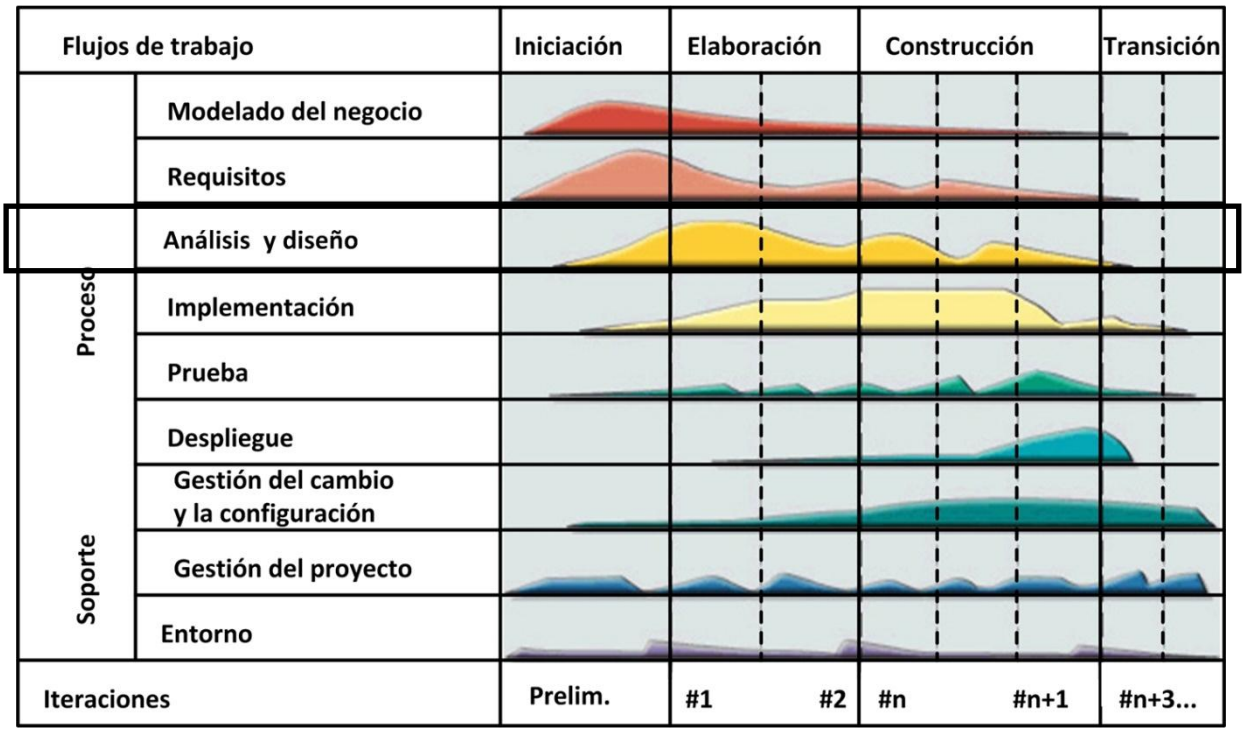
UML
(Lenguaje Unificado de Modelado)

U NIFIED M ODELING L ANGUAJE

Capítulo IV
Análisis de Requisitos con UML

1. Introducción al análisis

Cuan se lleva a cabo proyectos de desarrollo de software en los que se aplican la notación UML, estos son dirigidos por casos de uso, los requerimientos funcionales del sistema se modelan en términos de **actores** y **casos de uso**. Los actores representan a los usuarios del sistema o a otros sistemas que interactúan con el sistema bajo análisis, y los casos de uso representan secuencias de acciones (funciones) que el sistema ejecuta para aportar un resultado (salida) de valor a los actores. El conjunto de actores y casos de uso que forman el sistema conforman lo que se denomina "Modelo de casos de uso".



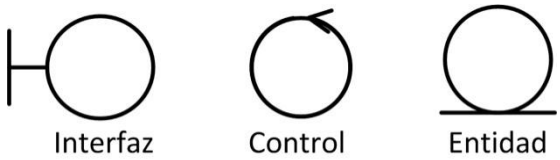
Cuando se ha obtenido una parte (o todo) el modelo de casos de uso, y sobre todo en aquellos casos en que el sistema que se está desarrollando no sea muy sencillo, es de mucha utilidad construir un "Modelo de análisis", esto significa "realizar los casos de uso", como un paso previo al diseño del sistema. Si el sistema es sencillo y los desarrolladores son experimentados, no es necesario realizar el "Modelo de análisis".

El "Modelo de análisis" es una especificación semi-detallada de los requerimientos y es una aproximación al diseño del sistema (un borrador de lo que será el diseño). Se utiliza para entender de forma más precisa los casos de uso y refinarlos expresándolos como colaboraciones entre objetos conceptuales (clasificadores o clases de análisis) que describen características estructurales y de comportamiento del sistema. El análisis se realiza durante el desarrollo del sistema, pero luego no se le realiza mantenimiento. Este se aplica al diseño, es decir que si en el futuro se realizan cambios en el sistema, se mantiene actualizado el diseño y no el análisis, esto es así debido a que el análisis se utiliza para entender el problema y poder construir el diseño, pero una vez construido este, no es necesario mantener el análisis.

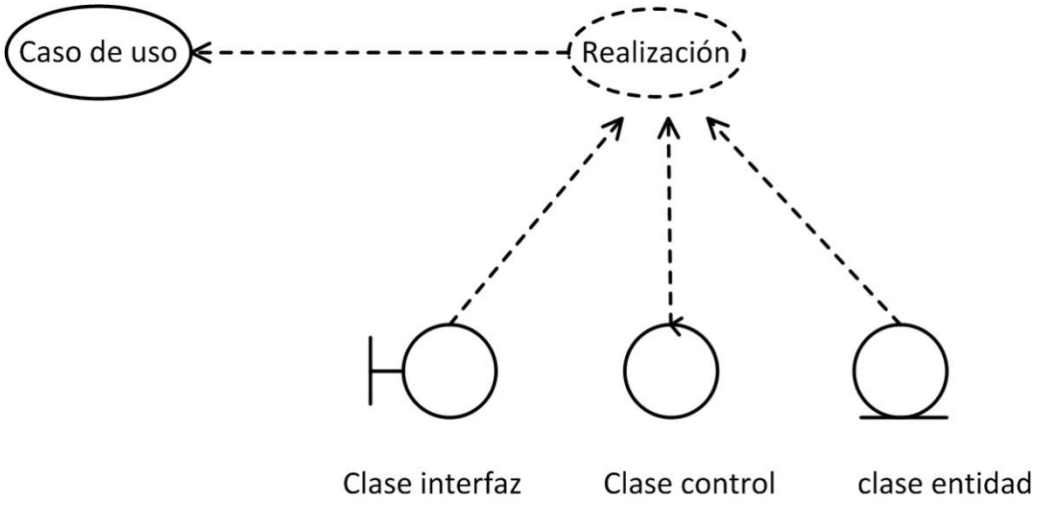
Por cada caso de uso se debe hacer una **realización**. Esta **realización** se asociada al caso de uso, y debe contener las descripciones de las interacciones, que se manifiestan en diagramas de colaboración o de secuencia, entre objetos durante la ejecución del caso de uso.

El tipo de objetos (clase a la que pertenecen) que participan de las interacciones, puede ser de los siguientes tres tipos y cada una tiene su correspondiente esteriotipo:

- **Interfaz (boudary):** relación entre el sistema y sus actores
- **Entidad (Entity)**: información utilizada en el sistema y que es persistente
- **Control (control)**: la lógica del sistema, este control puede ser las transacciones, el secuenciamiento y la coordinación entre los objetos.



La siguiente figura intenta síntetizar los conceptos anteriormente mencionados mediante su representación en UML (del inglés *Unified Modelling Language*, Lenguaje de Modelado Unificado):



Resumiendo, el Modelo de análisis no da una primera aproximación (acercamiento) a la estructura que tendrá el sistema, y la expresa en términos de objetos conceptuales de tipo: **Interfaz**, **Control** y **Entidad**; describiendo el comportamiento del sistema mediante la forma de interactuar de los objetos conceptuales, para ello hace uso de la "realizaciones" de los casos de uso.

2. Objetivo

El objetivo del modelo de análisis, es utilizarlo como un instrumento que permita a quienes desarrollan el sistema, poder documentar el análisis de los requerimientos (modelo de casos de uso). Su importancia radica en:

- Permite especificar con más precisión los requerimientos de los usuarios.
- Se utiliza el lenguaje de los desarrolladores (diagramas UML) y no el lenguaje del cliente como en los casos de uso. Además permite pensar en aspectos internos del funcionamiento del sistema.
- Facilita a los desarrolladores el entendimiento general del sistema estructurando los requerimientos.
- Proporciona un primer acercamiento al diseño del sistema, permitiendo encarar el problema desde lo "general" hacia lo "particular".

Por lo dicho anteriormente podemos decir que el modelo de análisis cumple la función de intermediario, lo que permite un pasaje más suave entre los requerimientos (modelo de casos de uso) y el diseño (modelo de diseño) del sistema.

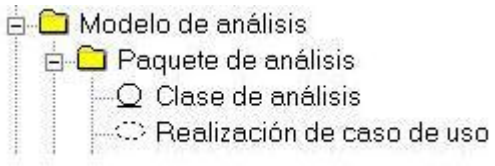
En la siguiente tabla se muestran las diferencias entre el modelo de casos de uso y el modelo de análisis.

Modelo de casos de uso	Modelo de análisis
Se describe en el lenguaje del cliente.	Se describe en el lenguaje de los desarrolladores.
Vista externa del sistema	Vista interna del sistema
Se estructura en casos de uso brindando una estructura a la vista externa del sistema.	Se Estructura con clases de análisis brindando una estructura a la vista interna del sistema
Es un "contrato" entre el cliente (o dueño de la idea) y los desarrolladores sobre lo que el sistema debe y no debe hacer.	Es usado por los desarrolladores para entender cómo funciona el sistema, que luego debe ser diseñado e implementado
Se permiten redundancias e inconsistencias entre requerimientos	No se permiten redundancias e inconsistencias entre requerimientos
Captura la funcionalidad del sistema	Define cómo realizar la funcionalidad dentro del sistema, funciona como una primera aproximación al diseño
Define los casos de uso para que posteriormente sean analizados en el modelo de análisis.	Define las **realizaciones** de casos de uso, cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

Cuzcano Quintin, S. (s.d).

