CICLO DE VIDA – GUIA 1

Un ciclo de vida debe:

- Determinar el orden de las fases del Proceso Software
- Establecer los criterios de transición para pasar de una fase a la siguiente

Un ciclo de vida, se elige en base a:

- La cultura de la corporación,
- El deseo de asumir riesgos,
- El área de aplicación,
- La volatilidad de los requisitos,
- Hasta qué punto se entienden los requisitos

CICLO DE VIDA EN CASCADA

Es una secuencia ordenada de transiciones de una fase a la siguiente según un orden lineal. Permite iteraciones (conexiones) durante su desarrollo, dentro de un mismo estado o de un estado a otro anterior. Una vez finalizado el desarrollo, todas las etapas tienen iteraciones entre ellas y se puede comenzar de nuevo y redefinir los requisitos del usuario.

REQUISITOS -> DISEÑO -> CODIFICACIÓN -> PRUEBA -> OPERACIÓN

Propiedades:

- Las etapas están organizadas de un modo lógico. No pueden pasar a la siguiente etapa hasta que tenga las decisiones tomadas.
- Cada etapa tiene un proceso de revisión. No pasará a la siguiente etapa hasta que se acepte el producto. Esto evita el menor número de errores de una etapa a la siguiente.
- El ciclo es iterativo. Pese que el flujo es de arriba hacia abajo, reconoce que los problemas en etapas inferiores afectan las etapas superiores.

Ventajas:

- Obliga a que las etapas estén definidas antes de pasar a la otra.
- Permite al líder del proyecto seguir y controlar los progresos de un modo más exacto.
- Requiere que el proceso genere documentos que luego pueden utilizarse para la validación y el mantenimiento del sistema.

Desventajas: Al asumir que los requisitos de un sistema pueden ser congelados antes de comenzar, durante el desarrollo se pueden tomar decisiones que den lugar a alternativas y este modelo no reconoce esta situación.

CICLO DE VIDA EN PROTOTIPADO

Es aconsejable usar cuando es necesario probarle al usuario la utilidad del mismo, ya que no tiene una idea detallada de lo que necesita o el ingeniero no está seguro de la viabilidad de los requisitos.

Ventajas:

- Ayuda a comprender los requisitos del usuario.
- El tiempo de desarrollo es reducido.
- Da la posibilidad de detectar fallas.
- El cliente puede probarlo y aportar nuevas ideas.

Desventajas: Al momento de elegir las funciones, se corre el riesgo de incorporar características secundarias y dejar de lado algunas características importantes.

Modelos del prototipado:

- Maqueta: Es un ejemplo visual del proyecto. Datos de prueba, no procesa datos.
- Prototipo desechable: Ayuda al cliente a identificar los requisitos. Se implanta solo los aspectos mal comprendidos o desconocidos. Todos los elementos, luego, se desecharán.
- Prototipo evolutivo: Modelo propuesto, fácilmente modificable y ampliable, aporta una representación física a los usuarios antes de ser implementados. Solo se implementan los requisitos comprendidos.

MODELO EN ESPIRAL

Permite una mejor visualización de las etapas del proceso software, con lo cual es más fácil detectar posibles riesgos, ya que se centra en la verificación del sistema.

Ventajas:

- Proporciona una combinación de los modelos existentes para un proyecto dado.
- Se presta atención a las opciones que permiten la reutilización de software existente.
- Se centra en la eliminación de errores.
- No establece una diferenciación entre desarrollo de software y mantenimiento del sistema.
- Proporciona un marco estable para desarrollos integrados hardware-software.

Desventajas:

- Modelo costoso.
- Genera mucho tiempo de desarrollo.
- Requiere experiencia en la identificación de riesgo.

PROCESO SOFTWARE

Los procesos software se usan como:

- Guía prescriptiva sobre la documentación a producir para enviar al cliente.
- Base para determinar qué herramientas, técnicas y metodologías de Ingeniería de Software serán más apropiadas para soportar las diferentes actividades.
- Marco para analizar o estimar patrones de asignación y consumo de recursos a lo largo de todo el ciclo de vida del producto software.
- Base para llevar a cabo estudios empíricos para determinar qué afecta a la productividad, el coste y la calidad del software.
- Descripción comparativa sobre cómo los sistemas software llegan a ser lo que son.

