



INGENIERÍA DEL SOFTWARE I

Práctica 6, Sesión 2

Modelado Arquitectura Diseño

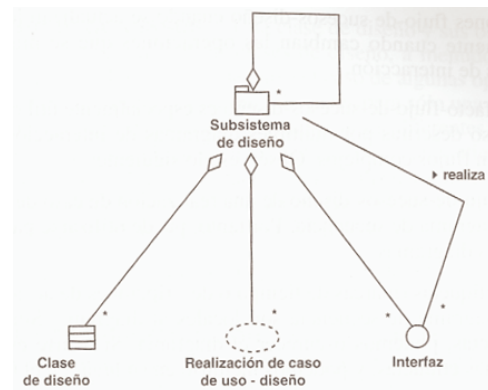
Univ. Cantabria – Fac. de Ciencias

María Sierra



Subsistema de Diseño

- **Organiza** los artefactos del modelo de diseño en piezas más manejables
- Cohesivos y débilmente acoplados
- **Representan:**
 - Separación de aspectos del diseño
 - Componentes de grano grueso en la implementación del sistema
 - Interfaces compuestas a partir de otros componentes de grano fino (ej los que especifican clases de implementación individuales, y que se convierten ellos mismos en ejecutables, ficheros binarios o entidades similares que pueden distribuirse en diferentes nodos)
 - Productos SW usados que han sido encapsulados en ellos (integración de productos SW utilizados)
 - Sistemas heredados (o parte de ellos) encapsulándolos



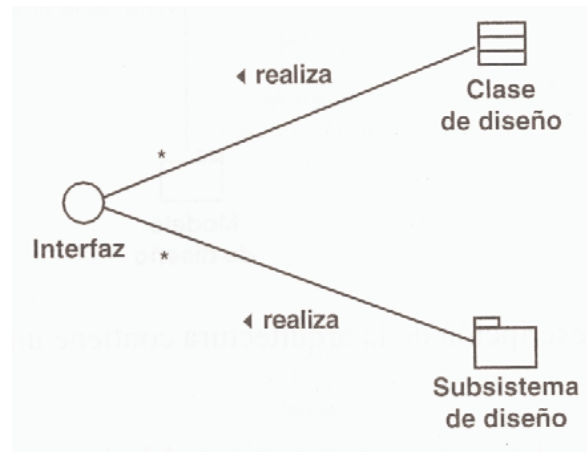
Consta de: clases del diseño, realizaciones de casos de uso, interfaces y otros subsistemas (recursivamente)

Puede proporcionar interfaces que representan la funcionalidad que exporta en términos de operaciones



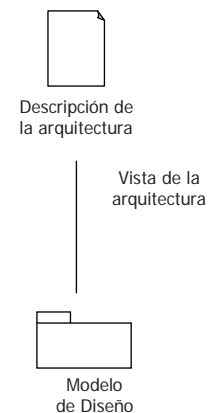
Interfaz

- Especifican las **operaciones** que **proporcionan** las **clases** y los **subsistemas** del diseño
- Una clase de diseño que proporcione un interfaz debe proporcionar también métodos que realicen las operaciones del interfaz.
- Un subsistema que proporcione un interfaz debe contener también clases de diseño u otros subsistemas (recursivamente) que proporcionen la interfaz
- Permite **separar** la **especificación** de la **funcionalidad** en términos de operaciones de sus implementaciones en términos de métodos. De esta forma la interfaz es independiente de su implementación



Descripción de la Arquitectura *(vista del modelo de diseño)*

- **Descomposición** del modelo de diseño en **subsistemas**, sus **interfaces** y las **dependencias** entre ellos
- **Clases de diseño** fundamentales:
 - Las que poseen una traza con las de análisis (clases activas)
 - Las generales y centrales (representan mecanismos de diseño genéricos y que tengan muchas relaciones con otras clases del diseño)
 - Basta con considerar las significativas para la arquitectura, clases abstractas, y no a sus subclasses, a menos que estas representen algún comportamiento interesante y significativo para la arquitectura diferente al de la clase abstracta
- **Realizaciones de casos de uso-diseño** que describan alguna funcionalidad importante y crítica que debe desarrollarse pronto dentro del ciclo de vida del SW:
 - Que impliquen muchas clases de diseño y por tanto tengan una cobertura amplia, posiblemente a lo largo de varios subsistemas, o que impliquen clases de diseño fundamentales





Modelo de Despliegue

- **Modelo de objetos** que describe la **distribución física del sistema** en términos de **cómo se distribuye la funcionalidad entre los nodos de cómputo**. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño
 - Cada **nodo** representa un **recurso de cómputo** (procesador o dispositivo HW similar)
 - Los **nodos** poseen **relaciones** que representan medios de **comunicación entre ellos** (Internet, Intranet, bus, etc...)
 - Puede **describir** diferentes **configuraciones** de red, incluidas configuraciones para pruebas y para simulación
 - La **funcionalidad** (los procesos) de un **nodo** se definen por los componentes que se distribuyen sobre ese nodo
 - El **modelo de despliegue** en sí mismo representa una **correspondencia** entre la **arquitectura SW** y la **arquitectura del sistema HW**



Descripción de la Arquitectura *(vista del modelo de despliegue)*

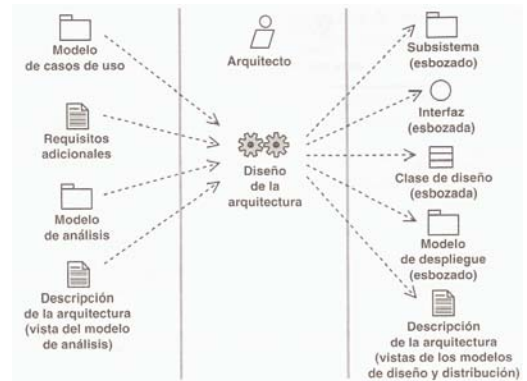
- En la vista arquitectónica, deberían mostrarse todos los aspectos del modelo de despliegue incluyendo la correspondencia de los componentes sobre los nodos tal como se identificó durante la implementación





Diseño de la Arquitectura

- **Objetivo** esbozar los modelos de diseño y despliegue y su arquitectura mediante la identificación de:
 - Nodos y sus configuraciones de red
 - Subsistemas y sus interfaces
 - Clases de diseño significativas para la arquitectura, como las activas
 - Mecanismos de diseño genéricos que tratan requisitos comunes, como los requisitos especiales sobre persistencia, distribución, rendimiento y demás, tal y como se capturaron durante el análisis sobre las clases y las realizaciones de caso de uso-análisis
- Se consideran posibilidades de **reutilización** (partes de sistemas parecidos, productos SW generales)
- Los subsistemas, interfaces u otros elementos de diseño se añadirán posteriormente



Diseño de la Arquitectura

- **Actividad 1: Identificación de nodos y configuraciones de red**
- **Actividad 2: Identificación de subsistemas y de sus interfaces**
- **Actividad 3: Identificación de clases de diseño relevantes para la arquitectura**
- **Actividad 4: Identificación de mecanismos genéricos de diseño**