3. Composición del modelo

Las herramientas CASE, como Rational Rose muestran el modelo de análisis expresado como una jerarquía de **paquetes** (del inglés *package*), que a su vez contienen **clases de análisis y realizaciones de casos de uso**. La imagen siguiente es una captura de pantalla de menú desplegable de la herramienta CASE, donde se puede ver esta jerarquía.



Un paquete de análisis es una abstracción que permite agrupar clases de análisis y las realizaciones de casos de uso que tengan funcionalidades similares. Los paquetes pueden llegar a representar subsistemas o posiblemente **capas** (*layers*) en el modelo de diseño.

Las clases que se obtienen en el análisis son abstracciones, por medio de las cuales se muestra la tanto estructura como el comportamiento del sistema, Este nivel de detalle es lo que podría denominarse "de grano grueso", es decir, que no se tiene en cuenta los atributos, operaciones y sus relaciones. Se habla más bien de responsabilidades y la interacción con las demás clases de análisis. Pero si se descubren los atributos es bueno ir tomando nota de ellos, aunque no es la misión del análisis.

Una realización de un caso de uso es una abstracción, en la cual se muestra cómo interactúan los objetos entre sí, que pertenecen a las clases de análisis, sin tener en cuenta la cardinalidad y la multiplicidad, para realizar las distintas secuencias de un caso de uso.

En esta etapa se recomienda generar una descripción de la arquitectura desde el punto de vista del análisis, que es un documento en el cual se resaltan los componentes más significativos desde el punto de vista arquitectónico.

3.1 Paquetes de análisis

En los paquetes de análisis se pueden agrupar las clases de análisis y realizaciones de casos de uso que tengan funcionalidades similares. Al separar el análisis en paquetes nos permite dividir el problema para ser analizado por diferentes grupos de analistas. Es decir que se debe tener las clases lo menos ligadas entre sí que se pueda.

Se recomienda además que estos paquetes debieran tener un bajo acoplamiento y una alta cohesión.

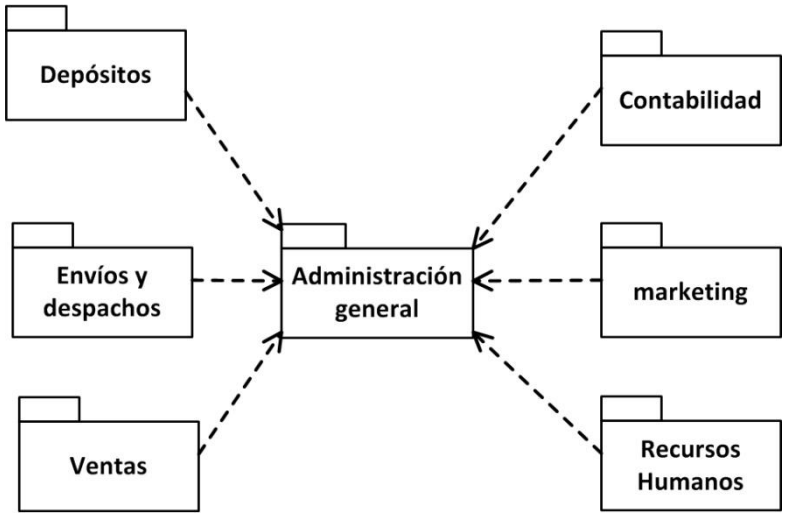
Alta cohesión: "cohesión es la medida en la que un componente se dedica a realizar solo la tarea para la cual fue creado, delegando las tareas complementarias a otros componentes", se entiende por alta cohesión a que los contenidos del paquete deben estar fuertemente relacionados desde el punto de vista funcional. La información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase.

Bajo acoplamiento: "El acoplamiento es la medida en que los cambios de un componente tiende a necesitar cambios de otro componente", acoplamiento es que las dependencias entre paquetes deben ser mínimas.

Para representar un paquete de análisis se utiliza el símbolo de paquete UML, este se puede observar en la figura siguiente.

Los paquetes de análisis por lo general coinciden con los paquetes que ya se

habían identificados en el modelo de casos de uso.



Ejemplo: el siguiente diagrama muestra la descomposición en paquetes de análisis para un sistema de administración general. En este sistema, se han identificado los siguientes paquetes:
Administración general: en este paquete se agrupan las clases vinculadas con la administración general del sistema.
Depósito: en este paquete se agrupan las clases vinculadas el mantenimiento (altas, bajas, modificaciones y consultas) de los artículos del depósito.
Envíos y despachos: contiene las clases vinculadas con la creación, modificación y eliminación de los despachos realizados.
Ventas: contiene las clases vinculadas con las ventas realizadas por le empresa.
Contabilidad: contiene las clases vinculadas las actividades contables de todos los movimientos de dinero realizado dentro de la empresa.
Marketing: contiene las clases vinculadas con las actividades de marketing de los productos producidos en la empresa.
Recursos humanos: contiene las clases vinculadas con el reclutamiento y el pago de sueldos de los empleados.

3.2 Clases de análisis

Características de las clases de análisis:

- Se orientan a los requerimientos funcionales, dejando de lado los requerimientos no funcionales para actividades posteriores. De esto se desprende que las clases de análisis tienen un mayor grado de abstracción, son más conceptuales, por lo tanto son de mayor granularidad que las clases de diseño.
- No se describen los métodos y sus operaciones, sino que su comportamiento se define mediante responsabilidades que se describen en forma textual.
- Los atributos son conceptuales, de alto nivel, (un ejemplo sería decir que el cliente tiene domicilio, sin importar como está compuesto el domicilio), esto difiere de los atributos de las clases de diseño, ya que en el diseño la definición de los atributos está dada por el lenguaje de programación que se utilizará en la implementación. Además es posible que algunos de los atributos de las clases de análisis, luego se convierten en clases en las actividades de diseño e implementación.
- Las relaciones en la que participan las clases son simples, es decir que no se tiene

en cuenta la navegabilidad, aunque se debieran reconocer las relaciones de herencia, si las hubiese.

- Las clases pertenecen a alguno de los tres estereotipos ya mencionados: clase de Interfaz, clase de Control, o clase de Entidad.

3.2.1 Clases de Interfaz

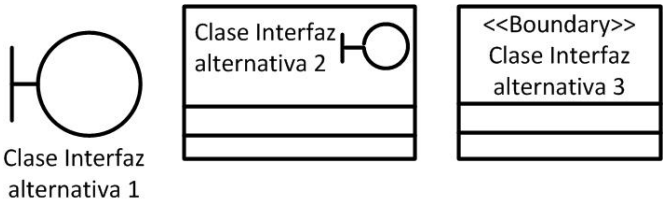
Las clases de interfaz son las que representan la interacción existente entre los actores y el sistema. La interacción actor-sistema representa el flujo de información entre estas partes y no necesariamente son pantallas. Esto es debido a que los actores no siempre son seres humanos. En otras palabras la entrada y salida de datos.

Por lo general estas clases representan abstracciones de pantallas (ventanas), formularios, impresoras, sensores, controladores, válvulas, otros equipos, o cualquier otra interfaz de aplicaciones, componentes, etc.

Las clases de análisis, son abstracciones de las interfaces reales, y son genéricas, es decir que no nos detenemos en los detalles de las mismas, lo que significa que no se debe modelar cada uno de sus componente en forma individual (esto se llevará a cabo en el diseño), En el análisis sólo describiremos las responsabilidades de la clase (los objetivos que se deben alcanzar) con la interacción actor-sistema. Es suficiente con describir en forma genérica la información que es provista por el actor, y los requerimientos que este necesita que el sistema le devuelva.

Por lo dicho hasta ahora podemos afirmar que clase de interfaz se debe relacionar con al menos un actor del sistema.

La siguiente figura muestra el símbolo UML de una clase interfaz y sus variantes de representación.



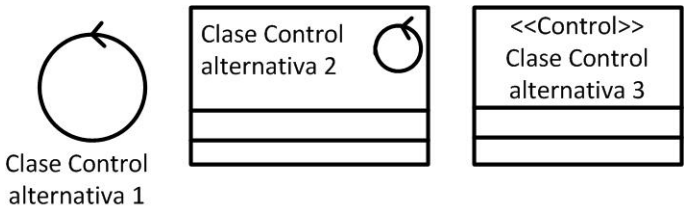
3.2.2 Clases de Control

Las clases de control son las responsables de coordinar los esfuerzos de otras clases. Representan coordinación, secuencia, transacciones y control de los objetos y se usan para encapsular el control de un caso de uso en concreto.

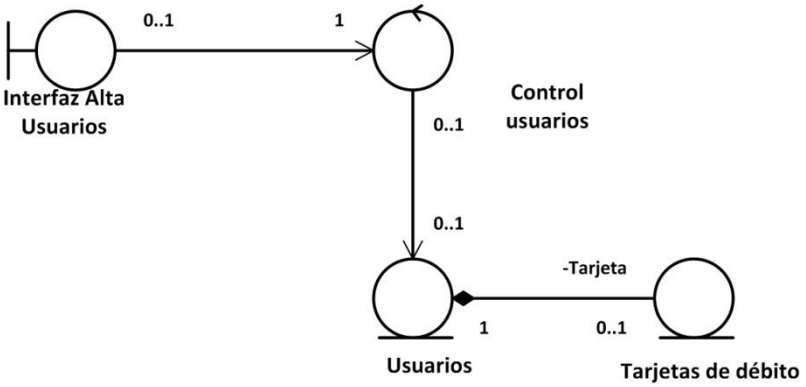
Es la que se encarga de modelar los cálculos y algoritmos complejos del sistema

Las clases de control se encargan de modelar la dinámica del sistema, además coordinan los procesos principales y los flujos de control, son las encargadas de asignar los trabajos a otros objetos de otras clases cuando sea requerido, como lo son objetos de Interfaz o de Entidad.

Se representan mediante el UML de la figura siguiente.