El proceso base de construcción del software consiste en analizar las necesidades de la organización en un dominio, desarrollar una solución que las satisfaga y posteriormente reinsertar la solución en el dominio, bajo un marco de gestión, seguimiento, control y gestión de calidad.

El proceso software está compuesto por:

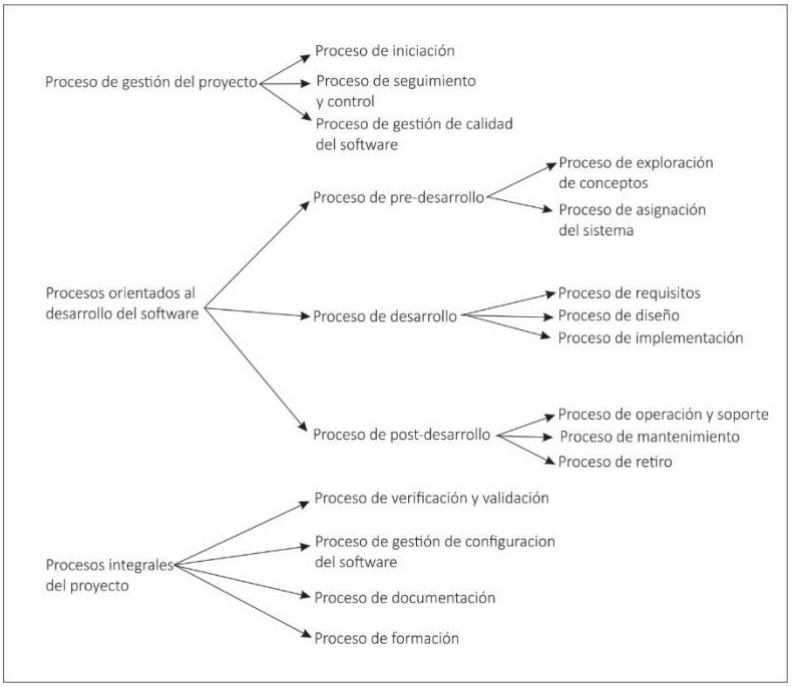
- **Proceso de Selección de un Modelo de Ciclo de Vida del Producto:** identifica y selecciona un ciclo de vida para el software que se va a construir.
- **Procesos de Gestión del Proyecto:** crean la estructura del proyecto y aseguran el nivel apropiado de la gestión del mismo durante todo el ciclo de vida del software.
- **Procesos Orientados al Desarrollo del Software:** producen, instalan, operan y mantienen el software y lo retiran de su uso. Se clasifican en procesos de pre-desarrollo, desarrollo y post-desarrollo.
- **Procesos Integrales del Proyecto:** son necesarios para completar con éxito las actividades del proyecto software. Aseguran la terminación y calidad de las funciones del mismo. Son simultáneos a los procesos orientados al desarrollo del software e incluyen actividades de no desarrollo.

Plan de gestión de proyecto software

- Establece el mapa de actividades para el modelo de ciclo de vida del software seleccionado.
- Asigna los recursos del proyecto
- Define el entorno del proyecto
- Planifica la gestión del proyecto

Plan de contingencia (Se documenta los riesgos identificados y su gestión)

- Analiza los riesgos
- Realiza la planificación de contingencia
- Gestionar el proyecto
- Archivar registros
- Implementar el sistema de informes de problemas.



Plan de Garantía de la calidad

Se documenta la planificación y administración de las acciones necesarias para proveer una confianza adecuada en la calidad de los productos software, satisfagan los requisitos técnicos establecidos.

VERIFICACION Y VALIDACIÓN DE SOFTWARE

VERIFICACIÓN: Conjunto de actividades para la comprobación de que un producto software está, técnicamente, **bien construido. (Que el producto funciona)**

VALIDACIÓN: Comprobar de que el producto software **construido es el que se deseaba construir.**

Objetivos de tipos de cambio que se dan durante el proceso del mantenimiento software

- **Corrección:** El mantenimiento correctivo cambia el software para **CORREJIR** los defectos que encuentre el cliente.
- **Adaptación:** El mantenimiento adaptativo consiste en **MODIFICAR** el software, para el que se desarrolló originalmente, y acomodarlo a los cambios de su entorno externo (Cambio de SO, CPU, periféricos).
- **Mejora:** El cliente puede descubrir funciones adicionales que pueda interesarles que pueda interesar en adicionar al software. **AMPLIA** el software más allá de sus funciones originales.

