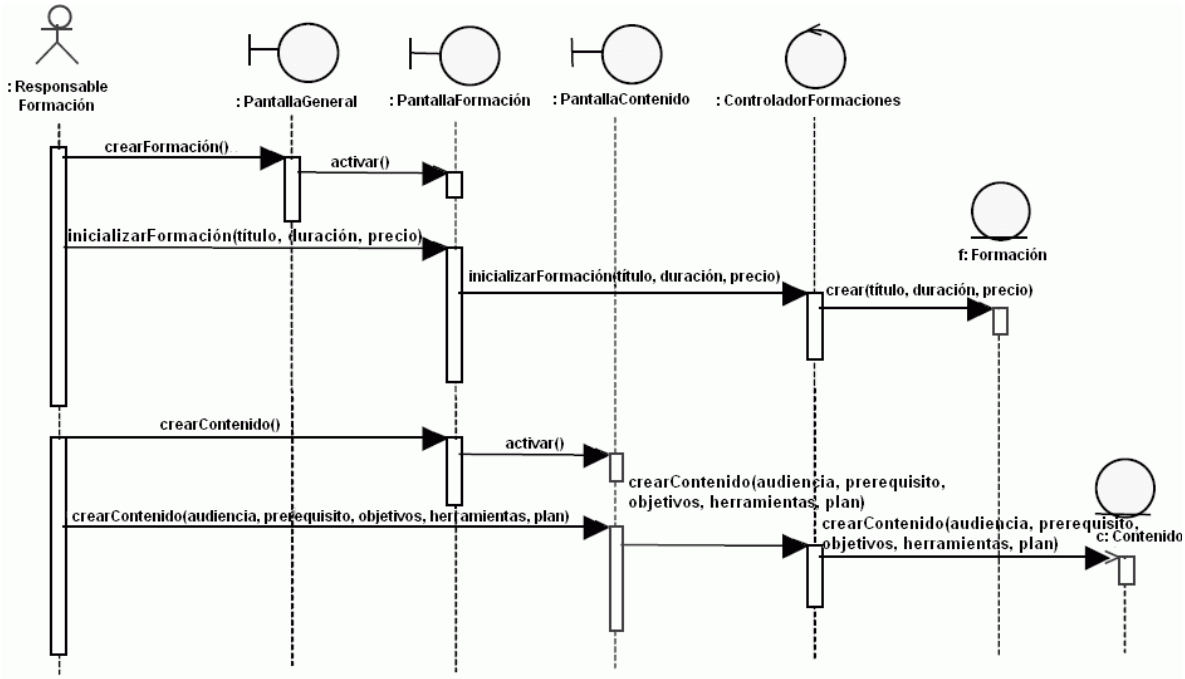


Figura 45: Diagrama de secuencia de la inicialización de f y de la creación de su contenido.

Se tendrá en cuenta que el diagrama de secuencia se vuelve cada vez menos legible a medida que añadamos objetos… Es por esta razón simple que el diagrama de comunicación es interesante en diseño: nos permite disponer nuestros objetos en las dos dimensiones con el fin de mejorar la legibilidad del esquema.

El diagrama de comunicación que corresponde al diagrama de secuencia precedente se muestra en la siguiente figura, como comparación (figura 46). Se tendrá en cuenta que el diagrama de comunicación presentado en la figura 46 permite diferenciar bien visualmente las capas de objetos. Prosigamos ahora con la creación sesiones.

MULTI-OBJETO

Para indicar que la formación f se vinculará con una colección de sesiones, utilizamos un multi-objeto. El multi-objeto es una construcción UML 1 que representa en un único símbolo varios objetos de la misma clase. Esto permite no añadir demasiado pronto clases de diseño detalladas, vinculadas al lenguaje de programación (como Vector de la STL C++ o ArrayList en Java, etc.). Un multi-objeto puede también representar la abstracción entera de una conexión a una base de datos.
Por otra parte habíamos olvidado en el diagrama de la figura 45 crear la colección vacía de sesiones en el momento de la creación de f. Es suficiente que a continuación de la creación de cada sesión, se añada a la colección. Utilizamos para esto una operación genérica add(): ver la figura 47.

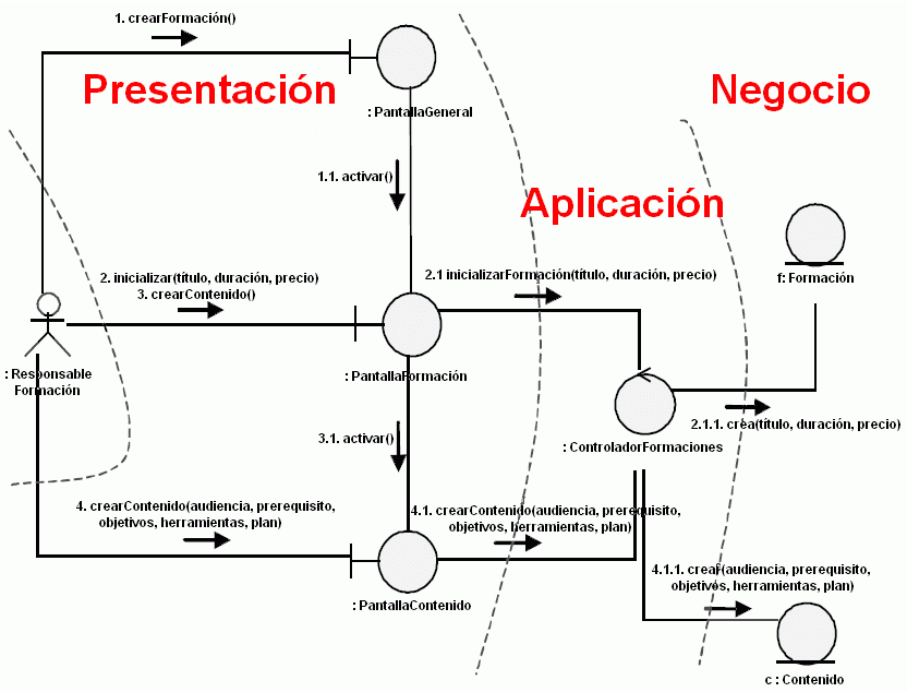


Figura 46: Diagrama de comunicación de la inicialización de f y de la creación de su contenido.

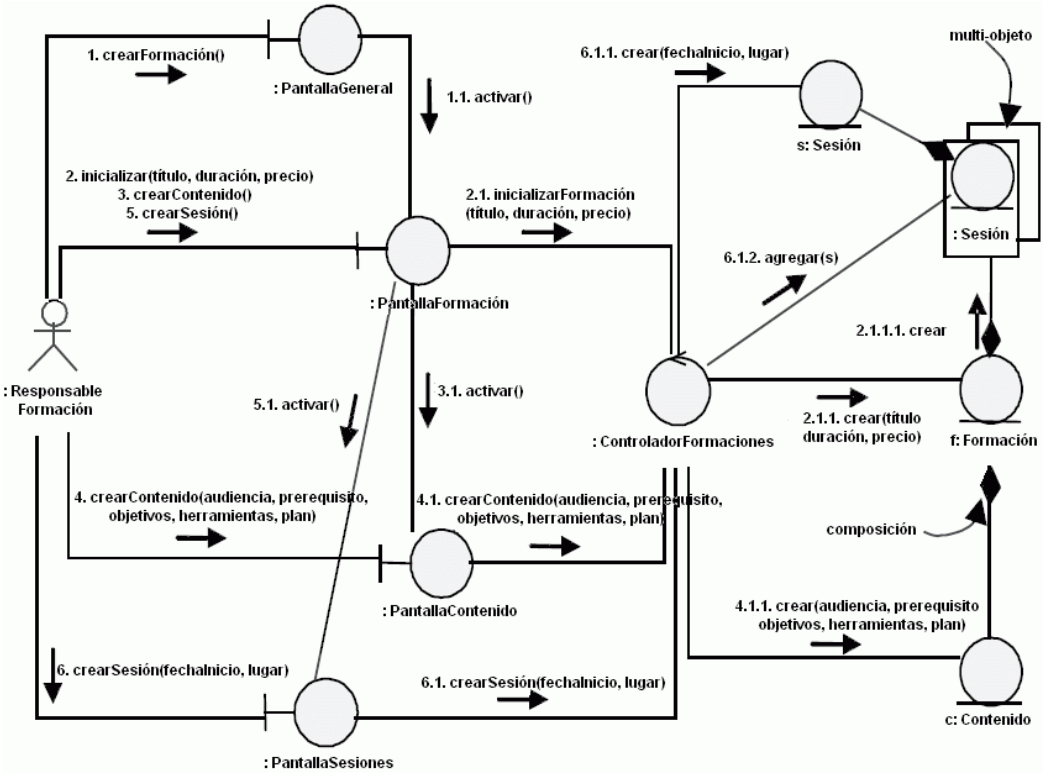


Figura 47: Diagrama de comunicación de la inicialización f, de la creación de su contenido y de una sesión.

El diagrama de comunicación presenta otra ventaja sobre el diagrama de secuencia: permite representar las relaciones estructurales entre los objetos. Por ejemplo mostramos los vínculos de composición alrededor del objeto formación f, para preparar mejor la trazabilidad con nuestro futuro diagrama de clases de diseño.

No nos queda más que vincular la formación f con un tema existente y validar la creación. Al comparar el trabajo realizado en las postcondiciones que se deseaban al principio de la respuesta, se puede constatar que no se tomó la siguiente en cuenta: «f se vinculó con el organismo proveedor». Lo añadimos simplemente a las responsabilidades del controlador, en el momento de la creación de la formación.

El diagrama de comunicación completo de la operación sistema crearFormación se muestra en la figura 48. Se puede observar la cantidad por de información, bastante considerable, que llega a representarse de manera más o menos legible sobre una única página. Sin embargo, alcanzamos allí los límites del diagrama de comunicación.

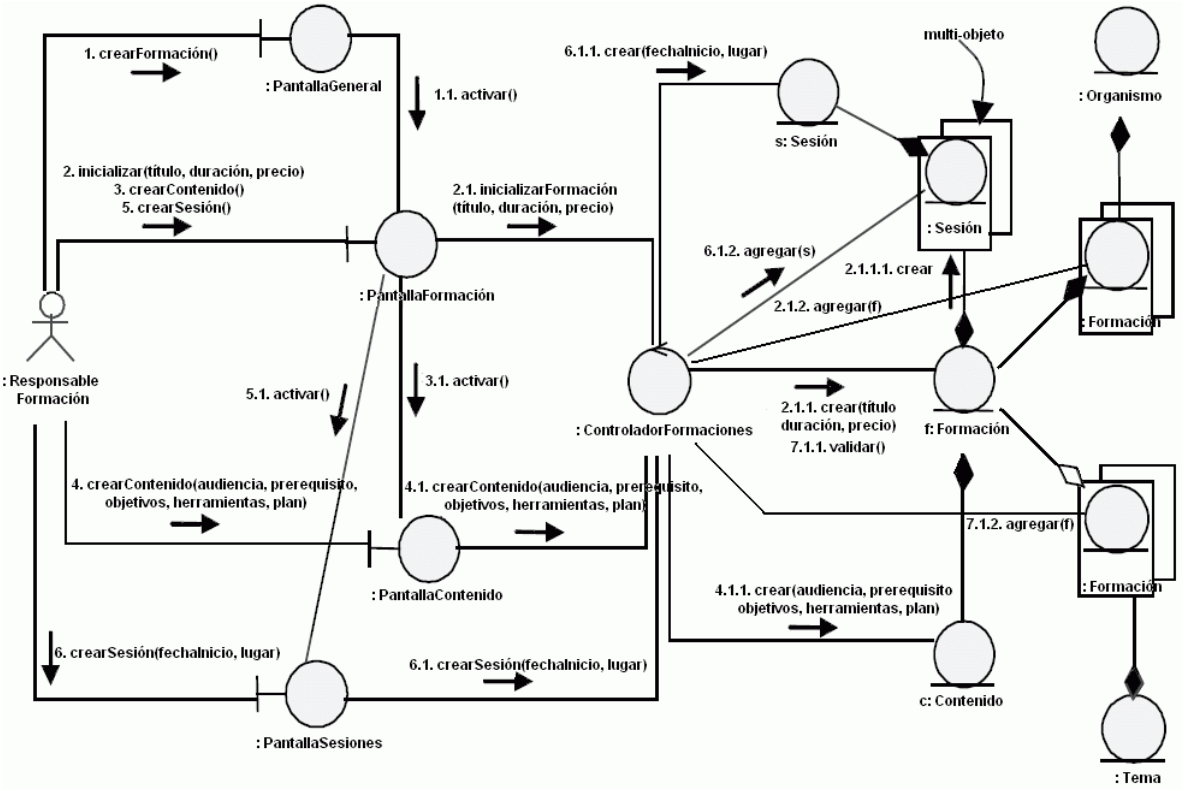


Figura 48: Diagrama de comunicación completo de la operación sistema crearFormación.

Una idea interesante para mejorar la legibilidad del diagrama consiste en recortarlo en dos tomando el objeto controlador como bisagra:

- una primera parte con el fin de especificar la cinemática de la interfaz humano computadora con los actores, los objetos «diálogo» y el objeto «control»;

- una segunda parte con el fin de especificar la dinámica de las capas aplicativas y de negocios con el objeto «control» y los objetos «entidad».

Los diagramas de comunicación parciales así obtenidos se muestran en las dos figuras que siguen.