Ejemplo:

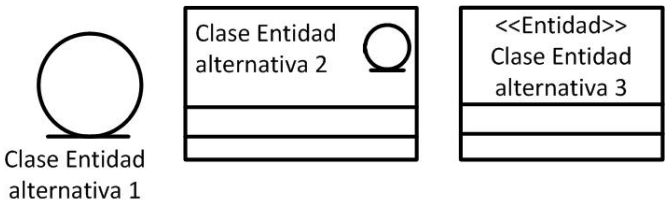


3.2.3 Clases de Entidad

Se utiliza para modelar los datos y esta debiera ser persistente, es decir perdura en el tiempo. Normalmente estas clases se pueden deducir fácilmente del dominio del problema en la etapa de la obtención de los requerimientos.

Al decir que las entidades "guardan" información, no implica que sean pasivas porque puede darse casos que pueden tener un comportamiento complejo relacionado con la información que representan.

Estas clases se representan con el símbolo de UML de la figura siguiente.



3.3 Realizaciones de caso de uso

Realizaciones de casos de uso de análisis: es una colaboración dentro del modelo de análisis que describe cómo se lleva a cabo y como se ejecuta un caso de uso determinado en términos de clases y sus objetos de análisis eniteraciones.

Describe el caso de uso en términos de colaboraciones entre objetos. Una realización de casos de uso vincula a los casos de uso del modelo de casos de uso con las clases y relaciones del modelo de diseño. Una realización de casos de uso especifica que clases deben construirse para implementar cada caso de uso. Es decir que cuando realizamos un caso uso estamos describiendo la manera en que un caso de uso específico se plasma en términos de clases de análisis y sus objetos correspondientes. Por lo tanto, las realizaciones nos permiten la trazabilidad de los componentes de análisis

con los casos de uso del modelo de casos de uso.

Las realizaciones se pueden documentar de las siguientes formas:

- descripciones textuales del flujo de eventos. La forma de describirlo es similar a la de los casos de uso, con la diferencia que se describe el comportamiento interno del sistema en lugar del comportamiento externo que se hace en los casos de uso.
- Diagramas de clases que muestran las relaciones de las clases de análisis,
- Diagramas de interacción (colaboración y/o secuencia), que muestran como colaboran los objetos entre sí. Estas colaboraciones representan los flujos de los eventos o un escenario particular de un caso de uso.

Las realizaciones se denominan igual que el caso de uso que realizan (es decir tienen el mismo nombre), y su forma de representar es con el símbolo de caso de uso UML, pero con líneas entrecortadas, este estereotipado se lo conoce como "realización". La siguiente figura se muestra una realización de caso de uso.

3.4 Descripción de la arquitectura

Cuando hablamos de las vistas de la arquitectura como se mencionó en la figura 35, el análisis hace su aporte a la "vista lógica", incluyendo los siguientes elementos:

- Los paquetes de análisis y las relaciones entre ellos. Los paquetes que aparecen en el análisis puede llegar a convertirse en subsistemas en las etapas de diseño e implementación, aunque no siempre es así. Esto se debe a que cuando se realiza el diseño, los subsistemas pueden cambiar debido a la cantidad de capas lógicas y físicas en las cuales se puede dividir el sistema.
- Las clases de análisis más importantes (consideradas clave), como por ejemplo las clases de entidad que guardan información persistente del sistema y por lo tanto son información fundamental del sistema, las clases de interfaz que son las responsables de comunicarse con los actores (humanos u otros sistemas o mecanismos), las clases de control que son las responsables de coordinar las secuencias de pasos que debe llevar a cabo el sistema.
- Finalmente aporta las realizaciones de los casos de uso, que ya hemos detallado en el capítulo anterior.

El documento que conforma la arquitectura del sistema recibe aporte de varias vistas, una de ellas es la vista lógica que es proporcionada por el análisis.

Vista lógica: captura el vocabulario del ámbito del problema como un conjunto de clases y objetos. El énfasis está en mostrar como los objetos y las clases que componen un sistema implementan el comportamiento requerido del sistema.

Aporte del modelo de análisis: en esta vista lógica se incluyen las partes más relevantes de este modelo, entre los que podemos mencionar están: las clases de análisis más importantes para el diseño, los paquetes de análisis que pueden llegar a transformase en subsistemas en la próxima etapa (diseño) y las realizaciones de los casos de uso priorizados como más importantes.

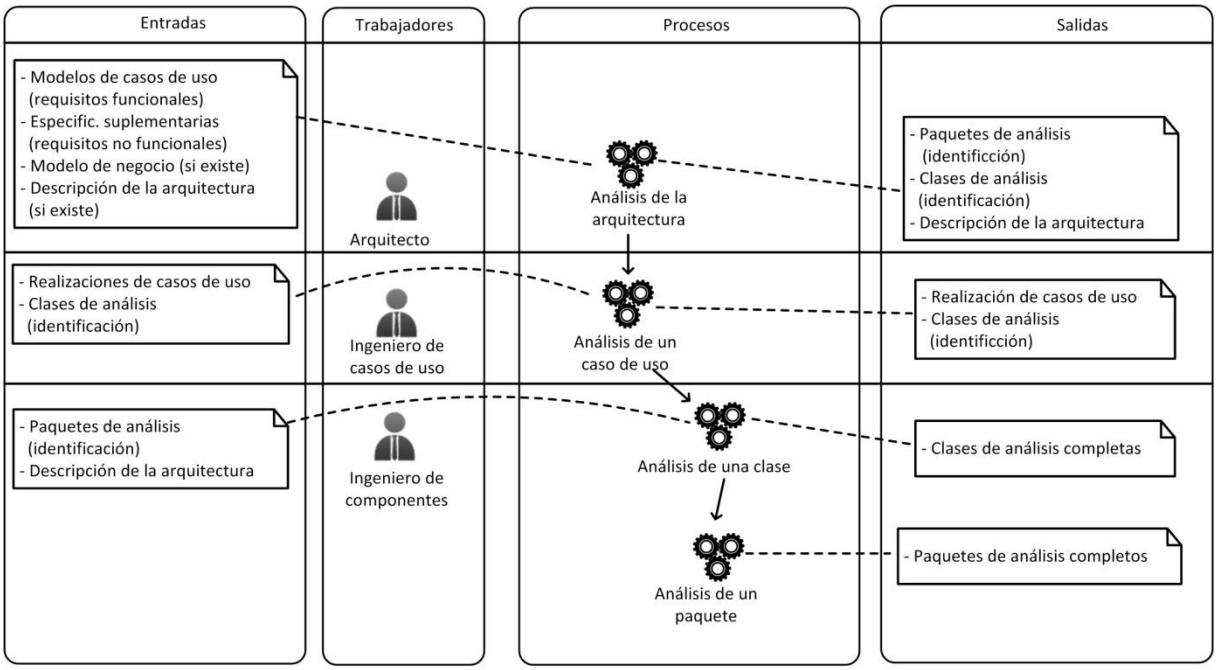
Aclaración: Cuando se consultan a distintos autores verán que hay diferencias en los criterios de cómo clasificar las distintas partes que forman la arquitectura 4 + 1. Por lo tanto aquí se hace esta aclaración. Además cuando se investiga el tema de la arquitectura en general, incluyendo otras arquitectura se pueden encontrar tantas clasificaciones como autores se consulte. No hay un estándar que clasifique las arquitecturas.

4. Actividades para la construcción del modelo de análisis

Como se puede observar en la imagen siguiente, las actividades a realizar son cuatro:

- la primera actividad para la creación del modelo de análisis es el **Análisis de la arquitectura**. Dicha actividad necesita de un conjunto de entradas que permitirán identificar los paquetes de análisis principales, las clases de entidad más obvias, y los requerimientos comunes a todo el sistema bajo análisis.
- La segunda, es el **Análisis de los casos de uso,** esto implica la realización de cada caso de uso. En esta actividad, se identifican las clases de análisis (de Interfaz, de Control y de Entidad), se detallan las interacciones entre las clases, y de ser necesario se realizan los diagramas de secuencias.
- En tercer lugar, se lleva a cabo el **Análisis de las clases**, en esta tarea se documentan las responsabilidades (que luego se transformarán en métodos en el diseño), los atributos y las relaciones entre las clases identificadas en la actividad anterior.
- Por último se realiza el **Análisis de los paquetes,** cuando nos encontramos realizando el análisis, es normal que se descubran nuevos paquetes de análisis, clases de análisis y requerimientos comunes. Por tal motivo se debe realizar el Análisis de los paquetes que ya tenemos encontrados hasta el momento, con el objetivo de refinar los paquetes de análisis identificados. Al realizar esta actividad hay que tener en cuenta los paquetes deben tener bajo acoplamiento y alta cohesión, además debemos determinar que clases de análisis constituirán estos paquetes. Es posible que debamos hacer una redistribución de las clases en los paquetes.

A continuación se muestra un diagrama de actividades UML que resume dichas actividades con sus productos de entrada y de salida.