FUNCIONES DE LA GESTION DE CONFIGURACIÓN

- Identificar la configuración de un sistema: Descripción documentada de las características reales del sistema en un determinado momento.
- Controlar la configuración: Establece la configuración inicial o básica y controla los cambios en los elementos de la misma.
- Informes del estado de la configuración.
- Auditorias de configuración: Revisiones independientes de la configuración para comprobar que los elementos de la configuración cumplen los requisitos de configuración establecidos.

MAPA DE ACTIVIDADES DE UN PROYECTO: Es una tabla donde se marcan qué actividades del proceso software se van a ejecutar para un determinado proyecto. Es una tabla de dos entradas. Por un lado, el proceso software con sus actividades y, por otro lado, el ciclo de Vida elegido descompuesto en sus partes.

DISEÑO ESTRUCTURADO – GUIA 3

MODELADO DE FUNCIONES

Diagrama de flujo de datos (DFD): Se centraliza en ver COMO funciona el sistema sin tener en cuenta CON QUIEN ni CON QUE.

Partes del DFD:

1. Diagrama de contexto (DC)
2. Tabla de eventos (TE)
3. Diagrama de sistemas (DS)

Diagrama de contexto (DC)

Objetivo:

- El alcance e interacción del sistema con el ambiente
- Las interfaces con otros sistemas
- Los eventos ante los cuales el sistema debe responder

Elementos:

- El sistema (forma de burbuja)
- Las entidades externas (un cuadrado con una letra)
- Los flujos de datos (flechas direccionales)

Tabla de eventos (TE)

Detectar e individualizar los eventos a que reacciona el sistema

EVENTO: Suceso que ocasiona que el sistema reaccione de alguna manera; el sistema ante determinada acción externa o interna genere respuestas para el sistema y/o las entidades externas. Se REPRESENTAN mediante la tabla de eventos. Cuenta con: las especificaciones de los eventos, los flujos de datos correspondientes al evento y la función del mismo.

Como se ESPECIFICAN los eventos:

- El tipo de evento (pudiendo ser externo o temporal)
- La entidad externa involucrada
- Descripción del evento.

Los flujos de datos se dividen en:

- Estimulo (los que causa)
- Los de respuesta (efecto que provoca el estímulo)

ELEMENTOS DE UN DFD

- Los procesos (se dibujan en una burbuja)
- Entidad externa (cuadrado con una letra identificadora)
- Los flujos de datos (flechas direccionales)
- Los flujos activadores: Son aquellos con que se inicia el proceso en cuestión (línea doble o gruesa)
- Las demoras Una idea desde un elemento del sistema hacia otro quede suspendida en el tiempo para su uso posterior. (rectángulo abierto del lado derecho con el numero de demora a la izquierda)

MODELADO DE DATOS

El DER (Diagrama Entidad-Relación) sirve para modelar una situación de negocio desde el punto de vista de los datos que necesita guardar. Representar el modelo lógico o conceptual de los datos de la organización.

ELEMENTOS DEL DER

- Una **entidad** refleja una idea, algo que necesita ser guardado.
- Las **relaciones** van a mostrar el vínculo que existe entre las ideas, es decir, entre dos entidades.
- Los **atributos** son las características particulares que tienen esa entidad.

Tipos de atributos:

- **Claves e identificadores:** Atributo que puede determinar de manera única y mínima al resto de los atributos.
- **Candidatos:** Aquellos atributos identificadores que pueden ser claves, pero no lo son. Ej: DNI cliente, CUIL cliente, etc.
- **No claves**: Aquellos que mencionan características de la entidad. Ej.: Nombre cliente, apellido cliente, domicilio.

Propiedades de una relación:

- **Cardinalidad:** Cantidad **máxima** de relaciones que tiene una entidad con respecto a la otra entidad. Hay dos formas de marcar la cardinalidad:
  - Indicar que se relaciona muchas veces, se hace a través del signo < o >
  - Indicar que se relaciona una vez con una línea que atraviesa la relación.
- **Modalidad:** Cantidad **mínima** de relaciones entre las entidades, por lo tanto, se pueden dar dos opciones: QUE EXISTA o QUE NO EXISTA. Con un 0 que no existe u opcional, con una línea cruzada que existe o es obligatoria.

