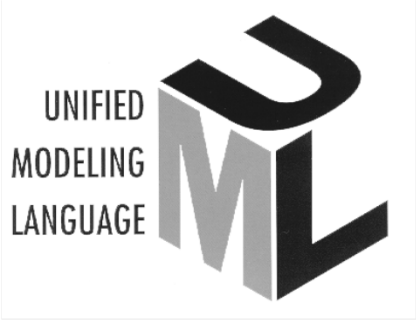
Introducción a UML



- Orientación a objetos
- Definir, organizar, visualizar
- Historia de UML
- Diagramas básicos UML
- Metodología de desarrollo

Orientación a Objetos

- Manera diferente de ver una aplicación
- Organizar la complejidad en microestructuras
- Componentes reutilizables
- Adaptabilidad a un entorno cambiante
- De la orientación a datos a las reglas de negocio
- Interdependencia
- Flexibilidad

Orientación a Objetos

Cambio de mentalidad

Mentalidad Procedural

- **¿Qué hace el sistema?**
- **¿Qué objetivos tiene ?**
- **¿Cómo diseño y codifico para conseguir los objetivos?**
- **Enfoque dirigido a los algoritmos**
- **Enfoque centrado en los datos**

Mentalidad O-O

- **¿Qué objetos configuran el sistema?**
- **¿Cual es la estructura y función de cada objeto?**
- **¿Cómo puedo precisar la dinámica del sistema a través del comportamiento o la interacción de sus objetos?**
- **Posponer las funciones algorítmicas**
- **Posponer el modelo de datos**

Orientación a Objetos

- **Encapsulación/ Encapsulamiento**
  - Empaquetamos dentro de un objeto una pieza de información con un comportamiento específico que actúa sobre esta información
  - Ventaja:
    - Limitamos los efectos de cambios sobre el sistema

Orientación a Objetos

- **Herencia**
  - Es un mecanismo que nos permite crear nuevos objetos basados en una progenie
  - Ventaja:
    - Facilidad de mantenimiento

Orientación a Objetos

- **Polimorfismo**
  - Capacidad de aplicar distintas implementaciones a una determinada funcionalidad
  - Ventaja:
    - Simplicidad y orden

Definir, Organizar, Visualizar

- **Lenguaje común**
  - Evitar la trampa del lenguaje
- **Modelo de referencia**
  - Evolución ordenada de los cambios
- **Trazabilidad**
  - Desde la funcionalidad al código
- **Reducción de costos**
  - Evitar los costos ocultos de mantenimiento

Agentes

- **Usuario**
  - Comprender que tipo de interacciones podrán realizar con el sistema
- **Analista**
  - Saber cuales son los objetos del sistema y como interactúan en distintos escenarios
- **Desarrollador**
  - Conocer la estructura y función de los objetos a implementar y qué recursos son necesarios

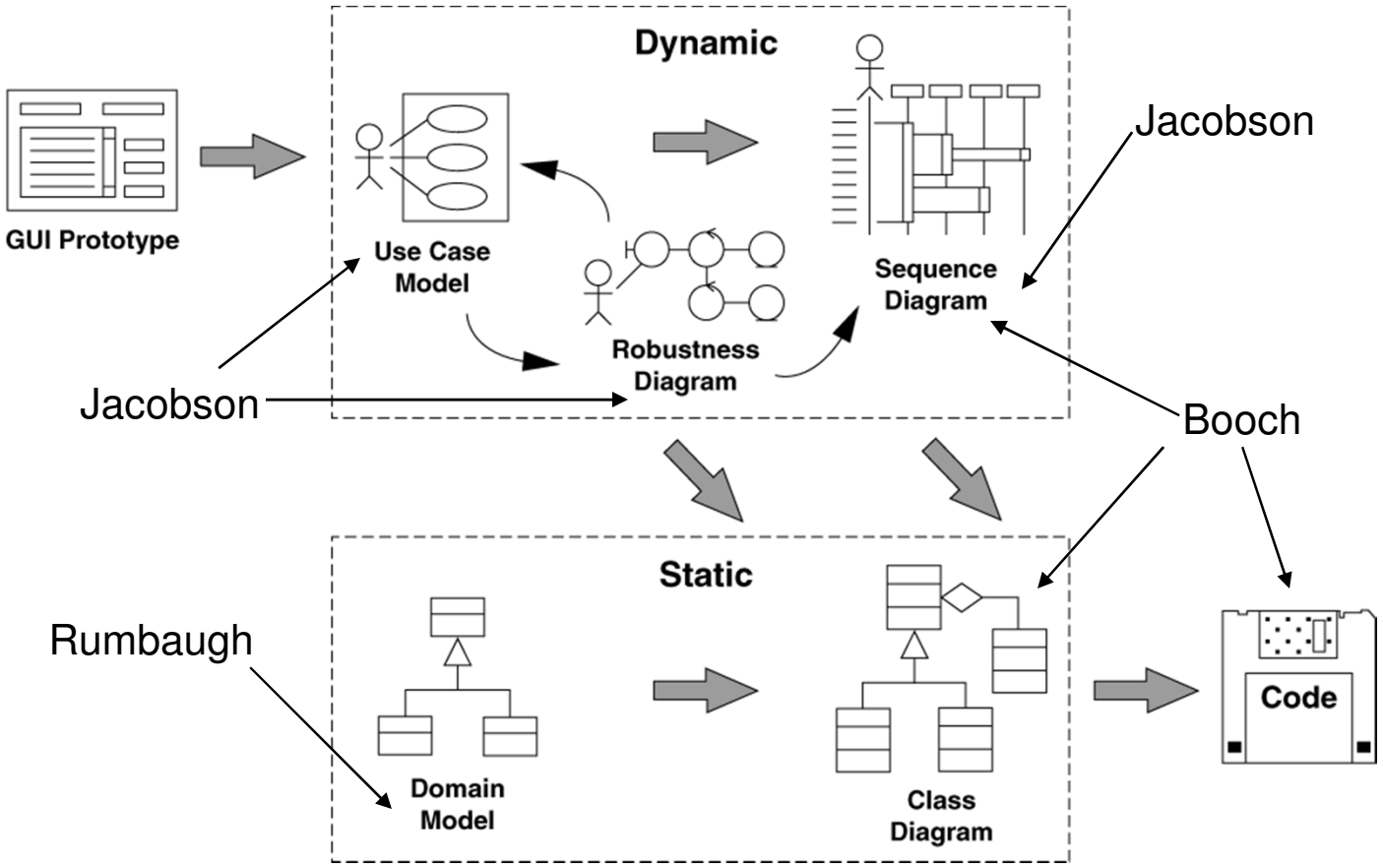
Agentes

- **Responsable de la certificación**
  - Preparar los tests de prueba a partir de las interacciones previstas entre objetos
- **Jefe de proyecto**
  - Entender la arquitectura del sistema y la interdependencia de sus componentes
- **Cliente**
  - Planificar el impacto del sistema dentro de la organización

Historia de UML

- **UML es una notación no una metodología**
- **Inicio: 1993 (Booch & Rumbaugh & Jacobson)**
- **1995 versión UML 0.8**
- **1997 versión UML 1.0**
- **1999 versión UML 1.3**
- **2005 versión UML 2.0**
- **2007 Versión UML 2.1**
- **2009 versión UML 2.2**
- **… actualmente versión 2.5 BETA2**
- **(http://www.omg.org/spec/UML/2.5/Beta2/PDF/)**

Historia de UML



Diagramas básicos UML

- Diagrama de Casos de Uso
- Diagrama de Actividad
- Diagrama de Secuencia
- Diagrama de Colaboración/ Comunicación
- Diagrama de Estado
- Diagrama de Clases
- Diagrama de Componentes
- Diagrama de Despliegue

Diagrama de Casos de Uso

- **Especificación de un CASO DE USO**
- **Matricular_Alumnos**

Propósito	Precondiciones	Activación
Realizar el proceso de matrícula a la universidad con las funciones de: - Identificación del Alumno - Validación de Requisitos - Tramitación del pago	Usuario tramitador habilitado Parámetros de la aplicación definidos	A discreción de un usuario habilitado ACTORES: Secretaria, Alumno

Diagrama de Casos de Uso

- **Especificación**

ESCENARIO DE ÉXITO
1. Usuario *activa ventana* de la aplicación de matrícula
2. Usuario *identifica al alumno* invocando al CU Identificar_Alumno.
3. El Sistema *valida requisitos* con CU Validar_Requisitos
4. Usuario ingresa al nuevo alumno
5. Sistema *muestra datos por defecto* como la fecha de matrícula.
6. Sistema *asigna ROL con CU_Generar Rol*
7. Sistema *genera el cálculo de la inscripción con CU calcular pago inscripción,* Usuario *registra la matrícula acad*émica
8. *Se genera el arancel con CU generar_arancel y se imprime un recibo llamando al CU Imprimir_recibo*

ESCENARIO ALTERNATIVO
Si el alumno es extranjero y no tiene rut se invoca a CU generar Rut ficticio
Postcondición: Se registra la Matrícula

