TEMA 56	Análisis y diseño orientado a objetos (BLOQUE INGENIERÍA DEL SOFTWARE)

I. INTRODUCCIÓN

A lo largo del desarrollo de un sistema software, desde el estudio de requisitos, hasta la implementación y puesta en marcha del mismo, se contemplan multitud de tareas complejas. Hoy en día existen metodologías de desarrollo muy definidas, las cuales han ido evolucionando en gran medida gracias a la ingeniería del software que pretende aplicar los principios de la ingeniería al desarrollo de software de forma que se sistematice la producción del mismo.

El organismo de estandarización IEEE define la ingeniería del software como la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software. Su objetivo es facilitar el mantenimiento del software y aumentar la calidad de éste.

En el paradigma de la orientación se integran dos aspectos de los sistemas de información que tradicionalmente, en el paradigma estructurado, se han analizado de forma separada, datos y procesos. Un sistema se concibe como un conjunto de objetos que se comunican entre sí mediante mensajes. A nivel conceptual un objeto es una entidad percibida en el sistema que se está desarrollando, mientras que a nivel de implementación, un objeto se corresponde con un encapsulamiento de un conjunto de operaciones o servicios. El encapsulamiento es un principio de abstracción que agrupa datos y procesos permitiendo ocultar a los usuarios detalles de implementación, ofreciendo una interfaz externa mediante la que interactuar con el objeto.

Un objeto se describe por sus propiedades, atributos, y por los servicios que puede proporcionar. El estado de un objeto viene determinado por los valores que toman los atributos. Se denomina clase a la implementación de un tipo de objeto.

En el paradigma orientado a objeto se puede distinguir entre dos grandes corrientes:

- Metodologías dirigidas por los datos, basadas en principalmente en la parte estructural de los objetos.

- Metodologías dirigidas por las responsabilidades, que representan un enfoque más purista y se centran en la responsabilidad de los objetos, es decir, las tareas que pueden llevar a cabo.

Hoy en día se ha impuesto UML, Unified Modeling Language), como lenguaje gráfico de modelado para visualizar, especificar, construir y documentar las especificaciones de un sistema software. La metodología MÉTRICA v3 tiene en cuenta la mayoría de las técnicas que contempla UML 1.2.

El tema se enfocará desde la visión de ingeniería del software con la metodología MÉTRICA v3, la cual fue creada en 1989 por el Consejo Superior de Informática, y cuyo objetivo fue la creación de un marco metodológico común para planificación y el desarrollo de sistemas de información de la Administración Pública Española.

II. DESARROLLO DEL TEMA

1. ANÁLISIS ORIENTADO A OBJETOS

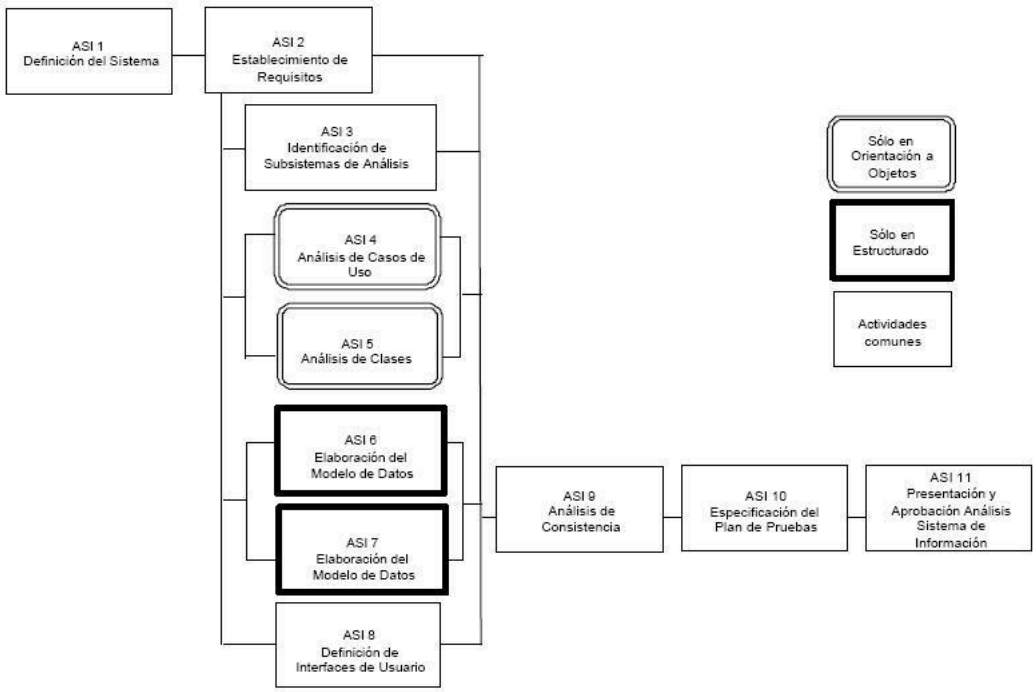
MÉTRICA v3 es una metodología que cubre tanto desarrollos estructurados como orientados a objeto. La metodología descompone el ciclo de desarrollo de software en procesos, cada uno de estos a su vez en actividades, y estas a su vez en tareas. Los procesos de la estructura principal de MÉTRICA v3 son:

- Planificación de Sistemas de Información.
- Desarrollo de Sistemas de Información.
- Mantenimiento de Sistemas de Información.