Otra propiedad es EL GRADO: Numero de entidades que participan en la relación, pueden ser:

- **Binarias:** Son las que se relacionan dos entidades
- **Unarias:** Aquellas que se relacionan con sí mismas.

Entidad **Tipo-Subtipo:** Es donde una entidad es el tipo y tiene relacionadas otras que son sus subtipos. El identificador en este tipo de relaciones es el identificados de la entidad tipo de mayor jerarquía. Los **subtipos** heredan todos los atributos del tipo y cada uno de ellos tiene atributos propios y distintos de los demás Subtipos.

La integración entre los modelos de datos (DER) y funciones (DFD), o balanceo de las técnicas es verificar que las entidades que se encuentran definidas en el Diagrama de Entidad Relación (DER) se encuentren usadas en el Diagrama de Flujo de Datos (DFD), si sobra o falta alguna de ellas entonces tendremos datos que no se utilizan o no están definidos, y, en consecuencia, los diagramas no balancean.

Para esto nos basaremos en dos reglas básicas:

- A cada entidad del DER le corresponde a una y solo una demora del DFD.
- A cada demora del DFD le corresponde una o más entidades del DER.

ESPECIFICACIÓN DE REQUISITOS – GUIA 2

La Ingeniería de requisitos (IR) trata de los principios métodos, técnicas y herramientas que permiten descubrir, documentar y mantener los requisitos para sistemas basadas en computadora de forma sistemática y receptible.

Contenido del documento de requisitos:

- Información acerca del problema.
- Propiedades y comportamiento del sistema.
- Restricciones del diseño y fabricación del producto.
- Descripciones acerca de cómo el futuro sistema ayudará a sus usuarios, a realizar mejor sus tareas.
- Restricciones acerca de la tecnología que será utilizada en la construcción del sistema (protocolos, SSOO, COTS, etc.)
- Restricciones acerca de las propiedades emergentes del sistema (requisitos no funcionales)

Tipo de requisitos:

- Requisitos que definen efectos sobre el entorno.
- Requisitos muy generales.
- Requisitos funcionales.
- Requisitos de rendimiento.
- Requisitos de usabilidad.

Requisitos en negativo:

- Dicen en donde NO emplear recursos.
- Especificar que no hará el sistema es fundamental para sistemas críticos. Se trata de especificar cómo el sistema evitará la aparición de situaciones potenciales peligrosas

Requisito funcional: Describen los servicios (funciones) que se esperan del sistema

Requisito no funcional: Son restricciones sobre los requisitos funcionales.

EDUCCIÓN DE REQUISITOS: Se refiere a la captura y descubrimiento de los requisitos.

Problemas de educción de requisitos:

- Los usuarios no pueden/saben describir muchas veces sus tareas.
- Mucha información importante no llega a verbalizarse.
- A veces hay que “inventar” los requisitos (sistemas orientados a miles de usuarios) Ej.: SO, videojuegos.
- La educción se afronta como un proceso pasivo, cuando debería ser un proceso cooperativo.

Técnicas de educción:

- Entrevistas
- Observaciones y análisis de datos
- Escenarios: Los requisitos se sitúan en el contexto de uso del sistema
- Prototipo: Útil cuando la incertidumbre es total acerca del futuro sistema

ANÁLISIS DE REQUISITOS Y SUBTAREAS ASOCIADAS: Consiste en detectar y resolver conflictos entre requisitos. Se realizan tres subtareas fundamentales:

- Clasificación de requisitos
- Modelización de requisitos
- Negociación

Criterios de clasificación de requisitos:

- En funcionales vs No funcionales
- Por prioridades
- Por coste de implementación
- Por niveles
- Según su volatilidad/estabilidad
- Si son requisitos sobre el proceso o sobre el producto

Negociación de requisitos: Aparece cuando los intereses de dos o más individuos se contradicen. De esta manera, la mejor opción sería una renegociación. Este proceso es en realizad algo positivo, puesto que los conflictos son fuentes de nuevos requisitos.