Diagrama de Casos de Uso

- **Procesos principales**

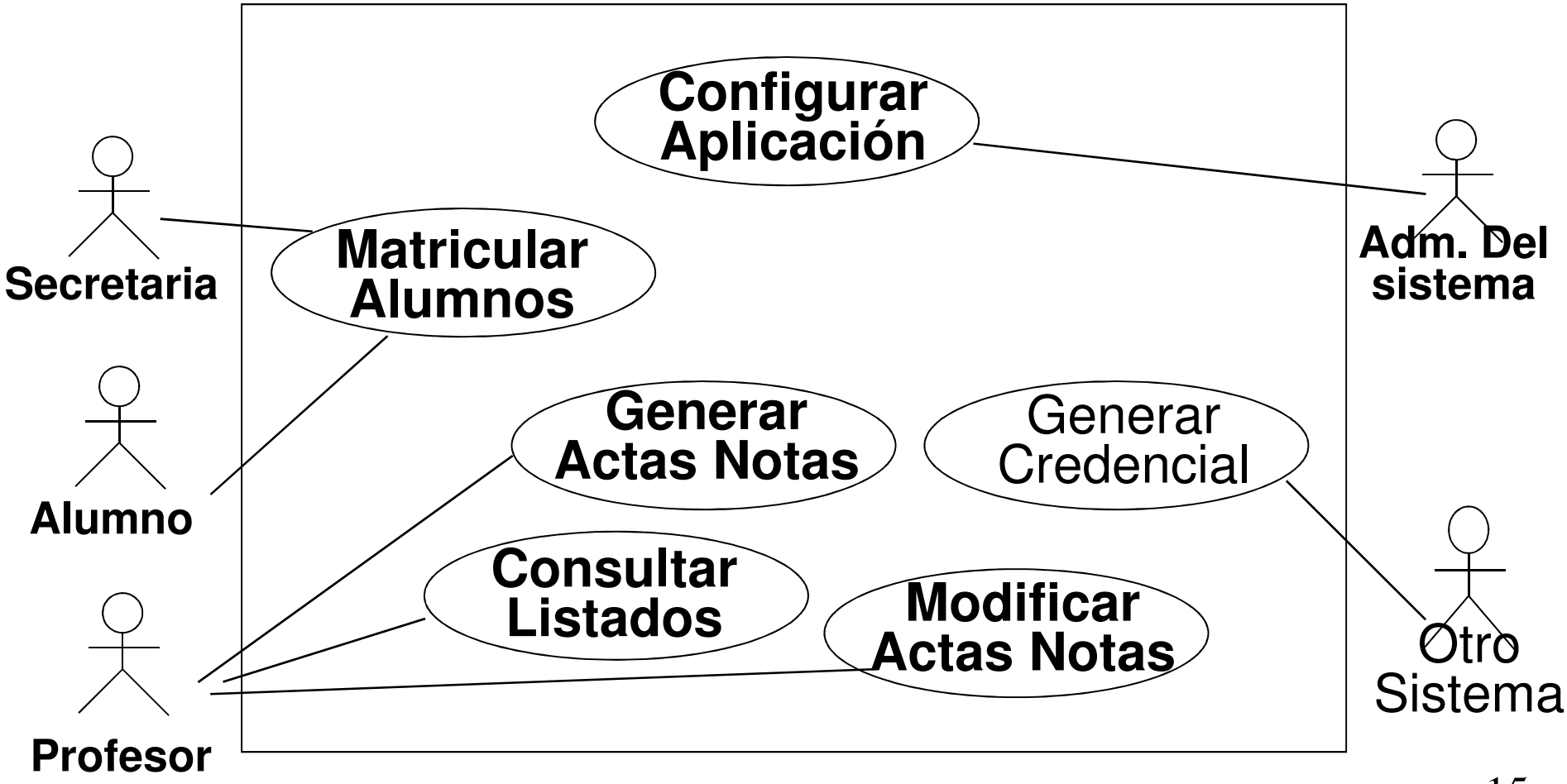


Diagrama de Casos de Uso

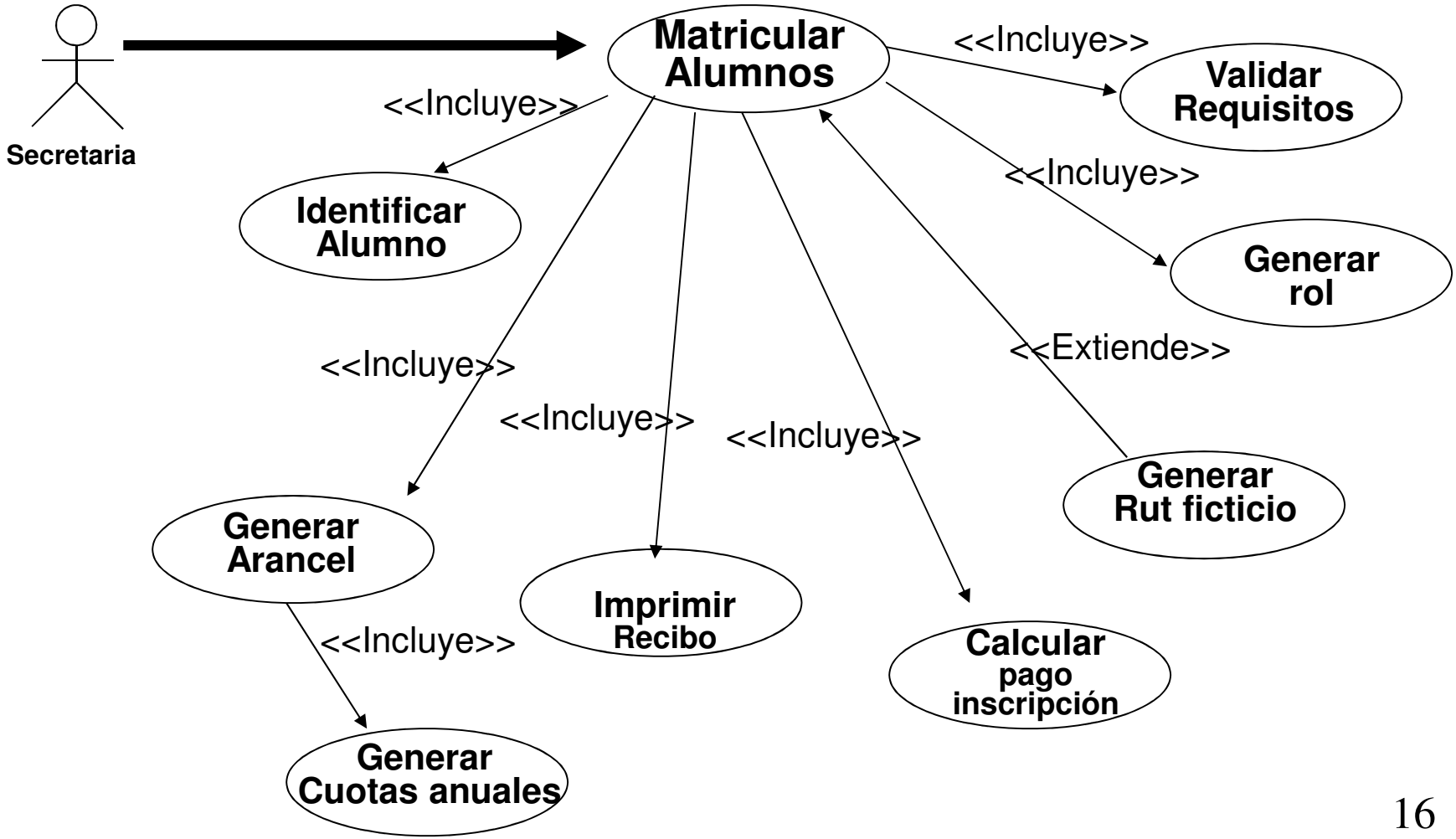


Diagrama de Casos de Uso

- **Muestran la granularidad del sistema en piezas de funcionalidad reutilizables**
- **Muestran la interacción de los Actores con la funcionalidad del Sistema**
- **Organizan visualmente los requerimientos del usuario**
- **Permiten certificar contractualmente la funcionalidad**
- **Formalizan el mapa de procesos de negocio**

Diagrama de Casos de Uso

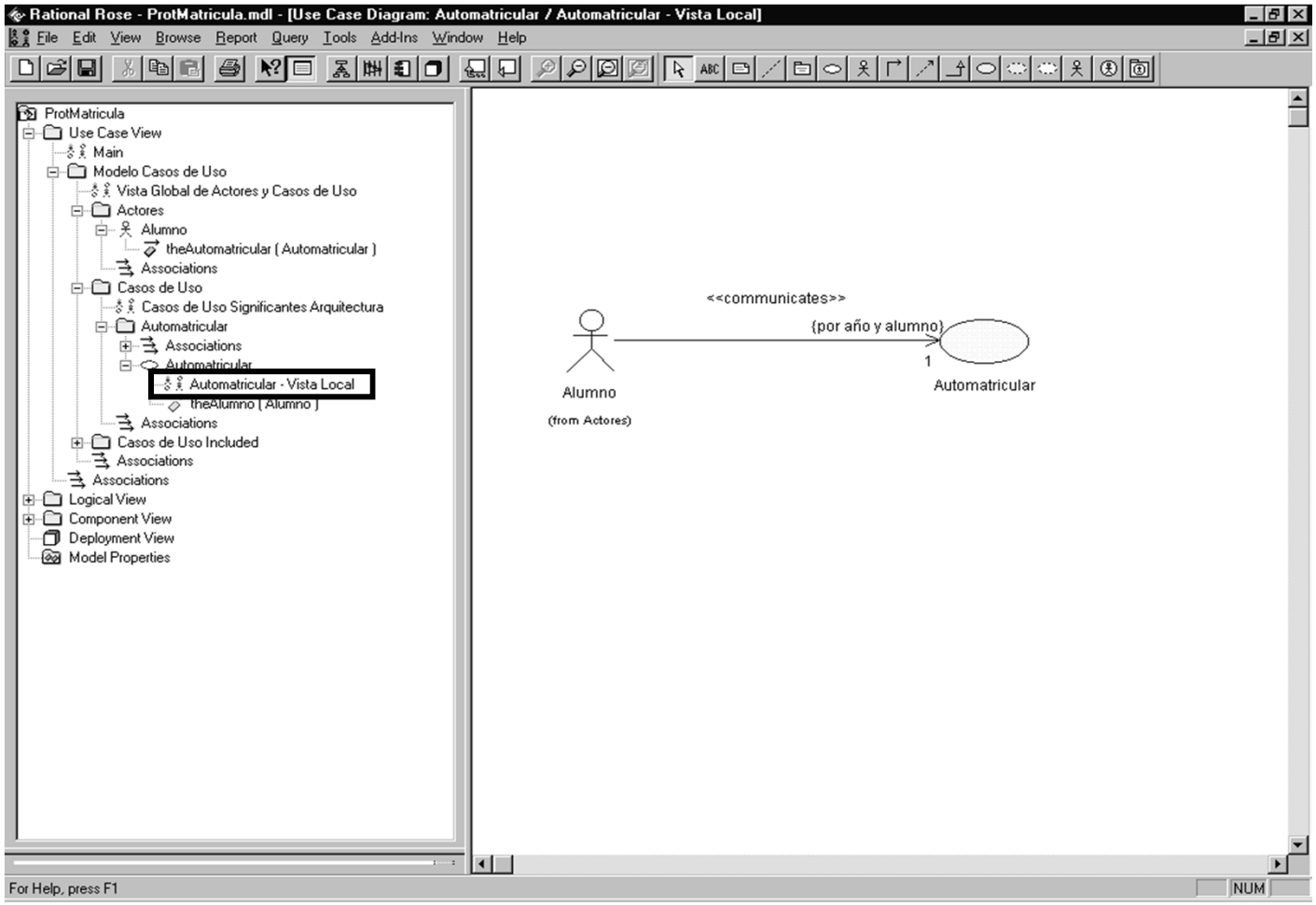
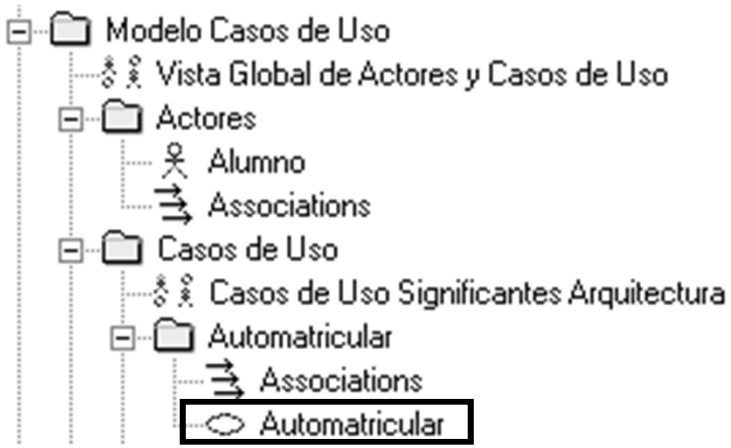
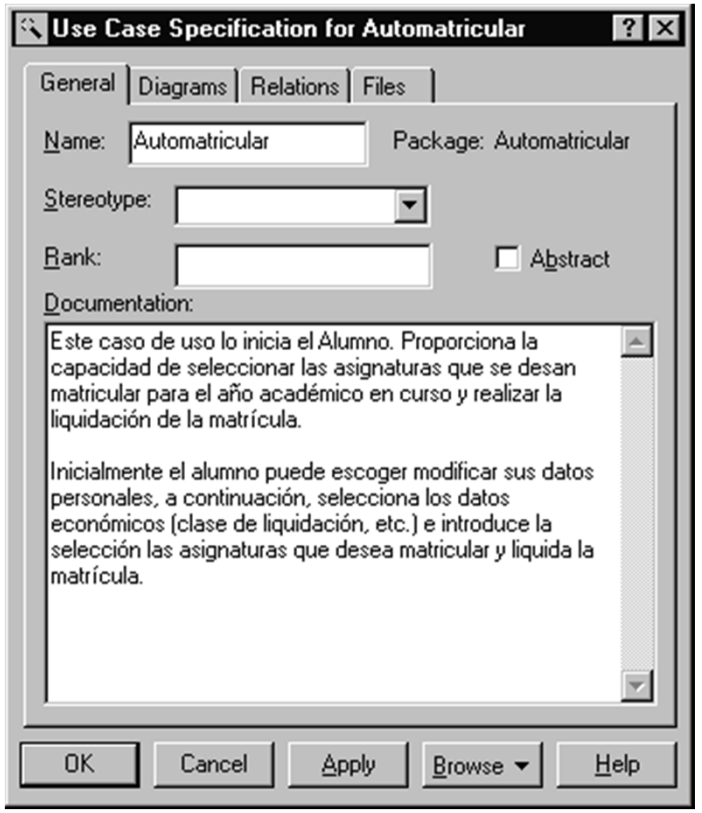


