Proyecto Certificador de Desarrollo de SW 1

Tema Nº5:

Metodología RUP y UML

Indicador de logro Nº5:Aplica la metodología RUP y la Notación Gráfica del Lenguaje de Modelado Unificado UML, en el desarrollo de software.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº5:**

METODOLOGÍAS RUP y UML

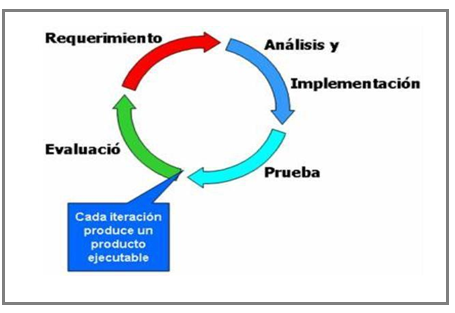
OBJETO DE LA EXPERIENCIA

Conoce la metodología RUP y su aplicación en el proceso de desarrollo de software en una Organización, utilizando el lenguaje unificado de modelado UML.

MARCO TEÓRICO

1. Metodología de desarrollo de software

La metodología RUP, abreviatura de Rational Unified Process (Proceso Unificado Racional), es un proceso propietario de la ingeniería de software creado por Rational Software, adquirida por IBM, ganando un nuevo nombre IRUP que ahora es una abreviatura Rational Unified Process y lo que es una marca en el área de software, proporcionando técnicas que deben seguir los miembros del equipo de desarrollo de software con el fin de aumentar su productividad en el proceso de desarrollo.



La metodología RUP utiliza el enfoque de la orientación a objetos en su desarrollo y está diseñado y documentado el uso de la notación UML (Unified Modeling Language), para ilustrar los procesos en acción. Utiliza técnicas y prácticas probadas comercialmente.

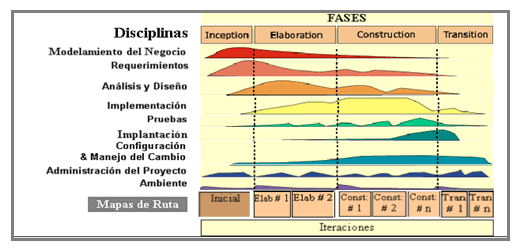
Es un proceso considerado pesado y preferentemente aplicable a grandes equipos de desarrollo y grandes proyectos, pero el hecho de que es ampliamente personalizable que permite adaptarse a proyectos de cualquier escala.

Para la gestión del proyecto, la metodología RUP proporciona una solución disciplinada como las tareas y responsabilidades señaladas dentro de una organización de desarrollo de software.

RUP es, en sí, un producto de software. Es modular y automatizado, y toda su metodología se apoya en varias herramientas de desarrollo integradas y vendidos por IBM a través de sus «Suites Racional»

1. Metodología RUP – Disciplinas y Fases

Hasta ahora estas líneas guía son generales, para ser adherido a pasar por la vida de un ciclo de proyecto. Las fases (ver figura abajo) indican el énfasis se da en el proyecto en un instante dado. Para capturar la dimensión temporal de un proyecto, RUP divide el proyecto en cuatro fases diferentes:



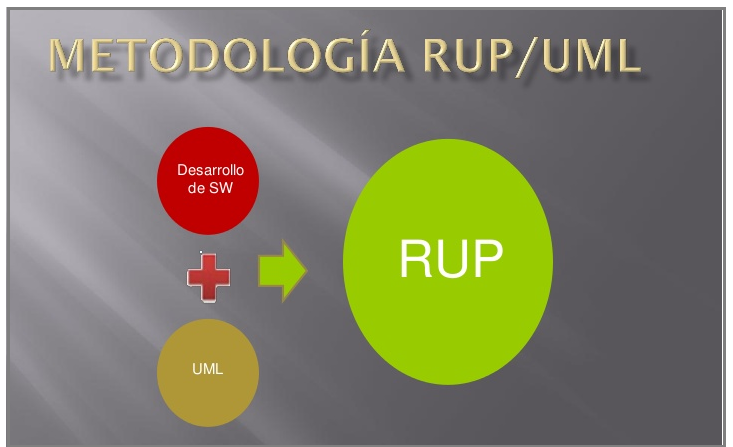
**Fases de la Metodología RUP**

* Inicio: Define el modelo del negocio y el alcance del proyecto
* Elaboración: Énfasis en la arquitectura
* Construcción: Énfasis en el desarrollo
* Transición: Énfasis en la aplicación