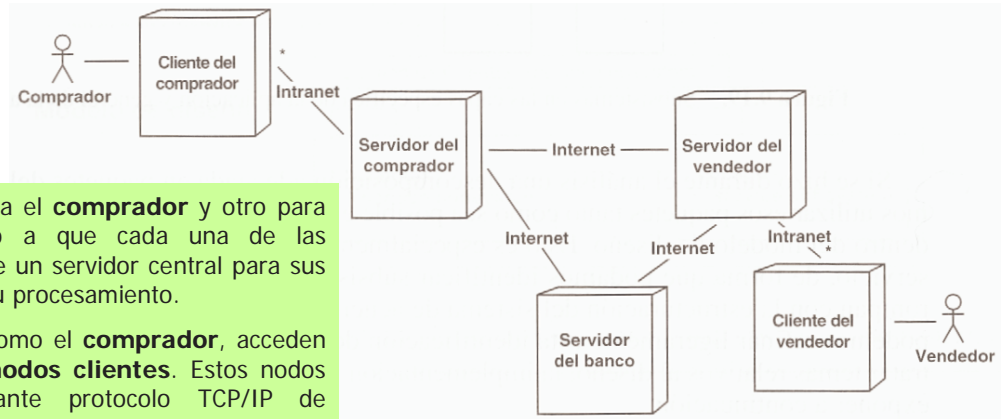
Diseño de la Arquitectura

- **Actividad 1: Identificación de nodos y configuraciones de red**
 - Las configuraciones físicas de la red tienen gran influencia sobre la arquitectura del SW, incluyendo las clases activas que se necesitan y la distribución de la funcionalidad entre los nodos de la red
 - Normalmente se emplea un **patrón a tres capas**: una para los **clientes** (interacciones de los usuarios), otra para la **funcionalidad de la B.D** y otra para la **lógica del negocio o de aplicación**. El patrón **cliente/servidor** es un caso especial de este patrón a tres capas (la lógica del negocio o aplicación se ubica en otra de las dos capas)
 - **Preguntarse:**
 - Nodos y su capacidad (potencia de proceso y tamaño de memoria)
 - Tipo de conexiones entre nodos, y protocolos de comunicación entre ellas
 - Características de conexiones y protocolos (ancho de banda, disponibilidad,...)
 - Se necesita capacidad de proceso redundante, modos de fallo, migración de procesos, mantenimiento de copias de seguridad de datos, etc...
 - Cada configuración de red (incluidas las de prueba y simulación) deben mostrarse en un diagrama de despliegue separado



Diseño de la Arquitectura

- **Actividad 1, Ejemplo: Identificación de nodos y configuraciones de red**
 - Ej: El sistema se ejecutará sobre tres nodos servidores y un cierto número de nodos cliente



Un nodo **servidor** para el **comprador** y otro para el **vendedor**, debido a que cada una de las organizaciones requiere un servidor central para sus objetos de negocio y su procesamiento.

Los usuarios finales, como el **comprador**, acceden al sistema mediante **nodos clientes**. Estos nodos se comunican mediante protocolo TCP/IP de Internet a Intranet.

Tendremos también un tercer servidor para el Banco, en él se producen los verdaderos pagos de facturas (las transferencias entre cuentas)



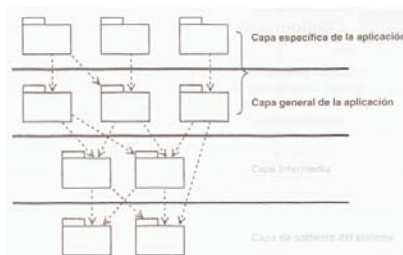
Diseño de la Arquitectura

• **Actividad 2: Identificación de subsistemas y sus interfaces**

- Permiten organizar el modelo de diseño en piezas manejables. Se identifican inicialmente como una manera de dividir el trabajo de diseño, o se pueden encontrar a medida que el modelo de diseño evoluciona y crece hacia una estructura que debe ser descompuesta
- No todos los subsistemas se implementan ya que algunos representan productos reutilizados y otros son recursos existentes en la empresa. Su inclusión e el modelo de diseño permite analizar y evaluar las alternativas de reutilización

■ **Identificación de subsistemas de aplicación**

- Se identifican a partir de los paquetes de análisis, pero puede ser necesario un refinamiento para tratar temas relativos al diseño, implementación y distribución del sistema:



- Una parte de un anterior paquete de análisis, compartida y utilizada por varios subsistemas
- Algunas partes de un anterior paquete de análisis se realizan por productos SW reutilizados (asignables a capas intermedias o subsistemas de SW del sistema)
- Los anteriores paquetes del análisis no representan una división de trabajo adecuada o la incorporación de un sistema heredado
- Los anteriores paquetes del análisis no están separados para una distribución directa sobre nodos. Puede que la descomposición de subsistemas tenga que tratar, en este caso, los aspectos de distribución de este tipo descomponiéndolos en subsistemas más pequeños de forma que cada uno pueda asignarse a un nodo determinado; después se han de refinar estos subsistemas más pequeños para minimizar el tráfico de la red

María Sierra - IS1

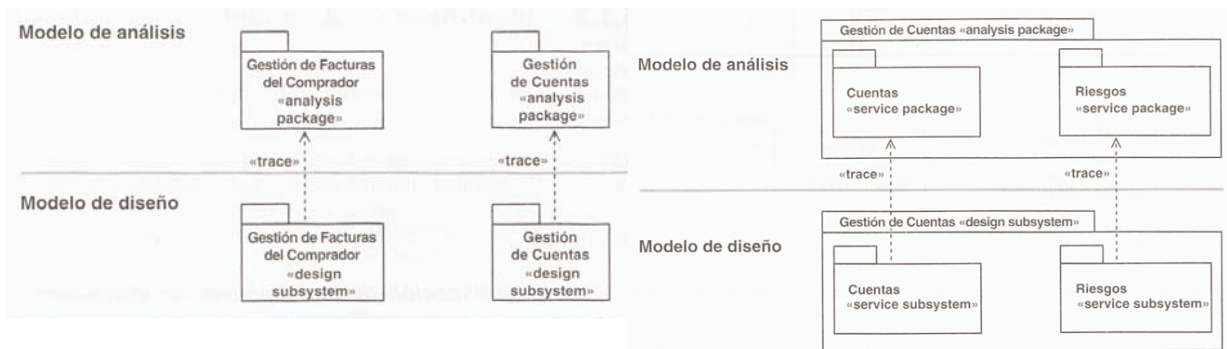
P6b.11



Diseño de la Arquitectura

• **Actividad 2, Ejemplo: Identificación de subsistemas diseño a partir de paquetes de análisis**

- Ej: Los paquetes "Gestión de Facturas de Comprador" y "Gestión de Cuentas" del modelo de análisis se utilizan para identificar los correspondientes subsistemas del modelo de diseño. Además los paquetes de servicio de "Cuentas" y "Riesgos" dentro del Paquete "Gestión de Cuentas" del modelo de análisis se utilizan para identificar los correspondientes subsistemas de servicio del modelo de diseño