Diagrama de Casos de Uso





Ventajas de los Casos de Uso

- **Lenguaje de comunicación entre usuarios y desarrolladores**
- **Comprensión detallada de la funcionalidad del Sistema**
- **Acotación precisa de las habilitaciones de los usuarios**
- **Trazabilidad desde los requerimientos al código ejecutable**

Ventajas de los Casos de Uso

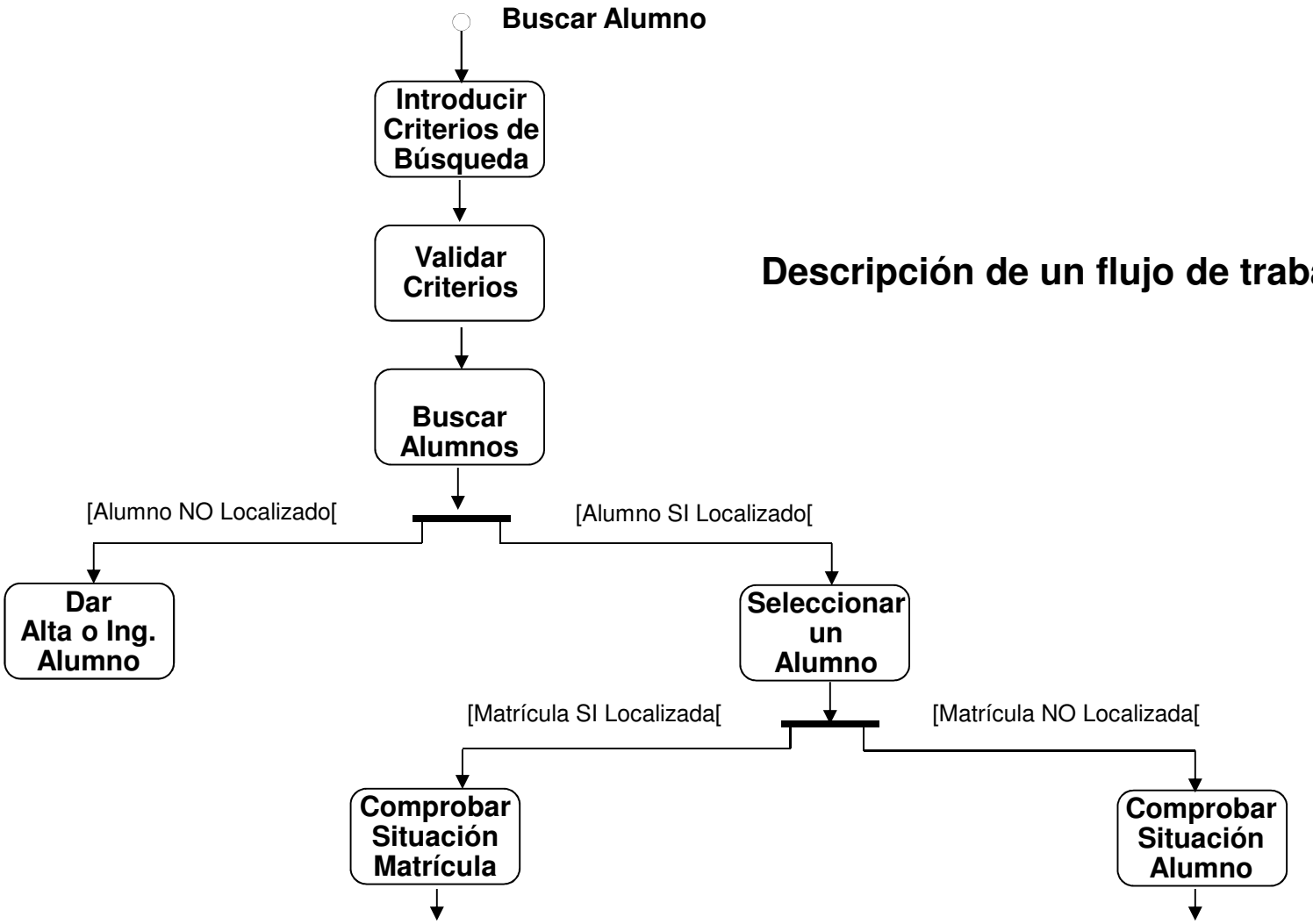
- **Gestión de riesgo para gobernar la complejidad de un sistema**
- **Planificación de iteraciones para su implementación**
- **Estimación precisa del esfuerzo para su implementación**
- **Documentación orientada al usuario: Manual de Procedimientos & Reglas de Negocio**

Diagrama de Actividad

- **Muestra la secuencia de actividades que se desarrollan en el flujo de trabajo de un Caso de Uso, como pieza de funcionalidad concreta**

- **Muestra el flujo de trabajo que se desarrolla en un proceso configurado como un paquete de Casos de Uso**

Diagrama de Actividad



Descripción de un flujo de trabajo

Diagrama de Actividad

- **Su objetivo no es relacionar actividad con objetos, sólo comprender qué actividades son necesarias y cuales son sus relaciones de dependencia**

- **Se utiliza para representar los distintos escenarios que comprende un Caso de Uso y permite describir tareas sincronizadas y responsabilidades**

Diagrama de Secuencia

- Describe la interacción de objetos que requiere la funcionalidad de los distintos escenarios de un Caso de Uso
- Los objetos son representados con su ciclo de vida dentro de una serie temporal
- Cada posible escenario de un Caso de Uso puede representarse con un diagrama de secuencia

Diagrama de Secuencia

Descripción de un escenario de Caso de Uso

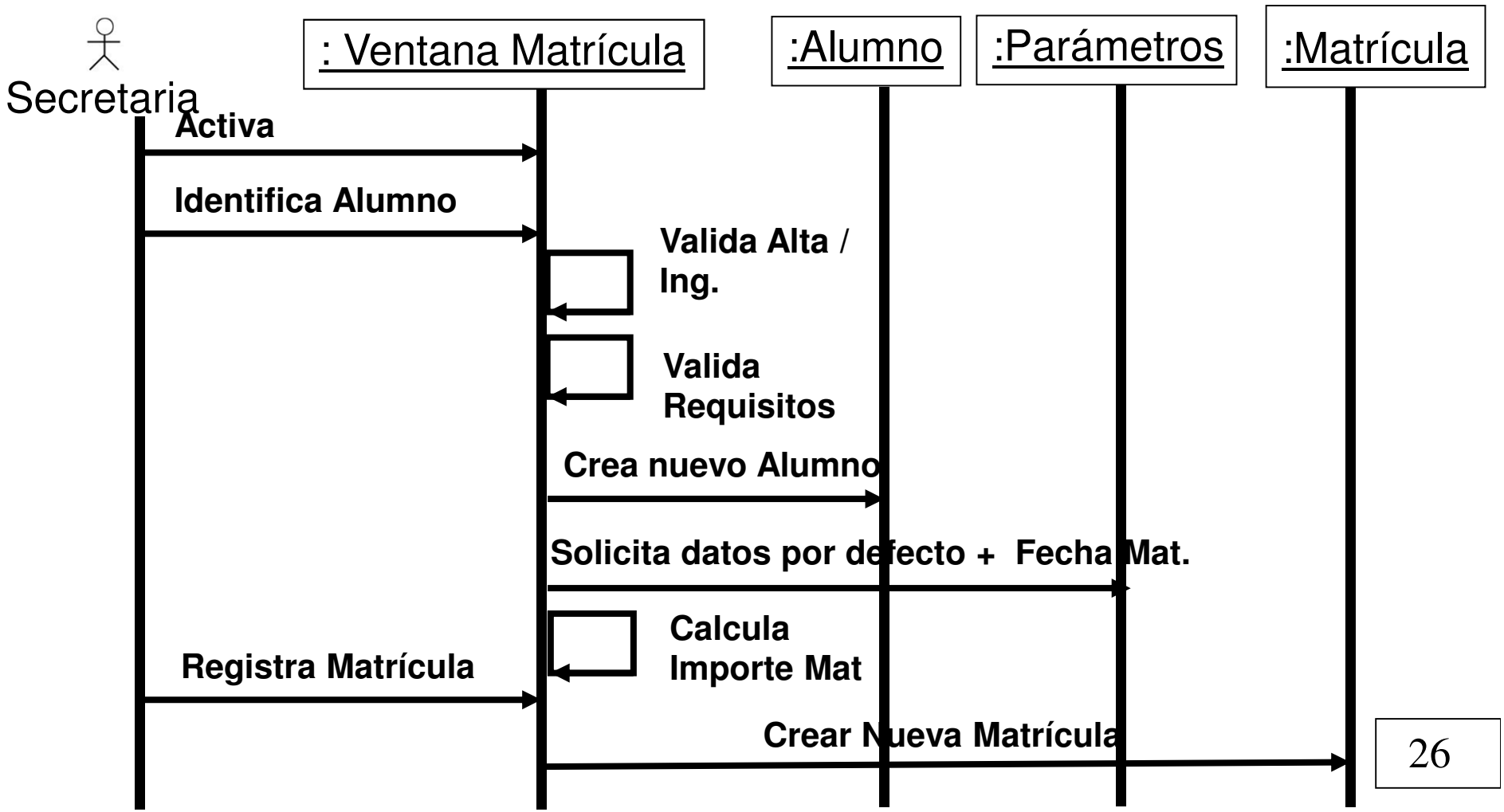


Diagrama de Colaboración/Comunicación

- **Muestra lo mismo que un diagrama de secuencia: cómo interaccionan los objetos dentro de un Caso de Uso**
- **A diferencia de un diagrama de secuencia no hay referencia a una serie temporal**
- **Su propósito es mostrar la topología del proceso distribuido entre los distintos objetos**