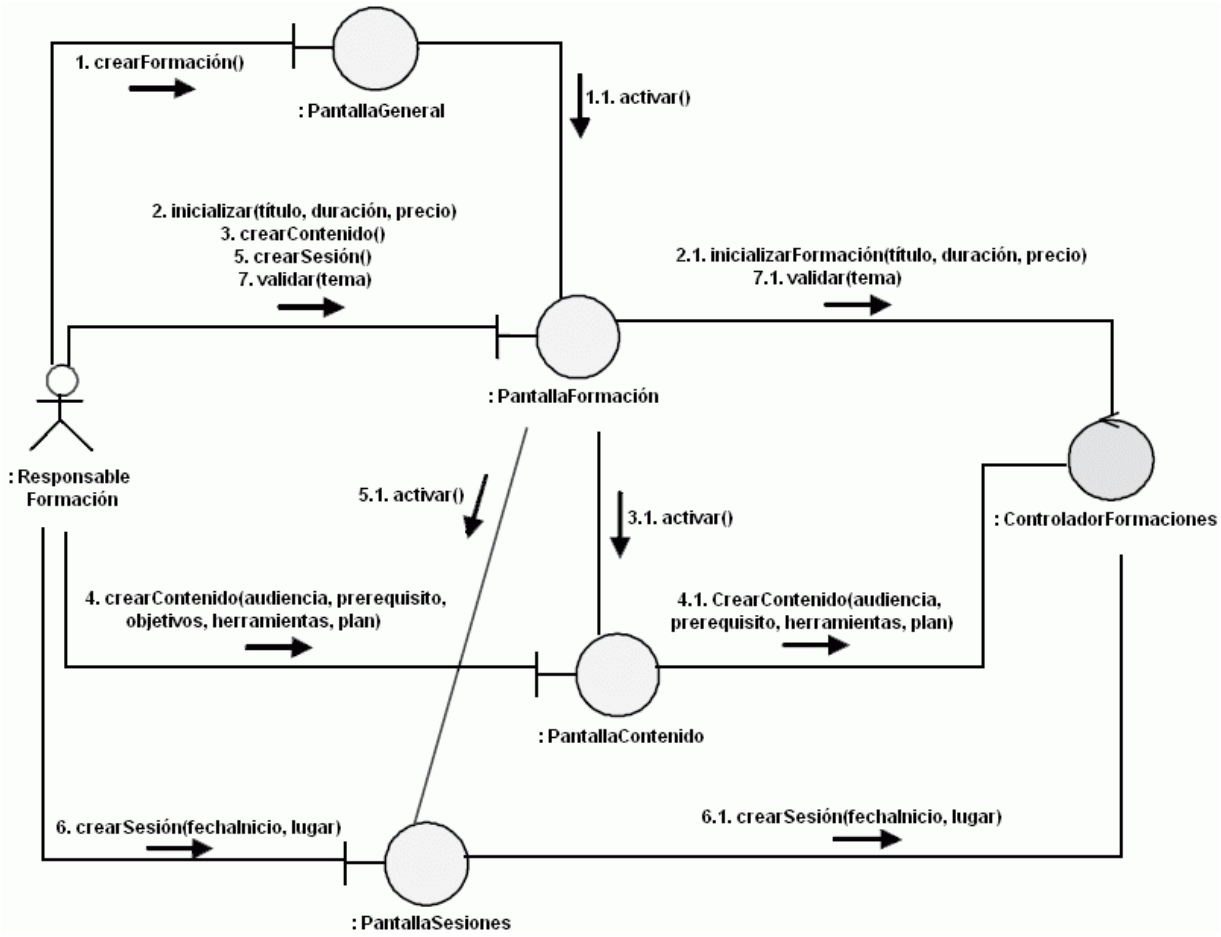


Figura 49: Diagrama de comunicación parcial de la operación sistema crearFormación: capa presentación y vínculo con la capa aplicativa.

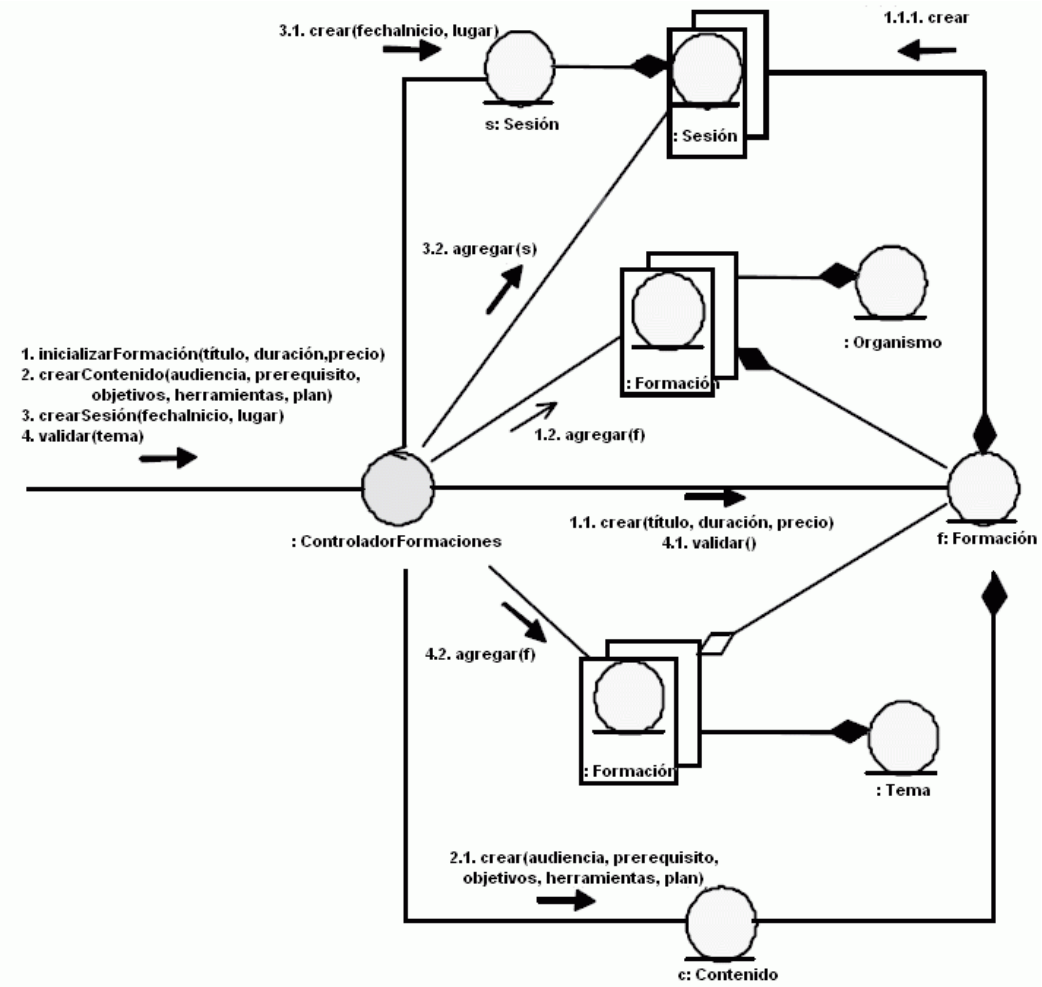


Figura 50: Diagrama de comunicación parcial de la operación sistema crearFormación: capa aplicativa y vínculo con la capa de negocios.

Etapa 9: Diagramas de clases de diseño (iteración #1)

Cada operación sistema va a dar lugar a un estudio dinámico en forma de un diagrama de interacción, como fue el caso de la operación crearFormación en la etapa anterior.

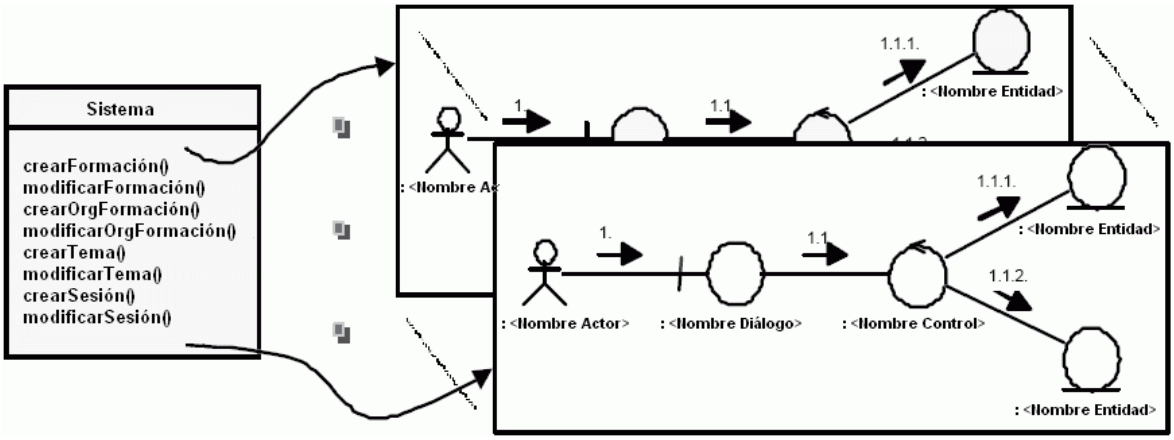


Figura 51: Planteamiento de diseño inicializado por las operaciones sistema.

Los diagramas de interacción así realizados van a permitir elaborar diagramas de clases de diseño, y esto añadiendo principalmente la información siguiente a las clases resultantes del modelo de análisis:

- las operaciones: un mensaje sólo puede ser recibido por un objeto si su clase declaró la operación pública correspondiente;

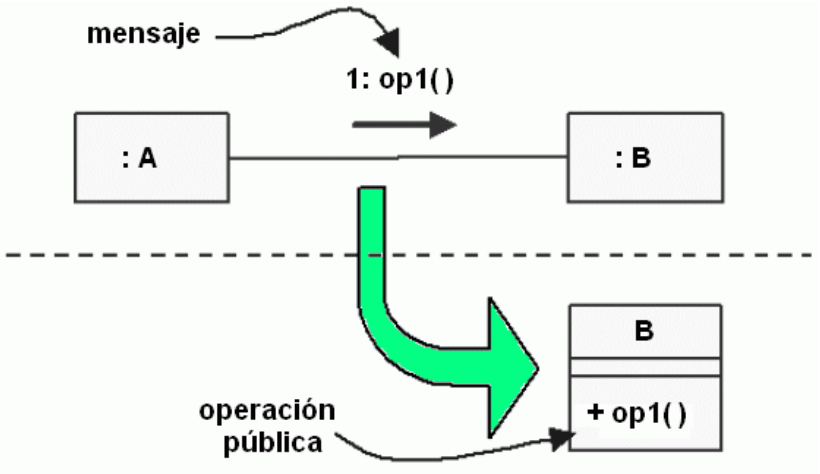


Figura 52: Relación entre mensaje y operación.

- la navegabilidad de las asociaciones o las dependencias entre clases, ya sea que los vínculos entre objetos sean duraderos o temporales, y en función del sentido de circulación de los mensajes.

VÍNCULOS DURADEROS O TEMPORALES

Un vínculo duradero entre objetos va a dar lugar a una asociación navegable entre las clases correspondientes; un vínculo temporal (por parámetro: «parámetro», o variable local: «local») va a dar lugar a una relación de dependencia.

En el ejemplo presentado a continuación, el vínculo entre el objeto :A y el objeto :B se convierte en una asociación navegable entre las clases correspondientes. El hecho de que el objeto :A reciba como parámetro de un mensaje una referencia sobre un objeto de la clase C induce una dependencia entre las clases en cuestión.

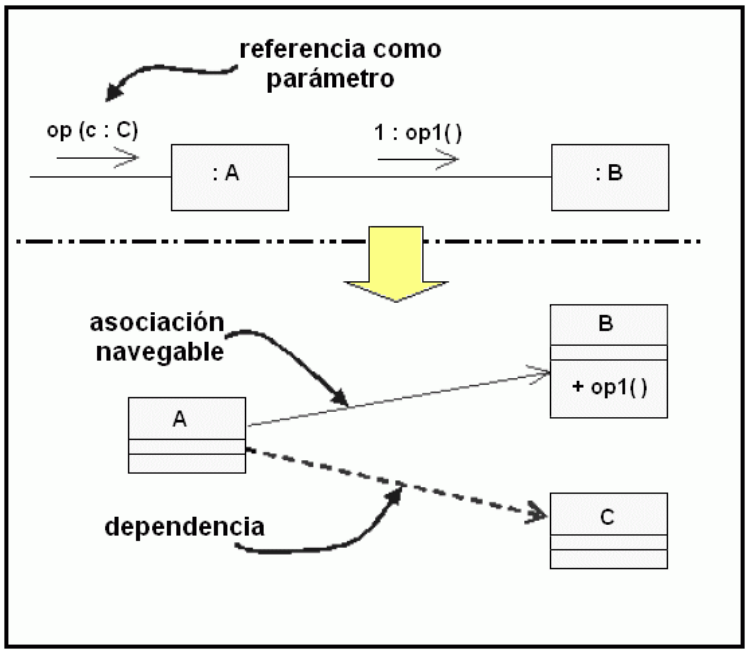


Figura 53: Relación entre los vínculos entre objetos y las relaciones entre clases.

Tengamos en cuenta que recomendamos no añadir las clases que corresponden a los multi-objetos en el diagrama de clases de diseño de tal modo que sigan siendo lo más lejanamente posible independiente del lenguaje de programación elegido.

Diagrama de clases de diseño

Aplicando las reglas enunciadas anteriormente, deseamos realizar un fragmento del diagrama de clases de diseño a partir del diagrama de comunicación parcial de la figura 50 (crearFormación).

El diagrama de comunicación presentado en la figura 50 nos permite en primer lugar añadir las operaciones en las clases como se muestra a continuación.