María Sierra - IS1

P6b.12



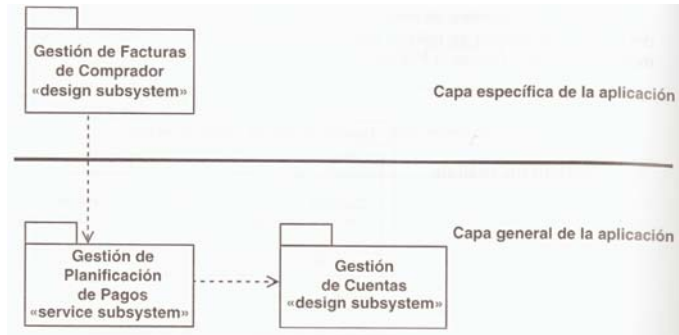
Diseño de la Arquitectura

- **Actividad 2, Ejemplo: Refinamiento de los subsistemas para tratar funcionalidades compartidas**

- Ej: Se considero implementar todos los paquetes de servicio para el pago de facturas en el subsistema "Gestión de Facturas de Comprador", donde parecían encajar tras el análisis de los casos de uso relacionados con la gestión de facturas

Los desarrolladores se dieron cuenta de que podían beneficiarse de la existencia de un subsistema de servicio de pagos general, por lo que **agruparon la funcionalidad de los pagos en un subsistema de servicio independiente llamado "Gestión de Planificación de Pagos"**

El subsistema "Gestión de Facturas de Comprador" utiliza la funcionalidad de este nuevo subsistema para planificar facturas. Cuando el pago planificado se hace efectivo, el subsistema "Gestión de Planificación de Pagos" utiliza el subsistema "Gestión de Cuentas" para la transferencia real del dinero de una cuenta a otra

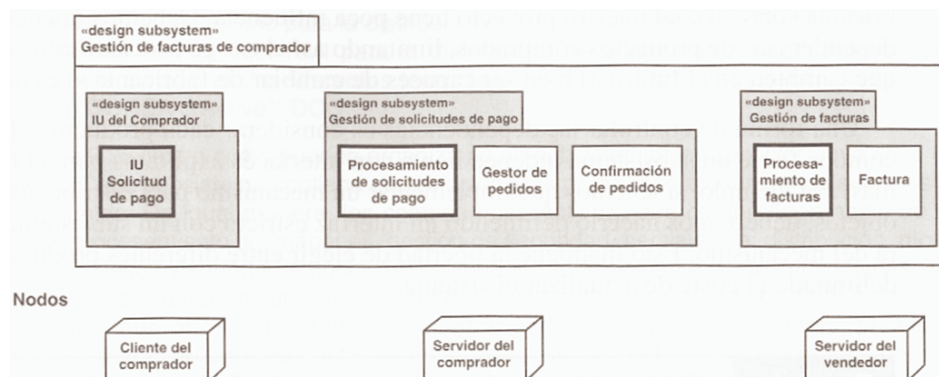


Diseño de la Arquitectura

- **Actividad 2, Ejemplo: Distribución de un subsistema entre los nodos**

- Ej: El subsistema "Gestión de Facturas de Comprador" debe distribuirse en varios nodos diferentes

Se descompone el subsistema en **tres subsistemas**: "IU del Comprador", "Gestión de Solicitudes de Pago" y "Gestión de Facturas". Los componentes generados a partir de cada uno de estos tres subsistemas más pequeños pueden ser distribuidos respectivamente en los **nodos** "Cliente del Comprador", "Servidor del Comprador" y "Servidor del Vendedor"

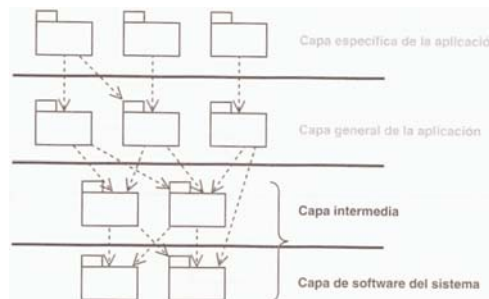




Diseño de la Arquitectura

Identificación de subsistemas intermedios y de Software del Sistema

- El SW del sistema y la capa intermedia constituyen los cimientos de un sistema, ya que toda la funcionalidad descansa sobre SW como Sistemas Operativos, Sistemas de Gestión de Bases de Datos, SW de comunicaciones, tecnologías de distribución de objetos, kits de diseño de IGU, y tecnologías de gestión de transacciones. La selección e integración de productos SW que se compran o se construyen son dos de los objetivos fundamentales durante las fases de inicio y elaboración.
- Precaución:** mantener una adecuada libertad de acción y evitar ser dependiente de un determinado producto o fabricante, se debe tratar de limitar las dependencias de productos comprados, limitando así el riesgo asociado a su uso (cambios futuros) y permitiéndonos cambiar de fabricante en cualquier momento.
- Se controlan las dependencias haciendo que cada producto SW comprado sea un subsistema independiente con interfaces explícitos para el resto de los sistemas.



Diseño de la Arquitectura

Actividad 2, Ejemplo: Utilización de Java para construir la capa intermedia

- Ej: Varias de las implementaciones del sistema ejemplo se deben ejecutar sobre diferentes tipos de máquinas, PC personales, estaciones de trabajo UNIX, por lo que deben de ser capaces de interoperar entre diferentes plataformas

Se implementa esta interoperación mediante middleware, concretamente con los paquetes Java AWT (Abstract Windowing Toolkit), RMI (Remote Method Invocation) y Applet. Estos paquetes de Java se representan como subsistemas que se ejecutarán sobre la máquina virtual de Java que debe ser instalada. Se utilizará un navegador de Internet para cargar páginas web que incorporan applets

A bajo nivel el sistema se construirá sobre SW del sistema, como el protocolo TCP/IP para la comunicación en Internet



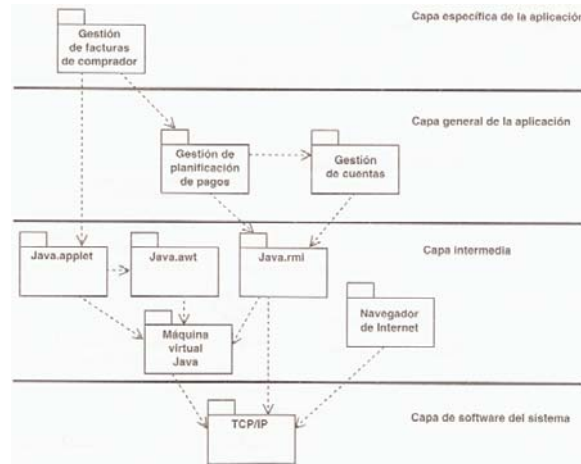
Cada subsistema contiene clases diseñadas para proporcionar ciertos servicios cuando se utilizan juntas. Los componentes de ActiveX (hojas de cálculo, multimedia, procesamiento de imágenes, gestión de seguridad, persistencia, distribución, motores de inferencia, procesos y soporte de concurrencia,...) pueden representarse como subsistemas. También suele ser práctico crear subsistemas para los tipos de datos estándar o las clases fundamentales que otros subsistemas pueden importar y utilizar (ej java.lang proporciona muchas clases fundamentales de Java como boolean, integer y float)



Diseño de la Arquitectura

- **Definición de dependencias entre subsistemas**
 - Definir dependencias entre subsistemas si sus contenidos tienen relación unos con otros. Dirección de la dependencia (navegabilidad) debe ser la misma que la de la relación.
 - Si consideramos las dependencias entre subsistemas antes de saber los contenidos, se considerarán las dependencias de los paquetes de análisis que se corresponden con las de diseño. Si además se utilizan interfaces entre subsistemas, las dependencias deberían ir hacia las interfaces, no hacia los propios subsistemas
- **Actividad 2, Ejemplo: Dependencias y Capas iniciales del Sistema ejemplo**