Diagrama de Colaboración

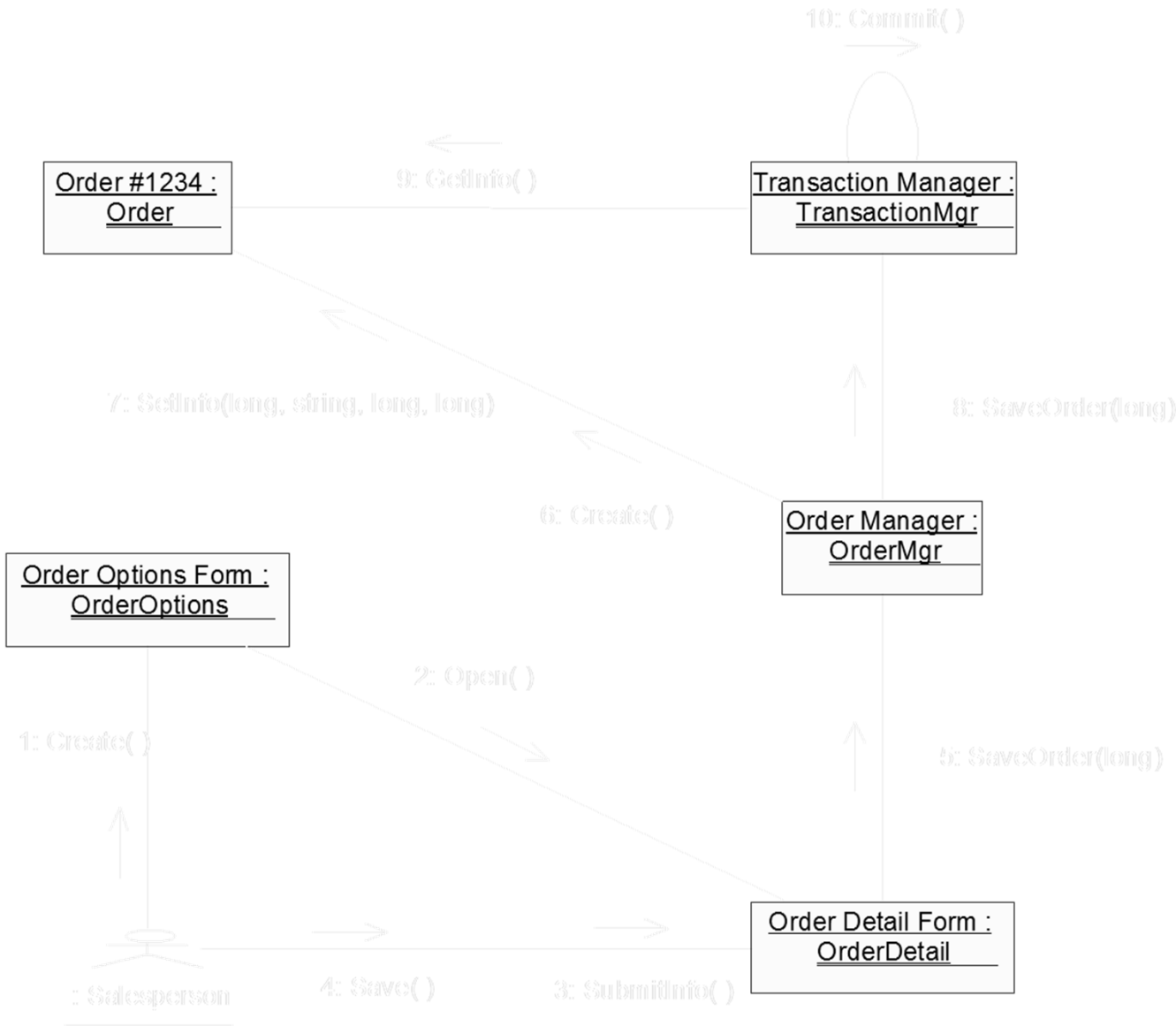
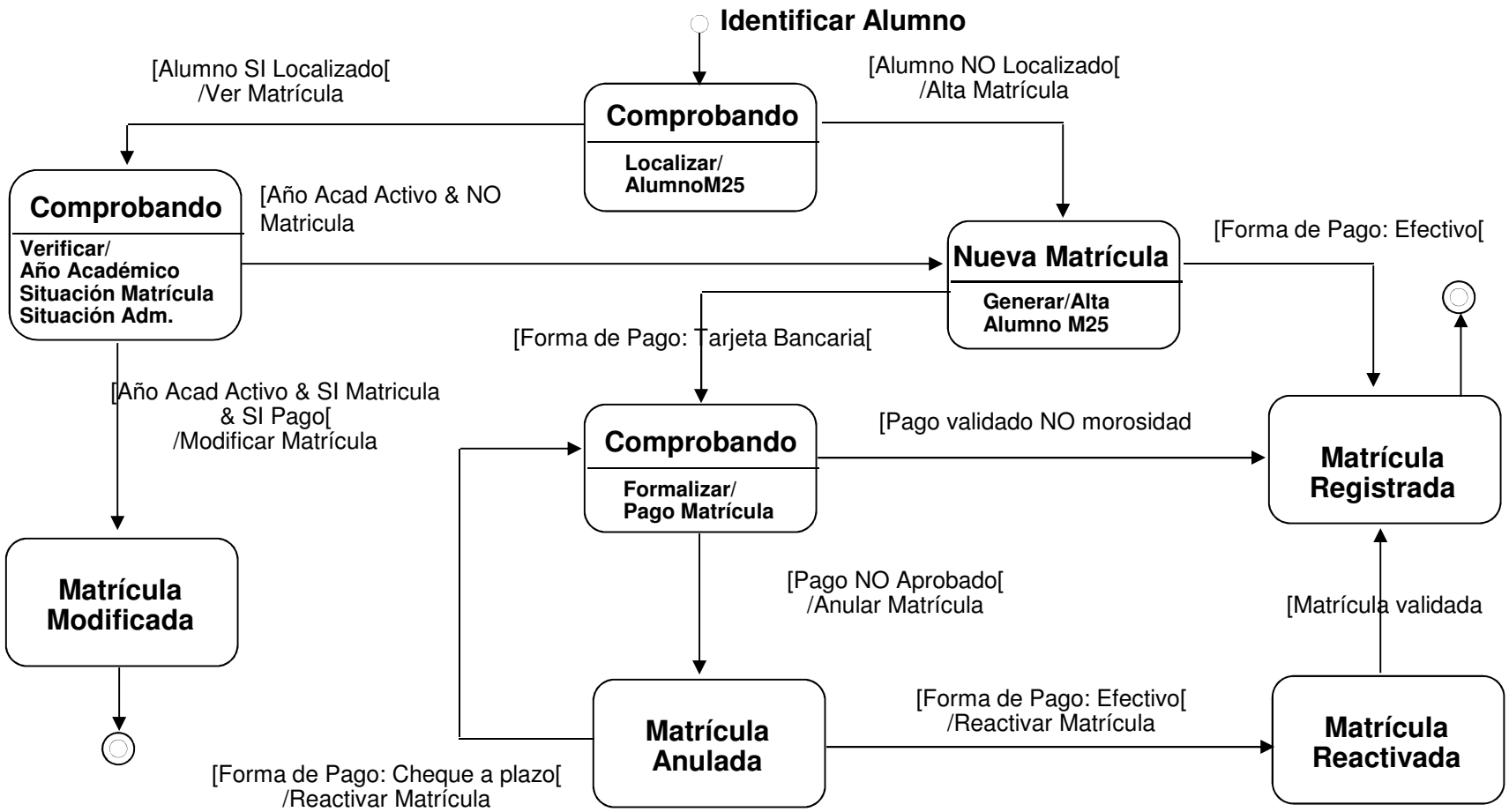


Diagrama de Estado

- **Muestra los distintos estados en que un objeto puede existir**
- **Presenta la visión dinámica del sistema**
- **Describe el comportamiento de un objeto, desde que nace hasta que muere**
- **Identifica todos los eventos necesarios para realizar la transición de un estado a otro**

Diagrama de Estado Transición



Dinámica del Sistema

Diagrama de Estado Transición

- **La dinámica de un sistema está determinada por:**
  - Todos los posibles estados de sus objetos
  - Todos los posibles eventos que afectan a los objetos
  - Todas las posibles transiciones de un estado

Diagrama de Estado Transición

- Un evento no es un objeto
- Un evento es “la causa” que justifica la existencia de un objeto
- Sólo podemos conocer que un evento ha ocurrido detectando “sus efectos”
- Sólo nos interesan los eventos que provocan un cambio de estado en los objetos
- Hay que distinguir un evento como tal, del objeto que representa el registro de sus efectos

Diagrama de Clases

- **Una Clase representa a un tipo de objetos que comparten:**
  - Las mismas propiedades (Atributos)
  - El mismo comportamiento (Métodos)
  - Las mismas relaciones con otros objetos (asociaciones y agregaciones)
  - La misma semántica dentro del sistema