RUP, se basa en la 4P

* Personas
* Diseño
* Producto
* Proceso

Todas las fases generan artefactos. Estos serán utilizados en la siguiente fase y documentar el proyecto y permite un mejor seguimiento.



**FASE DE INICIO**

La fase de inicio contiene los flujos de trabajo necesarios para el acuerdo de las partes interesadas – interesados – con los objetivos, la arquitectura y la planificación del proyecto. Si estos actores tienen un buen conocimiento, no será necesario analizar. De lo contrario, se requiere un análisis más elaborado.

En esta etapa, los requisitos esenciales del sistema se transforman en los casos de uso. El objetivo no es para cerrarlas en absoluto, sino sólo las que sean necesarias para dar forma a la opinión.

El paso es generalmente corto y se utiliza para definir si es factible para continuar con el proyecto y definir los riesgos y el coste de la última. Un prototipo se puede hacer para que el cliente apruebe. Como cita el RUP, lo ideal es realizar iteraciones, las cuales deben estar bien definida en cuanto a su importe y objetivos.

**FASE DE ELABORACIÓN**

La preparación será para el diseño del sistema, como complemento de la encuesta y / o documentación de casos de uso, frente a la arquitectura del sistema, revisar el modelo de negocio para el proyecto e iniciar la versión del manual del usuario. Uno debe aceptar:

¿Descripción del producto (aumento + integración) es estable? ¿El plan del proyecto es fiable?; Los costos son elegibles?

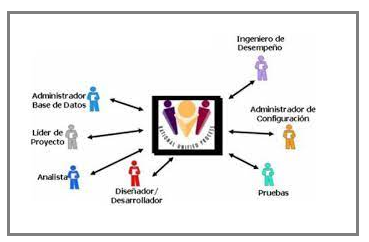
**FASE DE CONSTRUCCIÓN**

En la fase de construcción, el desarrollo físico del software se inicia, códigos de producción, pruebas alfa. pruebas beta se llevaron a cabo al inicio de la fase de transición.

Se debe aceptar las pruebas, procesos estables y de prueba, y el código del sistema son «línea de base».

**FASE DE TRANSICIÓN**

En esta fase es la entrega («despliegue») de software, que se lleva a cabo el plan de despliegue y entrega, el seguimiento y la calidad del software. Productos (lanzamientos, las versiones) se van a entregar, y coloque la satisfacción del cliente. Esta etapa también se lleva a cabo la formación de los usuarios.

****

DISCIPLINAS DE METODOLOGÍA RUP

SEÍS DISCIPLINAS DE INGENIERÍA DE SOFTWARE

**DISCIPLINA DEL MODELADO DE NEGOCIO**

Las organizaciones dependen cada vez más de los sistemas de TI, por lo que es imperativo que los ingenieros de sistemas de información saben cómo se integran las aplicaciones en el desarrollo de la organización. Las empresas invierten en TI, que entienden la ventaja competitiva del valor añadido por la tecnología.

El objetivo de modelado de negocio es establecer primero una mejor comprensión y comunicación entre ingeniería de negocios y la ingeniería de software.

Comprender el negocio significa que los ingenieros de software deben entender la estructura y la dinámica de la empresa objetivo (el cliente), los problemas actuales de la organización y las posibles mejoras. También deben asegurar una comprensión común de la organización de destino entre los clientes, usuarios finales y desarrolladores.

El modelado de negocios explica cómo describir la visión de una organización en la que se implementará el sistema y cómo utilizar esta visión como base para describir los procesos, funciones y responsabilidades.