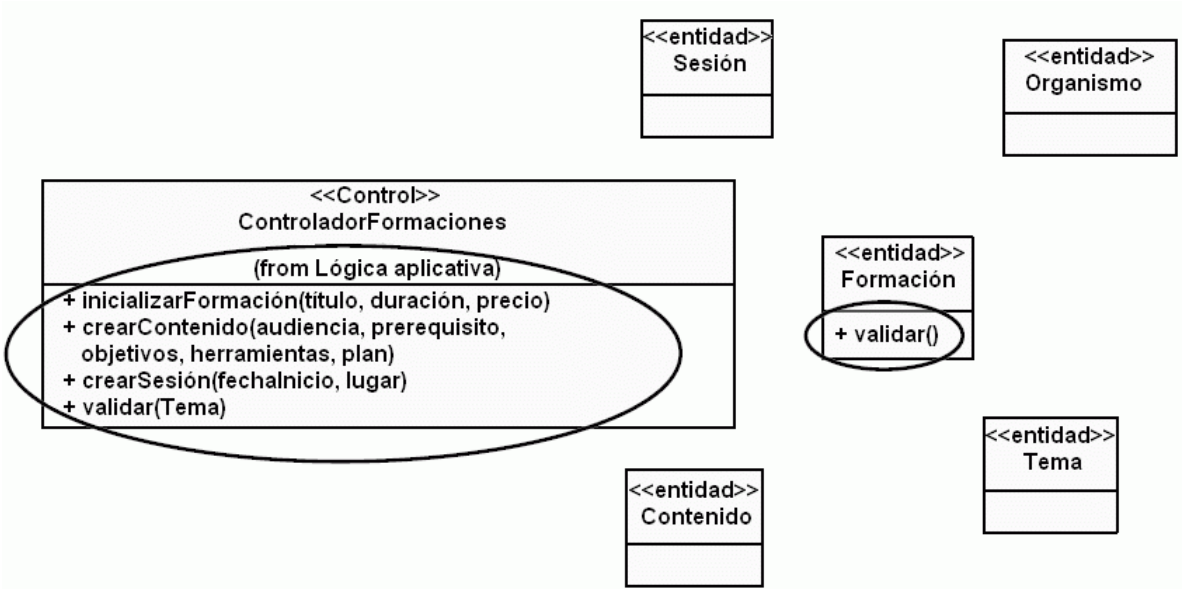


Figura 54: Operaciones en las clases de diseño.

Se tendrá en cuenta que hay pocas operaciones puesto que no las consideramos todas:

- Las operaciones de creación (mensaje crear),
- Las operaciones genéricas sobre las clases contenedores (add (), etc.).

Sin embargo, se puede observar un primer problema: cómo el objeto controladorFormaciones puede añadir la nueva formación f a los multi-objetos del organismo correspondiente sin poseer una referencia sobre este organismo?

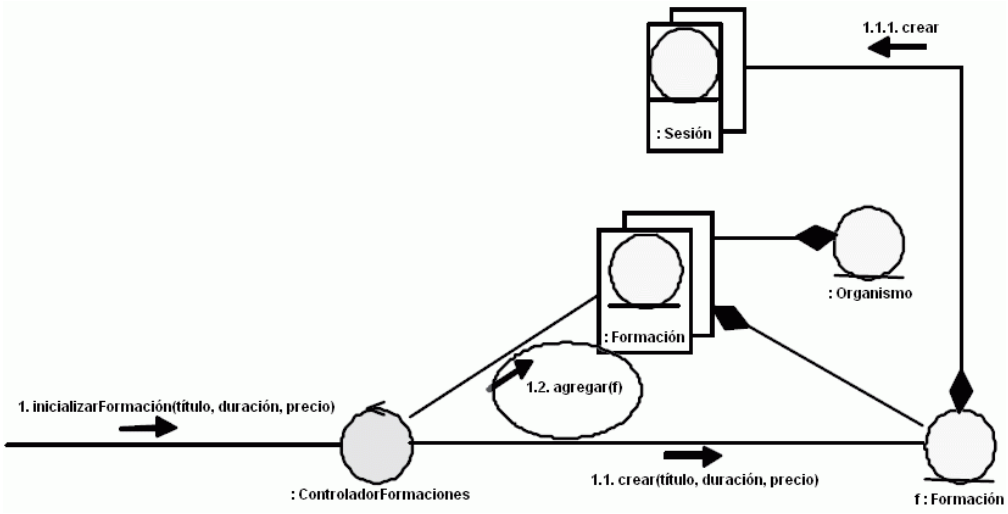


Figura 55: Diagrama de comunicación limitado al primer mensaje de la operación sistema crearFormación.

Esto significa que debemos añadir un parámetro a la operación inicializarFormación: una referencia hacia un organismo existente.

Si utilizamos también las palabras clave «parámetro» y «local» para indicar los vínculos temporales entre objetos, el diagrama de comunicación anterior es modificado como sigue:

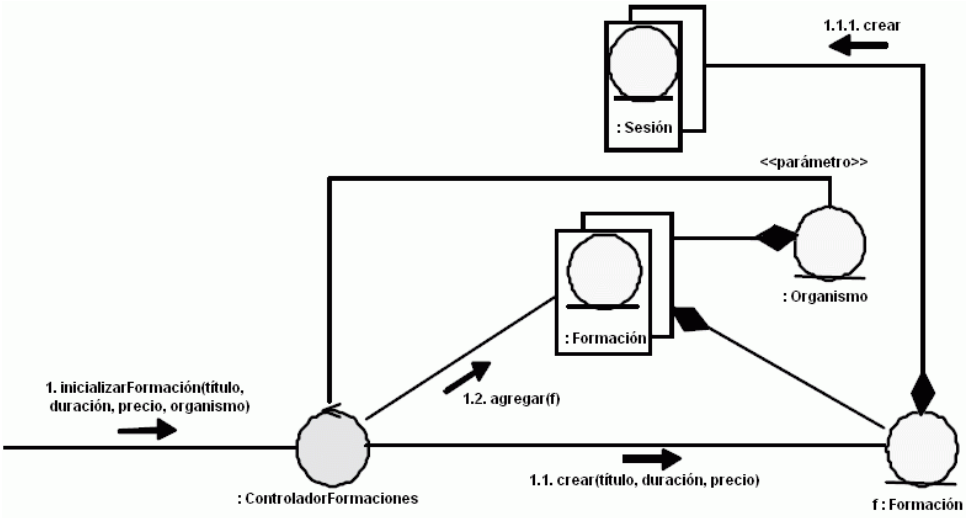


Figura 56: Diagrama de comunicación completo.

Vamos ahora a completar el diagrama de clases añadiendo las relaciones entre clases: asociación (con sus alternativas: agregación o composición) y dependencia. El trabajo se facilita en algo que ya habíamos comentado, los vínculos de composición y agregación sobre el diagrama de comunicación.

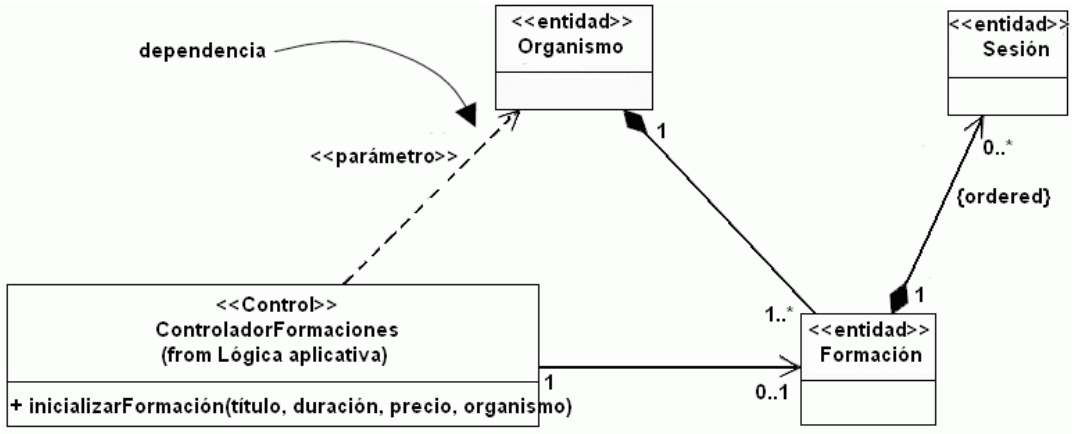


Figura 57: Diagrama de clases completo según el diagrama de comunicación anterior.

Se tendrá en cuenta la utilización de la palabra clave «parámetro» sobre la dependencia entre las clases ControladorFormaciones y Organismo, para reflejar el tipo de vínculo temporal que existe entre los objetos correspondientes en el diagrama de comunicación

Si aplicamos ahora el mismo planteamiento sobre el conjunto del diagrama de la figura 50, obtenemos el siguiente diagrama de clases de diseño. Hay que señalar que resaltamos los

atributos en las clases, pero no los parámetros de las operaciones (para simplificar el esquema).

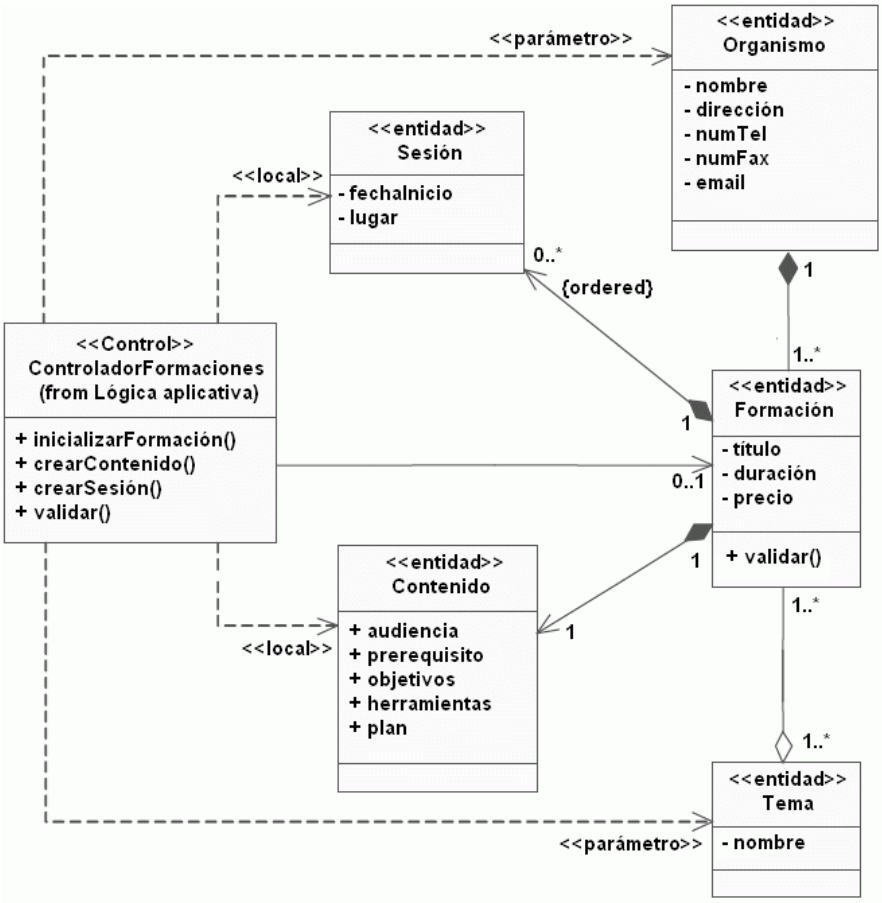


Figura 58: Diagrama de clases de diseño completo.

Por supuesto, este diagrama está aún en un estado totalmente provisional:

- las elecciones de navegabilidad de las asociaciones distan mucho de ser definitivas - podrán ser modificados por el estudio de las otras operaciones del sistema;
- las dependencias se transformarán quizá en asociaciones si los objetos requieren un vínculo duradero, y no un simple vínculo temporal, en el marco de otras operaciones del sistema.

Mejora al diseño

A partir de los diagramas realizados en la parte anterior, es preferible siempre proponer mejoras en el diseño orientado a objetos.

El diagrama de clases de la figura 58 presenta una clase ControladorFormaciones acoplada a todas las demás clases! Esta propiedad es totalmente contraria a un principio fundamental del diseño orientado a objetos, llamado generalmente «acoplamiento débil».

ACOPLAMIENTO DEBIL

El <<acoplamiento>> representa una medida de la cantidad de otras clases a las cuales se conecta una clase dada, de la cual tiene conocimiento, o de la que depende.

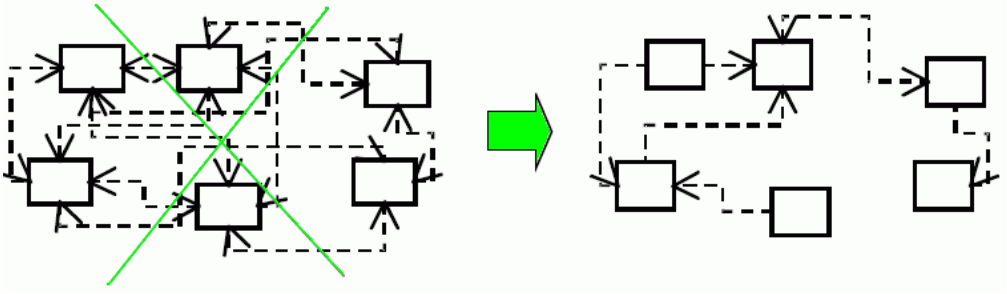


Figura 59: Acoplamiento entre clases.

Conservar un acoplamiento débil es un principio que es necesario considerar para todas las decisiones de diseño; es un objetivo subyacente que debe evaluarse de forma continua.

En efecto, utilizándolo, se obtiene en general una aplicación más evolutiva y más fácil de mantener.

La noción de objeto «controlador» que enumeramos anteriormente (ver figura 34) es un buen ejemplo de un medio aplicado para minimizar el acoplamiento entre las capas computacionales.

Intentemos ver si no existe un medio simple de reducir el acoplamiento de la clase ControladorFormaciones sin aumentar por lo tanto el de las otras clases.

Retomemos el diagrama de comunicación de la figura 50. El objeto ControladorFormaciones está bien colocado para crear los objetos Contenido y Sesión? No podría más bien delegar esta responsabilidad de creación al objeto Formación que va en cualquier caso a ser enseguida vinculado de una manera duradera a su contenido y a sus sesiones?

De esta manera, retiramos las dos dependencias entre ControladorFormaciones y Contenido y Sesión, sin añadir, puesto que Formación ya está acoplada a Contenido y Sesión por fuertes relaciones de composición.