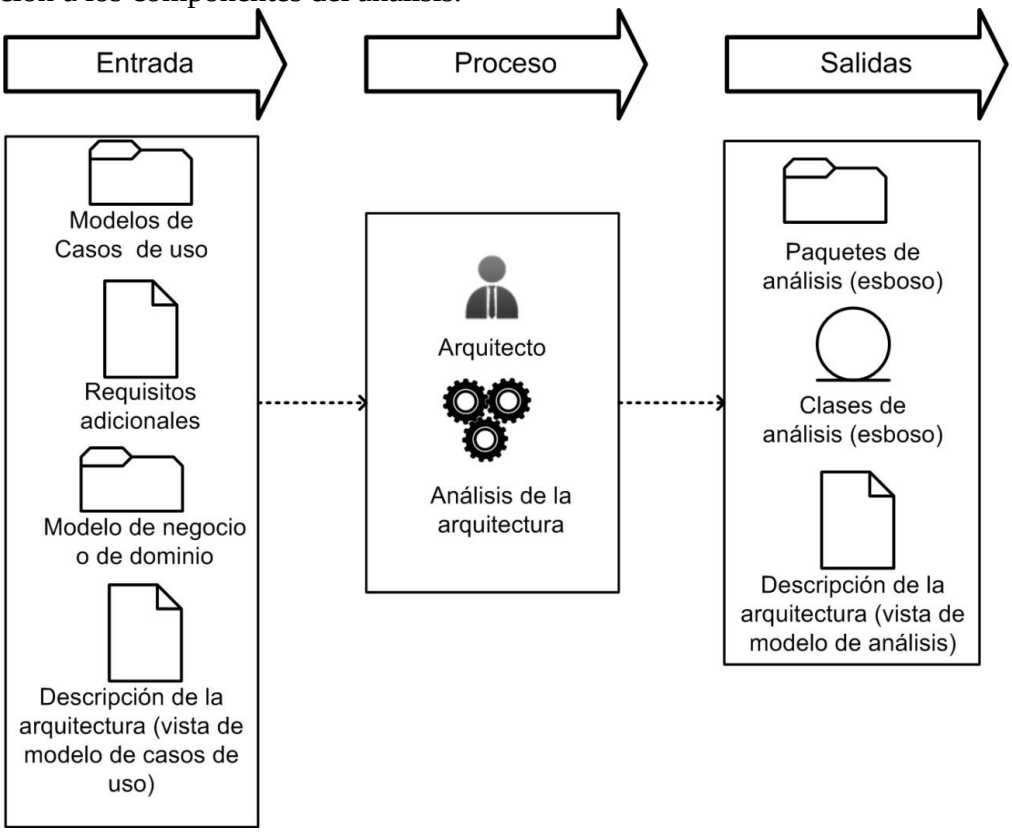
Entradas	Trabajadores	Procesos	Salidas
• Modelo de negocio (requisitos funcionales) • Especificaciones suplementarias (requisitos no funcionales) • Modelo de negocio (si existe) • Descripción de la arquitectura (si existe)	Arquitecto	Análisis de la arquitectura	• Paquetes de análisis (identificación) • Clases de análisis (identificación) • Descripción de la arquitectura
• Realizaciones de los casos de usos	Ingeniero de casos de uso	Análisis de casos de uso	• Realización de los casos de uso • Clases de uso (identificación)
• Paquetes de análisis (identificación) • Descripción de la arquitectura	Ingeniero de componentes	Análisis de una clase	• Clases de análisis completada
	Diseñador de interfaces del usuario	Análisis de un paquete	• Paquetes de análisis completos

Cabe aclarar que el orden asignado a las actividades no implica que deba terminarse una para pasar a la siguiente. Normalmente puede haber superposiciones entre ellas, sobre todo entre las actividades de **Análisis de los casos de uso y Análisis de las clases**, donde es posible que durante la primera actividad se vayan reconociendo responsabilidades, atributos y asociaciones entre las clases, que son tareas de la segunda actividad.

Asimismo, también puede darse la situación en que se efectúe parte del **Análisis de los paquetes** durante las actividades de **Análisis de los casos de uso y Análisis de las clases**, en el caso en que se reconozcan nuevos paquetes o se descubra que se debe reubicar clases durante las mismas.

4.1 Análisis de la arquitectura

El propósito del análisis arquitectónico es definir una arquitectura inicial para el sistema basada en la experiencia ganada de sistemas similares o dominios de problema semejantes. Es una primera aproximación al modelo de análisis en la cual identificaremos los paquetes de análisis, las clases de análisis más obvias, y los requerimientos no funcionales. Al "identificar" estaremos asignando nombre y realizar una breve descripción a los componentes del análisis.



Entradas	Trabajador / Proceso	Salidas
Modelo de casos de uso	Arquitecto: Análisis de la arquitectura	Paquete de análisis (esbozo)
Requisitos Adicionales		
Modelo de negocio o de dominio		Clases de análisis (esbozo)
Descripción de la arquitectura (vista del modelo de casos de uso)		Descripción de la arquitectura (vista del modelo de análisis)

4.1.1 Identificación de paquetes de análisis

Para realizar esta tarea se debe basar en los requerimientos funcionales expresados como casos de uso y el problema del dominio descripto en el documento de requerimientos, también conocidos como ERS (Especificación de Requisitos software). Una forma directa de identificar paquetes de Análisis es asignar las partes principales de varios Casos de Uso a un paquete específico y luego analizar la funcionalidad que le corresponde. La asignación apropiada de Casos de Uso a paquetes específicos incluye los que se requieran para soportar un proceso específico del negocio, un actor específico del sistema o los que están

relacionados por intermedio de generalizaciones y extensiones.

Cuando queremos asignar casos de usos a un paquete podemos tener en cuenta los siguientes puntos:

- Casos de uso que apoyan a un proceso específico del negocio
- Casos de uso que dan información a un actor específico del sistema
- Casos de uso que se relacionan a través de generalizaciones o relaciones de extensión (extend o include).

Esta inclusión de casos de uso en un paquete puede sufrir cambios luego de analizar las colaboraciones entre clases que permiten llevar a cabo los casos de uso, esto puede llevar a la generación de nuevos paquetes o a una distribución de las clases de los casos de uso entre los paquetes existentes.

4.1.2 Identificación de las clases de análisis más obvias

Las clases de análisis más obvias se obtienen a partir del documento de requerimientos o ERS, teniendo de no identificar demasiadas ya que cuando hayamos avanzado en el análisis, al establecer las distintas realizaciones de casos de uso de Análisis se encontrarán la mayor parte de ellas.

Ejemplo:

> La empresa ABC se encarga de vender sus productos en base a notas de pedidos que recibe de sus clientes (contienen el nombre del cliente, la fecha y los artículos que solicita) y luego confecciona las facturas de las mismas de acuerdo a los precios y la descripción de los artículos que se encuentran en el stock.
>
> Los pedidos son recibidos y cargados en el sistema verificando si el cliente existe en la base de datos, si no existe se lo da de alta o se pueden realizar modificaciones de los datos en caso que sea necesario.
>
> Los productos son cargados en el momento que se reciben del proveedor (se desea guardar el código del artículo, la descripción, el proveedor, el precio de compra, el precio de venta y la cantidad disponible.
>
> Las facturas pueden imprimirse en el momento que se carga la nota de pedido o con posterioridad.

Las entidades más obvias que encontramos en el enunciado anterior son las siguientes:
Productos: lista de todos los elementos que se venden.
Notas de pedidos: son los pedidos que realizan los clientes.
Clientes: personas que realizan los pedidos.
Facturas: comprobante de las ventas realizadas en base a los pedidos.
Proveedores: persona o empresa que provee de productos a la empresa ABC.

Como se puede ver las entidades más obvias son aquellas de las cuales queremos guardar información que sea persistente en el tiempo.

4.1.3 Identificación de requisitos no funcionales o requerimientos especiales comunes

Se denominan “requerimientos especiales”, “requerimientos especiales comunes” o “requisitos no funcionales” a aquellos que son detectados mientras se realiza el análisis, pero que no se resuelven inmediatamente, sino que deber atenderse en el diseño y en la implementación.

Entre estos requisitos que son restricciones impuesta al sistema se encuentran aquellos relacionados con los siguientes aspectos:

- Persistencia
- Distribución
- Concurrencia
- Seguridad
- Tolerancia a fallos
- Manejo de transacciones

Ejemplo:

Mientras se realiza el análisis del enunciado anterior se puede obtener el siguiente requisito: No se puede dar de baja a un cliente mientras tenga un pedido pendiente o se le haya realizado una venta con factura tipo "A".

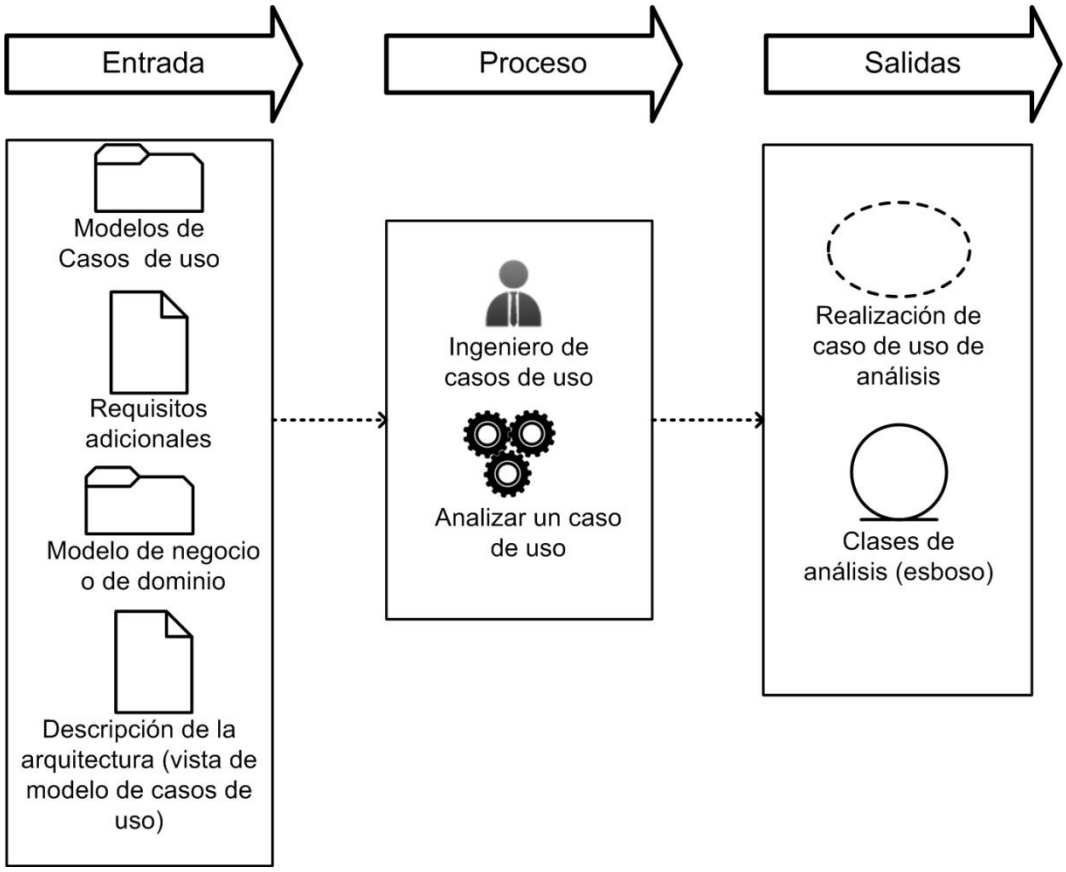
Este requisito será atendido en el diseño, y más precisamente en la implementación cuando se construya el código de "baja de cliente", es en ese momento que se deberá verificar si el cliente tiene pedidos pendientes o facturas del tipo "A", si esto es afirmativo, no se podrá dar de baja a un cliente. Pero si surge durante el análisis es conveniente registrarlo de manera de no pasarlo por alto.