Tipos de documentos de requisitos:

- Documento de requisitos de usuario (**DRU**): Se escribe desde el punto de vista del usuario/cliente/interesado. Los contenidos no poseen demasiado nivel de detalle. Se incluye la descripción del problema actual y metas que se esperan lograr con la construcción del nuevo sistema.
- Especificación de requisitos software (**ERS**): Desarrollo y detalla mucho más los contenidos de la DRU. Contiene la respuesta a la pregunta ¿Qué características debe poseer un sistema que nos permita alcanzar los objetos y evitar los problemas expuestos en la DRU?

Objetivo de la validación de requisitos: Descubrir problemas en el Documento de Requisitos antes de comprometer recursos a su implementación.

RECOPILACION DE INFORMACIÓN PARA LA ERS

Hay 3 métodos para obtener requisitos:

- Entrevistas
- JAD (Diseño Conjunto de Aplicaciones)
- Encuestas mediante cuestionarios

Las entrevistas se realizan a través de preguntas abiertas, cerradas y bipolares.

PREGUNTAS ABIERTAS: La respuesta puede ser de dos palabras o dos párrafos.

Ventajas

- El entrevistado se siente a gusto
- Permite al entrevistador entender el vocabulario del entrevistado.
- Proporciona detalles
- Abre a más preguntas que pueden haberse pasado por alto
- Hace más interesante la entrevista
- Permiten más espontaneidad
- Facilitan la forma de expresar al entrevistador
- Ayuda cuando el entrevistador no está preparado.

Desventajas

- Pueden dar detalles irrelevantes
- Posible pérdida de control de la entrevista
- Permiten respuestas más extensas de lo necesario
- Da la impresión de que el entrevistador es inexperto
- Puede dar la impresión de que el entrevistador no tiene un objetivo

PREGUNTAS CERRADAS: Solo puede contestar con un número finito como “Ninguno, uno o quince”

Ventajas

- Ahorrar tiempo
- Comparar las entrevistas fácilmente
- Ir al grano
- Mantener el control durante la entrevista
- Cubrir terreno rápidamente
- Conseguir datos relevantes

Desventajas

- Aburren al entrevistado
- No permiten tener detalles en las respuestas
- Olvidar las ideas principales por la razón anterior
- No ayuda a forjar una relación entre el entrevistador y el entrevistado

PREGUNTAS BIPOLARES: Limita más las opciones de respuesta del entrevistado. Solo permite dos tipos de respuesta: si o no, verdadero o falso, de acuerdo o desacuerdo.

SONDEO o seguimiento: Sirven para ahondar en las preguntas para conseguir respuestas más detalladas. El propósito es ir más allá de la respuesta inicial para conseguir mayor significado, clarificar, obtener y ampliar la opinión del entrevistado.

ESTRUCTURA PIRAMIDE: Comienza con preguntas cerradas, muy detalladas y luego continua a preguntas abiertas y respuestas más generalizadas. Se utiliza cuando se cree que el entrevistado necesita motivación para profundizar en el tema.

ESTRUCTURA EMBUDO: Comienza con preguntas abiertas y generales, y luego limitando las posibles respuestas utilizando preguntas cerradas**.** Esta estructura proporciona una forma sencilla y cómoda de empezar una entrevista. También es útil cuando el entrevistado tiene opiniones fuertes acerca del tema y necesita libertad de expresar sus emociones.

ESTRUCTURA DIAMANTE: Es una combinación de ambas estructuras anteriores. Implica comenzar de una manera específica, después se examinan los aspectos generales y finalmente se termina con una conclusión especifica. Tiene la ventaja de tomar mucho más tiempo que cualquiera de las otras entrevistas.

Diseño Conjunto de Aplicaciones (JAD): Es un método alternativo para entrevistar a los usuarios uno a uno, ya que las entrevistas personales requieren tiempo y están sujetas a error por su mala interpretación.

Razones de su uso:

- Reducir tiempo y costo para las entrevistas personales.
- Mejorar la calidad de los resultados de la evaluación de los requerimientos de la información.
- Generar una mayor identificación del usuario con los nuevos sistemas de información como resultado de los procesos participativos.