El diagrama de comunicación puede entonces modificarse como seo muestra en la figura 60.

El diagrama de clases de diseño reduce dos dependencias, simplemente porque el objeto ControladorFormaciones supo delegar una parte de sus responsabilidades al objeto Formación.

En realidad, este ejemplo simple es totalmente representativo del trabajo iterativo de evaluación y mejora que debe hacer todo diseñador en la parte de diseño orientado a objetos.

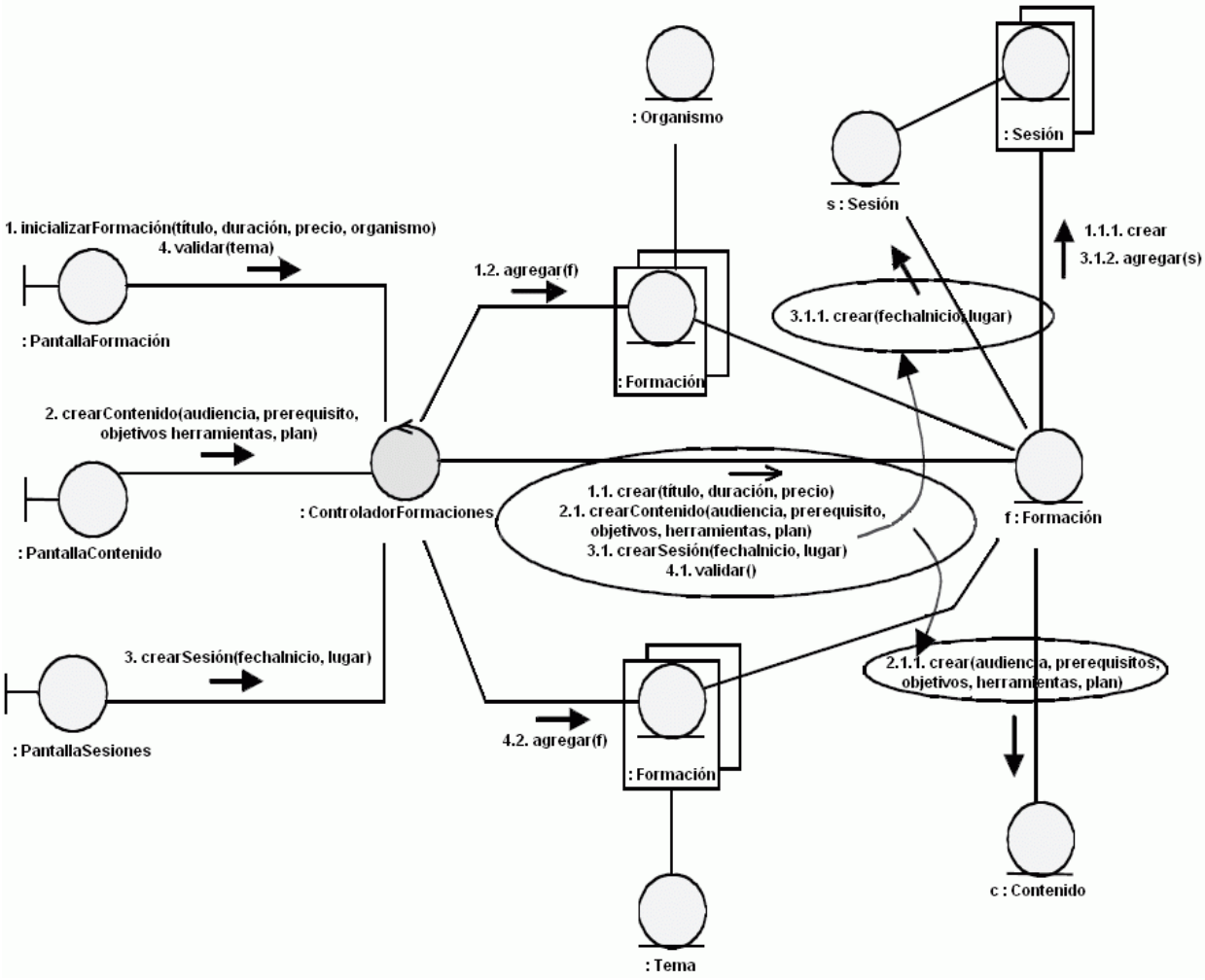


Figura 60: Diagrama de comunicación mejorado de la operación del sistema crearFormación.

Para terminar, vamos completar el diagrama de clases mejorado por los tipos de los atributos, y proceder a la firma completa de las operaciones (parámetros con su tipo).

Es necesario tener bien en cuenta que utilizamos tipos simples del lenguaje Java (como int y short), clases básicas de Java (como String y Fecha), de las clases «primitivas» usuario (como Número y Email), que será necesario definir precisamente, y finalmente las clases del modelo (como Tema y Organismo).

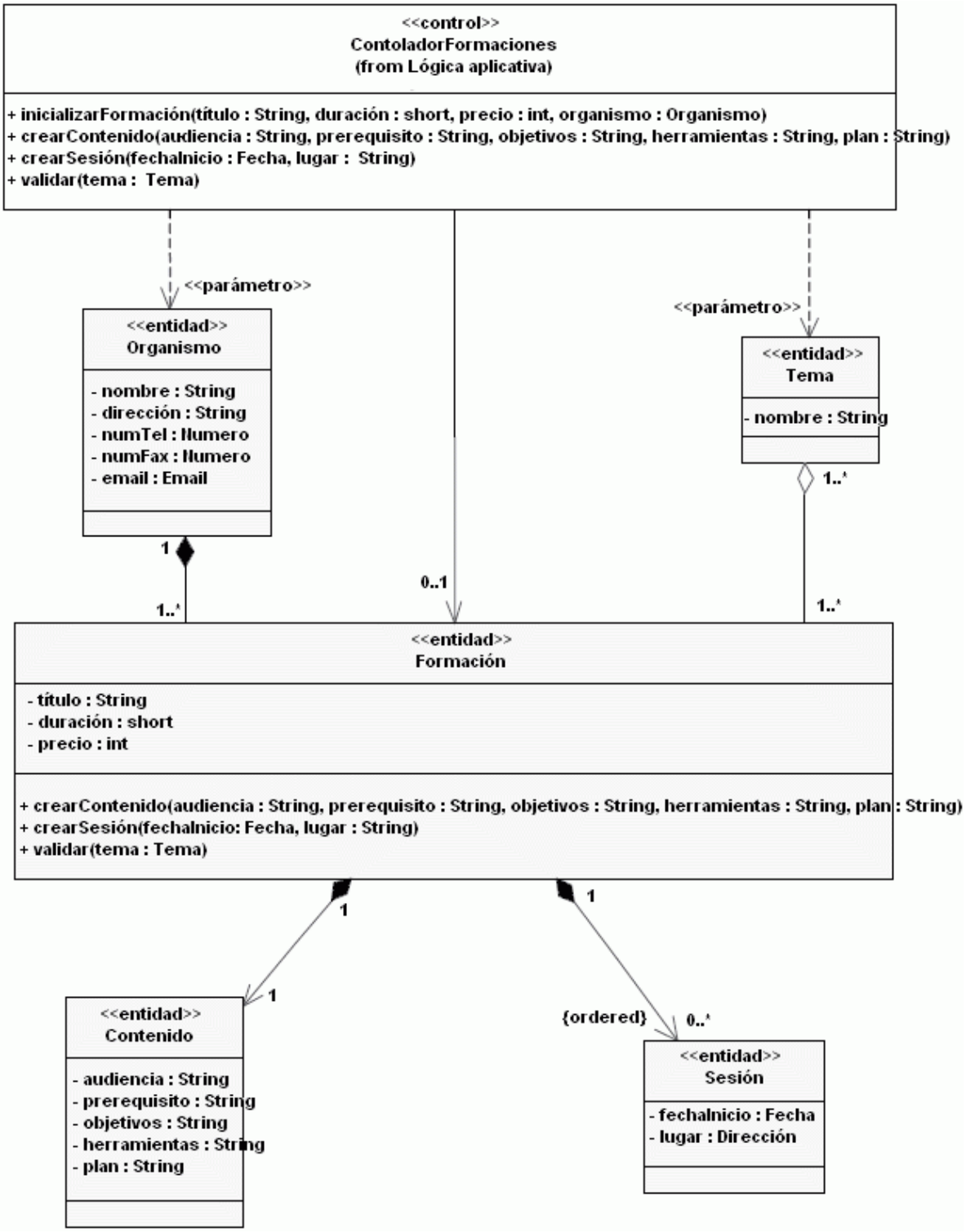


Figura 61: Diagrama de clases de diseño mejorado.

Etapa 10: Definición de las operaciones del sistema (iteraciones #2 y #3)

En esta fase, consideramos como punto de partida que la iteración 1 se realizó con éxito. Los casos de uso Consultar el catálogo y Actualizar el catalogo se concibieron, se implementaron y se probaron. El paquete de negocios Catálogo de formaciones se refinó y se enriqueció en consecuencia. Un estado posible de su diagrama de clases de diseño (que no muestran más que las clases «entidad») se presenta en la siguiente figura.

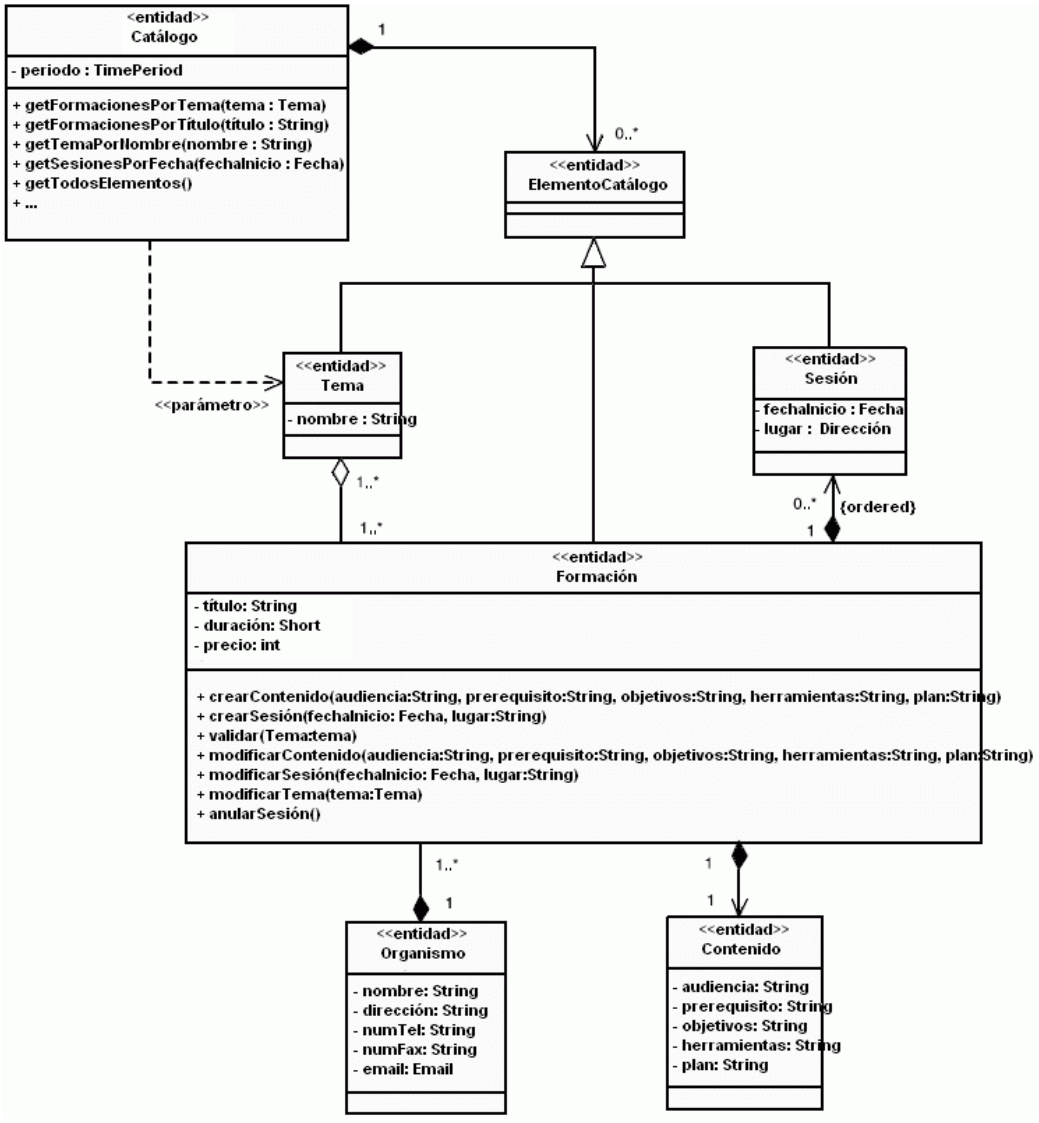


Figura 62: Diagrama de clases de diseño del paquete Catálogo.

Se podrá notar que muchas operaciones se han agregado, así como una clase abstracta ElementoCatálogo que engloba los temas, las formaciones y las sesiones, en la óptica de la creación de una solicitud de formación para un empleado a partir de un elemento cualquiera del catálogo de formaciones.

El almacenamiento del catálogo en una base de datos relacional es operacional, así como la interfaz hombre máquina de los dos casos de uso. El mecanismo de autentificación está igualmente disponible en una versión que nos permitirá continuar con nuestro trabajo.