Diagrama de Clases

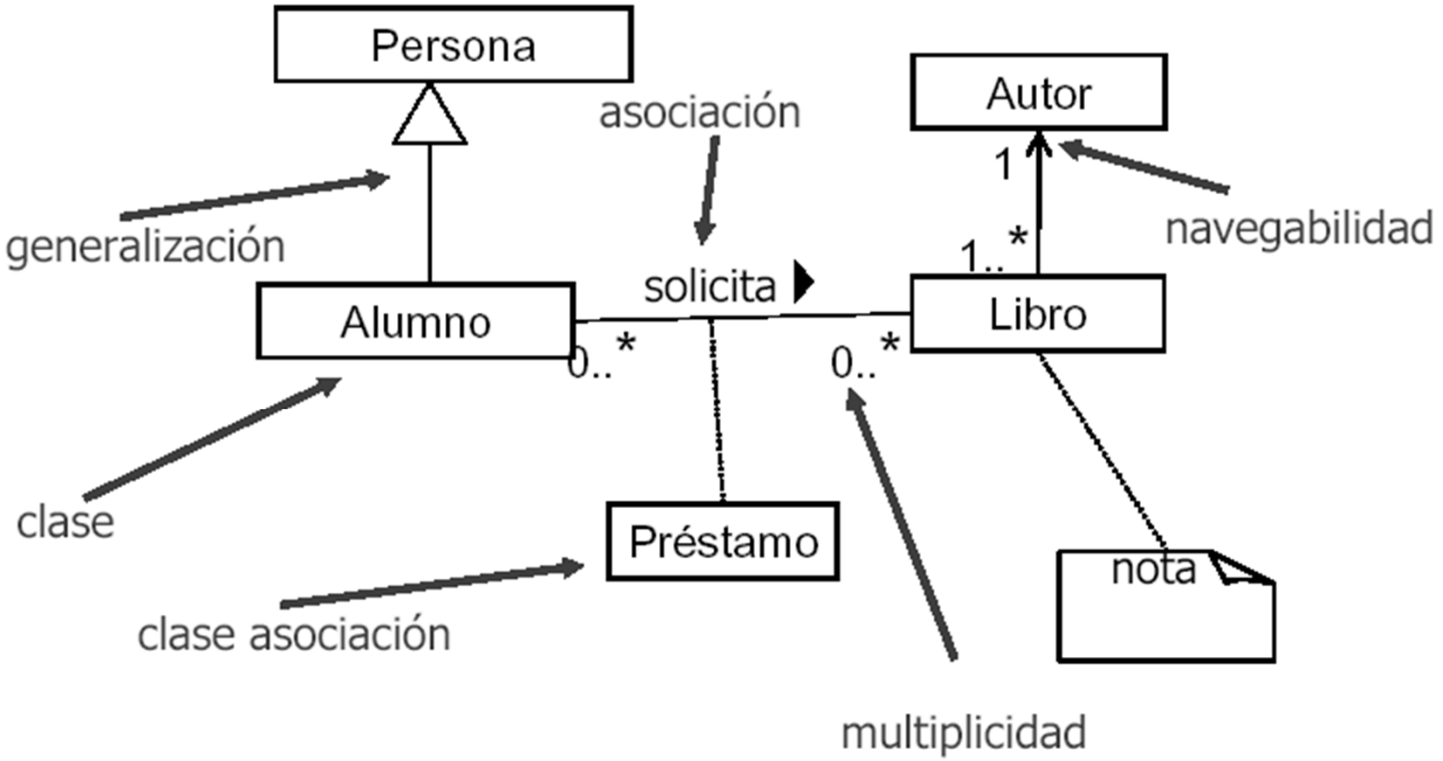


Diagrama de Clases

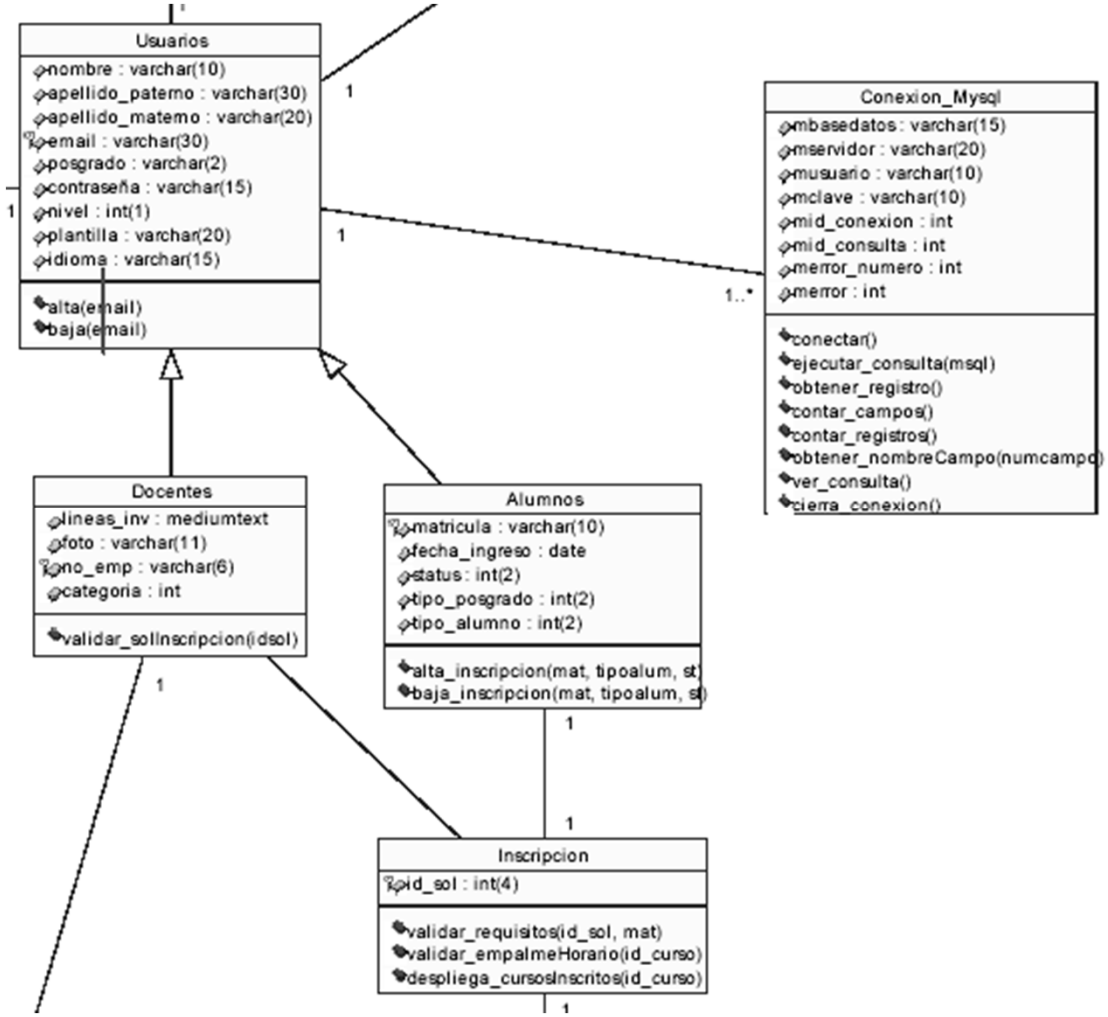


Diagrama de Clases

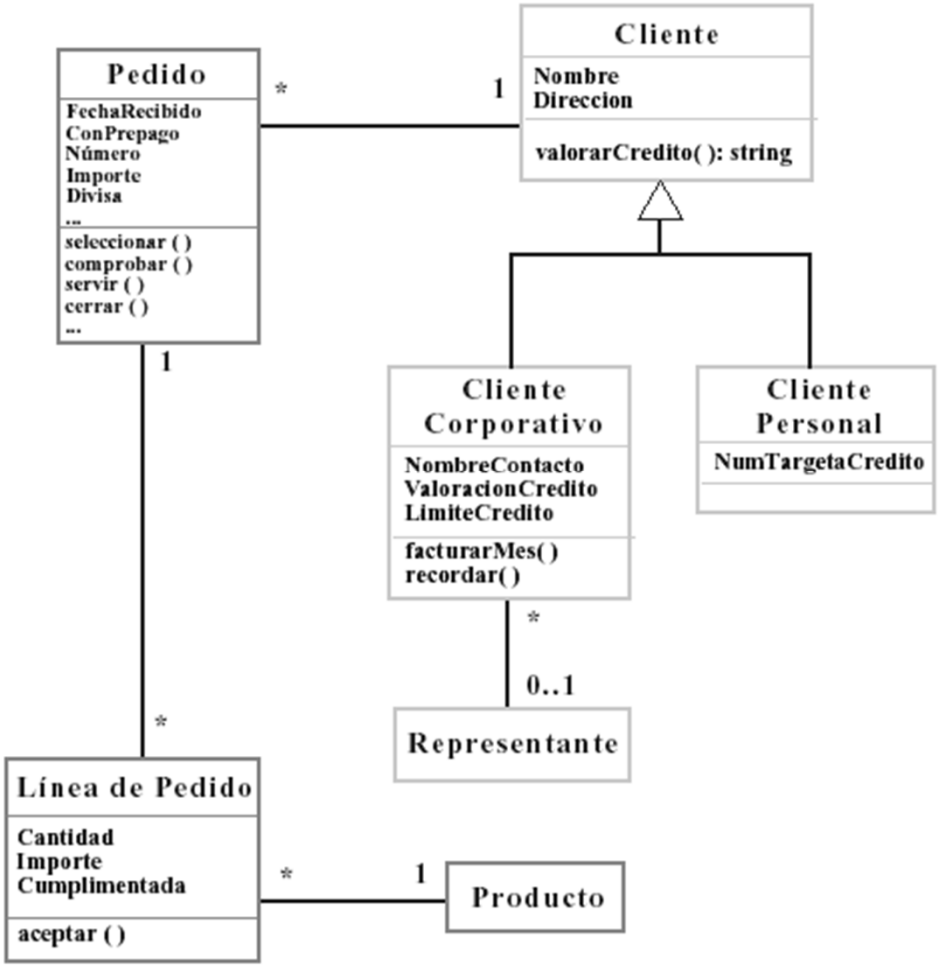
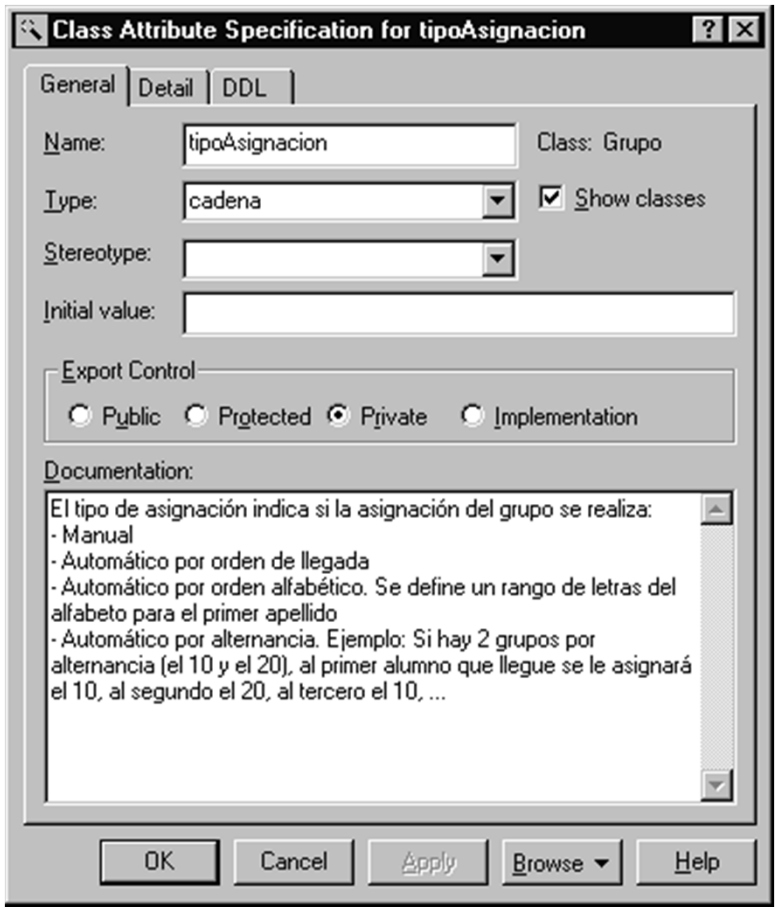


Diagrama de Clases

- **Un Objeto representa a una entidad del mundo real o inventada**
- **Es un concepto que dispone de una definición (intensión) y de una aplicabilidad (extensión)**
- **Es la instancia de una Clase**

Definir las Clases

Operaciones



Atributos

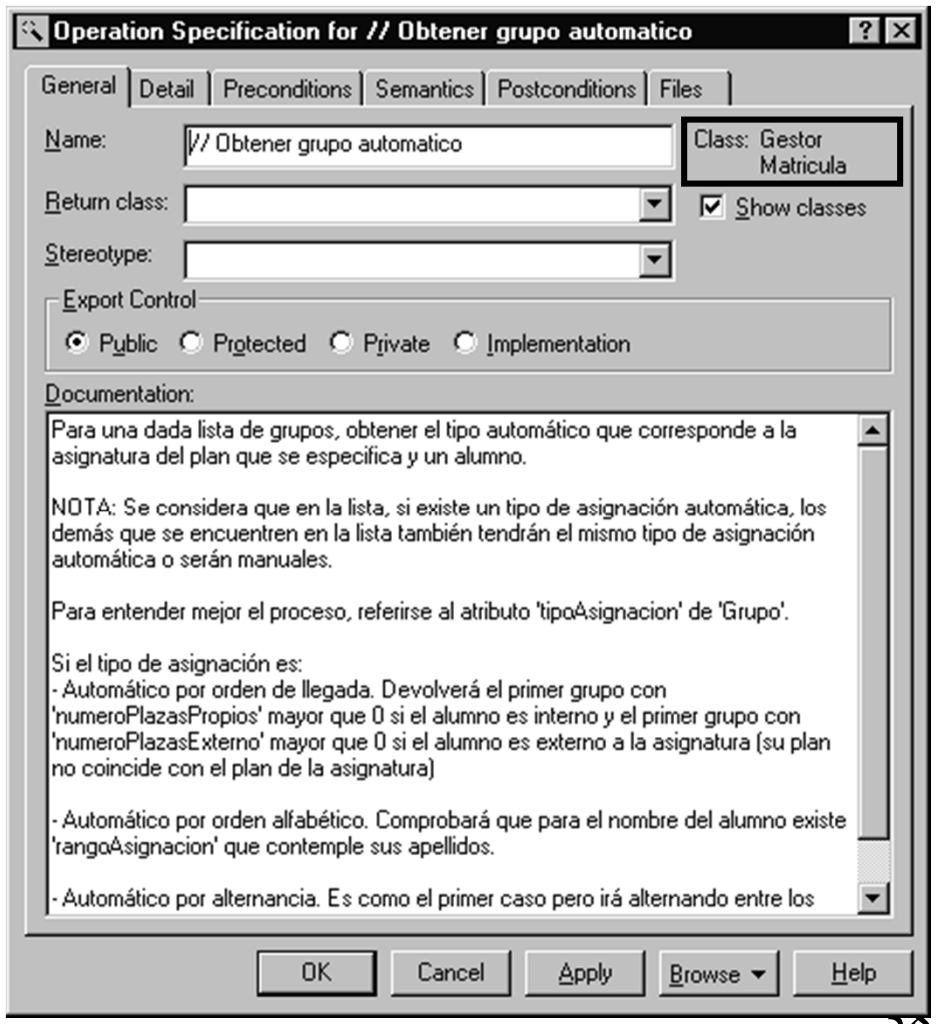


Diagrama de Componentes

- **Muestra la vista física del modelo**
- **Muestra los componentes de software que configuran el sistema y su interdependencia**
- **Presenta dos tipos de componentes:**
  - Ejecutables
  - Librerías de código
- **Cada clase del modelo es mapeada con el código fuente de un componente**