Las dependencias, sobre todo las de las capas intermedias, pueden ser más o menos implícitas en el entorno de implementación (Java en este caso), pero si **tienen impacto en la arquitectura del sistema**, especialmente cuando **atravesan capas**, es útil hacerlas explícitas en el modelo de diseño y mostrarlas en diagramas de clases



María Sierra - IS1

P6b.17



Diseño de la Arquitectura

- **Identificación de interfaces entre subsistemas**
 - Las interfaces proporcionadas por un subsistema **definen operaciones** que son **accesibles "desde fuera" del subsistema**. Estas interfaces las proporciona o bien clases o bien otros subsistemas
 - Para esbozar los interfaces sin conocer el contenido de los subsistemas se consideran las dependencias entre subsistemas identificadas en el paso anterior. Cuando un subsistema tiene una dependencia que apunta hacia él, es probable que deba proporcionar una interfaz. Además si existe un paquete de análisis que podamos obtener mediante una traza desde el subsistema, entonces cualquier clase del análisis referenciada desde el exterior del paquete puede implicar una interfaz candidata
- **Consideraciones:**
 - Las interfaces identificadas permiten refinar las dependencias entre los subsistemas (para que sean dependencias entre interfaces y no entre subsistemas)
 - En los interfaces de las capas inferiores (capas intermedias y de SW) la identificación de interfaces es más sencillo, los subsistemas de estas capas encapsulan productos SW que suelen tener interfaces predefinidas
 - No es suficiente con identificar las interfaces, es necesario identificar las operaciones que deben definirse sobre cada una de ellas (mediante el diseño de casos de uso en términos de subsistemas y sus interfaces)

María Sierra - IS1

P6b.18



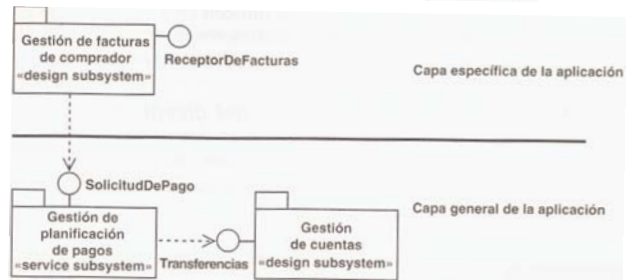
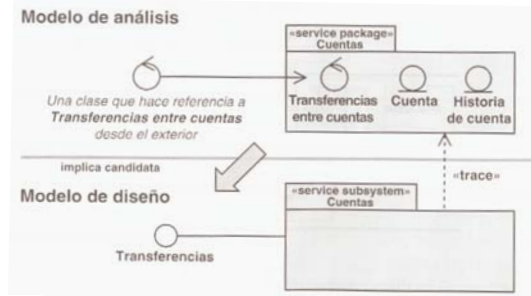
Diseño de la Arquitectura

• **Actividad 2, Ejemplo: Identificación de interfaces candidatas a partir del modelo de análisis**

- Ej: El paquete de servicio "Cuentas" contiene una clase del análisis llamada "Transferencias entre cuentas" a la cual se hace referencia desde fuera del paquete

Podemos por tanto identificar una interfaz inicial, llamada "Transferencias", proporcionada por el correspondiente subsistema de servicio "Cuentas" del modelo de diseño.

El subsistema "Gestión de Cuentas" proporciona la interfaz "Transferencias" para transferir dinero entre cuentas. Esta es la misma interfaz que proporciona el subsistema de servicio "Cuentas" dentro de "Gestión de Cuentas". El subsistema "Gestión de Planificación de Pagos" proporciona la interfaz "SolicitudDePago" que se utiliza para planificar los pagos. El subsistema "Gestión de Facturas de Comprador" proporciona la interfaz "ReceptorDeFacturas" para recibir nuevas facturas de un vendedor. Esta interfaz se utiliza en el caso de uso "Enviar Factura al Comprador", en el cual se manda una nueva factura al comprador



María Sierra - IS1

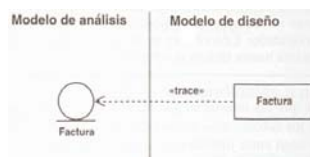
P6b.19



Diseño de la Arquitectura

• **Actividad 3: Identificación de clases de Diseño**

- Estudiar las clases de análisis que participan en la correspondiente realización de caso de uso-análisis. Identificar las clases de diseño que poseen una traza hacia esas clases
 - Ej: La clase de diseño "Factura" se esboza a partir de la clase entidad de análisis "Factura"



- Estudiar requisitos especiales de la correspondiente realización de caso de uso-análisis. Identificar las clases de diseño que los realizan.
- Identificar las clases de diseño necesarias que faltan
- Las clases de diseño se recogen en un diagrama de clases asociado con la realización y que muestra las relaciones que se utilizan en la realización del caso de uso
- Nota: esbozo de las más significativas

María Sierra - IS1

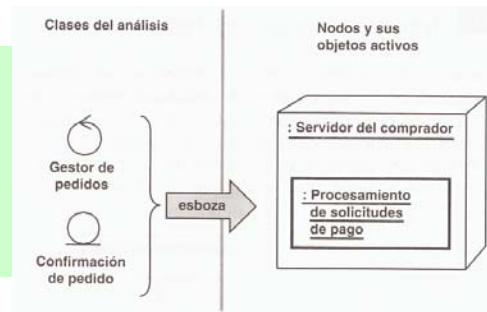
P6b.20



Diseño de la Arquitectura

- **Identificación de clases activas:** aquellas necesarias en el sistema considerando los requisitos de concurrencia del mismo:
 - Requisitos de rendimiento, tiempo de respuesta y disponibilidad que tienen los diferentes actores en su interacción con el sistema.
 - Pej: un actor tiene requisitos exigentes en cuanto a tiempo de respuesta podríamos dedicar un objeto activo a la gestión de ese actor, que tome sus entradas y le envíe las respuestas (un objeto que no se pare debido a que otros objetos tienen una carga alta)
 - La distribución del sistema sobre nodos. Los objetos activos deben soportar la distribución sobre varios nodos diferentes, que pueden requerir, por ejemplo, al menos un objeto activo por nodo y objetos activos aparte para gestionar la intercomunicación entre nodos
 - Otros requisitos: sobre el arranque y terminación del sistema, progresión, evitación de interbloqueo, evitación de inanición, reconfiguración de nodos y capacidad de los nodos

Ejemplo: El sistema debe ser distribuido en nodos: cliente del comprador, servidor del comprador, servidor del banco y demás. Clases de Análisis identificadas: IU Solicitud de Pago, Gestor de Pedidos, Confirmación de Pedidos, Factura, etc. Un comprador está interesado en la funcionalidad de las clases de análisis Gestor de Pedidos, Confirmación de Pedidos, pero esta funcionalidad requiere más potencia de cálculo que la que tiene el nodo cliente del comprador. Por tanto las partes principales de esas dos clases de análisis deben ubicarse en el Servidor del comprador, y deben ser gestionadas por una clase activa separada (Procesamiento de solicitudes de pago) en ese nodo.