Operaciones del sistema

Vamos ahora a concebir y aplicar la segunda y tercera iteraciones. Comencemos pues por el caso de uso *Solicitar una formación*. Su descripción de alto nivel se realizó en la etapa 1. La citamos para recordarla: «El empleado puede consultar el catálogo, y seleccionar un tema, o una formación, o incluso una sesión particular. La solicitud es registrada automáticamente por el sistema y la remite al responsable formación por correo electrónico». Para el segundo caso, *Tratar las solicitudes*, la descripción era la siguiente: «El responsable de formación va a utilizar el sistema para indicar a los empleados su decisión (acuerdo o descuerdo)». En caso de acuerdo sobre una sesión precisa, el sistema va a enviar automáticamente por fax una solicitud de inscripción en forma de orden de pedido al organismo en cuestión. Vamos en primer lugar a realizar un diagrama de secuencia del sistema de la situación nominal del caso de uso *Solicitar una formación*. Suponemos que el empleado puede efectuar varias solicitudes, de ahí el marco *loop*. Las selecciones de una formación y de una sesión son opcionales, de ahí los dos marcos *opt* imbricados.

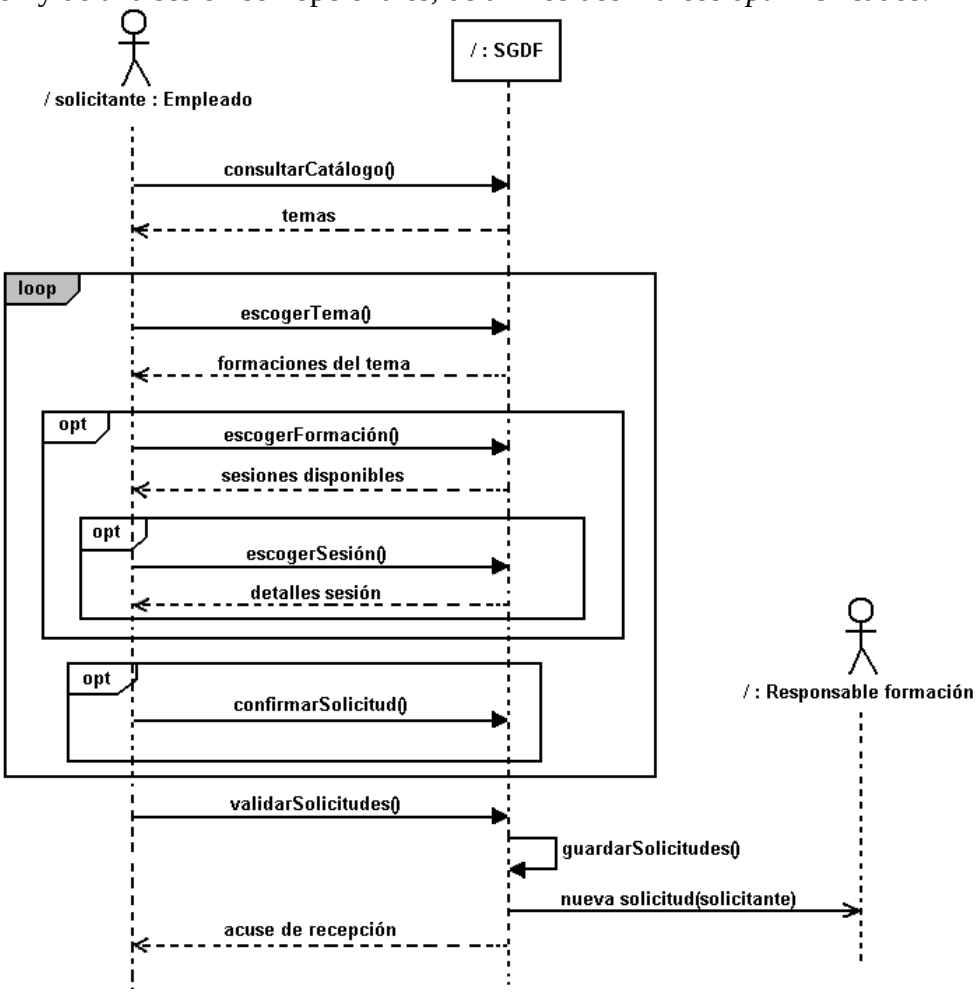


Figura 63: Diagrama de secuencia del sistema de Solicitar una formación.

Vamos a continuación a realizar un diagrama de secuencia sistema simplificado (sin tratar el caso de aceptación de solicitud incompleta) del caso de uso *Tratar las solicitudes*.

Consideremos la posibilidad que ofrece UML 2 para representar gráficamente las precondiciones en lo alto de la línea de vida del sistema

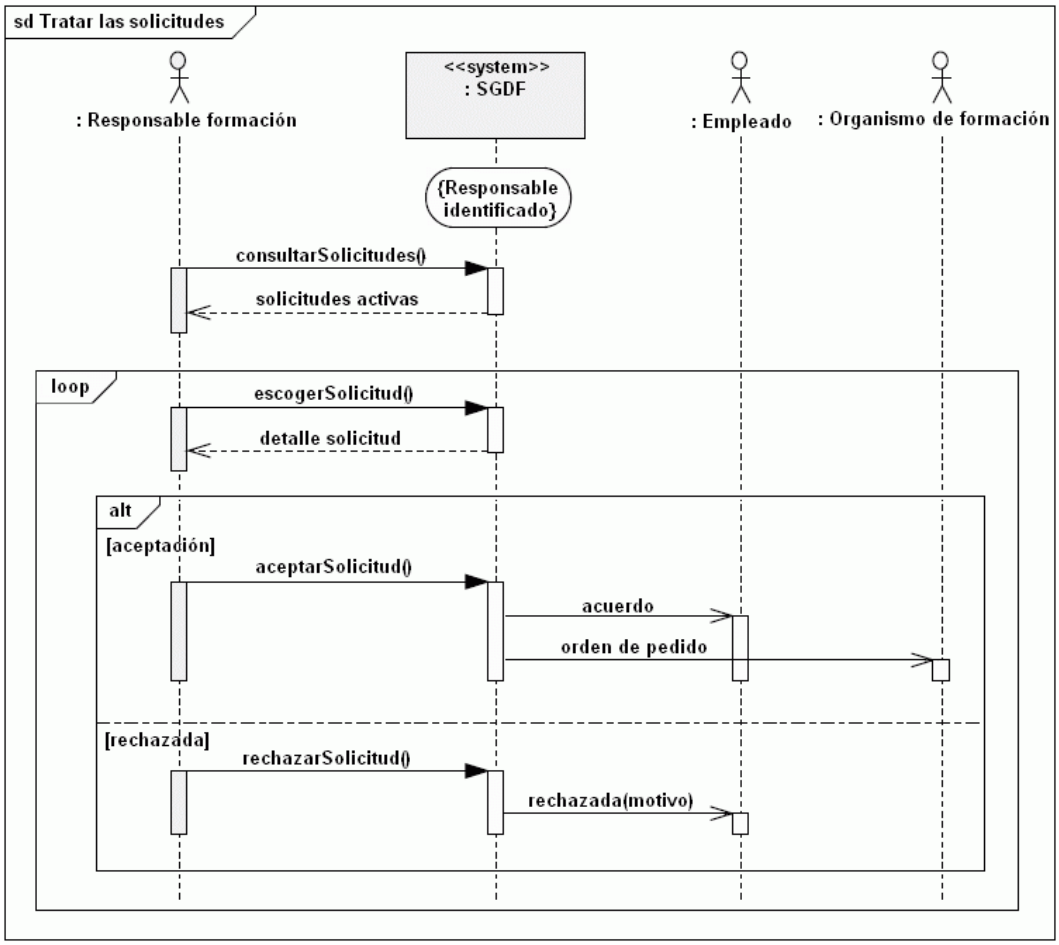


Figura 64: Diagrama de secuencia del sistema de Tratar las solicitudes.

Las principales operaciones sistema para los casos de uso Solicitar una formación y Tratar las solicitudes se listan en los dos esquemas anteriores (flechas que entran en el SGDF).

Etapa 11: Contratos de operaciones (iteraciones #2 y #3)

En primer lugar, vamos a extraer del diagrama de clases del paquete Solicitudes de formación (ver figura 26) la parte de interés para nuestra cuestión. Es decir, las operaciones del sistema validarSolicitud y rechazarSolicitud van a actuar sobre objetos y vínculos que emanan del siguiente diagrama.

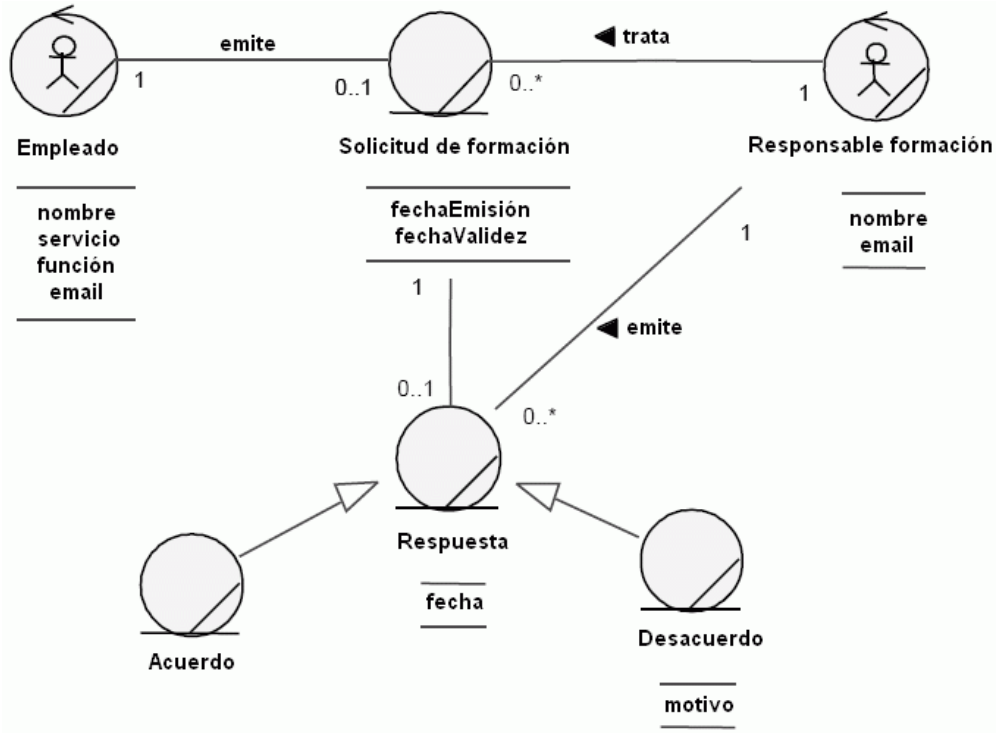


Figura 65: Extracto del diagrama de clases deducido de la modelización de negocios.

Establezcamos en primer lugar el contrato de la operación validarSolicitud:

- Nombre
  validarSolicitud
- Responsabilidades
  Crear una solicitud inicial según los elementos del catálogo y transmitirla al responsable de formación para instrucción.
- Referencias
  Caso de uso *Solicitar una formación*
- Precondiciones
  - el catálogo de formación existe;
  - el empleado está conectado en el intranet;
  - un objeto e representa al empleado que existe en la aplicación.
- Postcondiciones
  - una solicitud de formación sdf se ha creado;
  - los atributos fechaValidez y fechaEmisión de sdf se han inicializado;
  - sdf se ha vinculado al empleado e;

  - sdf se ha vinculado a un elemento del catálogo de formación (es un aspecto que faltaba en el diagrama de modelización de negocios);
  - un e-mail que contiene sdf se ha enviado al responsable de formación.
- Excepciones
  - El empleado puede anular su creación de solicitud en cualquier momento antes de validar.

Prosiguiendo con el contrato de la operación rechazarSolicitud:

- Nombre
  rechazarSolicitud
- Responsabilidades
  Rechazar una solicitud enviada por un empleado y devolver el motivo del rechazo.
- Referencias
  Caso de uso *Tratar las solicitudes*
- Precondiciones
  - una solicitud de formación sdf existe;
  - el responsable está conectada en la intranet;
  - un objeto e representa al empleado que existe en la aplicación y está vinculado a sdf.
- Postcondiciones
  - la solicitud de formación sdf se ha destruido;
  - un objeto Desacuerdo d se ha creado;
  - los atributos fecha y motivo de f se han inicializado;
  - un e-mail que contiene d se ha enviado al empleado e.
- Excepciones
  - ninguna

Etapa 12: Diagramas de interacción (iteraciones #2 y #3)

Como en la etapa 8 para las iteraciones 1 y 2, vamos a proseguir nuestro trabajo de diseño realizando un diagrama de comunicación.

El planteamiento es similar al que se adoptó anteriormente (diagramas de interacción de diseño). El diagrama de comunicación que representa la inicialización de la solicitud de formación por el empleado es totalmente similar al de la figura 44.

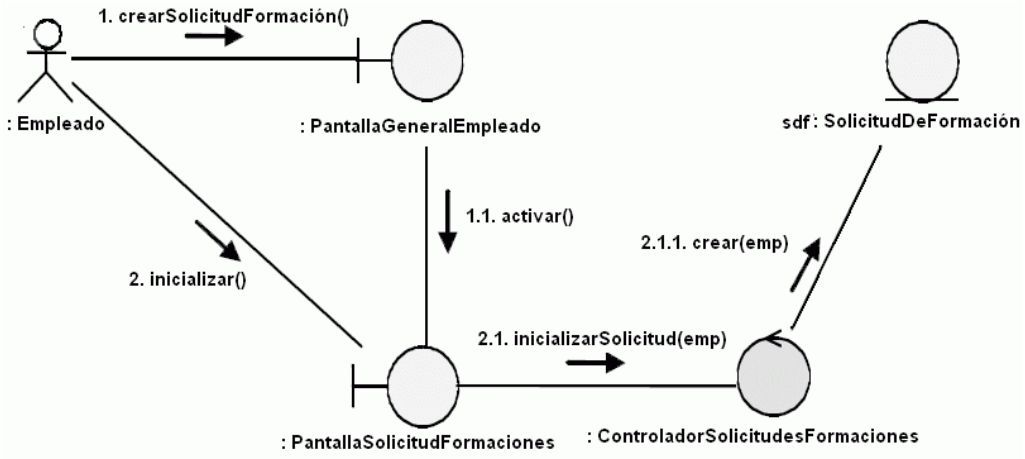


Figura 66: Diagrama de comunicación de la inicialización de sdf.