El proceso de "Desarrollo de Sistemas de Información" se descompone en cinco procesos para facilitar su comprensión:

- Estudio de Viabilidad del Sistema (EVS).
- Análisis del Sistema de Información (ASI).
- Diseño del Sistema de Información (DSI).
- Construcción del Sistema de Información (CSI).
- Implantación y Aceptación del Sistema (IAS).

El proceso de "Análisis del Sistema de Información (ASI)" tiene como objetivo conseguir una especificación detallada del sistema de información que satisfaga las necesidades de información de los usuarios.



1.1 ACTIVIDAD ASI1: DEFINICIÓN DEL SISTEMA

La primera tarea del proceso lleva a cabo la descripción del sistema de información, a partir de los productos generados en el proceso "Estudio de Viabilidad del Sistema (EVS)". Se delimita el alcance del sistema y se genera un catálogo de requisitos generales y se describe el sistema mediante modelos de alto nivel. También se identifican los usuarios que participan en el proceso de análisis y se elabora el plan de trabajo a seguir.

1.2 ACTIVIDAD ASI2: ESTABLECIMIENTO DE REQUISITOS

A continuación de la actividad ASI1 se realiza la definición de requisitos del nuevo sistema. El objetivo de esta actividad es elaborar un catálogo de requisitos detallado, que permita describir con precisión el SI, y que además sirva de base para comprobar que es completa la especificación de los modelos obtenidos en actividades posteriores, Identificación de Subsistemas de Análisis (ASI3), Análisis de Casos de Uso (ASI4), Análisis de Clases (ASI5) y Definición de Interfaces de Usuario (ASI8).

Para la obtención de requisitos se toma como punto de partida los resultados de la actividad ASI1 y se competan mediante sesiones de trabajo con los usuarios. Como técnica de obtención de requisitos se propone la especificación de casos de uso de la orientación a objetos.

1.3 ACTIVIDAD ASI3: IDENTIFICACIÓN DE SUBSISTEMAS DE ANÁLISIS

La siguiente actividad trata de estructurar el SI en subsistemas de análisis, para facilitar la especificación de los distintos modelos y la traza de requisitos. En paralelo se generan los distintos modelos que sirven de base para el diseño, actividades ASI4, ASI5 y ASI8.

Se asume la necesidad de una realimentación y ajuste continuo con respecto a la definición de los subsistemas, sus dependencias e interfaces.

1.4 ACTIVIDAD ASI4: ANÁLISIS DE LOS CASOS DE USO

Su objetivo es identificar las clases cuyos objetos son necesarios para realizar un caso de uso y describir su comportamiento mediante la interacción de dichos objetos. Esta actividad se lleva a cabo para cada uno de los casos de uso contenidos en cada subsistema identificado en la actividad ASI3, obteniéndose los diagramas de clases y el diagrama de interacción de objetos.

Las tareas de esta actividad se realizan en paralelo, con continuas realimentaciones entre ellas.

- ASI4.1 Identificación de Clases Asociadas a un Caso de uso: aplica el Diagrama de Clases para generar un modelo de clases de análisis.

- ASI4.2 Descripción de la Interacción de Objetos: aplica el Diagrama de Interacción de Objetos (secuencia o colaboración) para obtener el análisis de la realización de los casos de uso.

1.5 ACTIVIDAD ASI5: ANÁLISIS DE CLASES

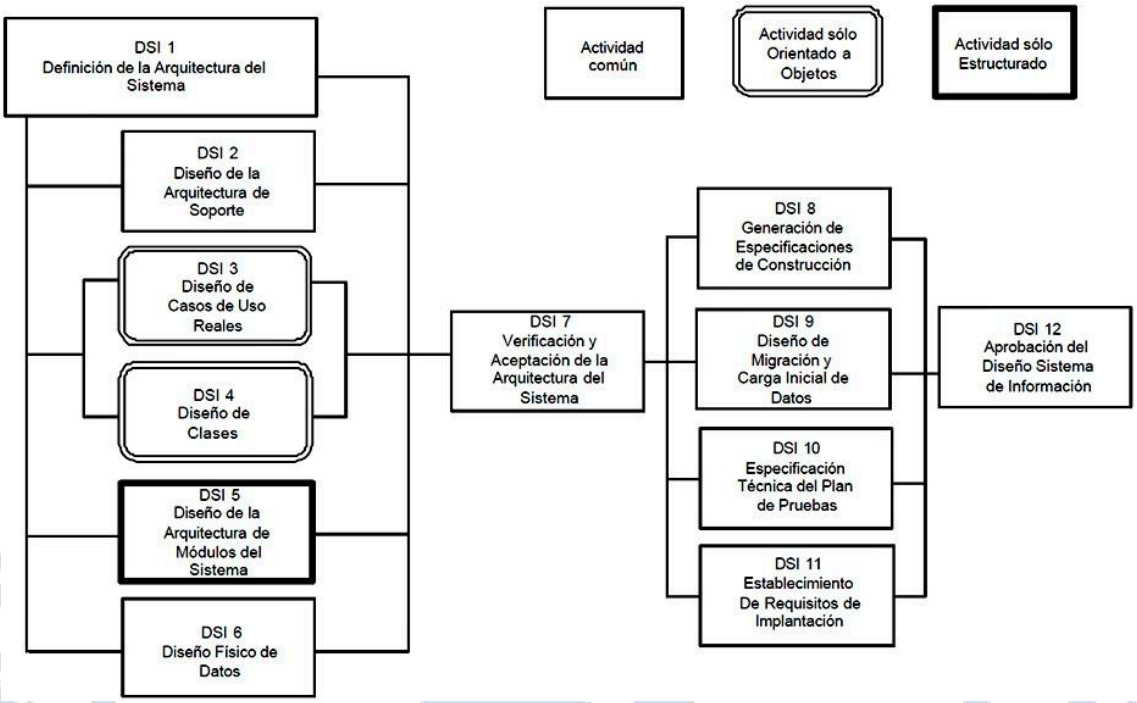
Describe cada una de las clases que han surgido, identificando las responsabilidades que tienen asociadas, sus atributos, y las relaciones entre ellas. Teniendo en cuenta las clases identificada en la actividad ASI4 se realiza el modelo de clases para cada subsistema. Las tareas que se llevan a cabo en esta actividad son:

- ASI5.1 Identificación de Responsabilidades y Atributos: se hace uso del Diagrama de Clases y Diagrama de Transición de Estados para completar el Modelo de Clases de Análisis y el Comportamiento de Clases de Análisis.
- ASI5.2 Identificación de Asociaciones y Agregaciones: se apoya en el Diagrama de Clases y aporta información al Modelo de Clases de Análisis.
- ASI5.3 Identificación de Generalizaciones: se apoya en el Diagrama de Clases y aporta información al Modelo de Clases de Análisis.

2. DISEÑO ORIENTADO A OBJETOS

El objetivo del proceso "Diseño del Sistema de Información (DSI)" es la definición de la arquitectura del sistema y del entorno tecnológico que le va a dar soporte, junto con la especificación detallada de los componentes del sistema de información.

A partir de dicha información se generan las especificaciones de construcción relativas al propio sistema, así como la descripción técnica del plan de pruebas, la definición de requisitos de implantación y el diseño de los procedimientos de migración y carga inicial.



2.1 ACTIVIDAD DSI1: DEFINICIÓN DE LA ARQUITECTURA DEL SISTEMA

Se establece un particionamiento físico del sistema de información, así como su organización en subsistemas de diseño, la especificación del entorno tecnológico, y sus requisitos de operación, administración, seguridad y control de acceso. Se completan los catálogos de requisitos y normas.

2.2 ACTIVIDAD DSI2: DISEÑO DE LA ARQUITECTURA DE SOPORTE

Incluye el diseño detallado de los subsistemas de soporte, el establecimiento de las normas y requisitos propios del diseño y construcción, así como la identificación y definición de los mecanismos genéricos de diseño y construcción. En paralelo se realiza el diseño detallado del sistema de información a través de las actividades DSI3 y DSI4.

2.3 ACTIVIDAD DSI3: DISEÑO DE CASOS DE USO REALES

Con el diseño detallado del comportamiento del sistema de información para los casos de uso, el diseño de la interfaz de usuario y la validación de la división en subsistemas.

2.4 ACTIVIDAD DSI4: DISEÑO DE CLASES

Con el diseño detallado de cada una de las clases que forman parte del sistema, sus atributos, operaciones, relaciones y métodos, y la estructura jerárquica del mismo.

2.5 ACTIVIDAD DSI6: DISEÑO FÍSICO DE DATOS

Una vez finalizado el modelo de clase se comienza con el diseño físico en esta actividad. Incluye el diseño y optimización de las estructuras de datos del sistema, así como su localización en los nodos de la arquitectura propuesta.

3. TÉCNICAS DE ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS

3.1 CASOS DE USO

Los casos de uso proporcionan un modo claro y preciso de comunicación entre cliente y desarrollador. Proporcionan una visión de "caja negra" del sistema, es decir, cómo aparece el sistema desde el exterior sin necesidad de entrar en los detalles de su construcción. Los objetivos de los casos de uso son:

- Capturar los requisitos funcionales del sistema y expresarlos desde el punto de vista del usuario.
- Guiar todo el proceso de desarrollo del sistema de información.

Un caso de uso es una secuencia de acciones realizadas por el sistema, que producen un resultado observable y valioso para un usuario en particular, es decir, representa el comportamiento del sistema. Se pueden completar incluyendo precondiciones y poscondiciones del sistema.

Los diagramas de casos de uso presentan dos tipos de elementos fundamentales:

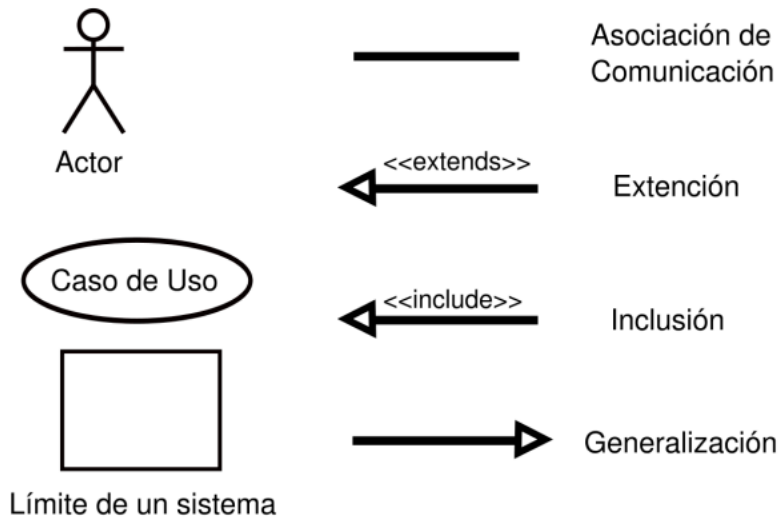
- Actores: es algo o alguien que se encuentra fuera del sistema y que interactúa con él.
- Casos de uso: representa el comportamiento que ofrece el sistema de información desde el punto de vista del usuario

Además de estos elementos, un diagrama de casos de uso presenta relaciones, entre actores y casos de uso o entre casos de uso. La relación entre un actor y un caso de uso es una relación de comunicación, y normalmente el actor aporta información para la realización del caso de uso o recibe información como resultado del mismo.