María Sierra - IS1

P6b.21



Diseño de la Arquitectura

- **Actividad 4: Identificación de Mecanismos Genéricos de Diseño**
 - Se estudian **requisitos comunes** (como los especiales identificados en análisis) y decidimos **como tratarlos**, teniendo en cuenta las tecnologías de diseño e implementación disponibles. El **resultado** es un **conjunto de mecanismos genéricos de diseño** que pueden manifestarse como clases, colaboraciones o subsistemas. Los requisitos a tratar suelen estar relacionados con aspectos como:
 - Persistencia
 - Distribución transparente de objetos
 - Características de seguridad
 - Detección y recuperación de errores
 - Gestión de transacciones
 - A veces no se identifican a priori sino a medida que se exploran las realizaciones de los casos de uso y las clases de diseño

María Sierra - IS1

P6b.22



Diseño de la Arquitectura

• **Actividad 4, Ejemplo: Un mecanismo de diseño para la distribución transparente de objetos**

- Ej: Algunos objetos como los objetos "Factura", deben ser accesibles desde varios nodos, por lo que deben diseñarse para un sistema distribuido

Se implementa esta distribución haciendo que cada clase distribuida sea una subclase de una clase abstracta de Java "java.rmi.UnicastRemoteObject" que soporta RMI (técnica empleada por Java para obtener una distribución transparente de los objetos). Objetos distribuidos de manera que el cliente no tiene conocimiento de dónde reside el objeto.



• **Actividad 4, Ejemplo: Mecanismos de diseño para la persistencia**

- Ej: Algunos objetos como los objetos "Pedido" y "Factura", deben ser persistentes

Emplear un sistema de gestión de BD orientado a Objetos, o bien uno relacional, o incluso ficheros binarios. La mejor opción depende de cómo debemos acceder y actualizar los objetos, y de la facilidad de implementación y evolución futura de cada una de las opciones

- Una BD relacional da mejor rendimiento para datos tabulados, mientras que una OO lo hace para estructuras de objetos complejas
- Los sistemas de gestión de BD relacionales son una tecnología más madura que las BDOO, pero migrar a un gestor de objetos de un sistema implementado anteriormente con una BD relacional puede ser muy costosa

Cualquier decisión ha de ser documentada con un mecanismo de diseño genérico

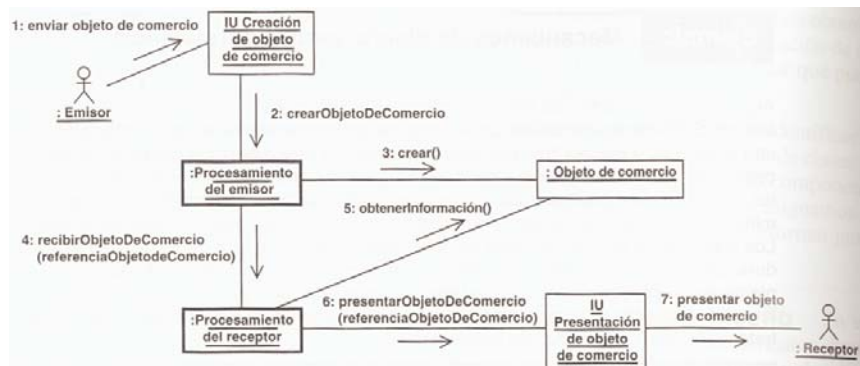


Diseño de la Arquitectura

• **Actividad 4, Ejemplo: Una colaboración genérica utilizada en varias realizaciones de casos de uso**

- Ej: Se identifica un patrón en el cual un actor crea un objeto de Comercio, como por ejemplo un "Pedido" o una "Factura", y lo envía a otro actor:
 - Cuando un comprador decide pedir ciertos bienes o servicios a un vendedor, el comprador invoca el caso de uso "Pedir Bienes o Servicios". El caso de uso permite al comprador el especificar y enviar electrónicamente un pedido al vendedor.
 - Cuando un vendedor decide enviar una factura a un comprador, invoca el caso de uso "Enviar Factura al Comprador", el cual envía electrónicamente una factura al comprador.

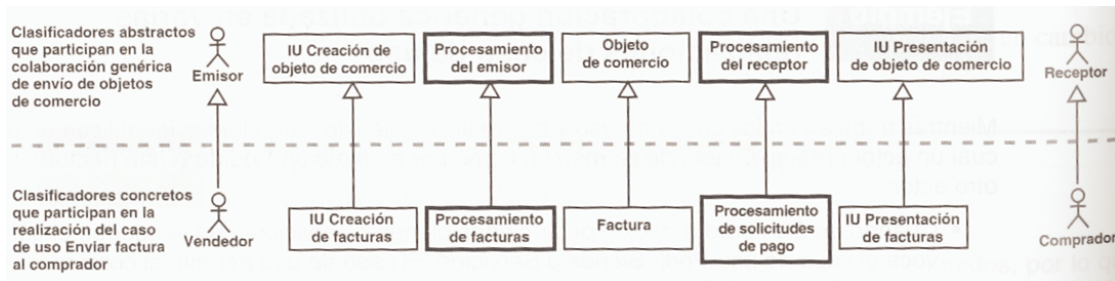
Comportamiento común que puede representarse por una colaboración genérica





Diseño de la Arquitectura

- **Actividad 4, Ejemplo: Una colaboración genérica utilizada en varias realizaciones de casos de uso**



Según el esquema descrito en el diagrama de colaboración, cuando se lleva a cabo el caso de uso "Pagar Factura al Comprador", por ejemplo, podemos hacer que ciertos clasificadores concretos sean subtipos de cada uno de los clasificadores abstractos que participan en la colaboración genérica

La realización del caso de uso "Pagar Factura al Comprador" sólo debe hacer referencia a la colaboración genérica, sin necesidad de duplicarla o especializarla, suponiendo que las operaciones (virtuales) de los clasificadores abstractos se realicen mediante métodos de los clasificadores concretos que son sus subtipos

Se puede emplear la misma técnica para realizar el caso de uso "Pedir Bienes o Servicios"

La **generalización no es la única forma de emplear una colaboración genérica**, ej: los patrones que son colaboraciones parametrizadas (con clases parametrizadas) también son genéricos y pueden utilizarse asociando clases concretas con los parámetros

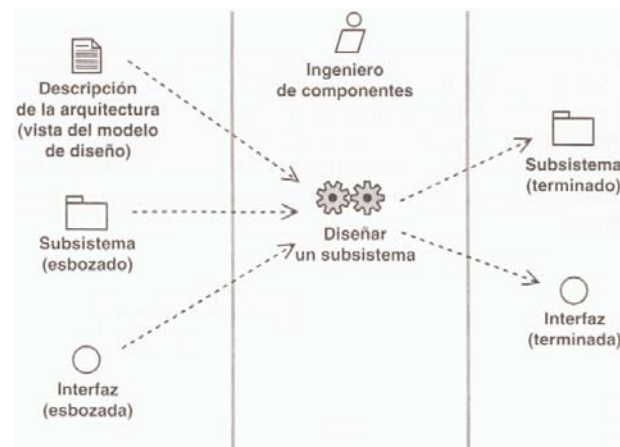
Los **mecanismos genéricos deberían ser identificados y diseñados en la fase de elaboración**, si se ha ce con cuidado estos mecanismos darán solución a los aspectos más difíciles del diseño