Continuando con el establecimiento del vínculo con un elemento del catálogo de formación, luego por la localización de los atributos fechaValidez y fechaEmision, y finalmente el envío del mensaje al responsable.

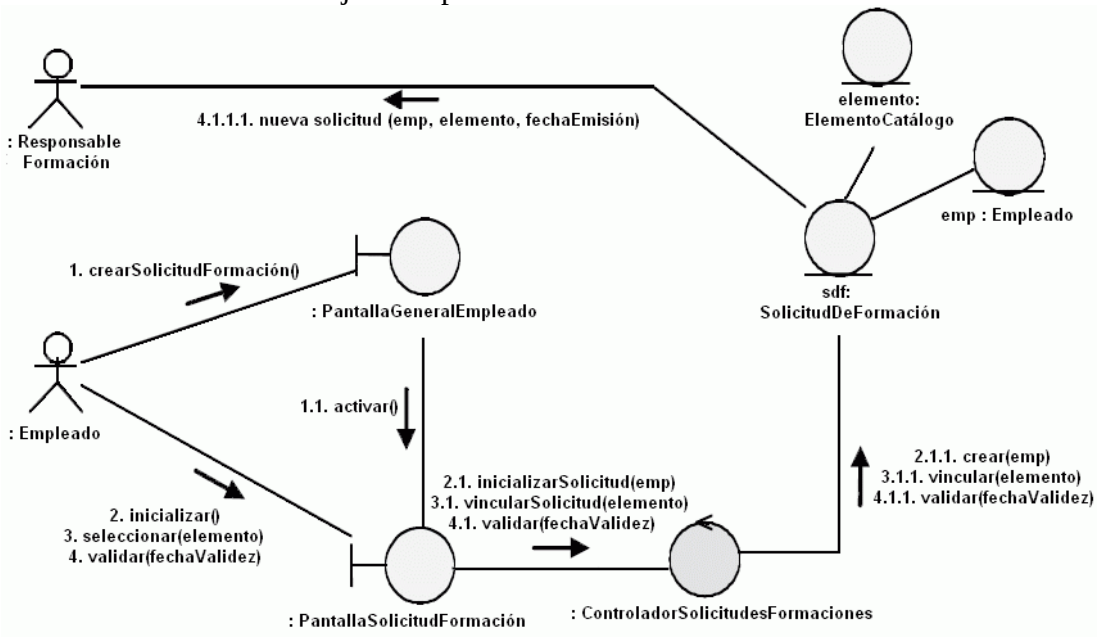


Figura 67: Diagrama de comunicación completo de la operación validarSolicitud.

Etapa 13: Diagramas de clases de diseño (iteraciones #2 y #3)

Considerando el modelo de la figura 62 y extrapolando a partir de la respuesta anterior, y también apoyándonos en el conocimiento del tema, tenemos que realizar un diagrama de clases de diseño del paquete Solicitudes.

Ya examinamos las sutilezas del diagrama de clases en ejemplos anteriores. La clase de asociación Inscripción no será una sorpresa. Tendremos en cuenta la forma en que se completa el compartimento operaciones de la clase SolicitudDeFormación, en particular con las operaciones privadas, necesarias para el envío de los mensajes a los actores.

Se tendrá en cuenta que destacamos en el diagrama las clases Sesión y ElementoCatálogo aunque no pertenecen al paquete actual. Es decir, es importante mostrar sus relaciones con clases del paquete Solicitudes para justificar enseguida el sentido de las dependencias entre los paquetes que engloban. Precisamente, sólo es necesario representar a las asociaciones navegables, las dependencias o las generalizaciones que señalan hacia las clases externas a las del paquete en cuestión.

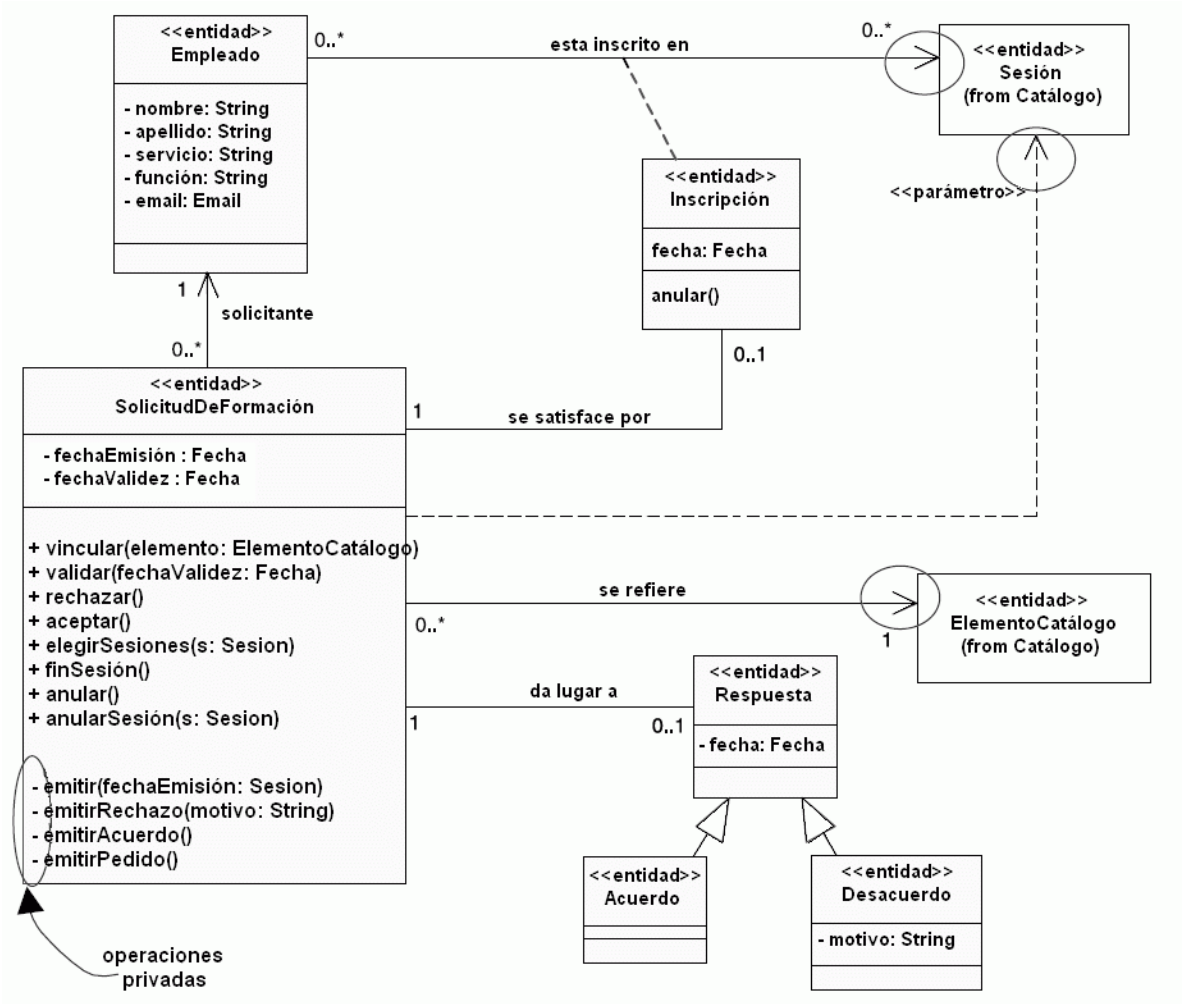


Figura 68: Diagrama de clases de diseño del paquete Solicitudes.

Etapa 14: Regreso a la arquitectura

Diagrama de paquetes

Retomemos la figura 36 que representaba la arquitectura en capas del sistema y mostraba todas las clases que definimos dentro de los paquetes correspondientes. No se tomaron en cuenta la capa de servicios técnicos, más que las clases básicas de Java.

Basta con contabilizar todas las clases que utilizamos en nuestros distintos diagramas, y representarlas dentro del paquete adecuado. La arquitectura lógica detallada de las tres primeras capas se muestra en la figura siguiente.

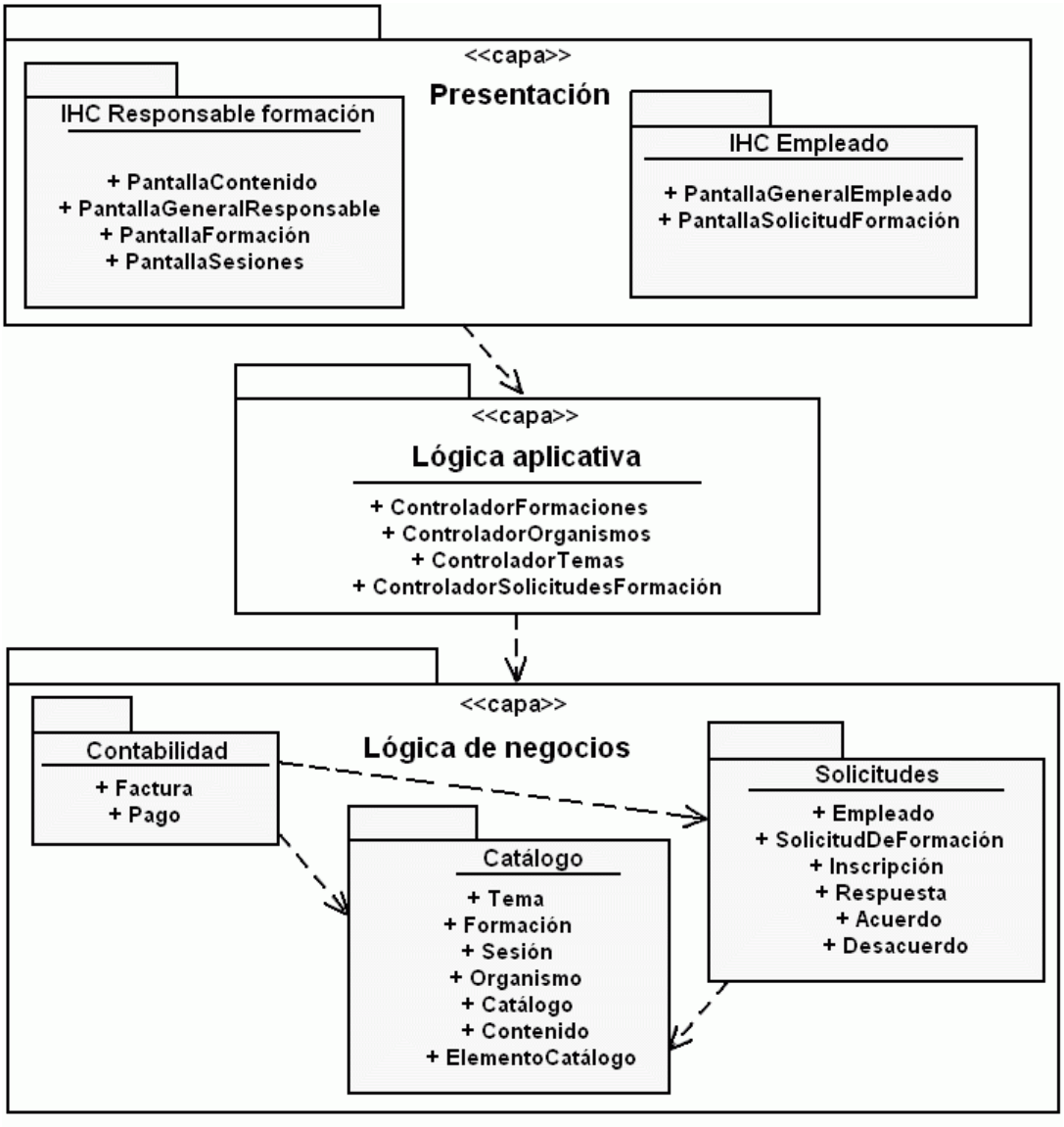


Figura 69: Detalle de la arquitectura en capas de las tres primeras iteraciones.

Etapa 15: Paso a código objeto

Los modelos de diseño que realizamos permiten producir, de una manera fácil, el código en un lenguaje de programación orientado a objetos como Java o C#:

- Los diagramas de clases permiten describir el esqueleto del código, a saber todas las declaraciones.

PRODUCCIÓN DEL ESQUELETO DE CÓDIGO JAVA (O C#) A PARTIR DE LOS DIAGRAMAS DE CLASES

El primer enfoque:

- la clase UML se convierte en una clase Java (igualmente en C#);
- los atributos UML se convierten en variables de instancias Java (de la misma forma para C#);
- las operaciones UML se convierten en métodos Java (igual en C#).

Se tendrá en cuenta que los roles navegables producen variables de instancias, al igual que los atributos, pero con un tipo usuario en vez de un tipo simple. El constructor por defecto está implícito.

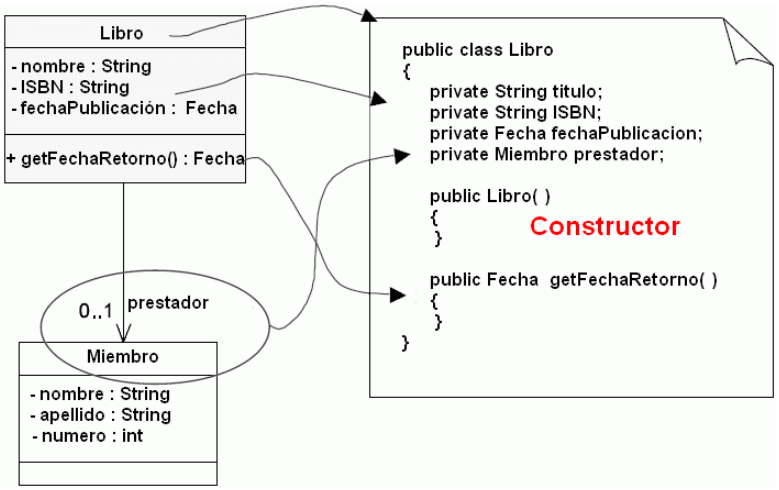


Figura 70: Esqueleto del código Java de la clase Libro.