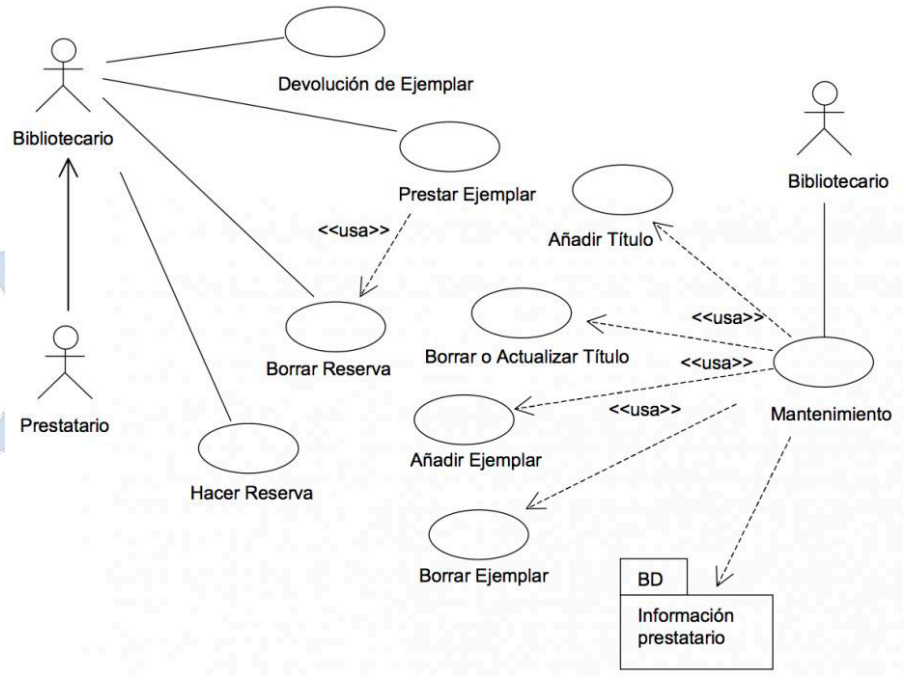
La relación entre casos de uso es una relación unidireccional, y puede ser de dos tipos: de uso o inclusión, o extensión.

- La relación de uso o inclusión se utiliza cuando se quiere reflejar un comportamiento común entre varios casos de uso.
- La relación de extensión se utiliza cuando se quiere reflejar un comportamiento opcional de un caso de uso.

El diagrama de casos de uso es un grafo de actores, casos de uso y relaciones, que hace uso de la notación que se muestra a continuación.



Un ejemplo de una aplicación que se encarga de la gestión de los préstamos y reservas de libros y revistas en una biblioteca.



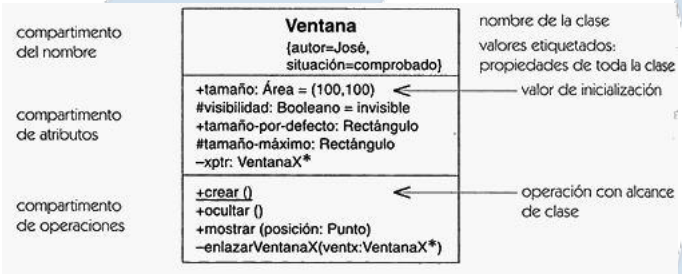
3.2 DIAGRAMA DE CLASES

Su principal objetivo es la representación de los aspectos estáticos del sistema, utilizando diversos mecanismos de abstracción (clasificación, generalización, agregación). El diagrama recoge las clases de objetos y sus asociaciones, pero en ningún caso los comportamientos temporales de las clases.

Sus elementos básicos son: clases, relaciones e interfaces.

3.2.1 CLASES E INTERFACES

Una clase describe un conjunto de objetos con propiedades (atributos) similares y un comportamiento común. Los objetos son instancias de las clases. Aunque no suele haber un procedimiento inmediato para identificar las clases, estas suelen corresponderse con sustantivos que hacen referencia al ámbito del sistema de información y que se encuentran en los documentos de las especificaciones de requisitos y los casos de uso.