Diagrama de Componentes

- Son utilizados por el responsable de compilar el sistema
- Describen en qué orden han de ser compilados los componentes
- Muestran qué componentes run-time serán creados como resultado de la compilación
- Muestran el mapeo de las clases con los componentes implementados

Diagrama de Componentes

Vista de los componentes ejecutables

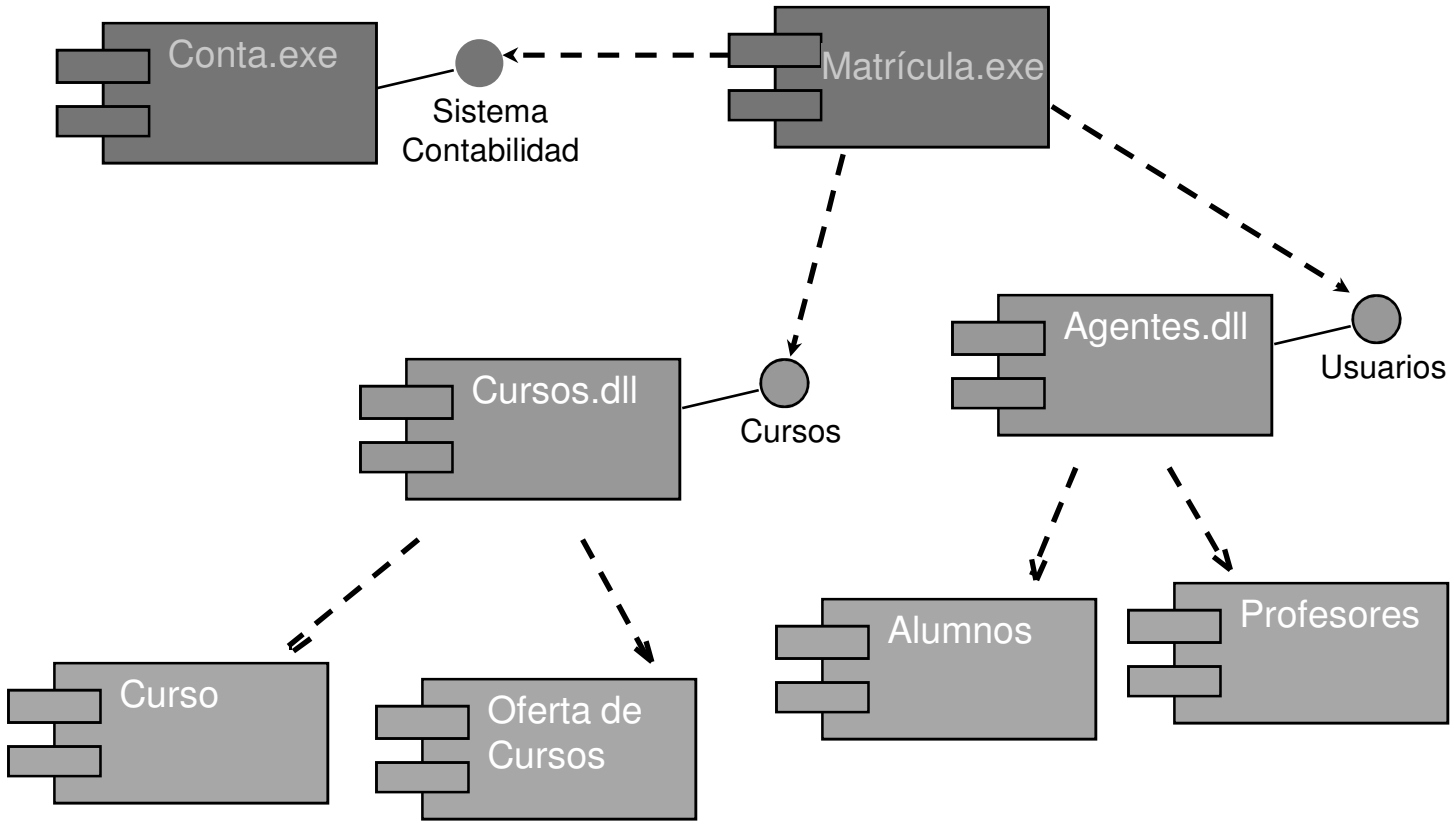
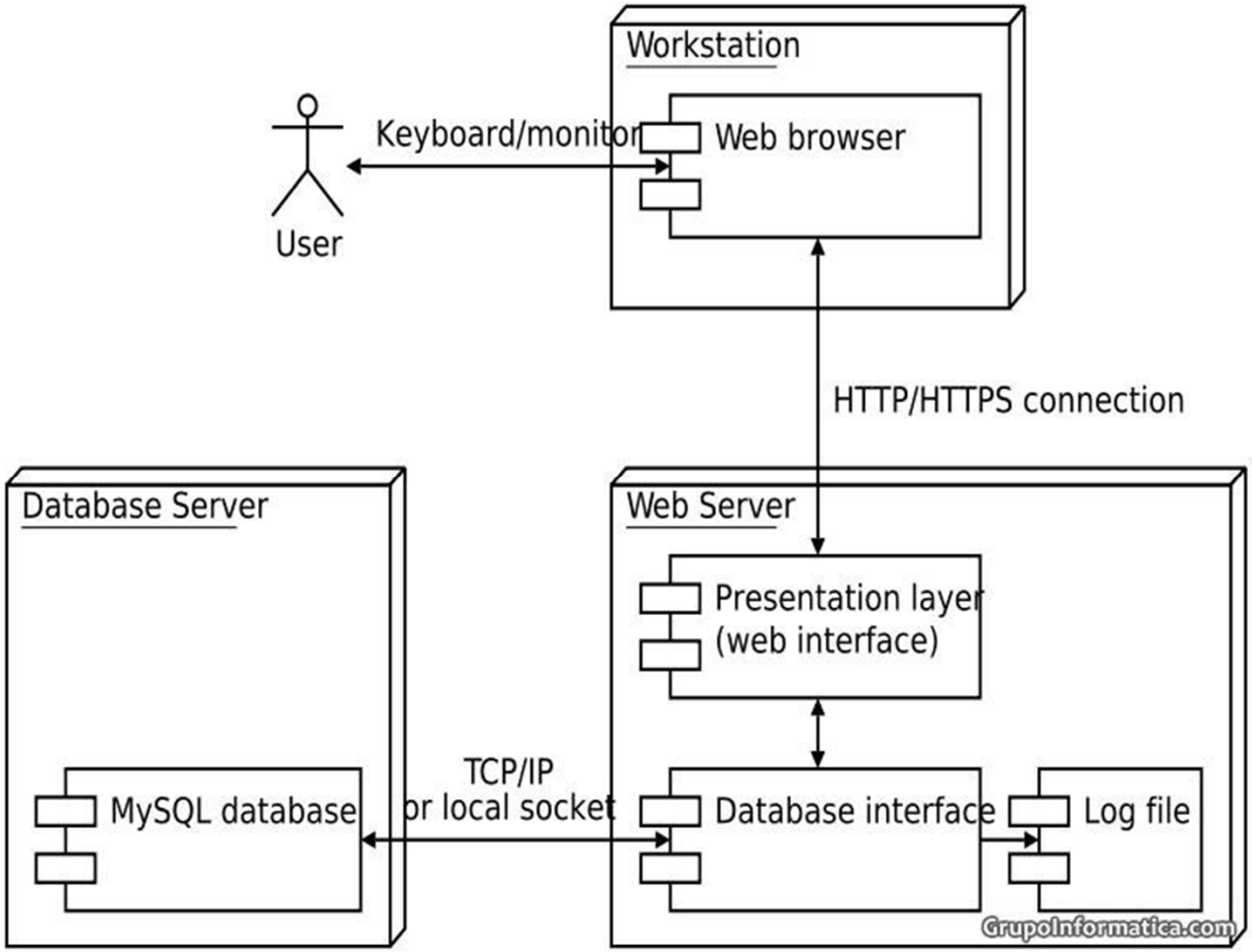


Diagrama de Despliegue

- **Muestra la distribución física de los componentes en nodos locales y remotos de la red**
- **Un nodo puede representar una pieza de hardware, desde un periférico a un servidor**
- **Presenta los distintos componentes de una arquitectura en tres capas (3Tier)**
  - Servidor de datos
  - Servidor de aplicaciones
  - Cliente

Diagrama de Despliegue

Vista de la distribución física de nodos de proceso



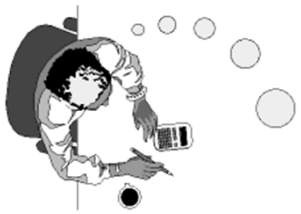
Agentes

- **Usuario**
  - Comprende el modelo conceptual de su dominio
- **Analista y Diseñador**
  - Definen la arquitectura del sistema
- **Desarrollador**
  - Organiza el código de manera simple y ordenada y traza el mapeo con la base de datos

Agentes

- **Arquitecto**
  - Supervisar el cumplimiento de los requerimientos no funcionales
    - Disponibilidad del sistema
    - Rendimiento
    - Escalabilidad
- **Implementador**
  - Comprender mejor la topología de un sistema distribuido