4.2 Análisis de los casos de uso

Al realizar el análisis de caso de uso, estamos realizando los casos de uso. Durante esta actividad muestra como las instancias de las clases de análisis interactúan entre sí, para llevar adelante la funcionalidad del sistema.

Los propósitos de esta actividad son los siguientes:

- Encontrar que clases de análisis interactúan entre sí para realizar el comportamiento del caso de uso.
- Encontrar las responsabilidades que tienen cada una de las clases del caso de uso.
- Detectar los atributos (genéricos) de cada clase que forma el caso de uso



Entradas	Trabajador / Proceso	Salidas
Modelo de casos de uso	Arquitecto: Analizar de la arquitectura	Paquetes de análisis (esbozo)
Requisitos Adicionales		
Modelo de negocio o de dominio		Clases de análisis (esbozo)
Descripción de la arquitectura (vista del modelo de casos de uso)		Descripción de la arquitectura (vista de modelo de análisis)

4.2.1 Identificación de clases de análisis

Para cada caso de uso que se desea realizar se deben identificar las clases de Interfaz, de Control y de Entidad que este necesite para su funcionamiento, asignándoles un nombre representativo y una breve descripción.

Al momento de identificar los tres tipos de clases se deben tener en cuenta los siguientes factores:

Clases de Entidad: estudiar en detalle la descripción del caso de uso que se describe según la plantilla de apartado 2.2.1. y la documentación del modelo de negocio (si existe) y por el modelo de dominio (si existe). Para encontrar estas clases hay que tener en mente la información que debería ser almacenada o manipulada en la realización del caso de uso. Se debe prestar atención de evitar registrar como una entidad a un atributo, si bien el atributo contiene una información que es persistente, este forma parte de una clase entidad junto con

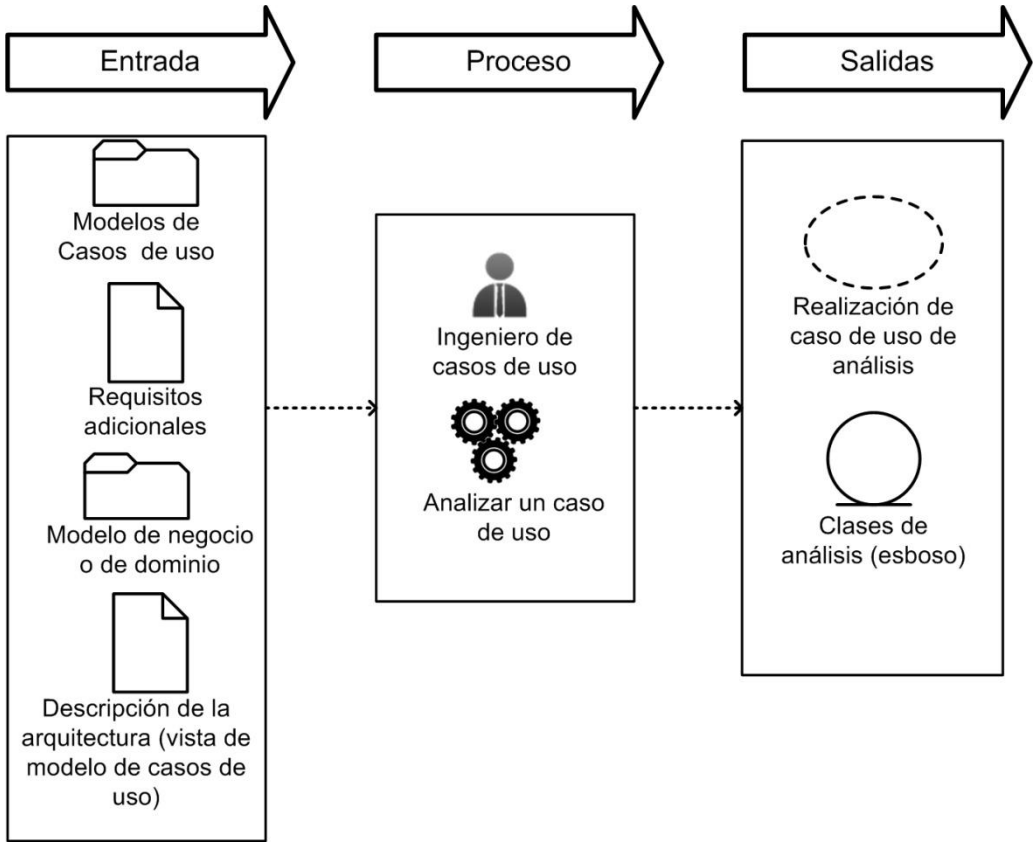
otros atributos, por ejemplo “apellido”, “nombre” y “nro-cliente” guardan información que es persistente, pero no son una entidad, sino atributos de una clase “clientes”.

Clases de Interfaz:

- Para cada actor humano se debe identificar una clase de interfaz central, que representa una pantalla (ventana) principal que le permite al actor interactuar con el caso de uso realizado.
- En general se puede considerar a estas clases como agregados de otras clases de interfaz más primitivas.
- El posible que este mismo actor ya interactúe con otras interfaces, en ese caso tratar de ver si las interfaces existentes no se pueden reutilizar, disminuyendo de esa manera las tareas de programación.
- Identificar los actores no humanos y asignar una interfaz a cada uno, por ejemplo un servidor de correo, cuya interfaz deberá enviar y recibir e-mail. (sistema externo, puede ser software, hardware, terminales, dispositivos, etc.).
- Estas clases representan las interfaces de comunicaciones con el sistema externo.

Clases de control: una vez identificadas las clases de control (ver 3.2.2), se debe a continuación refinar de acuerdo a los requerimientos del caso de uso y se pueden dar los siguientes casos:

- Si la mayoría de la responsabilidad cae sobre el actor, el control puede estar en la clase de interfaz.
- Existen casos en que la clase de control debido a la gran cantidad de trabajo que realiza es muy compleja, Dado este caso se puede descomponer en varias clases de control más específicas, distribuyendo la carga de trabajo.
- Si fuese el caso de que una misma clase de control es utilizada por dos o más actores diferentes (donde cada clase de control tiene una interfaz), suele ser recomendable descomponer la clase de control en dos clases, aislando los cambios en los requerimientos de cada actor.



Entradas	Trabajador / Proceso	Salidas
Modelo de casos de uso	Ingeniero de casos de uso: Analizar un caso de uso	Realización de casos de uso de análisis
Requisitos Adicionales		
Modelo de negocio o de dominio		Clases de análisis (esbozo)
Descripción de la arquitectura (vista del modelo de casos de uso)		

Ejemplo:
En la figura siguiente se muestra un diagrama con las clases de análisis identificadas para el caso de uso Crear usuario para la empresa ABC, dicha empresa se encarga de vender sus productos en base a notas de pedidos (ver punto 4.1.2).

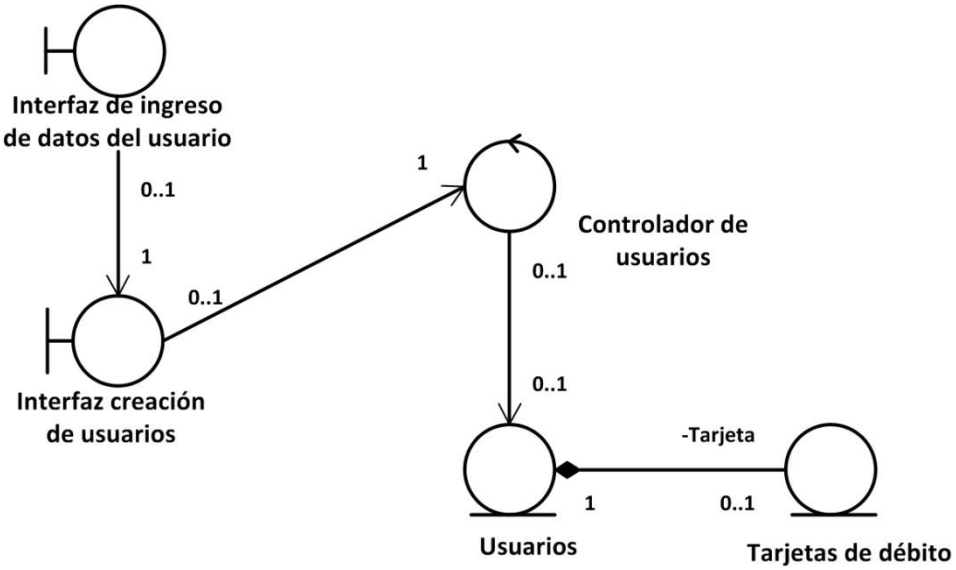
Interfaz de ingreso del usuario (Interfaz): pantalla (o ventana) que permite el ingreso al sistema. Una vez ingresado al sistema se puede activar la Interfaz creación usuarios.

Interfaz creación usuarios (Interfaz): pantalla o ventana que permite ingresar los datos para la creación de usuarios.

Controlador de usuarios (Control): es el controlador encargado de la creación, localización, modificación, eliminación y consultas de usuarios.

Usuario (Entidad): contiene la información persistente de los usuarios del sistema.

Tarjeta de débito (Entidad): contiene la información de las tarjetas de débito de los usuarios.



4.2.2 Descripción de las interacciones entre objetos

Luego de haber identificado todas las clases que pertenecen a la realización de un caso de uso, la siguiente tarea es describir como estos objetos interactúan entre sí. Este proceso se realiza a través de los diagramas de interacción de UML. Entre estos diagramas están los "diagramas de colaboración" o "diagramas de secuencia".