**ANALISIS Y DISEÑO DE LA DISCIPLINA DE DISEÑO**

El propósito del análisis y diseño es mostrar cómo se llevará a cabo el sistema. El objetivo es construir un sistema que:

Ejecutar en un entorno de ejecución específica, las tareas y las funciones especificadas en las descripciones de casos de uso

Satisfacer todas sus necesidades

Es fácil de mantener cuando no son cambios en los requisitos funcionales, los resultados del proyecto en un modelo de análisis y diseño tiene opcionalmente un modelo de análisis. El modelo de diseño sirve como una abstracción del código fuente, es decir, el proyecto sirve como modelo de «retroalimentación» de la forma en que el código fuente está estructurado y escrito.

El modelo de diseño consta de clases de diseño estructurados en paquetes y subsistemas con interfaces bien definidas, en representación de lo que se convertirá componentes de la aplicación. También contiene descripciones de cómo los objetos de estas clases colaboran para llevar a cabo el diseño de casos de uso.

**DISCIPLINA DE IMPLEMENTACIÓN**

Los efectos de la aplicación son:

* Para configurar el código de la organización en términos de subsistemas de aplicación organizados en capas
* Para llevar a cabo las clases y objetos en términos de componentes (archivos de código fuente, binarios, ejecutables, etc.)
* Para probar los componentes desarrollados como unidades
* Incorporar los resultados producidos por los ejecutores individuales (o equipos), en un sistema ejecutable

Los sistemas se logran a través de los componentes de la aplicación. El proceso describe cómo reutilizar componentes existentes o implementar nuevos componentes con responsabilidades bien definidas, haciendo que el sistema sea más fácil de mantener y aumentar las posibilidades de reutilización.

**PRUEBA DE DISCIPLINA**

Los fines de disciplina prueba son:

Comprobar la interacción entre los objetos  
Comprobar la correcta integración de todos los componentes de software  
Compruebe que todos los requisitos han sido ejecutadas correctamente  
Identificar y asegurar que los defectos se tratan antes de la implementación de software  
Asegúrese de que todos los defectos son corregidos, revisados y cerrados

El Rational Unified Process propone un enfoque iterativo, lo que significa que debería estar probando el proyecto en su totalidad. Esto le permite encontrar defectos tan pronto como sea posible, lo que reduce drásticamente el costo de reparar el defecto.

Las pruebas se realizan a lo largo de cuatro dimensiones de la calidad: fiabilidad, funcionalidad, rendimiento de las aplicaciones y el rendimiento del sistema. Para cada una de estas dimensiones de la calidad, el proceso se describe cómo a pasar la prueba de la planificación, diseño, implementación, ejecución y evaluación.

**DISCIPLINA DE IMPLANTACIÓN**

El propósito del despliegue es producir lanzamientos de productos exitosos y entregar el software a los usuarios finales. Abarca una amplia gama de actividades, incluyendo la producción de versiones de software externos, el envase de aplicaciones de software y de negocios, distribución de software, instalación de software y proporcionar ayuda y asistencia a los usuarios.

Aunque las actividades de despliegue se centran principalmente en torno a la transición, muchas de las actividades se deben incluir en las etapas anteriores para preparar la aplicación, al final de la fase de construcción. Los procesos («flujos de trabajo») de «Implementación y Medio Ambiente» RUP contienen menos detalles que otros flujos de trabajo.

TRES DISCIPLINAS SOPORTE/SERVICIO DE LA METODOLOGÍA RUP

**DISCIPLINA DEL MEDIO AMBIENTE**

El medio ambiente se centra en las actividades necesarias para configurar el proceso para un proyecto. En él se describen las actividades necesarias para desarrollar directrices para apoyar un proyecto.

La propuesta de las actividades ambientales es proporcionar a los procesos de organización de desarrollo de software y herramientas que apoyarán al equipo de desarrollo. Si los usuarios no entienden que RUP es un marco de proceso, pueden percibirlo como un proceso engorroso y costoso. Sin embargo, un concepto clave en las RUP era la lata RUP y, a menudo debe ser refinado.