Una clase se representa como una caja, separada en tres zonas por líneas horizontales:

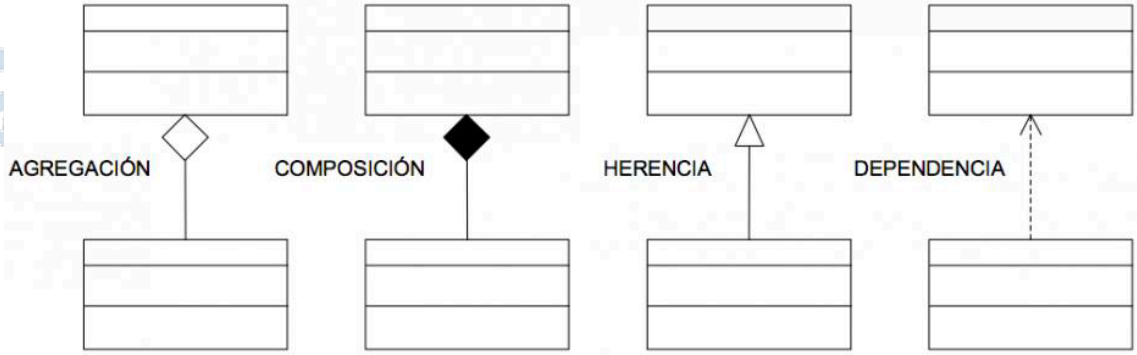
- Zona superior: se muestra el nombre de la clase y propiedades generales como el estereotipo o propiedades de clase.
- Zona central: contiene una lista de atributos, incluyendo para cada uno de ellos, el nombre, su tipo, valor por defecto, visibilidad, etc.
- Zona inferior: contiene una lista de operaciones o métodos, donde se indica el nombre del método, visibilidad y atributos esperados.

Las interfaces son especificaciones semánticas de un conjunto de operaciones de una clase o paquete que son visibles desde otras clases o paquetes. Se representan igual que las clases, incluyendo el estereotipo <<Interface>>, y con la zona de atributos vacía u omitida.

3.2.2 RELACIONES

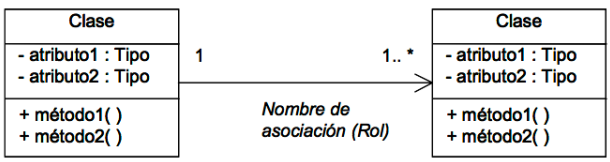
Los tipos más importantes de relaciones estáticas entre clases son:

- Asociación: representan enlaces entre objetos o instancias de clases del tipo más general y denotan dependencia semántica. Se pueden presentar elementos adicionales como el rol o nombre de la asociación y la multiplicidad que describe la cardinalidad de la relación.
- Herencia: representan jerarquías de generalización y especialización. Se representan con una línea continua con una flecha hueca en el extremo que apunta a la superclase.
- Agregación: se trata de un tipo especial de relación jerárquica entre un objeto que representa la totalidad de ese objeto y las partes que lo componen. Se representa con un rombo hueco en la clase cuya instancia es una agregación de las instancias de la otra.
- Composición: es una forma de agregación donde la relación de propiedad es más fuerte. En este caso se representa con un rombo lleno.
- Dependencia: indica que una clase requiere de otra para proporcionar alguno de sus servicios. Se representa con una línea discontinua con una flecha apuntando a la clase cliente.



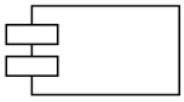
Las diferentes propiedades de la relación se pueden representar con la siguiente notación:

- Multiplicidad: puede ser un número concreto, un rango o un conjunto de números.
- Orden: se puede especificar si las instancias guardan un orden mediante la palabra clave {ordered}.
- Navegabilidad: con una flecha que apunte de una clase a otra se indica el sentido de la navegación.
- Rol o nombre de la asociación: se coloca junto al extremo de la línea que está unida a una clase.