Diseño de un subsistema

- **Debe Garantizar que**

- Es tan independiente como sea posible de otros subsistemas y/o de sus interfaces
- Proporciona las interfaces correctos
- Cumple su propósito de ofrecer una realización correcta de las operaciones tal y como se definen en las interfaces que proporciona





Diseño de un subsistema

- **Actividad 1: Mantenimiento de las dependencias entre subsistemas**
 - Deben definirse y mantenerse dependencias en un subsistema respecto a otros. Si esos otros subsistemas proporcionan interfaces, las dependencias (de uso) deberían declararse hacia estas en lugar de hacia los propios subsistemas. Es mejor ser dependiente de un interfaz que de un subsistema
 - Minimizar las dependencias entre subsistemas y/o interfaces, tratando de reubicar las clases contenidas en un subsistema que sean demasiado dependientes de otros subsistemas
- **Actividad 2: Mantenimiento de interfaces proporcionadas por el subsistema**
 - Las operaciones definidas en los interfaces de los subsistemas deben soportar todos los roles que cumple el subsistema en las diferentes realizaciones de casos de uso. Pueden ser refinadas por el ingeniero de componentes a medida que evoluciona el modelo de diseño y se van diseñando los casos de uso (un subsistema y sus interfaces se pueden utilizar dentro de varias realizaciones de casos de uso)
- **Actividad 3: Mantenimiento de los contenidos de los subsistemas**
 - Un subsistema cumple sus objetivos cuando ofrece una realización correcta de las operaciones tal y como están descritas por las interfaces que proporciona. Pueden ser refinados por el ingeniero de componentes a medida que evoluciona el modelo de diseño:
 - Por cada interfaz que proporcione el subsistema, debería haber clases de diseño y otros subsistemas dentro de este que también proporcionen la interfaz
 - Para clarificar cómo el diseño interno de un subsistema realiza cualquiera de sus interfaces o casos de uso, podemos crear colaboraciones en términos de elementos contenidos en el subsistema



Diseño de la Arquitectura

- **Ejemplo: Una clase de diseño que proporciona una interfaz dentro de un subsistema**
 - Ej: El subsistema "Gestión de Facturas del Comprador" proporciona la interfaz "Factura"

El ingeniero de componentes responsable del subsistema decide hacer que la clase "Factura" realice esa interfaz

Una realización alternativa podría haber sido hacer que alguna otra clase del diseño realizase la interfaz, y que a su vez usase la clase Factura

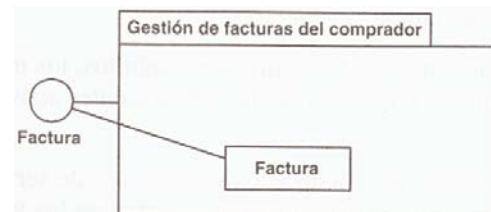




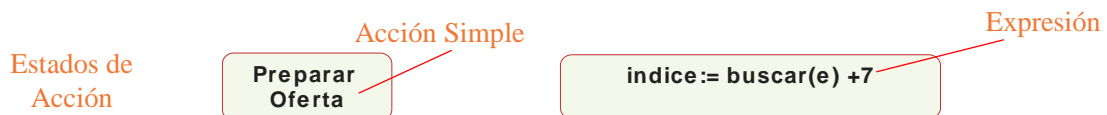
Diagrama de Actividad: **Introducción**

- Un diagrama de actividades se utilizan para **modelar los aspectos dinámicos** de un sistema
- Las **actividades** son similares a las acciones, pero tienen duración y se ejecutan dentro de un estado de un objeto
- Un diagrama de actividades muestra el flujo de actividades
- Una **actividad** es una ejecución no atómica en curso, dentro de una máquina de estado
- Las actividades producen acciones



Diagrama de Actividad: **Elementos**

- **Estados de Acción:** Unidades ejecutables y atómicas que representan la ejecución de acciones
 - No se pueden descomponer y aunque existan eventos no se interrumpirá la ejecución del estado de acción



- **Estados de Actividad:** Elementos compuestos de otros estados de actividad y de acción

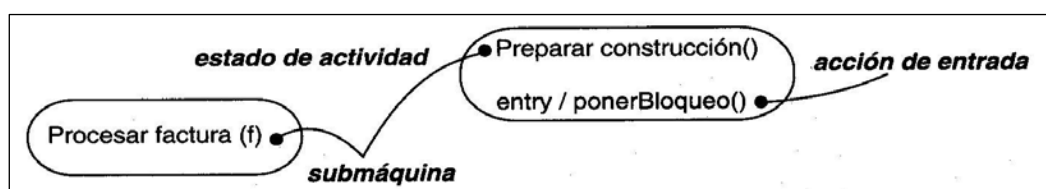
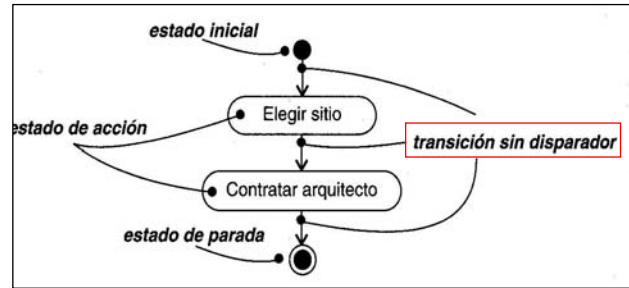




Diagrama de Actividad: Elementos

- Transiciones:** Paso de una actividad a otra

- Se representa mediante una flecha.
- Tiene un principio (punto negro) y un fin (círculo que rodea a un punto negro)



- Bifurcaciones:** Modelan alternancias en el camino controlada por una expresión booleana.

- Se representa con un rombo
- Puede tener una transición entrada y dos o más de salida

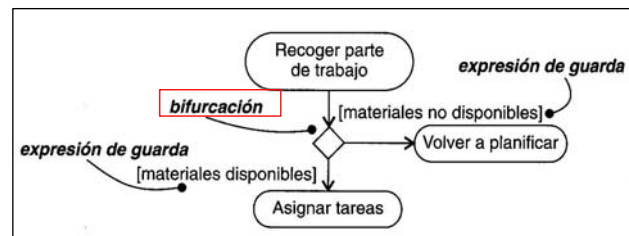
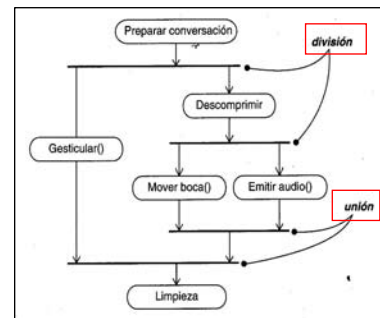


Diagrama de Actividad: Elementos

- Divisiones y Uniones:** Se realizan en el flujo de acciones para agrupar o no las mismas

- Una **División** puede tener una entrada y dos o más salida
- Una **Unión** puede tener dos o más entradas y una salida



- Calles (Swimlanes) :** Sirven para dividir el diagrama de actividad en grupos relacionados, donde cada uno representa la parte de la organización responsable de esas actividades