Esto se hizo inicialmente de forma manual, es decir, un documento «caso del complejo» escrito que especifica el proceso de refinado para ser utilizado. Más tarde, IBM producto Compositor Método Racional fue creado para ayudar a hacer este paso simple, ingenieros de proceso y los directores de proyectos podría más fácilmente personalizar el RUP para sus necesidades del proyecto.

Muchas de las variaciones posteriores de las regiones ultra periféricas, incluidas OpenUP / Basic, el código abierto, versión ligera de RUP, se presentan ahora como procesos separados en su propio derecho, y atender a los diferentes tipos y tamaños de proyectos, tendencias y tecnologías de desarrollo de software.

Históricamente, como el RUP es a menudo la medida para cada proyecto por un experto en procesos RUP, el éxito global del proyecto puede ser algo dependiente de la capacidad de esta persona.

**CONFIGURACIÓN DE LA DISCIPLINA Y DE LA GESTIÓN**

La disciplina de la gestión del cambio en el negocio con RUP abarca tres tratamientos específicos: configuración, solicitudes de cambio, y el estado y de medida.

La gestión de configuración: gestión de la configuración es responsable de la estructuración sistemática de productos. Los artefactos tales como documentos y modelos necesitan estar bajo el control de versiones y estos cambios deben ser visibles. También realiza un seguimiento de las dependencias entre los artefactos de manera que todos los artículos relacionados se actualizan cuando se realizan cambios

La gestión del cambio de solicitud: Durante el proceso de desarrollo del sistema con muchos artefactos que existen varias versiones. El CRM realiza un seguimiento de los cambios propuestos

El estado y la medición de la gestión: las solicitudes de cambio tienen los estados: nuevo, conectado, aprobado, asignado y completa. La solicitud de cambio también tiene atributos como la causa raíz, o la naturaleza (como el incumplimiento y recuperación), prioridad, etc.

Estos estados y atributos se almacenan en la base de datos para producir informes útiles sobre el progreso del proyecto. Racional también tiene un producto para mantener las solicitudes de cambio llamados ClearQuest. Esta actividad tiene procedimientos a seguir.

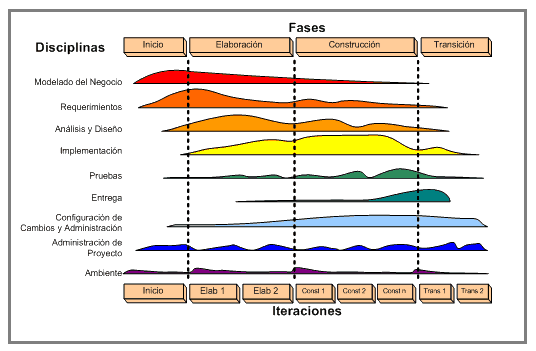
**PROYECTO DE GESTIÓN DE LA DISCIPLINA**

La planificación del proyecto en el RUP se produce en dos niveles. Hay un grano fino o planes de fase que describe el proyecto en su totalidad, y un número de alta granularidad o planes de iteración que describe los pasos iterativos.

Este curso se centra principalmente en los aspectos importantes de un proceso de desarrollo iterativo: La gestión de riesgos; La planificación de un proyecto iterativo a través del ciclo de vida y para una iteración en particular; Y el proceso de seguimiento de un proyecto iterativo, la métrica. Sin embargo, esta disciplina de RUP no pretende cubrir todos los aspectos de la gestión de proyectos.

Por ejemplo, no cubre cuestiones tales como:

* Gestión de personas: contratación, formación, etc.
* Presupuesto general: definición, asignación, etc.
* Gestión de contratos: con los proveedores, clientes, etc.