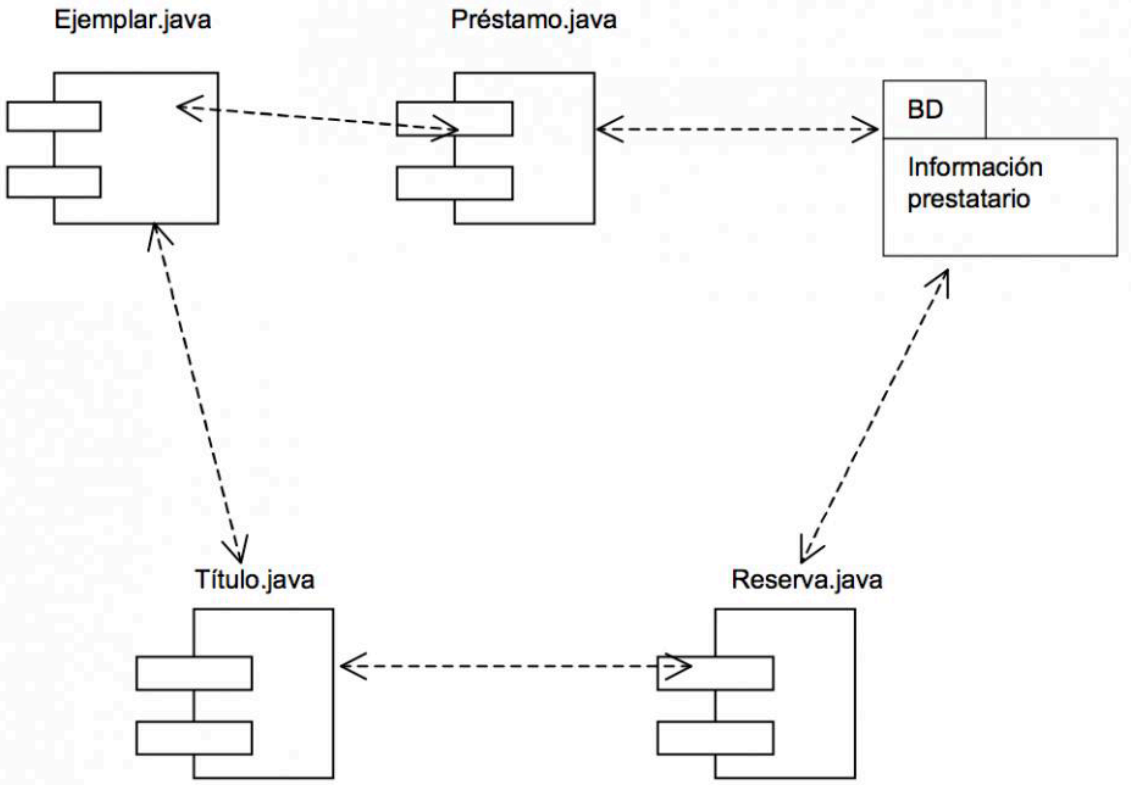


3.3 DIAGRAMA DE COMPONENTES

Proporciona una visión física de la construcción del SI. Muestra la organización de los componentes software, sus interfaces y las dependencias entre ellos. Un componente es un módulo de software que puede ser código fuente, código binario, un ejecutable, o una librería con una interfaz definida. Además, se representan las dependencias entre componentes o entre un componente y la interfaz de otro.

Componente: se representa como un rectángulo, con dos pequeños rectángulos superpuestos perpendicularmente en el lado izquierdo.	
Interfaz: se representa con un pequeño círculo situado junto al componente que lo implementa y unido a él por una línea continua.	
Relación de dependencia: se representa mediante una línea discontinua con una flecha que apunta al componente o interfaz que provee el servicio al otro.	

Ejemplo de un sistema encargado de la gestión de préstamos y reservas de libros y revistas en una biblioteca.



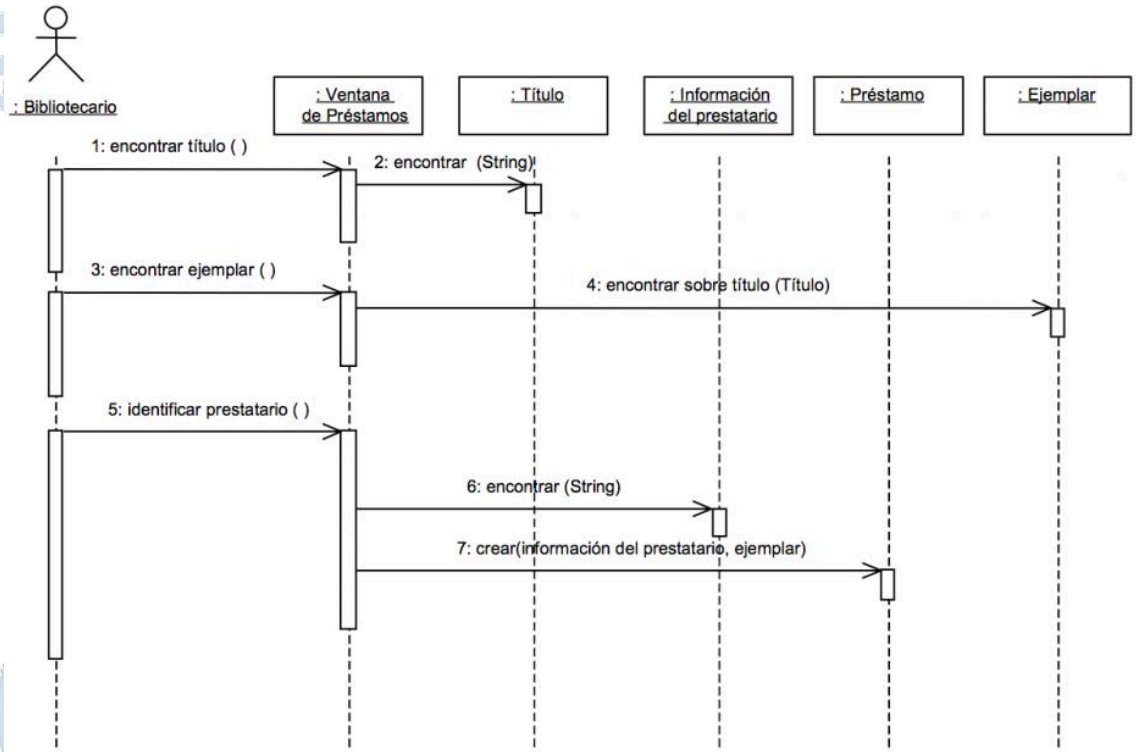
3.4 DIAGRAMA DE INTERACCIÓN

El objetivo de esta técnica es describir el comportamiento dinámico del SI mediante el paso de mensajes entre los objetos del mismo. Describe en detalle un determinado escenario de un caso de uso. Los elementos que lo componen son:

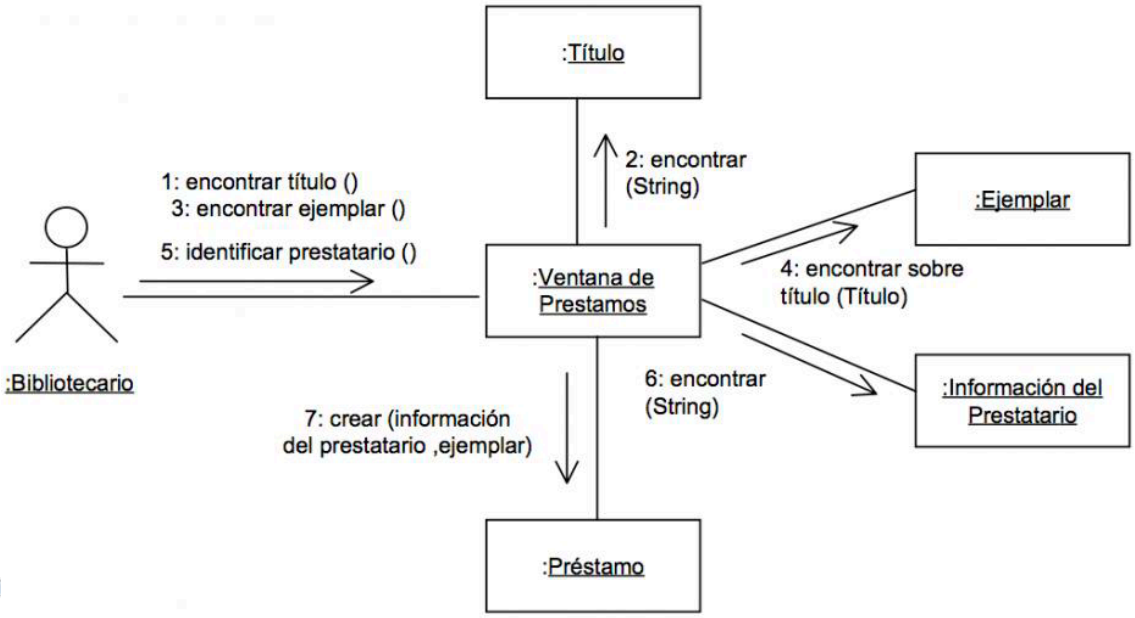
- Objetos, que representan a cada entidad que tiene un estado, un comportamiento e identidad.
- Mensajes, que serán las comunicaciones entre los objetos.

Hay dos tipos de diagramas de interacción, los diagramas de secuencia y los diagramas de colaboración. Los primeros muestra de forma explícita la secuencia de los mensajes intercambiados por objetos. Mientras que los diagramas de colaboración muestran de forma más clara cómo colaboran los objetos, es decir, con qué otros objetos tienen vínculos o intercambia mensajes un determinado objeto.

Ejemplo de un diagrama de secuencia para el caso de uso "Prestar un ejemplar de una aplicación encargada de los préstamos y reservas de una biblioteca":



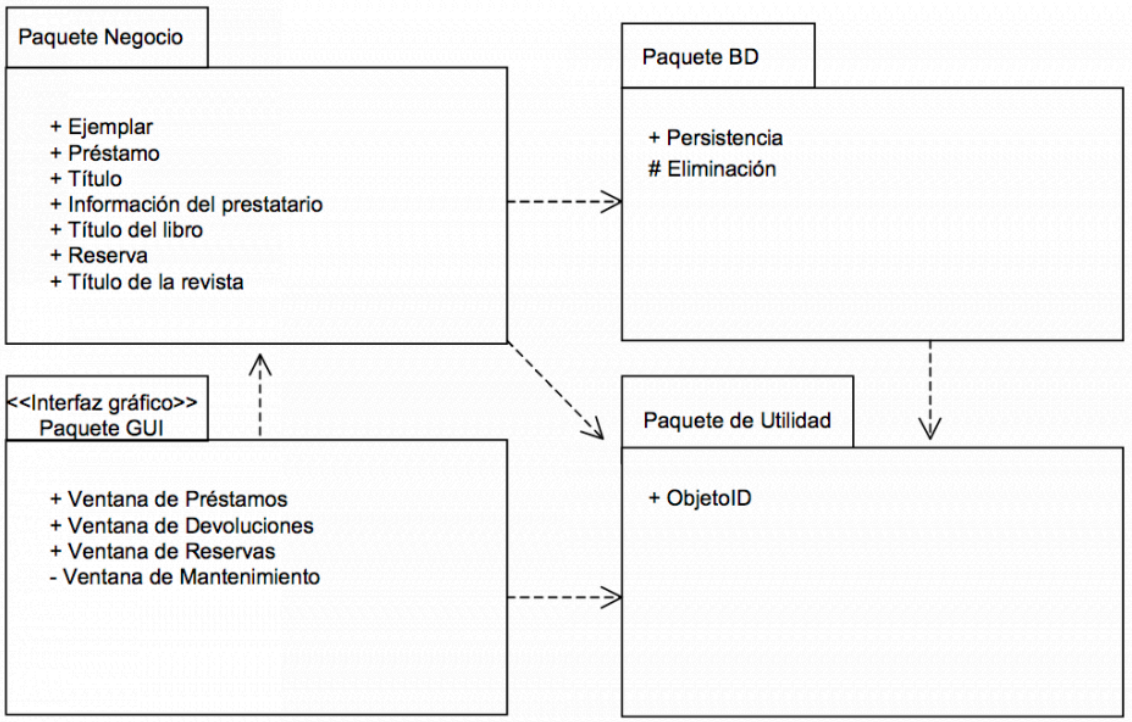
Ejemplo de diagrama de colaboración para el mismo caso de uso:



3.5 DIAGRAMA DE PAQUETES

Obtienen una visión del SI organizándolo en subsistemas, agrupando los elementos del análisis, diseño o construcción y detallando las relaciones de dependencia entre ellos. El mecanismo de agrupación se denomina Paquete.