- Cada calle tiene un nombre único dentro del diagrama
- Cada responsable se agrupa en una calle o lane
- Cada calle puede llegar a ser representada por una o más clases
- Cada actividad pertenece a una única calle, pero las transiciones pueden cruzar las calles

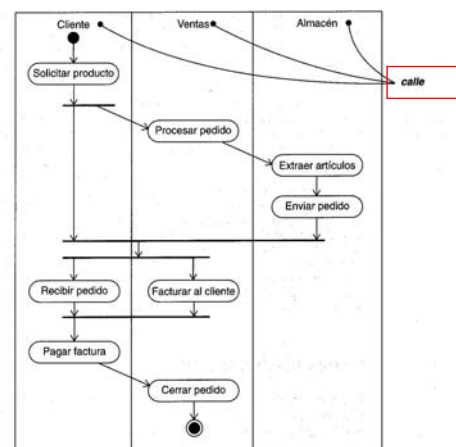
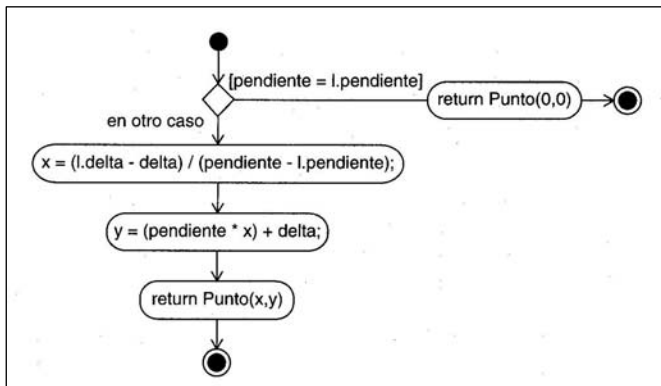




Diagrama de Actividad: Usos comunes

• Modelado de una Operación

- Las operaciones son los elementos a los que más se asocian los diagramas de actividades
- Los diagramas de actividades son en estos casos diagramas de flujo de las acciones de una operación.

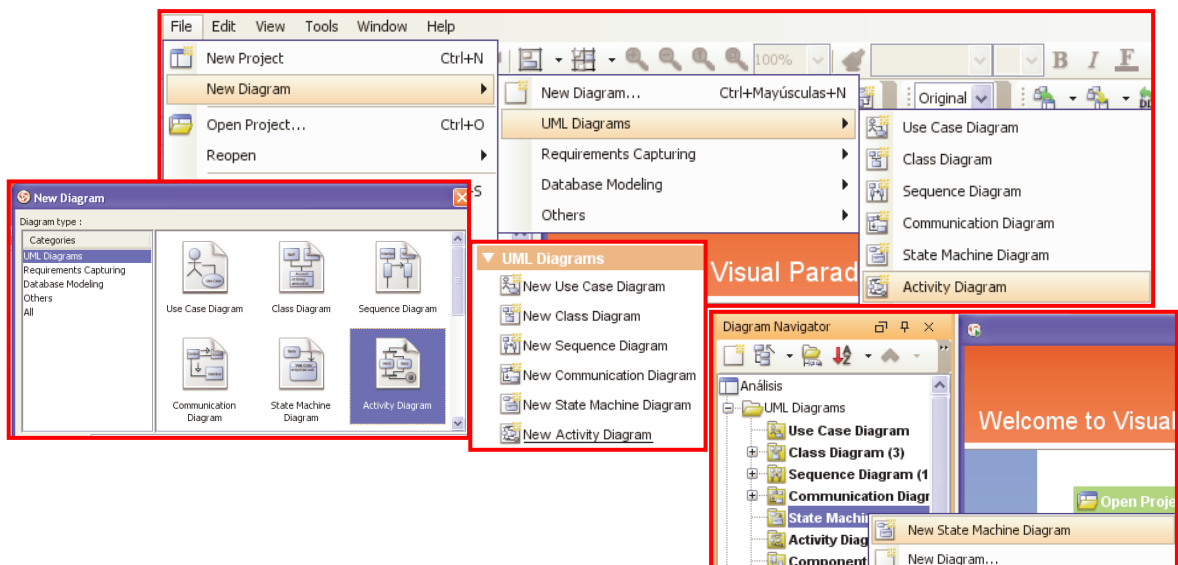


- Reunir los datos importantes de la operación (parámetros, tipos, retornos, atributos, etc.)
- Identificar precondiciones en el estado inicial y poscondiciones en el estado final. También las invariantes que se deben cumplir mientras se ejecuta la operación
- Especificar actividades y acciones, representándolas como estados de actividad o estados de acción.
- Usar bifurcaciones
- Usar divisiones y uniones para especificar flujos paralelos



Diagramas de Actividad con VP

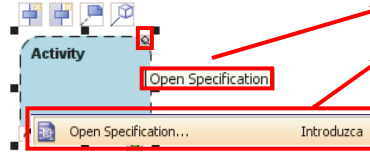
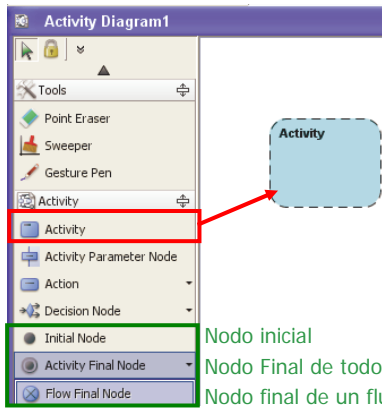
Crear Diagrama



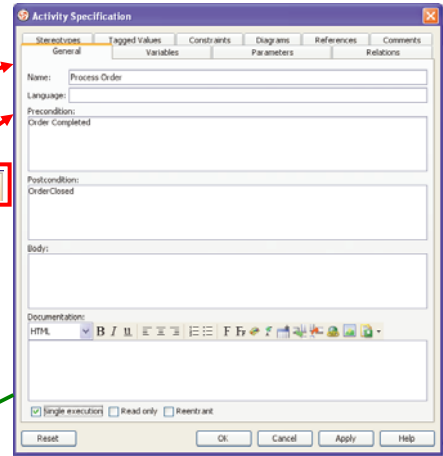


Diagramas de Actividad con VP: Elementos

Actividades

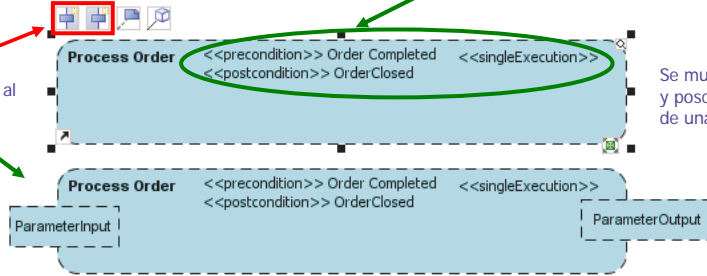


Especificación de Actividades



- Nodo inicial
- Nodo Final de todos los flujos de la actividad
- Nodo final de un flujo de la actividad

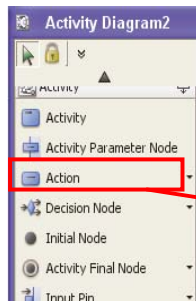
Parámetros de entrada / salida al nodo de Actividad