Al momento de realizar estos diagramas se deben tener en cuenta los siguientes elementos:

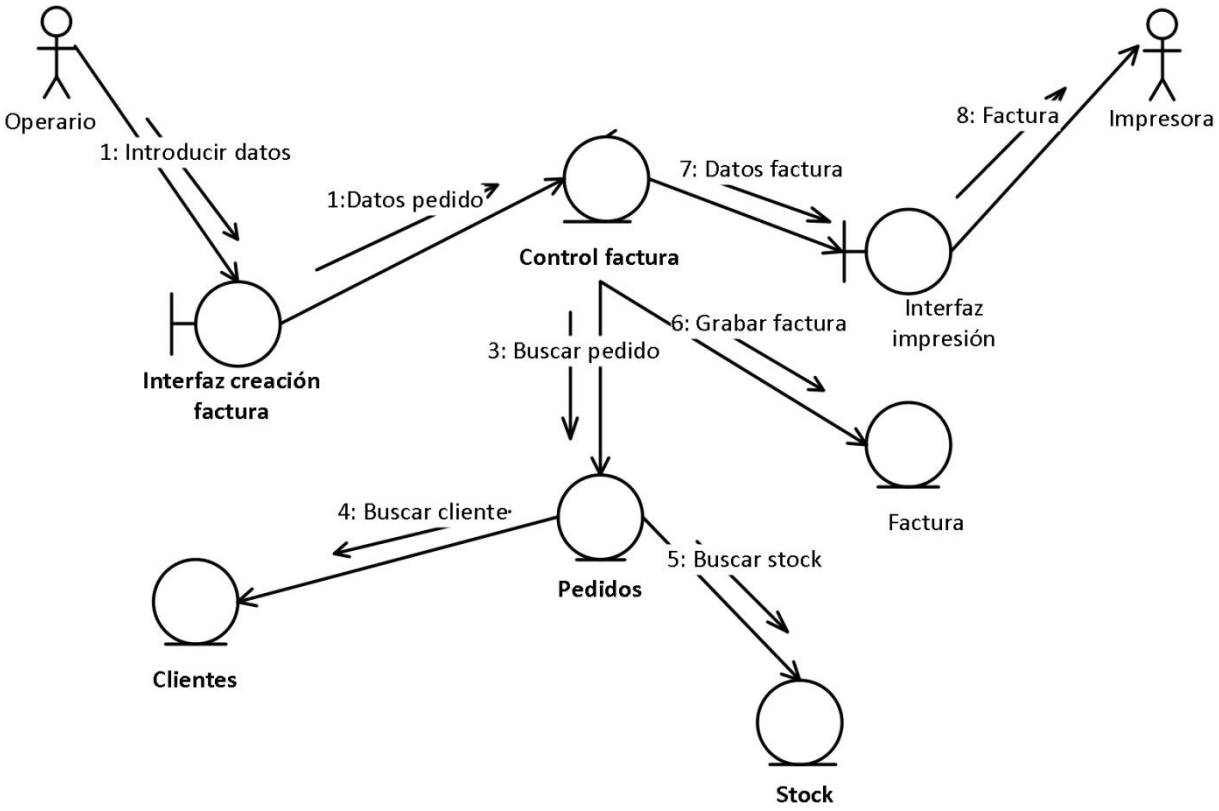
- La mayoría de las veces un caso de uso comienza por un mensaje proveniente desde un actor a un objeto de interfaz.
- Cada clase debe tener al menos un objeto, de lo contrario la clase es superflua y debería ser eliminada.
- En el análisis los mensajes no están asociados a operaciones (métodos), sino solamente se definen sus responsabilidades. El nivel de detalle de las operaciones quedan para el diseño.

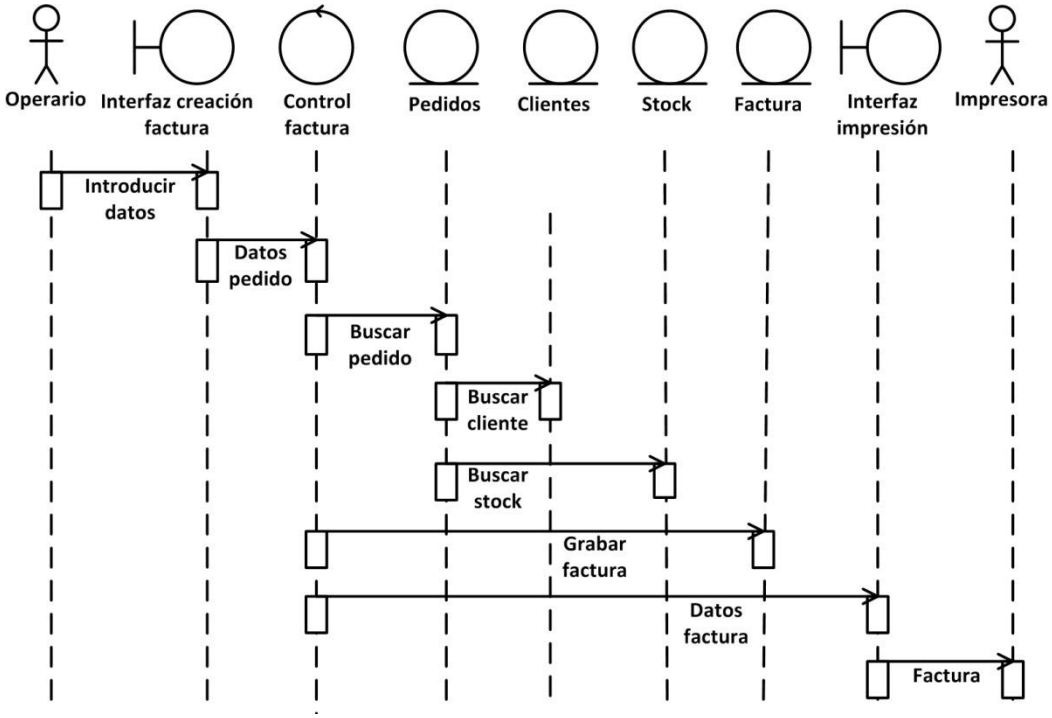
Cuando se necesiten explicaciones para un diagrama se puede agregar un párrafo que lo explique.

Ejemplo:
En la siguiente figura se muestra la interacción entre objetos de un caso de uso llamado "Actualizar usuario". La misma interacción se muestra mediante un "diagrama de colaboración" y mediante un "diagrama de secuencia".

En ambos casos, la interacción se puede describir de la siguiente manera:

1. El operario modifica sus datos en la interfaz y le da la orden de actualizar.
2. La interfaz transmite los datos del pedido al control factura.
3. Control factura busca los datos del pedido de acuerdo a los datos del cliente ingresado.
4. La entidad factura busca los datos del cliente.
5. La entidad factura busca los ítems del stock.
6. El control factura, genera la factura y la almacena en la entidad factura.
7. El control factura envía los datos a la interfaz de impresión de la factura.
8. La interfaz impresión genera la impresión y envía la factura a la impresora.



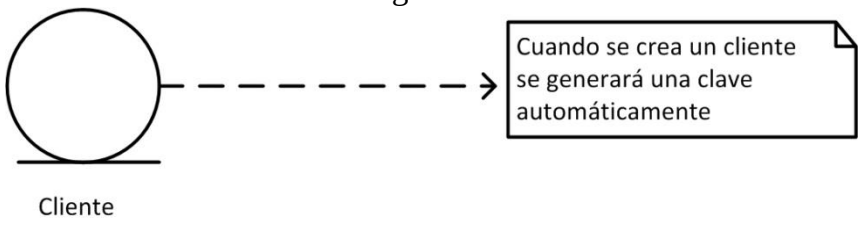


4.2.3 Identificación de requerimientos especiales

Mientras se está trabajando con las clases de análisis y la descripción de las interacciones entre objetos se pueden descubrir requerimientos no funcionales (o requisitos especiales), estos requerimientos se tratarán en las actividades de diseño e implementación. Pero estos requerimientos deben ser capturados (en un documento o en notas y comentarios sobre las clases de análisis y los diagramas), y de ser posible se deben vincular con los requerimientos especiales comunes identificados durante la

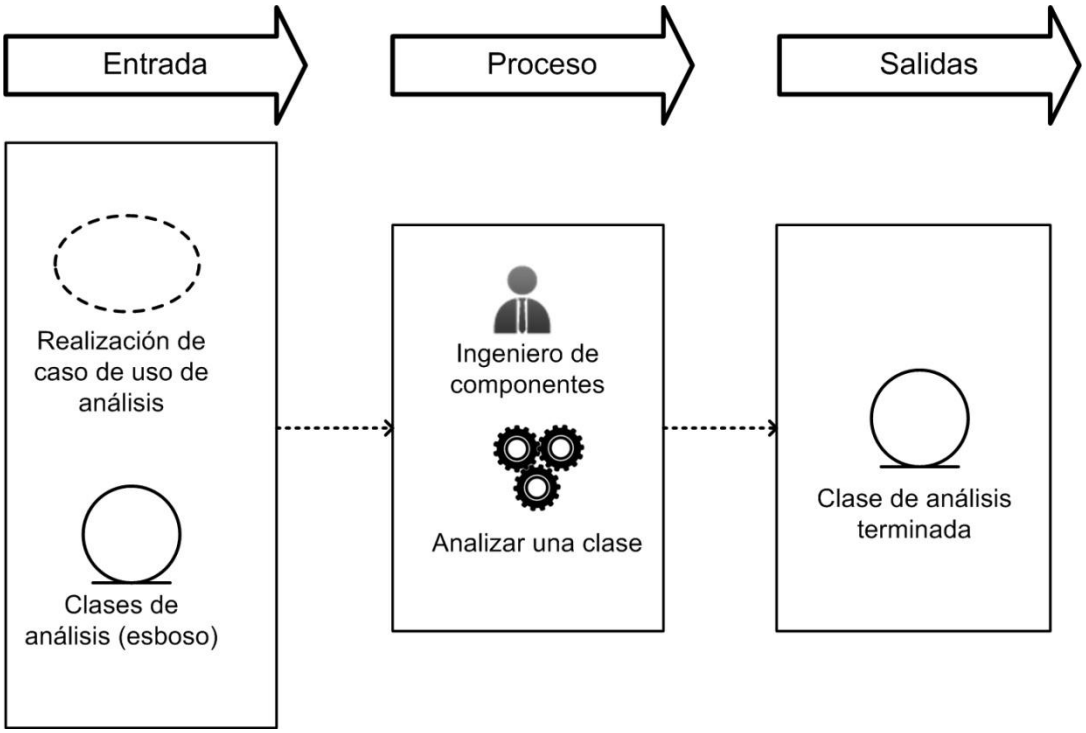
actividad de Análisis arquitectónico.