Metodología de desarrollo



Aprobación Anteproyecto

Realización Plan Director

Documentación Especificaciones

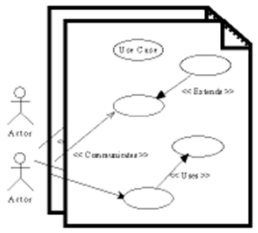
Interfaces Bases de Datos

Implementación Certificación

Código

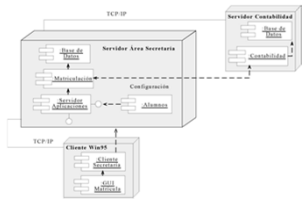
Matrícula Proyecto

Procesos Principales



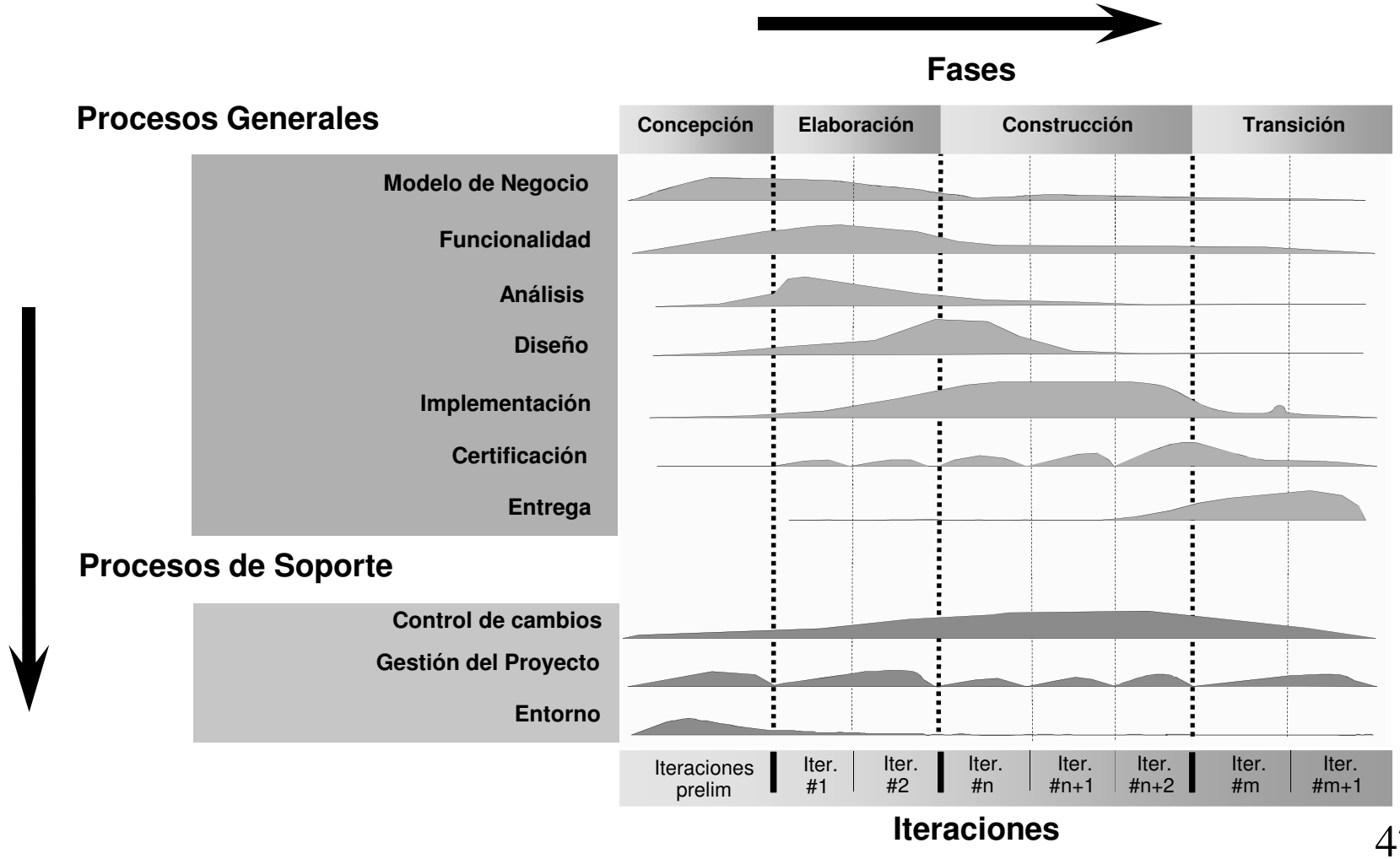
Funcionalidad

Análisis Diseño



Componentes Despliegue

Metodología de desarrollo



Concepción

- Misión del proyecto
- Inscripción del proyecto
- Glosario de conceptos
- Estimación de esfuerzo y cronograma
- Apoyo en patrones de funcionalidad y análisis
- Umbral de riesgo
- Aprobación del anteproyecto
- Proceso secuencial no iterativo

Elaboración

- Funcionalidad
- Priorización de los Casos de Uso
- Plan Director de Proyecto: Iteraciones
- Especificación de los Casos de Uso
- Análisis
- Diseño
- Pruebas de certificación
- Proceso iterativo

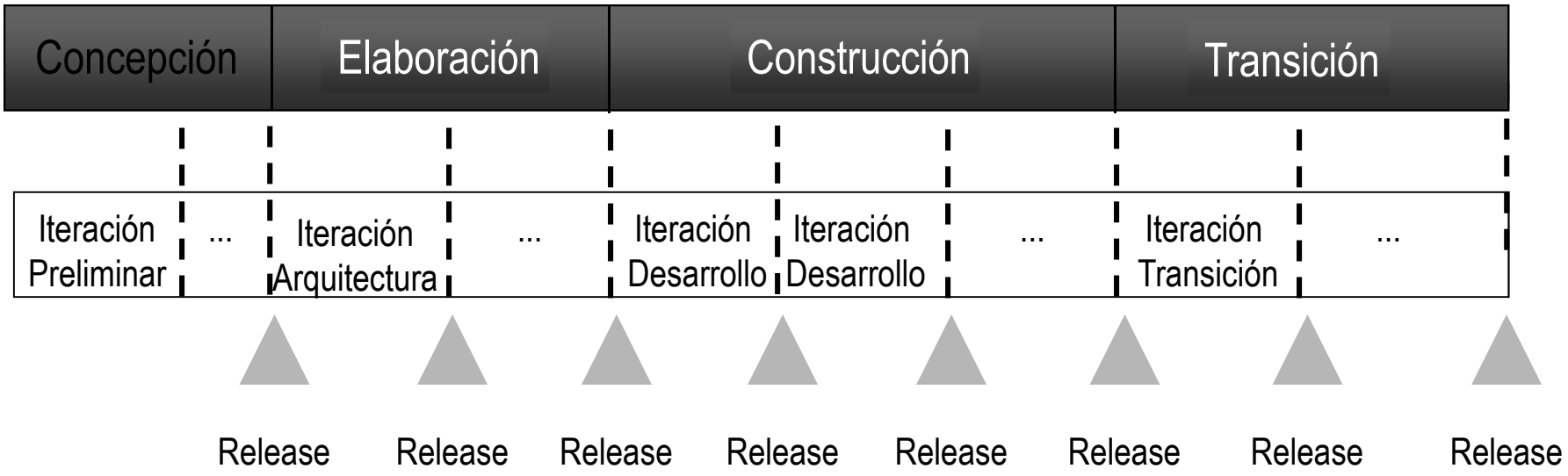
Construcción

- Poner el diseño en acción
- Desarrollo de código
- Refactoring
- Mapeo de la base de datos
- Interface gráfica de usuario: Navegación
- Pruebas de certificación
- Proceso iterativo

Transición

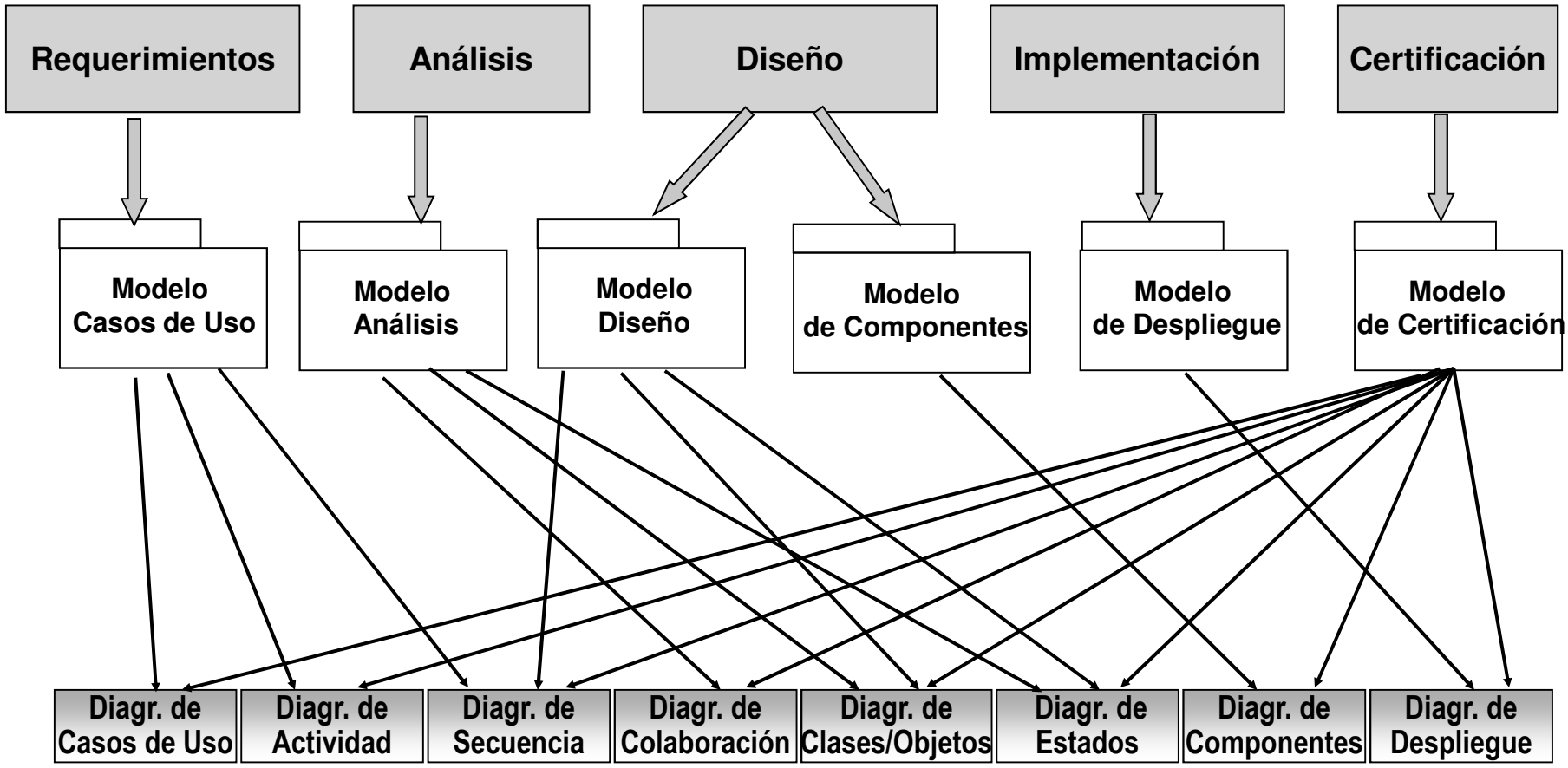
- Compilación y despliegue de componentes
- Pruebas de certificación
- Actualización del modelo de referencia
- Actualización del diseño y otros diagramas
- Documentación de usuario
- Documentación de administrador de sistema
- Plan de formación
- Plan de soporte

Metodología de desarrollo



Iteración : Secuencia de actividades con un **Plan Director** establecido y un criterio de **certificación** que finaliza con una versión ejecutable

Metodología de desarrollo

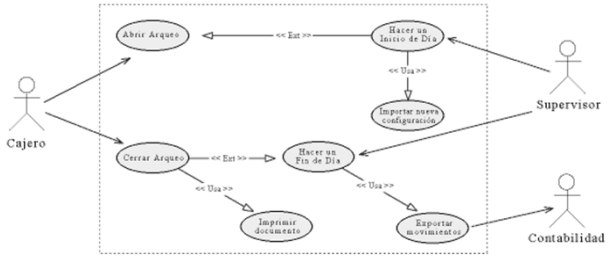


Plan de certificación

Certificación de la Funcionalidad

UC Cursar_pedidos

Flujo Principal	Variaciones
1. Usuario ***prepara un pedido de n items***, con su código de producto y cantidad, para un cliente.	
2. Sistema ***comprueba cada línea*** del pedido con la situación de ítems del producto en stock.	a. SI existen suficientes ítems del producto en el stock, sistema ***asigna ítems al pedido***. b. NO existen suficientes unidades del producto en el stock, sistema ***aparca el pedido*** del cliente y ***genera un pedido de reposición*** a proveedor para reponer las unidades necesarias a stock y ***servir el pedido*** del cliente
3. Sistema ***genera expedición del pedido*** al cliente.	