Nota: Se puede igualmente utilizar el concepto C# de property, que permite una mejor encapsulación. Hay que tener cuidado, el concepto del atributo existe en C++, pero no significa la misma cosa que en UML.

- Con los diagramas de interacción, es fácil escribir el cuerpo de los métodos, en particular la secuencia de llamadas de métodos sobre los objetos que colaboran.

PRODUCCIÓN DEL CUERPO DE LOS MÉTODOS A PARTIR DE LOS DIAGRAMAS DE INTERACCIÓN

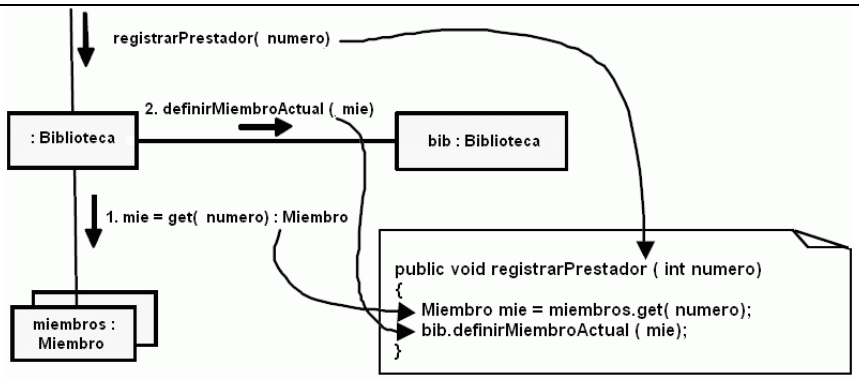


Figura 71: Cuerpo del método registrarPrestador en Java(se puede igualmente producir para C#).

Producción de un esqueleto de código Java

A partir de la figura 68, se propone un esqueleto de código Java para la clase SolicitudDeFormación. Retiramos lo que no se refiere a la clase SolicitudDeFormación. Las reglas anteriores bastan para producir el esqueleto de la clase en Java. La única dificultad que se tiene, es que no es necesario olvidar la directiva de importación para las relaciones con las clases que pertenecen a otros paquetes, así como para las clases básicas de Java.

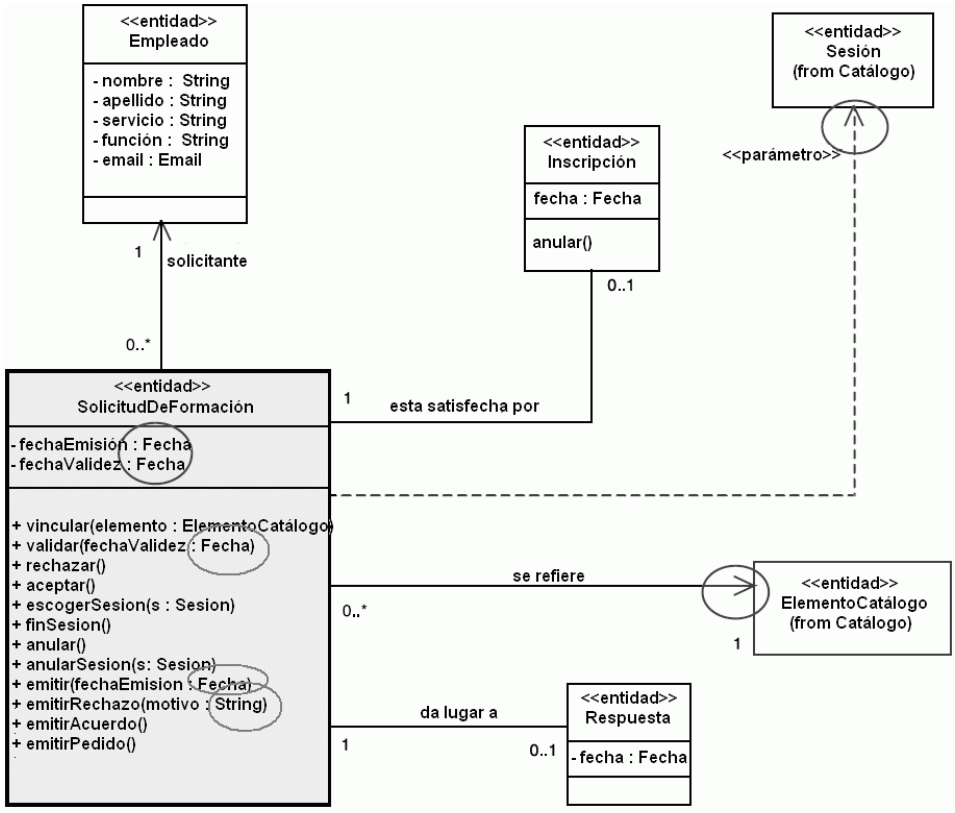
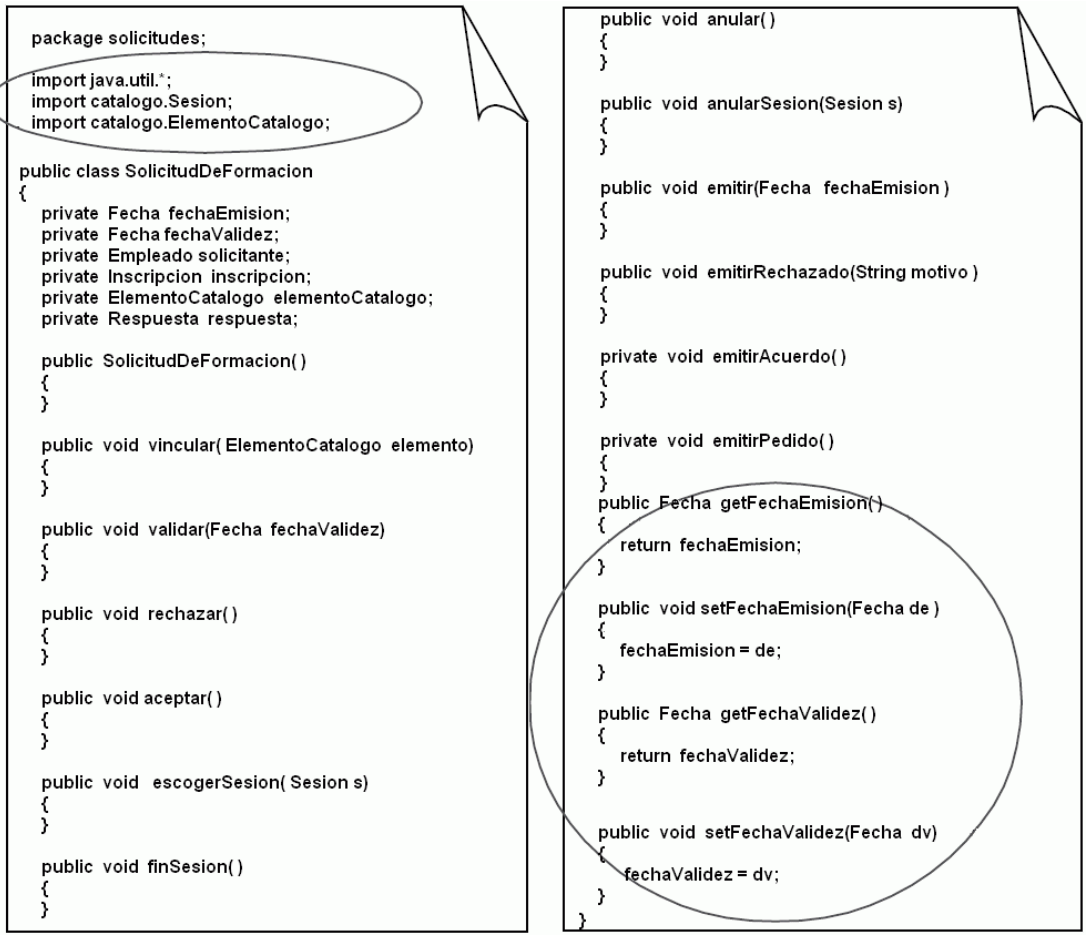


Figura 72: La clase SolicitudDeFormación y sus relaciones.

El código Java correspondiente se muestra en la figura 73. Se observarán las directivas de importación así como los cuatro últimos métodos que permiten el acceso en lectura (get) y en escritura (set) a los atributos, para respetar el principio de encapsulación.



```
package solicitudes;

import java.util.*;
import catalogo.Sesion;
import catalogo.ElementoCatalogo;

public class SolicitudDeFormacion
{
   private Fecha fechaEmision;
   private Fecha fechaValidez;
   private Empleado solicitante;
   private Inscripcion inscripcion;
   private ElementoCatalogo elementoCatalogo;
   private Respuesta respuesta;

   public SolicitudDeFormacion( )
   {
   }

   public void vincular( ElementoCatalogo elemento)
   {
   }

   public void validar(Fecha fechaValidez)
   {
   }

   public void rechazar( )
   {
   }

   public void aceptar( )
   {
   }

   public void escogerSesion( Sesion s)
   {
   }

   public void finSesion( )
   {
   }

   public void anular( )
   {
   }

   public void anularSesion(Sesion s)
   {
   }

   public void emitir(Fecha fechaEmision )
   {
   }

   public void emitirRechazado(String motivo )
   {
   }

   private void emitirAcuerdo( )
   {
   }

   private void emitirPedido( )
   {
   }
   public Fecha getFechaEmision( )
   {
      return fechaEmision;
   }

   public void setFechaEmision(Fecha de )
   {
      fechaEmision = de;
   }

   public Fecha getFechaValidez( )
   {
      return fechaValidez;
   }

   public void setFechaValidez(Fecha dv)
   {
      fechaValidez = dv;
   }
}
```

Figura 73: Esqueleto del código Java de la clase SolicitudDeFormación.

Producción del esqueleto de código C#

Para mostrar que nuestro diseño permite enseguida fácilmente derivar el código en cualquier lenguaje orientado a objetos, a continuación utilizaremos el lenguaje C# (de la plataforma .NET) en lugar de Java.

Tengamos en cuenta que en C#, se utilizaría probablemente el concepto de propiedad (property) para codificar los accesos de manera más elegante, por ejemplo:

```
Public DateTime  FechaValidez
{
    get {return m_FechaValidez; }
    set {m_FechaValidez=value; }
}
```

Producción de un esqueleto de código Java (continuación)

A partir de la figura 62, proponemos un esqueleto de código Java para la clase Formación.

Retomando el esquema 62, eliminamos lo que no se refiere a la clase Formación.

Algunas dificultades suplementarias aparecen, con relación a la cuestión anterior:

- la relación de generalización con ElementoCatálogo,
- las multiplicidades «1..* » con Tema y «0. * {ordered}» con Sesión.

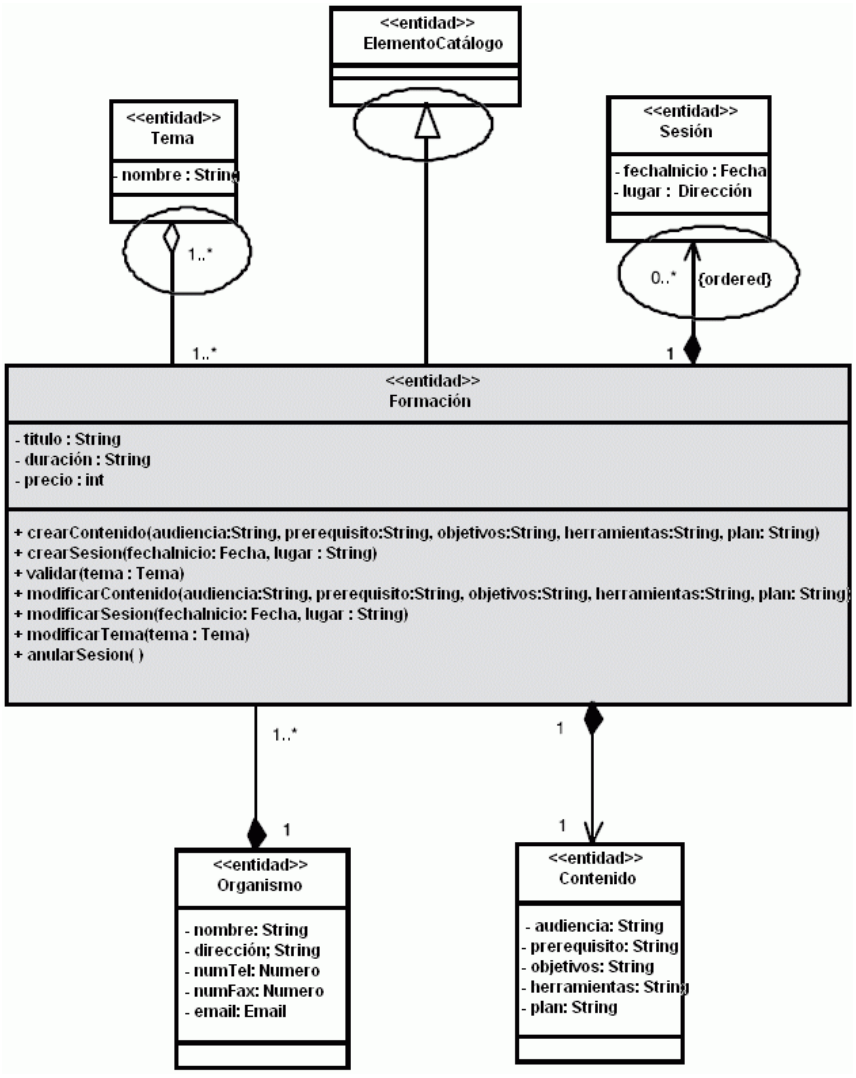


Figura 75: La clase Formación con sus relaciones.