Ejemplo:
En la figura siguiente muestra un requerimiento especial sobre la clase Usuario, documentado mediante una nota en un diagrama de clases.



4.3 Análisis de las clases

Cuando se realiza en análisis de cada una de las clases lo que estamos realizando es la identificación de las responsabilidades, los atributos, las asociaciones y las generalizaciones de las clases de análisis.



Entradas	Trabajador / Proceso	Salidas
Realización de casos de uso de análisis	Ingeniero de componentes: Analizar una clase	Clases de análisis terminada
Clases de análisis (esbozo)		

4.3.1 Identificar las responsabilidades

Al referirnos a las responsabilidades que posee un objeto nos estamos refiriendo a las acciones que este objeto puede llevar a realizar y el conocimiento que este objeto guarda de sí y que puede proveer a otros objetos.

Todas las clases de análisis tienen un conjunto de responsabilidades. Si tuviese una única responsabilidad es una clase demasiado sencilla, sin embargo otras pueden tener muchas responsabilidades por lo tanto puede ser demasiado compleja. En estos casos

puede ser recomendable dividir esta clase en dos o más clases más sencillas.

Para identificar que responsabilidades tienen las clases se debe examinar todos los diagramas de interacción en los que el objeto está (clase) está presente, analizando los mensajes que reciben esos objetos (representados por flechas). Estos mensajes definen las responsabilidades que tendrá la clase.

Cada una de las responsabilidades encontradas se debe documentar con un nombre y una descripción textual.

Ejemplo:

Si tomamos como ejemplo la empresa ABC y en ella nos detenemos en el objeto Stock veremos que recibe la flecha "Buscar Stock" lo que nos indica que su responsabilidad será buscar en el stock.

4.3.2 Identificar los atributos

Un atributo es una especificación que define una propiedad de un Objeto. Estos atributos son condición necesaria para llevar a cabo las responsabilidades de una clase.

Al momento de identificar los atributos, hay que tener en cuenta lo siguiente:

- Los atributos siempre son sustantivos.
- Durante el análisis los tipos de los atributos deben ser conceptuales. Por lo tanto no deben describirse como tipos de datos de lenguajes de programación, como por ejemplo: números, palabras, etc.; en lugar de *integer*, *char*, *string*, etc.
- Es posible que una clase de análisis se vuelva muy compleja debido a la cantidad de atributos que posee, en este caso se puede dividir la clase en dos clases, distribuyendo los atributos.
- Las clases de entidad tiene la característica de que sus atributos resultan obvios, por ser la información que manejan (almacenan).
- En las clases de interfaz con actores humanos, los atributos normalmente representan elementos de información que son manejados por los actores, tales como campos para el ingreso de datos.
- En las interfaces que interactúan con otros sistemas, los atributos, normalmente representan propiedades de una interfaz de comunicaciones.
- Las clases de control raramente poseen atributos, debido a su corta vida (existen en momento de ejecución). A pesar de ello, pueden contener atributos que representen valores acumulados o derivados durante la realización de un caso de uso.

Ejemplo: Una clase de entidad Usuario de la figura 82 se identificaron los siguientes atributos.

Id: identificador del usuario.
Usuario: nombre de usuario
Contraseña: clave de acceso al sistema.
Mail: dirección de correo electrónico.
Nombre: nombre del cliente
Estado: estado del usuario (habilitado o inhabilitado).

La figura siguiente muestra la clase con sus atributos y sus responsabilidades.

4.3.3 Identificar las asociaciones

Los mensajes que se envían entre objetos son los medios que utilizan para comunicarse. Estos mensajes se ven reflejados en los diagramas de interacción (secuencia y colaboración). Esta comunicación entre los objetos permite la realización del caso de uso.

Criterios para tener en cuenta para encontrar asociaciones son los siguientes:

- La cantidad de asociaciones entre clases de análisis debe ser mínima. Por lo tanto sólo se deben asociar las clases cuando estas tenga objetos que se comunican entre sí, y evitando de establecer asociaciones cuando se cuando existe la posibilidad que "podrían ser necesarias" más adelante.
- No dedicar tiempo a la optimización de todas las asociaciones.

Una asociación especial es la agregación. Se utiliza en las siguientes situaciones:

- Cuando un concepto que contienen a otros conceptos, como por ejemplo: un ómnibus que contiene a sus pasajeros.
- Cuando un concepto está formado por otros conceptos, como por ejemplo: un ómnibus está compuesto por un motor, un chasis y ruedas.
- Cuando los conceptos que forman un conjunto de otros conceptos, como por ejemplo un grupo familiar que a su vez se compone por un padre, una madre y sus hijos.

Para documentar las asociaciones se utilizan diagramas de clases (de análisis primeramente y luego de diseño). En los diagramas de clases de análisis no es necesario hacerlos en forma detallada, pero si se puede describa el rol y la multiplicidad que tiene cada clase que participa en una asociación sería bueno que lo hagamos.

Ejemplo: en el diagrama de clases de la figura 82, se pueden apreciar asociaciones simples entre las clases de análisis participantes, y una agregación entre la clase Usuario y la clase Tarjeta de débito.

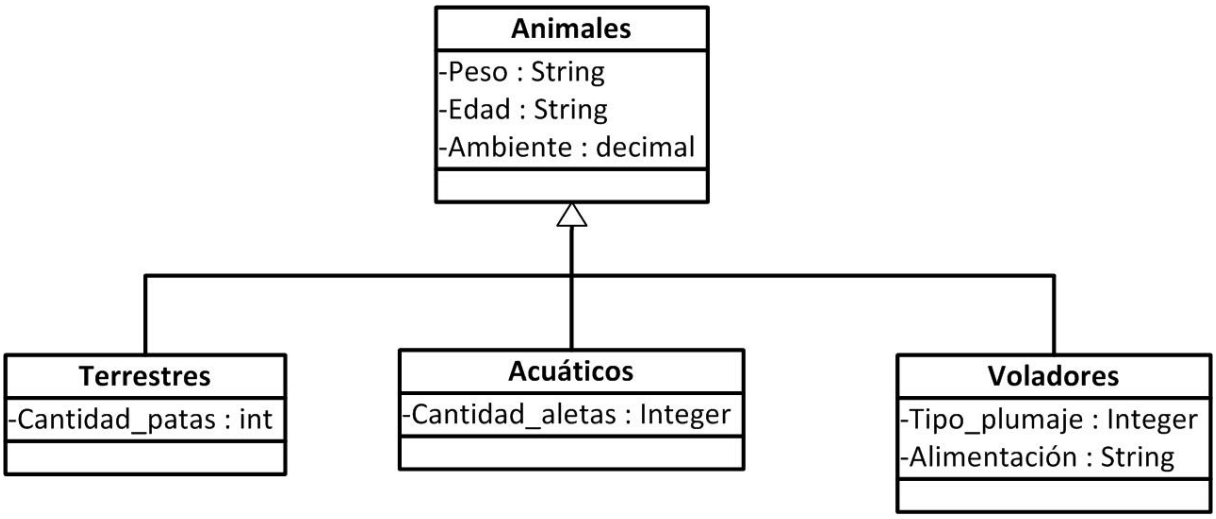
4.3.4 Identificación de generalizaciones (herencia)

Las herencias (generalizaciones) nos permiten extraer el comportamiento común entre clases de análisis.

La generalización se da cuando tenemos varias (dos o más) clases de análisis que comparten entre ellas comportamientos comunes. Si esto ocurre se debe asignar a una nueva clase de análisis ese comportamiento común, permitiendo que las dos primeras sean especializaciones (clases derivadas) de la nueva clase.

Cuando una clase es especialización de otra, hereda el comportamiento y los atributos de la clase madre o superclase.

Ejemplo: la figura siguiente se muestra una clase madre Animales que tiene como propiedades el tener un peso, una edad y un ambiente donde viven. Las clases terrestres, acuáticos y voladores son refinamientos de la misma, o sea, también son animales pero pueden agregar nuevos atributos y comportamientos. Por ejemplo, el terrestre agrega el atributo Cantidad_patas.

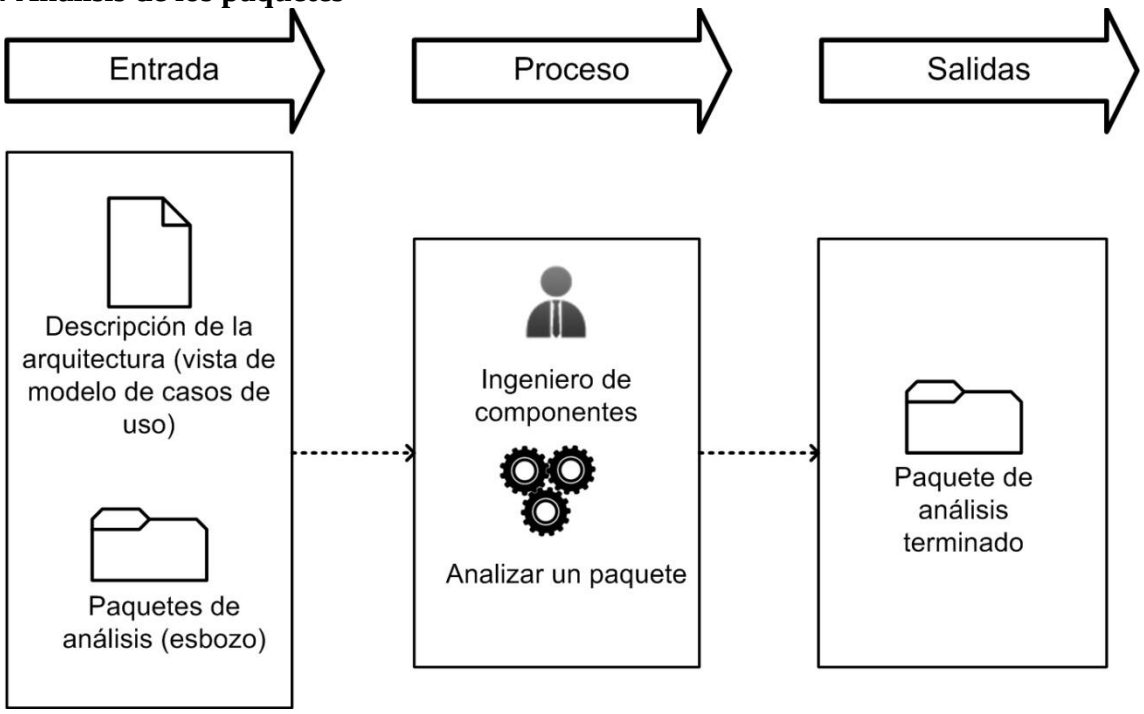


4.3.5 Identificación de requerimientos especiales

Como ya se mencionara en capítulos anteriores, existen requerimientos funcionales y no funcionales (especiales), y es posible que durante el análisis de las clases se descubran nuevos requerimientos no funcionales (o especiales), que es tarea del diseño e implementación atenderlos, pero si aparecen en el análisis los debemos registrar. Por lo tanto estos requerimientos deben ser registrados y documentados.