Desventajas

- Requiere que los participantes dediquen una gran cantidad de tiempo. Ya que requiere un compromiso de dos a cuatro días.
- Si la preparación de las sesiones del JAD es inadecuada o si el informe de seguimiento y la documentación de especificaciones están incompletas, los diseños resultantes podrían ser poco satisfactorios.

CUESTIONARIOS: Es una técnica de recopilación de información que permite a los analistas estudiar las actitudes, creencias, comportamiento y características de muchas personas importantes en la organización que podrían resultar afectadas por los sistemas actuales y los propuestos. Se utilizan preguntas cerradas. Las Encuentas se hacen a través de correo electrónico o la web y se puede utilizar software para convertir las respuestas en tablas de análisis de datos. Los cuestionarios pueden ayudar a detectar problemas o poner de manifiesto cuestiones importantes antes de que se realicen las entrevistas.

ORIENTACION A OBJETOS – GUIA 4

El análisis y diseño orientado a objetos surge como respuesta a las nuevas necesidades que surgen conforme que avanzan los síntomas de información. Ofrece un enfoque que habilita métodos lógicos necesarios para crear nuevos sistemas complejos como respuesta al entorno cambiante de un negocio.

Objetos: Son personas, lugares o cosas que son relevantes para el sistema bajo análisis. Los objetos se representan y agrupan en clases. Una clase define el conjunto de atributos y comportamientos compartidos por cada objeto de la clase. Cada clase debe tener un **nombre** que la distinga de todas las demás. Ej.: Auto (Comienza con Mayúscula).

Atributos: Describe alguna propiedad de todos los objetos de la clase. Ej.: Tamaño, color, marca y modelo.

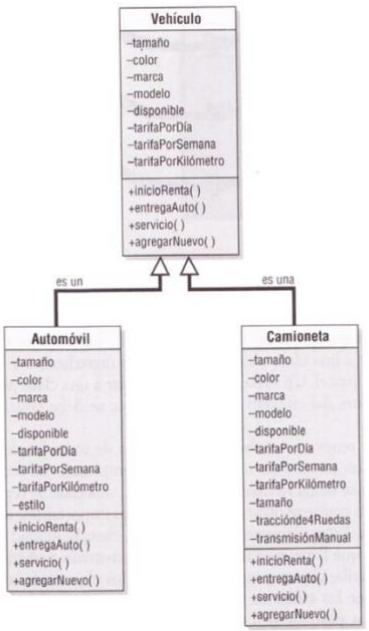
Un **método de una clase** es una acción que se puede solicitar a cualquier objeto de la clase. Son los procesos que una clase sabe cómo realizar.

Otro concepto importante de los sistemas orientados a objetos es la **herencia**. Las clases pueden tener **hijos**; es decir, una clase se puede crear a partir de otra clase. La **clase original** - o **madre-** se conoce como **clase base**. La **clase hija** se denomina **clase derivada.** Está se puede crear de tal manera que herede todos los atributos y comportamientos de la clase base. Sin embargo, una clase derivada podría tener atributos y comportamientos adicionales. Ej.: Una clase Vehículo que tenga atributos como tamaño, color y marca. Las derivadas podrían ser Automóvil o Camioneta.

DIAGRAMAS DE CLASE

Es un tipo de estructura que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones y las relaciones entre los objetos.

- Signo resta (-): Atributos **privados** o disponibles **solo para el objeto**, no se comparten con otra clase.
- Signo suma (+): Atributos **públicos**, **podrían ser invocados por otras clases.**
- Signo numeral (#): Atributos **protegidos**, están **ocultos** para **todas las clases,** excepto para las subclases inmediatas



La **UML** (Lenguaje Unificado de Modelación) proporciona un conjunto estandarizado de herramientas para documentar el análisis y diseño de un sistema software. La UML incluye diagramas que permiten a las personas visualizar la construcción de un sistema orientado a objetos.

TABLA DE COMPONENTES.