Las reglas anteriores no bastan ya. Vimos un ejemplo de transformación de una asociación navegable de multiplicidad «1» (o «0..1»), pero cómo traducir las asociaciones navegables de multiplicidad «*»?

TRADUCCIÓN EN JAVA DE LAS ASOCIACIONES CON MULTIPLICIDAD "*"

El principio es relativamente simple: una multiplicidad «*» va a traducirse en un atributo de tipo colección de referencias de objetos en vez de una simple referencia sobre un objeto.

La dificultad consiste en elegir la buena colección entre las muy numerosas clases básicas que propone Java. Aunque sea posible crear tablas de objetos en Java, no es inevitablemente la buena solución. En la materia, se prefiere más bien recurrir a colecciones, entre las cuales las más utilizadas son ArrayList (antiguamente Vector) y HashMap (antiguamente HashTable). Si se utiliza ArrayList se debe respetar un orden y recuperar los objetos a partir de un índice entero; si se utiliza HashMap se desea recuperar los objetos a partir de una clave arbitraria.

Nota, el JDK 1.5 introduce las colecciones con tipos llamadas «generics». Vamos a poder declarar en adelante listas de sesiones y maps de temas, como se ilustra en la cuestión siguiente. Se tienen algunos ejemplos de las soluciones que deben adoptarse para efectuar una elección pertinente:

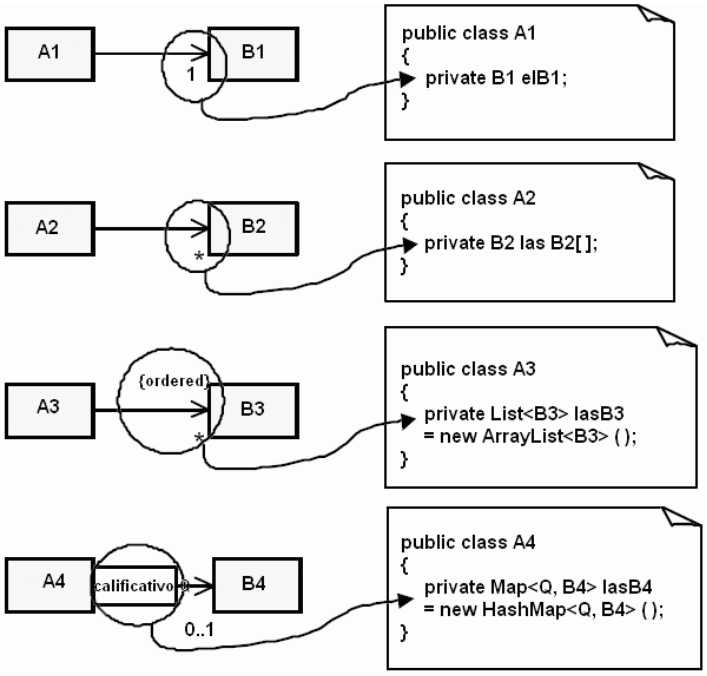


Figura 76: Traducciones posibles de las asociaciones en Java 1.5.

Para la clase Formación, vamos a utilizar:

- un ArrayList para la asociación ordenada con la clase Sesión;
- un HashMap para la asociación con la clase Tema, más bien que una simple tabla: nos serviremos del nombre de tema como calificativo.

Todas estas explicaciones nos conducen al código siguiente para la clase Formación.

```
package catalogo;

import  java.util.*;

public  class Formacion   extends ElementoCatalogo
{
   prívate  String  titulo;
   prívate  short  duracion;
   prívate  int  precio;
   prívate  List<Sesion> sesiones = new ArrayList<Sesion>( );
   prívate  Map<String, Tema>  temas = new HashMap<String, Tema>( );
   prívate  Contenido  contenido;
   prívate  Organismo  organismo;

   public  Formacion( )
   {
   }

   public  void  crearContenido(String audiencia, String prerequisito, String objetivos,
     String herramientas, String plan)
   {
   }

   public  void  crearSesion(Fecha fechaInicio, String lugar)
   {
   }

   public  void  validar(Tema  tema)
   {
   }

   public  void  modificarContenido(String audiencia, String prerequisito, String objetivos,
     String herramientas, String plan)
   {
   }

   public  void  modificarSesion(Fecha fechaInicio, String lugar)
   {
   }

   public  void  modificarTema(Tema  tema)
   {
   }

   public  void  anularSesion( )
   {
   }

   public  String  getTitulo( ) {return  titulo;}
   public  void  setTitulo(String  t) {titulo = t;}
   public  short  getDuracion( ) {return  duracion;}
   public  void SetDuracion(short  d) {duracion = d;}
   public  int  getPrecio( ) {return  precio:}
   public  void  setPrecio(int  p) {precio = p;}

}
```

Figura 77: Esqueleto del código Java 1.5 de la clase Formación.

Producción de un esqueleto de código en C# (continuación)

TRADUCCIÓN A C# DE LAS ASOCIACIONES CON MULTIPLICIDAD "*"

El principio es idéntico al de Java: una multiplicidad «*****» va a traducirse en un atributo de tipo colección de referencias de objetos en vez de una simple referencia sobre un objeto.

Como en Java, se prefiere recurrir a colecciones, entre las cuales las más utilizadas son ArrayList, SortedList y HashTable. Utilizamos ArrayList si se debe respetar un orden y recuperar los objetos a partir de un índice entero; se utiliza HashTable o SortedList si se desea recuperar los objetos a partir de una clave arbitraria. Se muestran algunos ejemplos de las soluciones que deben adoptarse para efectuar una elección pertinente:

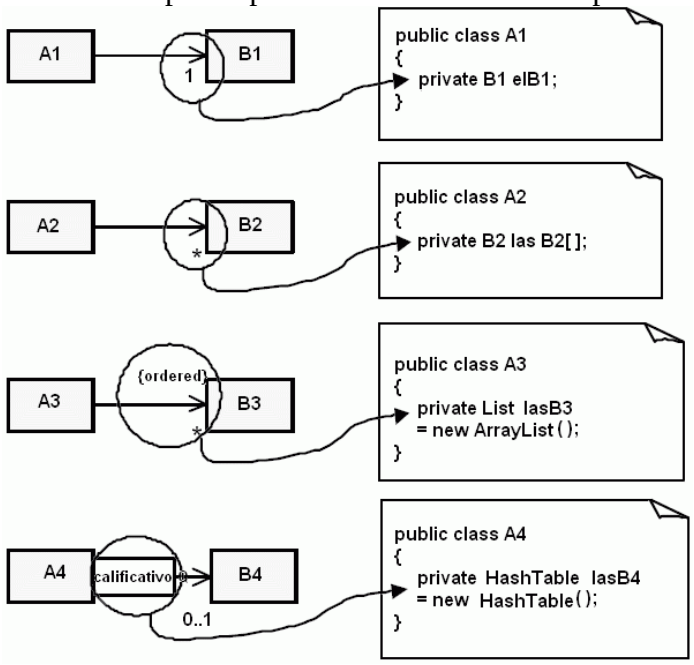


Figura 78: Traducciones posibles de las asociaciones en C#.

Hay que tener en cuenta que las colecciones parametrizadas se anunciaron también en la próxima versión de C# (.NET 2.0). El tercer caso daría por ejemplo una lista genérica como sigue:

```
public class A3
{
private List<B3>   lasB3 = new ArrayList ();
}
```

Para la clase Formación, vamos a utilizar:

- un ArrayList para la asociación ordenada con la clase Sesión;
- un HashTable para la asociación con la clase Tema, más bien que una simple tabla: nos serviremos del nombre de tema como calificativo.

Todas estas explicaciones nos conducen al código siguiente para la clase Formación.

```
namespace  Catalogo;
{
   using System;

   public class Formacion  :  ElementoCatalogo
   {
      private  string  Titulo;
      private  short  Duracion;
      private  int  Precio;
      private  List  Sesiones = new ArrayList( );
      private  HashTable  Temas = new HashTable( );
      private  Contenido  Cont;
      private  Organismo  Org;

      public  Formacion( )
      {
      }

      public  void  crearContenido(string  Audiencia, string  Prerequisito, string
        Objetivos, string  Herramientas, string  Plan)
      {
      }

      public  void  crearSesion(Fecha fechaInicio, string  Lugar)
      {
      }

      public  void  validar(Tema  T)
      {
      }

      public  void  modificarContenido(string  Audiencia, string  Perequisito, string
       objetivos, string  Herramientas, string  Plan)
      {
      }

      public  void  modificarSesion(Fecha fechaInicio, string  Lugar)
      {
      }

      public  void  modificarTema(Tema  T)
      {
      }

      public  void  anularSesion( )
      {
      }

      public  string  getTitulo( ) {return  Titulo;}
      public  void  setTitulo(string  T) {Titulo = T;}
      public  short  getDuracion( ) {return  Duracion;}
      public  void SetDuracion(short  D) {Duracion = D;}
      public  int  getPrecio( ) {return  Precio:}
      public  void  setPrecio(int  P) {Precio = P;}
   }
}
```

Figura 79: Esqueleto del código C# de la clase Formación.

Etapa 16: Despliegue de la aplicación

Vamos finalmente a describir la implantación física de nuestra aplicación de gestión de las solicitudes de formación gracias a un último tipo de diagrama propuesto por UML: el diagrama de despliegue.

DIAGRAMA DE DESPLIEGUE

El diagrama de despliegue muestra la configuración física de distintos materiales que participan en la ejecución del sistema, así como los artefactos que soportan.

Este diagrama está constituido por «nodos» conectados por vínculos físicos. Los símbolos de los nodos pueden contener artefactos (y no componentes como en UML 1).

ARTEFACTO

Un artefacto modela una entidad física como un archivo. Se le representa por un rectángulo que contiene la palabra clave «artefacto». Se aclara la presencia de un artefacto sobre un nodo de despliegue imbricando su símbolo en el del nodo que engloba.

Si un artefacto implementa un componente o una clase, se dibuja una flecha punteada con la palabra clave «manifiesto» que va del símbolo del artefacto al símbolo del componente que implementa.

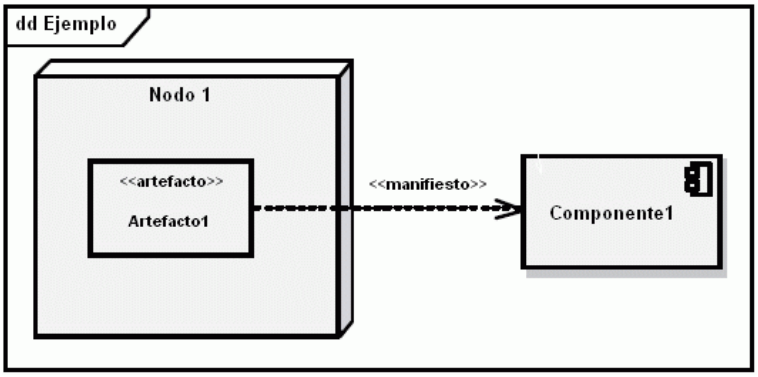


Figura 80: Notación del diagrama de despliegue en UML 2.

El diagrama de despliegue se modificó notablemente con la nueva versión UML 2. Esto viene principalmente de la redefinición del concepto de componente, más lógico en UML 2 y en consecuencia menos físico. La introducción del nuevo concepto de artefacto permite separar mejor la manifestación física del componente de su representación lógica.

Propondremos un diagrama de despliegue realista para las tres primeras iteraciones del sistema de gestión de las solicitudes de formación.

Cada actor tiene su propio puesto cliente que es una «PC» conectada al servidor intranet de la empresa, él cual por sí mismo es un servidor PC NT. Este servidor de la intranet contiene en particular la aplicación de autenticación.

Cada uno de los dos actores tiene por otro lado su propia interfaz humano-computadora, materializada por una página JSP. Estas dos JSP utilizan un mismo servicio de autenticación general, contenido por el servidor intranet. El catálogo se almacena en una base de datos específica, así como la de los empleados.

El servidor de negocios alberga por su parte las otras aplicaciones así como las bases de datos. Se trata de una máquina Unix y esto por razones históricas…

El diseño, realizado con la herramienta Enterprise Architect que proporciona sus propios iconos para los distintos tipos de nodo, se utiliza para generar la siguiente figura.

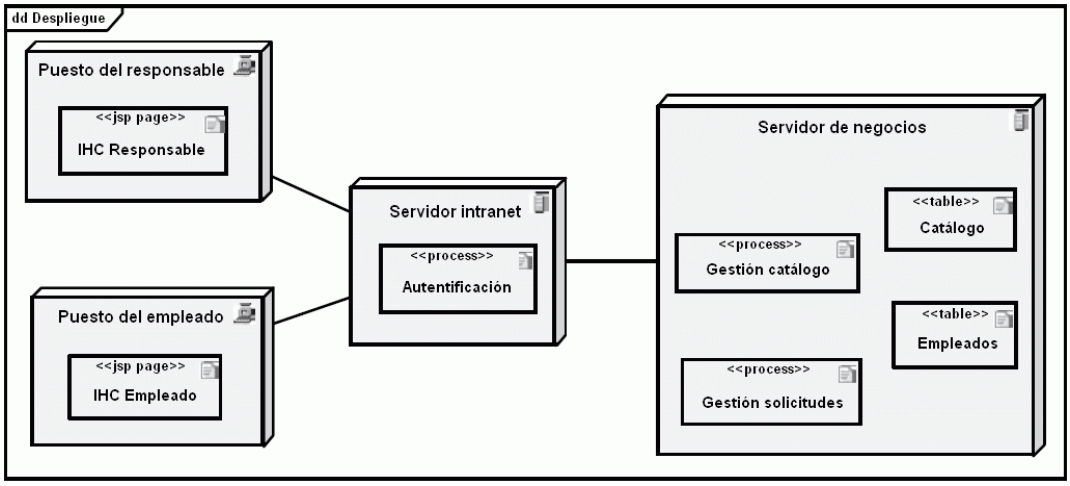


Figura 81: Diagrama de despliegue correspondiente a las tres primeras iteraciones.