Diagramas Casos de Uso

Especificación Casos de Uso

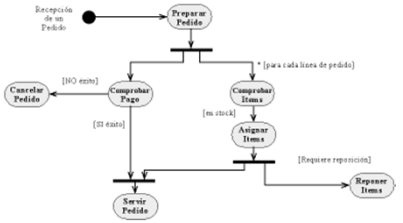


Diagrama de Actividad

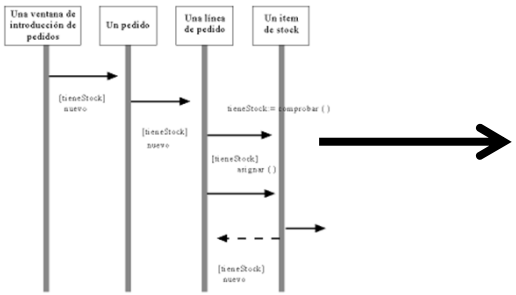
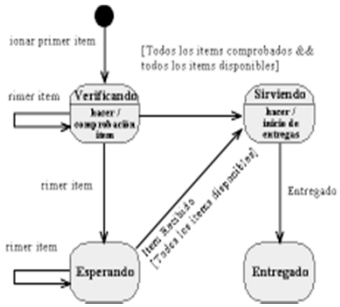


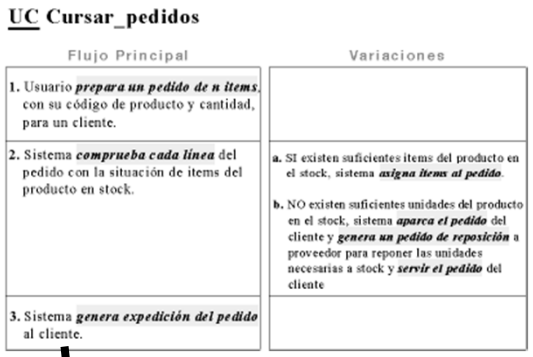
Diagrama de Secuencia



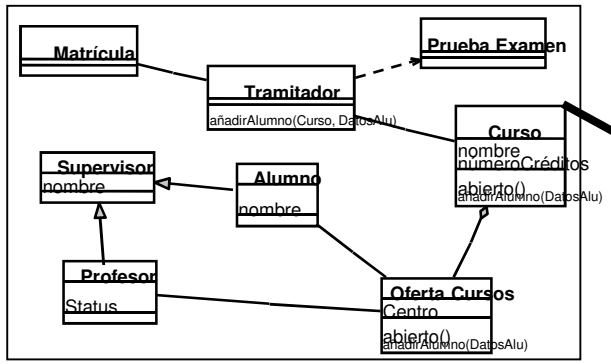
Dinámica Estados

Plan de certificación

Certificación del Modelo de Referencia (TRAZABILIDAD)



Especificación Casos de Uso



Diagramas de Clases

Interfaces

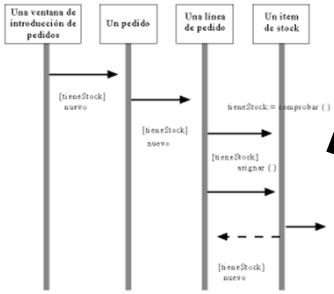


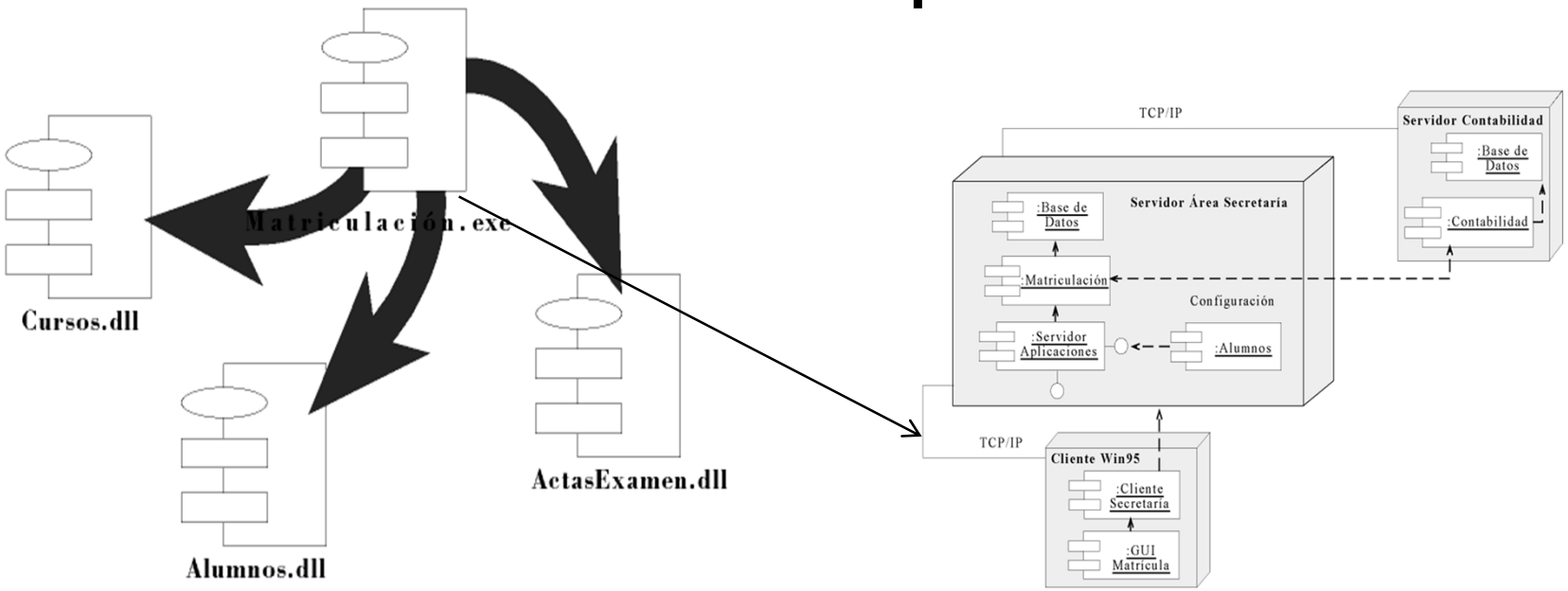
Diagrama de Secuencia

Modelo de Datos

Plan de certificación

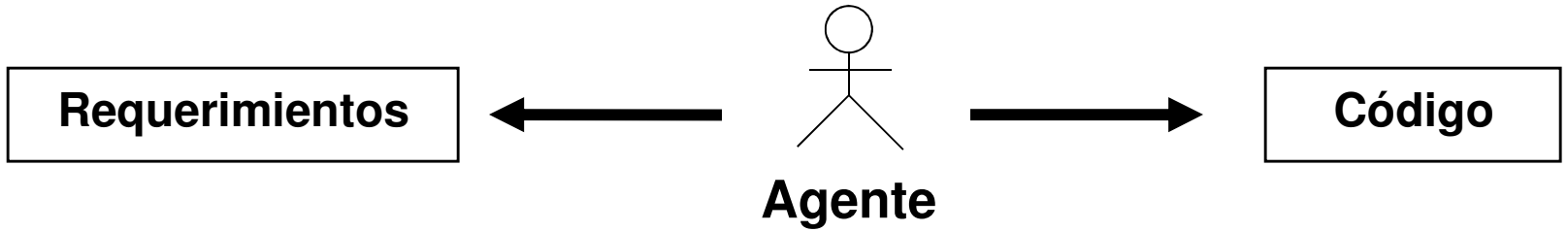
Certificación de Componentes (SW / HW)

Arquitectura

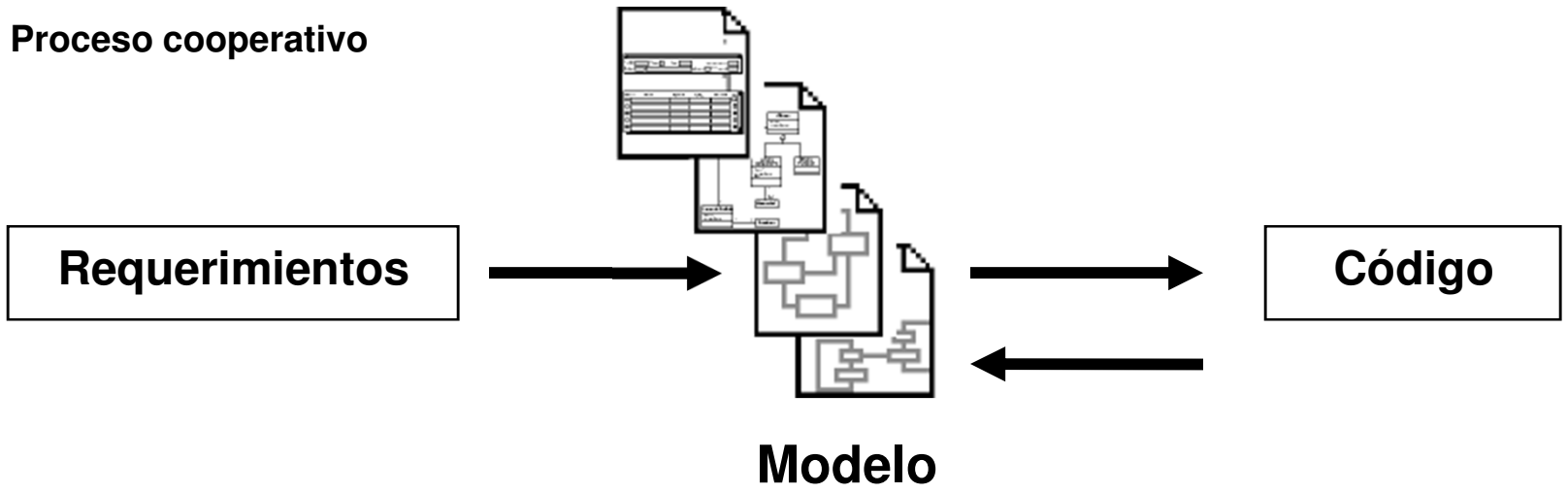


Agentes del Modelo

Método tradicional



Proceso cooperativo



Agentes del Modelo

- **Cliente y Jefe de proyecto**
  - Usaran los diagramas de Casos de Uso para visualizar la globalidad del sistema y delimitar el alcance del proyecto

- **Jefe de proyecto**
  - Usará los diagramas de Casos de Uso y la documentación asociada para descomponer el proyecto en un Plan Director de Iteraciones

Agentes del Modelo

- **Analista y Cliente**
  - Usaran la documentación asociada a los Casos de Uso para comprender mejor y delimitar la funcionalidad del sistema

- **Documentalista**
  - Usará la documentación asociada a los Casos de Uso para redactar los manuales de usuario y definir el plan de formación

Agentes del Modelo

- **Analista y Desarrollador**
  - Usaran los diagramas de secuencia y colaboración para visualizar la lógica del sistema, y el flujo de mensajes entre los objetos que lo componen

- **Controller**
  - Usará la documentación asociada a los Casos de Uso y los diagramas de secuencia y colaboración para diseñar las pruebas de certificación

Agentes del Modelo

- **Desarrollador**
  - Usará los diagramas de Clases y los diagramas de Estado Transición para visualizar la estructura de todas las piezas claves del sistema y la dinámica de su comportamiento

- **Implementador**
  - Usará los diagramas de Componentes y los diagramas de Despliegue para visualizar los ejecutables, ficheros DLL y otros componentes, así mismo la distribución de su despliegue en la red

Agentes del Modelo

- **Todos los Agentes**
  - Usaran el modelo de referencia para garantizar la trazabilidad entre los requerimientos y el código, y para asegurar la trazabilidad entre el código y la funcionalidad