Categoría UML	Elementos de UML	Detalles específicos de UML
Cosas	Cosas estructurales	Clases Interfaces Colaboraciones Casos de uso Clases activas Componentes Nodos
	Cosas de comportamiento	Interacciones Máquinas de estado
	Cosas de agrupamiento	Paquetes
	Cosas de anotación	Notas
Relaciones	Relaciones estructurales	Dependencias Agregaciones Asociaciones Generalizaciones
	Relaciones de comportamiento	Comunica Incluye Extiende Generaliza
Diagramas	Diagramas estructurales	Diagramas de clase Diagramas de componentes Diagramas de despliegue
	Diagramas de comportamiento	Diagramas de caso de uso Diagramas de secuencias Diagramas de colaboración Diagramas de gráfico de estado Diagramas de actividades

COSAS ESTRUCTURALES: Permiten al usuario describir relaciones.

COSAS DE COMPORTAMIENTO: Describen como funcionan las cosas.

COSAS DE AGRUPAMIENTO: Se usan para definir límites.

COSAS DE ANOTACIÓN: Para que podamos agregar notas en los diagramas.

Las **relaciones** son el pegamento que une a las cosas.

RELACIONES ESTRUCTURALES: Se usan para enlazar las cosas en los diagramas estructurales. En esté incluye **dependencias, agregaciones, asociaciones y generalizaciones.**

RELACIONES DE COMPORTAMIENTO: Se usan en los diagramas de comportamiento. En esté hay cuatro tipos: **comunica, incluye, extiende y generaliza.**

DIAGRAMAS ESTRUCTURALES: Se usan para describir las relaciones entre las clases. Incluyen diagramas de **clase**, diagramas de **objetos**, diagramas de **componentes** y diagramas de **despliegue**.

DIAGRAMAS DE COMPORTAMIENTO: Se usan para describir la interacción entre las personas y la cosa que nos referimos como caso de uso.

Incluyen

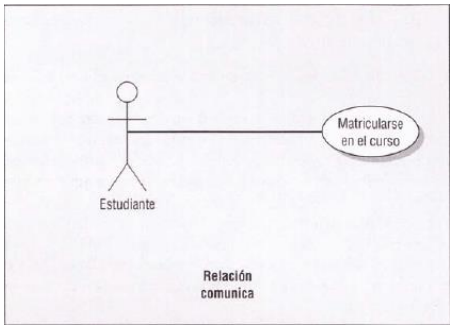
- diagrama de **caso de uso**
- diagrama de **secuencias**
- diagramas de **colaboración**
- diagrama de **gráfico de estado**
- diagrama de **actividades**.

MODELO DE CASO DE USO

Describe lo que hace un sistema sin describir **COMO** lo hace. Técnica de análisis orientada a objetos.

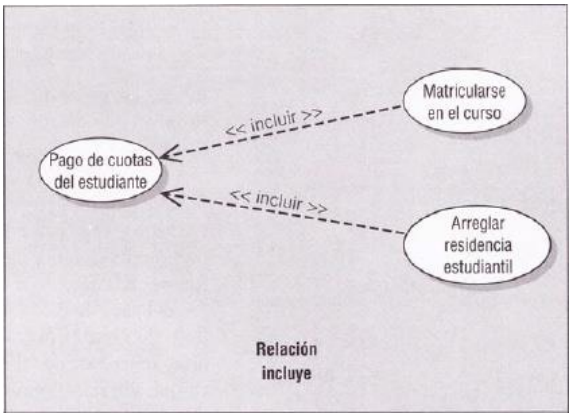
SÍMBOLOS DEL CASO DE USO

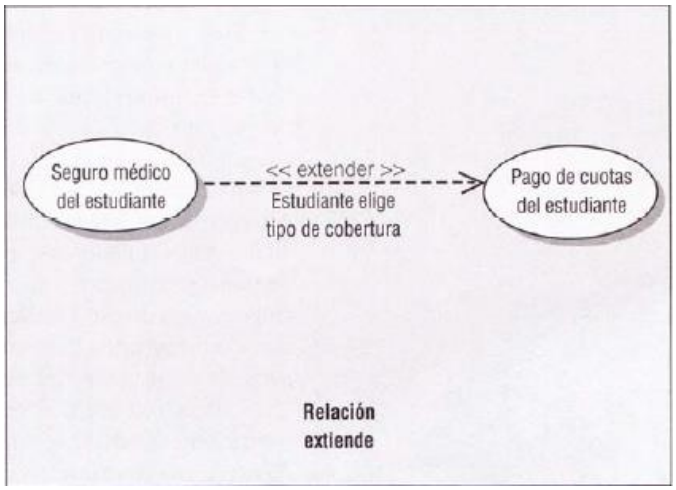
- **Actor:** Son parecidos a las entidades externas; existen fuera del sistema. Este término se refiere a un pape particular de un usuario del sistema. El actor existe fuera del sistema e interactúa con éste de una forma específica. Los actores pueden iniciar una instancia de un caso de uso. Los actores pueden interactuar con uno o más casos de uso u viceversa. Un actor se divide en dos:
  - Los actores **principales** proporcionan datos o reciben información del sistema
  - Los actores **secundarios** ayudan a mantener el sistema en ejecución o proporcionan ayuda.
- **Evento:** Es una entrada al sistema que pasa en un tiempo y lugar específicos y ocasiona que el sistema haga algo.
- **Caso de uso:** siempre describe tres cosas:
  - un actor que inicia el evento
  - el evento que activa el caso de uso
  - y el caso de uso que desempeña las acciones activadas por el evento.