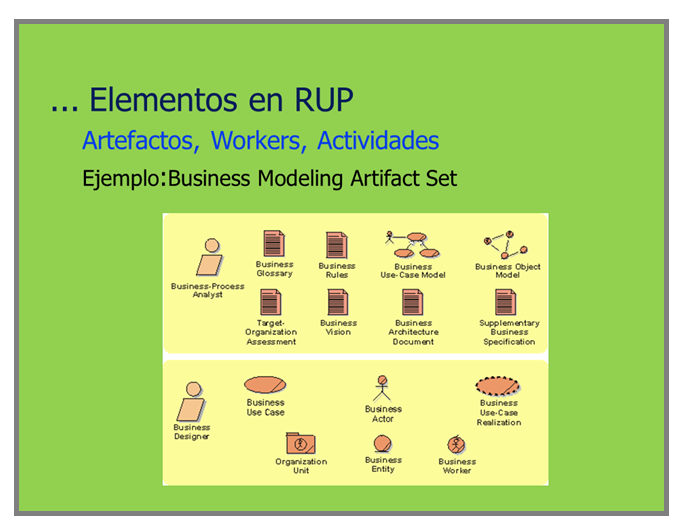


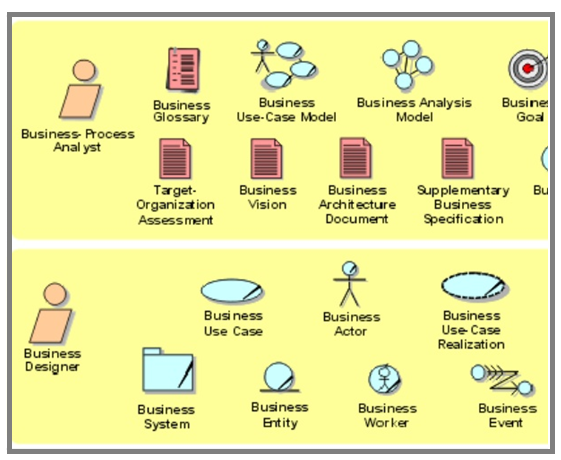
ARTEFACTOS RUP

En RUP en cada una de sus fases realizan una serie de artefactos para saber mejor la función y estructura de un programa.

Un artefacto puede ser:

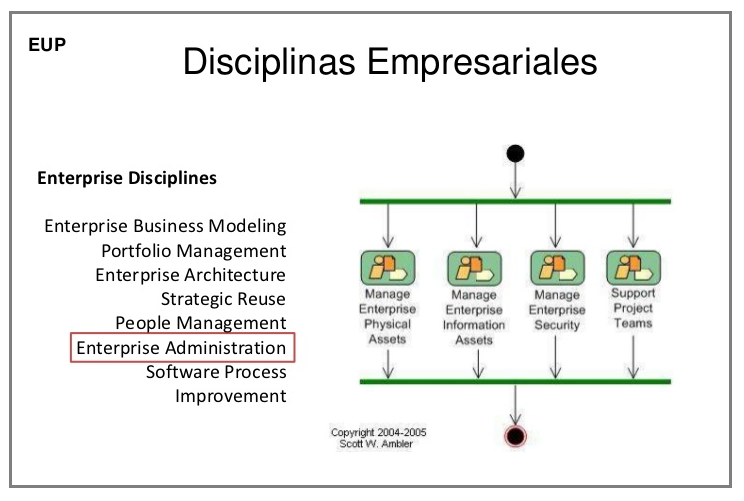
* Un documento: como un Caso de Negocio o un documento de la arquitectura del Software.
* Un modelo: como un modelo de caso de uso.
* Un elemento de un modelo: como una sola clase de todo el Diagrama de Clases.

****



1. Metodología EUP

Significa "Proceso unificado de empresa". EUP es una metodología de desarrollo de software que ayuda a las empresas a crear software de manera estructurada y organizada. Es una extensión de la [Proceso racional unificado (RUP)](https://techlib.net/definition/rup.html), agregando dos nuevas fases de desarrollo: Producción y Retiro. Como el RUP incluye cuatro fases, el EUP consta de seis fases:



**COMIENZO**

Se plantea la idea del proyecto. El equipo de desarrollo determina si vale la pena seguir el proyecto y qué recursos se necesitarán.

**ELABORACIÓN**

La arquitectura del proyecto y los recursos necesarios se evalúan más a fondo. Los desarrolladores consideran posibles aplicaciones del software y los costos asociados con el desarrollo

**CONSTRUCCIÓN**

El proyecto está desarrollado y completado. El software está diseñado, escrito y probado.