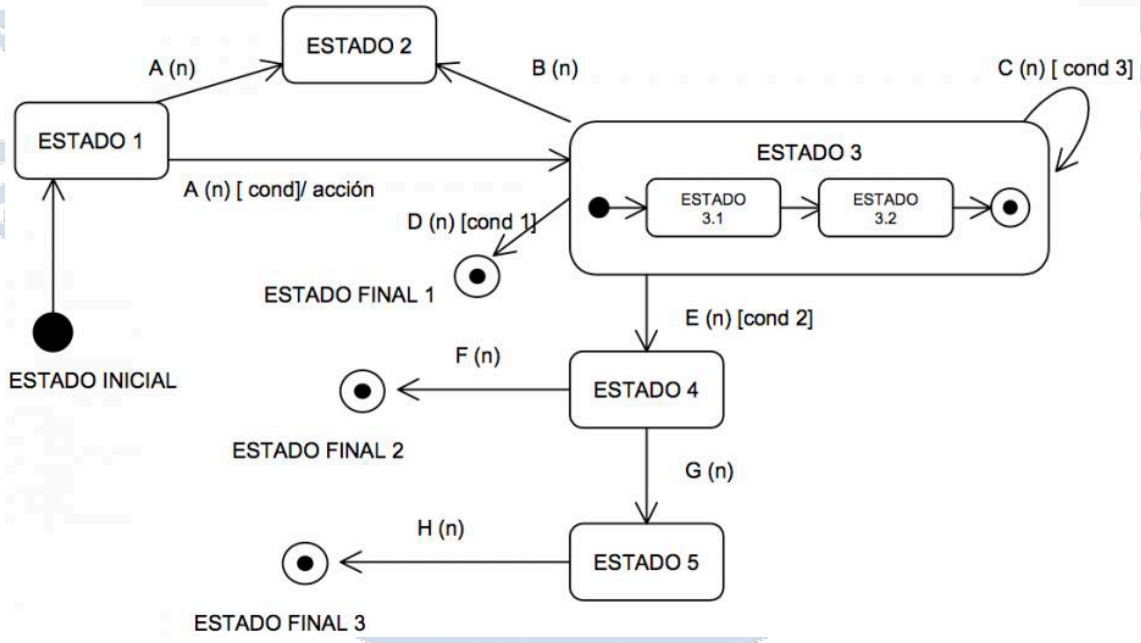
En MÉTRICA v3, el diagrama de paquetes es tratado como una técnica aparte, que se aplica en el análisis para la agrupación de casos de uso o de clases, en el diseño de la arquitectura para la agrupación de clases y en el diseño detallado para agrupar componentes.



3.6 DIGRAMA DE TRANSICIÓN DE ESTADOS

Muestra el comportamiento dependiente del tiempo de un sistema de información. Representa los estados que puede tomar un componente o un sistema y muestra los eventos que implican el cambio de un estado a otro. Los elementos principales son los estados y las posibles transiciones entre ellos.

- El estado de un componente o sistema representa algún comportamiento que es observable externamente y que perdura durante un periodo de tiempo finito. Se representa mediante un rectángulo con las esquinas redondeadas.
- Una transición es un cambio de estado producido por un evento y refleja los posibles caminos para llegar a un estado final desde un estado inicial. Se representa mediante una flecha continua que une dos estados. Junto a una etiqueta que debe tener al menos el nombre del evento que provoca la transición.



III. PROPUESTA DIDÁCTICA PARA EL DESARROLLO DEL TEMA EN EL MARCO ESCOLAR

El tema versa sobre aspectos fundamentales de programación. Dentro del sistema educativo dichos contenidos están desarrollados legislativamente en los ciclos formativos de la familia profesional de Informática y Comunicaciones:

- CFGS: Desarrollo de aplicaciones multiplataforma

  - Real Decreto 450/2010.
  - Orden de 16 de junio de 2011.

- CFGS: Desarrollo de aplicaciones web
  - Real Decreto 686/2010.
  - Orden de 16 de junio de 2011.

Su aplicación en el aula se realiza en los módulos profesionales:

- Programación (CFGS DAM y DAW)
- Entornos de desarrollo (CFGS DAM y DAW)

IV. CONCLUSIÓN

El análisis y diseño orientado a objeto no está tan estudiado como el estructurado. Las técnicas son más inconexas, y no hay un enfoque tan sistematizado como para la programación estructurada, aunque la reutilización de las librerías y la proliferación de los IDE ha hecho que el desarrollo del paradigma orientado a objeto se dispare.

Hasta hace pocos años había una gran cantidad de notaciones diferentes, para las representaciones y el modelado, hoy en día se ha impuesto UML, lenguaje gráfico de modelado para visualizar, especificar, y construir y documentar las especificaciones de un sistema software.

La metodología MÉTRICA v3 tiene en cuenta la mayoría de las técnicas que contempla UML 1.2

V. BIBLIOGRAFÍA

- Cabrera, G. (2005). Análisis y diseño detallado de aplicaciones informáticas de gestión. McGraw-Hill.
- Piattini, M., Calvo-Manzano, JA., Cervera, J. & Fernández, L. (1996). Análisis y diseño detallado de aplicaciones informáticas de gestión. Ra-Ma.
- Pressman, R. (2005). Ingeniería del software. Un enfoque práctico, 5ª Ed. McGraw-Hill.
- Sommerville, I. (2005). Ingeniería del software. Pearson-Prentice Hall.