La forma de documentar puede ser a través de etiquetas, en forma de texto luego de cada diagrama o en un documento donde están todos los requerimientos no funcionales haciendo referencia a que clases corresponden.

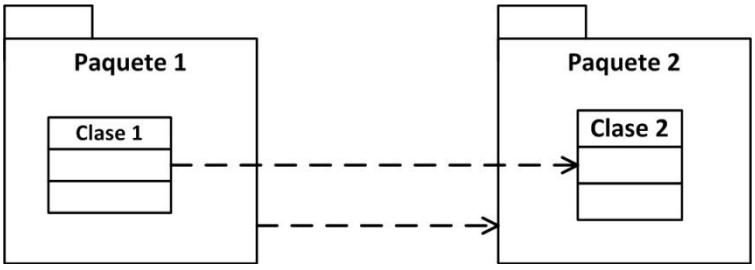
4.4 Análisis de los paquetes



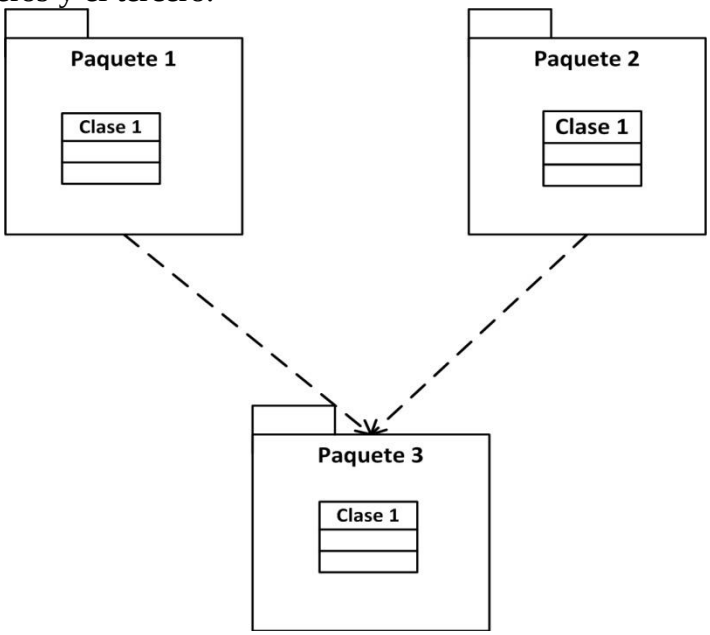
Entradas	Trabajador / Proceso	Salidas
Descripción de la arquitectura (vista del modelo de casos de uso)	Ingeniero de componentes: Analizar un paquete	Paquete de análisis terminado
Paquetes de análisis (esbozo)		

Al finalizar el análisis es momento de realizar la actividad de refinar la estructura de paquetes que se había identificado al inicio del análisis, y de ser necesario ampliarla. Esto implica las siguientes actividades:

- Se debe definir cuáles son las dependencias que existen entre los paquetes con el objetivo de facilitar los cambios futuros en el sistema.
  Cuando hay dos paquetes que contienen clases de análisis que están asociadas entre sí, ambos paquetes tienen que tener una relación de dependencia entre sí. Por ejemplo, si la clase 1 del paquete 1 está asociada con la clase 2 del paquete 2, existe una dependencia entre los paquetes 1 y 2.
  La dependencia tendrá un sentido de navegabilidad que estará dado por la navegabilidad de la asociación entre las clases. Esto se muestra en la figura siguiente:

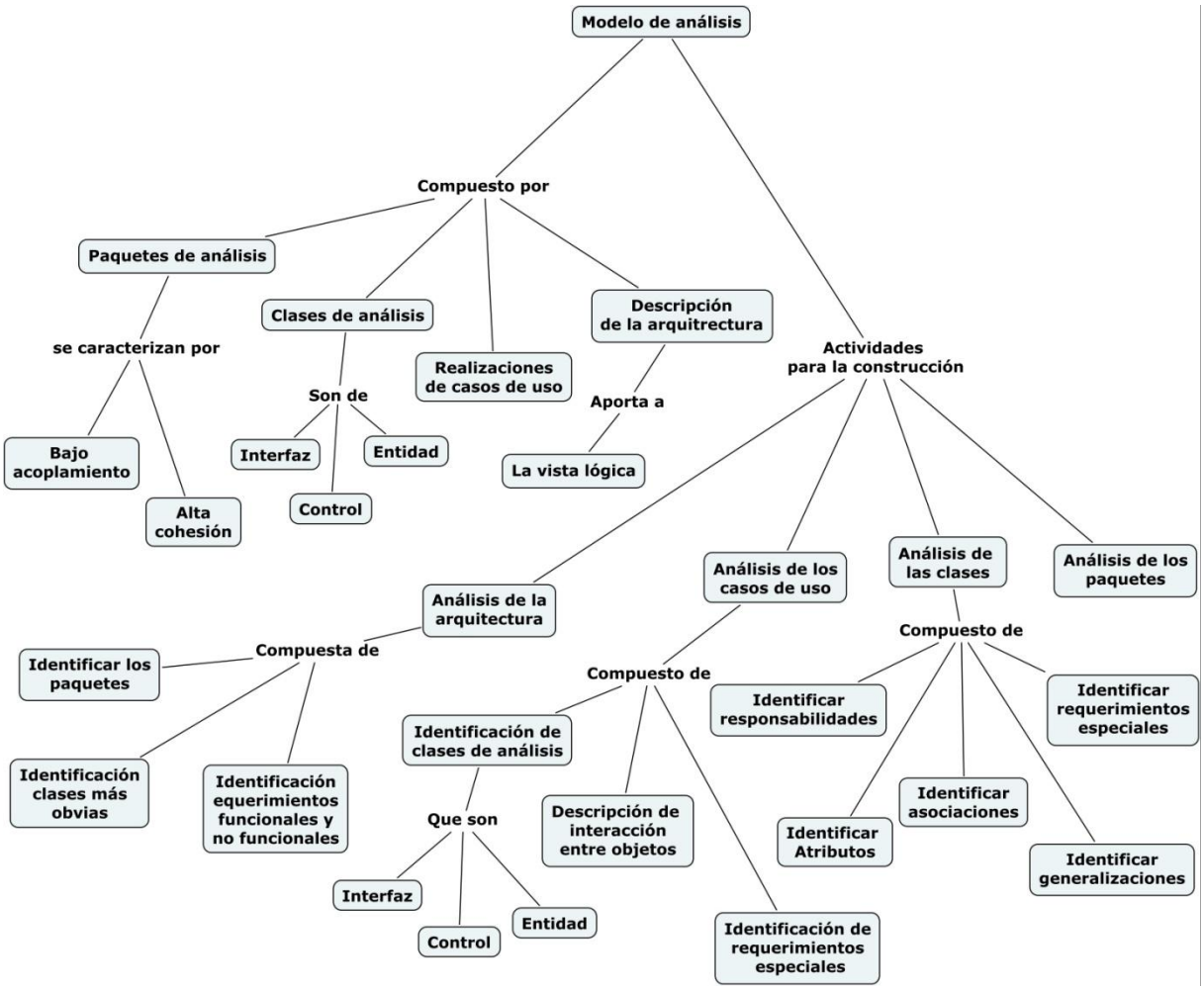


- Es importante que cada paquete de análisis sea tan independiente de los demás como sea posible (bajo acoplamiento entre paquetes).
  Esto se logra tratando de minimizar las dependencias entre paquetes, para ello posiblemente deberemos reubicar clases de análisis.
  Cuando se realiza esta revisión al finalizar el análisis es normal que se encuentren clases que pertenecen a más de un paquete de análisis. Si esto ocurre, se debe extraer esas clases repetidas de ambos paquetes, crear un nuevo paquete y reubicar las clases en este tercer paquete, y establecer una relación de dependencia entre los dos primeros y el tercero.



- Finalmente asegurarse de que las clases de cada paquete contenga clases de análisis fuertemente relacionadas funcionalmente (alta cohesividad).

1. **Resumen a través de un mapa conceptual**


Modelo de análisis
Compuesto por
Paquetes de análisis
se caracterizan por
Bajo acoplamiento
Alta cohesión
Clases de análisis
Son de
Interfaz
Entidad
Control
Realizaciones de casos de uso
Descripción de la arquitrectura
Aporta a
La vista lógica
Actividades para la construcción
Análisis de la arquitectura
Compuesta de
Identificar los paquetes
Identificación clases más obvias
Identificación equerimientos funcionales y no funcionales
Análisis de los casos de uso
Compuesto de
Identificación de clases de análisis
Que son
Interfaz
Control
Entidad
Descripción de interacción entre objetos
Identificación de requerimientos especiales
Análisis de las clases
Compuesto de
Identificar responsabilidades
Identificar Atributos
Identificar asociaciones
Identificar generalizaciones
Identificar requerimientos especiales
Análisis de los paquetes