**TRANSICIÓN**

El software se lanza al público. Los ajustes o actualizaciones finales se realizan en función de los comentarios de [los usuarios finales](https://techlib.net/definition/enduser.html).

**PRODUCCIÓN**

El software se mantiene útil y productivo después de ser lanzado al público. Los desarrolladores se aseguran de que el producto continúe ejecutándose en todos los sistemas compatibles y el personal de asistencia proporciona asistencia a los usuarios.

**JUBILACIÓN**

El producto se retira de la producción, a menudo llamado "desmantelamiento". Puede ser reemplazado o simplemente ya no es compatible. El lanzamiento de una nueva versión de software a menudo coincide con la fase de retiro de una versión anterior.

1. Notación Gráfica – UML

**UML**, Es un lenguaje gráfico que permite visualizar, especificar, construir y documentar un sistema de información.

**UML** (Unified Modeling Language), es el lenguaje de modelado de software más conocido y utilizado en la actualidad, normado por la OMG (Object Magament Group), [www.omg.org](http://www.omg.org), UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.



Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar, también metodologías ágiles como SCRUM.

CREDAORES DE UML

El lenguaje UML comenzó a gestarse en octubre de 1994, cuando Rumbaugh se unió a la compañía Rational fundada por Booch (dos destacados investigadores en el área de metodología del software).

El objetivo de ambos era unificar dos métodos que habían desarrollado: el método Booch y el OMT (Object Modelling Tool). El primer borrador apareció en octubre de 1995. En esa misma época otro conocido investigador, Jacobson, creador de OOSE (Object Oriented Software Engineer) se unió a Rational y se incluyeron ideas suyas.

* James Rumbaugh
* Ivar Jacobson
* Grady Booch

Estas tres personas son conocidas como los “tres amigos”. Además, este lenguaje se abrió a la colaboración de otras empresas para que aportaran sus ideas. Todas estas colaboraciones condujeron a la definición de la primera versión de UML.

EMPRESAS PARTICIPANTES

IBM

ORACLE

MICROSOFT

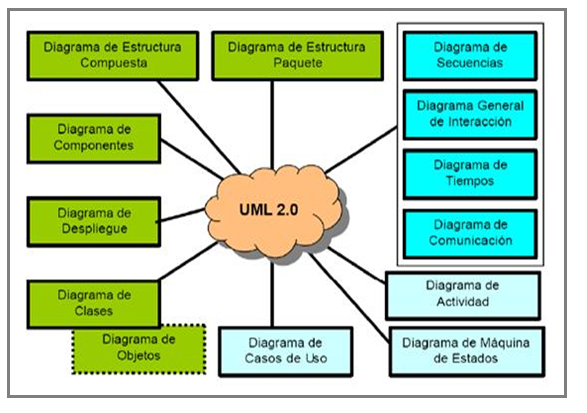
HP

CORBA

1. Taxonomía de Diagramas UML 2.0

Uno de los objetivos principales de la creación de UML era posibilitar el intercambio de modelos entre las distintas herramientas CASE orientadas a objetos del mercado. Para ello era necesario definir una notación y semántica común.