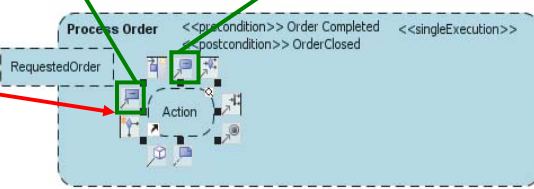
Se muestran las precondiciones y poscondiciones: restricciones de una acción



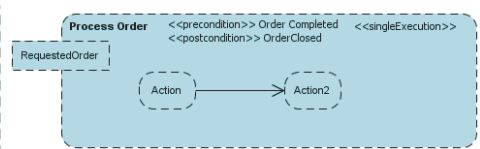
Diagramas de Actividad con VP: Elementos



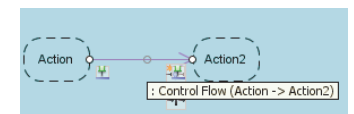
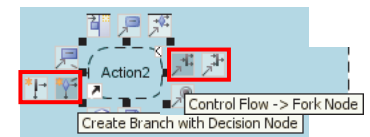
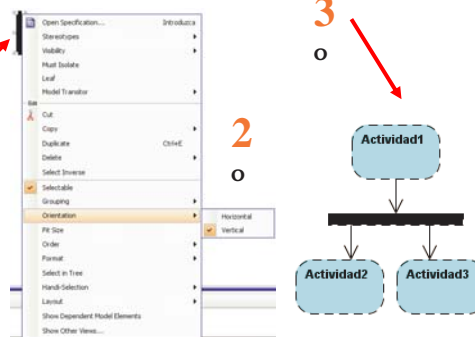
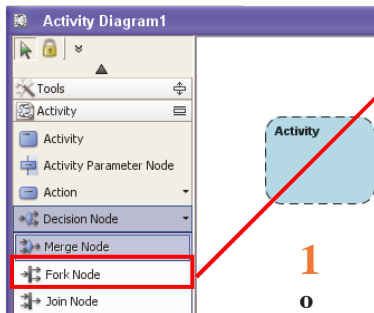
Flujo de Objeto Flujo de Control



Acciones

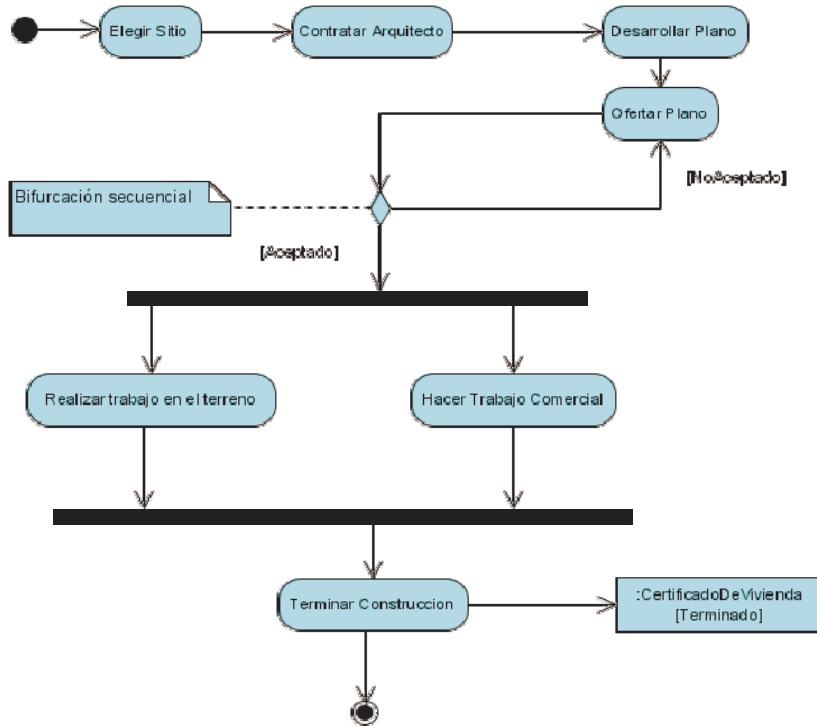


Nodos de Unión/Bifurcación

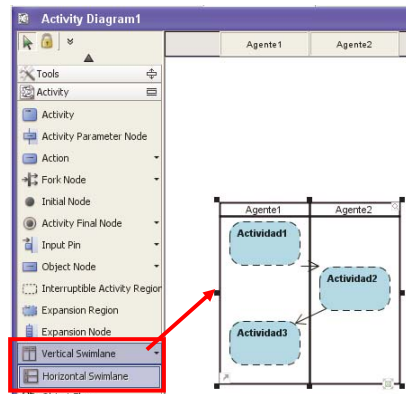




Diagramas de Actividad con VP: Elementos

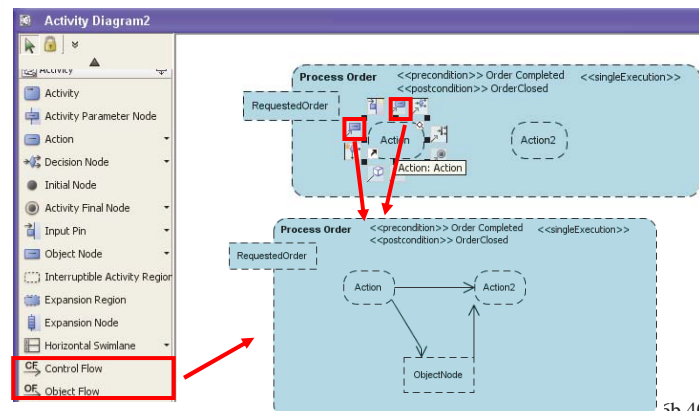


Diagramas de Actividad con VP: Elementos



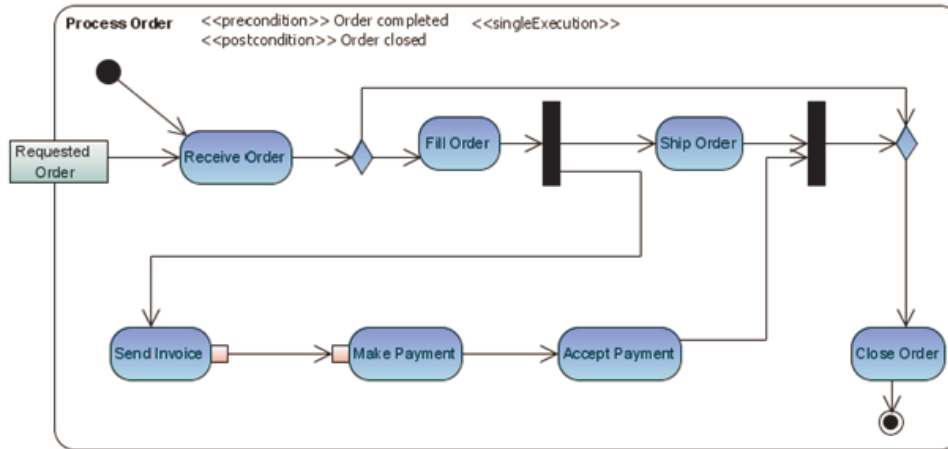
Swimlanes

Flujo de Control y de Objetos





Diagramas de Actividad con VP: Ejemplo



Diagramas de Actividad con VP: Ejemplo