**VERSIONES:** UML 1.X, 2.X, 2.5 (2012, una versión en proceso, que fue formalmente liberada en junio de 2015), 3.x (Evolución de la versión 2.x)

~~~~

TIPOS DE DIAGRAMAS UML

1. **Por su Estructura**

Diagrama de Clases, Diagrama de Objetos, Diagrama de Componentes, Diagrama de Estructura Compuesta, Diagrama de Paquetes, Diagrama de Despliegue.

1. **Por su Comportamiento**

Diagrama de Casos de Uso, Diagrama de Actividades, Diagrama de Estados.

1. **Diagramas de Interacción**

Diagrama de Secuencias, Diagrama de Comunicación, Diagramas de Tiempo, Diagrama de interacción.

Por su Estructura

**Diagrama de clase:** Describe los diferentes tipos de objetos en un sistema y las relaciones existentes entre ellos. Dentro de las clases, muestra las propiedades y operaciones, así como las restricciones de las conexiones entre objetos.

**Diagrama de objetos:** (También llamado Diagrama de instancias), plantilla de los objetos en un sistema en un momento del tiempo.

**Diagrama de paquetes:** Muestra la estructura y dependencia entre paquetes, los cuales permiten agrupar elementos (no solamente clases) para la descripción de grandes sistemas.

**Diagrama de despliegue:** Muestra la relación entre componentes o subsistemas software y el hardware donde se despliega o instala.

**Diagrama de estructura compuesta:** Descompone jerárquicamente una clase mostrando su estructura interna.

**Diagrama de componentes:** Muestra la jerarquía y relaciones entre componentes de un sistema software.

Por su Comportamiento

**Diagrama de casos de uso**: Permite capturar los requerimientos funcionales de un sistema.

**Diagrama de estado:** Permite mostrar el comportamiento de un objeto a lo largo de su vida.

**Diagrama de actividad:** Describe la lógica de un procedimiento, un proceso de negocio o workflow.

Diagrama de Interacción

Describen cómo los grupos de objetos colaboran para producir un comportamiento. Son los siguientes:

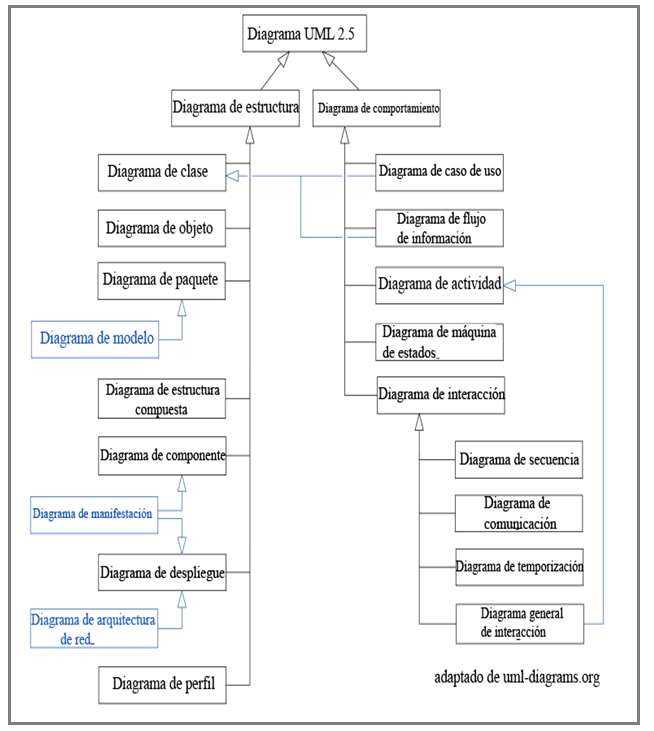
**Diagrama de secuencias:** Muestra los mensajes que son pasados entre objetos en un escenario.

**Diagrama de comunicación:** Muestra las interacciones entre los participantes haciendo énfasis en la secuencia de mensajes.

**Diagrama de colaboración:** Solamente en UML 1.X) Muestra las interacciones organizadas alrededor de los roles.

**Diagrama de (visión de conjunto o resumen de)** interacción: Se trata de mostrar de forma conjunta diagramas de actividad y diagramas de secuencia.

**Diagrama de tiempo:** Pone el foco en las restricciones temporales de un conjunto de objetos

~~~~

**Actividad:**

Ingresa a la plataforma virtual, luego desarrolla la siguiente actividad propuesta:

1. **CUESTIONARIO TÉCNICO**
2. ¿Cuál es la importancia y aplicación de la metodología RUP?
3. ¿Cómo la metodología RUP asigna de forma disciplinada tareas y responsabilidades?
4. ¿Cuándo recomendaría usted utilizar RUP, en proyectos informáticos?
5. ¿Para qué sirve UML, en el desarrollo de software?
6. ¿Cuantos tipos de diagrama UML existen y dar un ejemplo del tipo de diagramas?
7. ¿Qué es un artefacto en el desarrollo de software?

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_