COMUNICA: Se usa para conecta a un actor con un caso de uso. Ej.: Un ESTUDIANTE se comunica con MATRICULARSE EN EL CURSO. Se representa con una línea, el actor con el caso de uso.

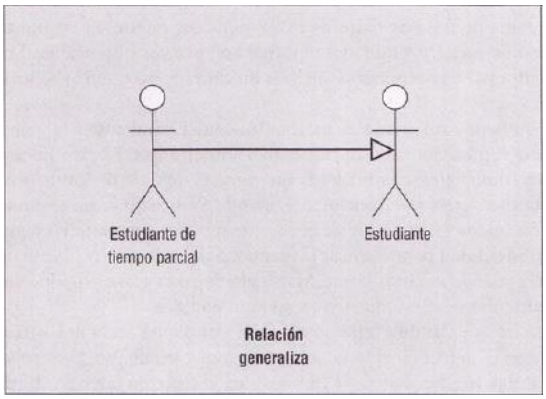
INCLUYE: Describe una situación en que un caso de uso, contiene un comportamiento que es común para más de un caso de uso. Un caso común se incluye en otros dos casos de uso. Una flecha punteada apunta al caso de uso común.





EXTIENDE: Describe la situación en la que un caso de uso posee el comportamiento que permite al nuevo caso de uso manejar una variación o excepción del caso de uso básico. Va del caso de uso extendido al básico.

GENERALIZA: Implica que una cosa es más típica que otra. Esta relación podría existir entre dos actores o dos casos de uso. La flecha va del caso de uso extendido al básico.



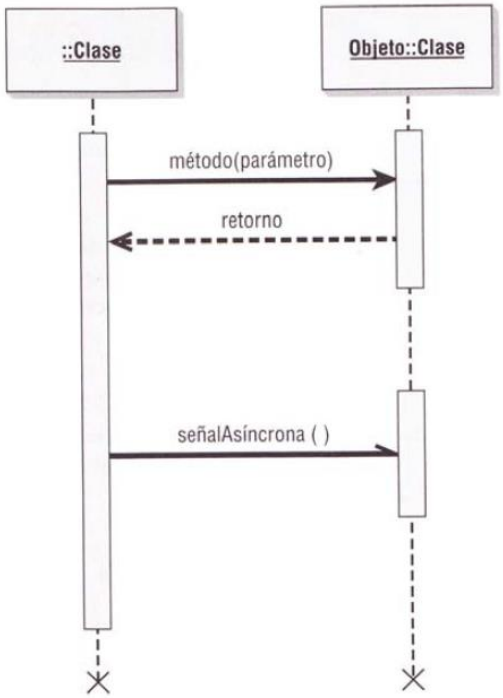
Hay 3 áreas de un escenario de caso de uso:

- Identificadores e iniciadores de caso de uso: Contienen el nombre de caso de uso y una ID única, el área de aplicación, los actores, una breve descripción de lo que logra el caso de uso y la activación del evento con el tipo de activación, externo (empezado por un actor) o temporal (que se activan o empiezan por tiempo).
- Pasos desempeñados: Representan el flujo estándar de eventos y los pasos tomados para la realización exitosa del caso de uso.
- Condiciones, suposiciones y preguntas: Incluye las precondiciones o la condición del sistema antes de que se pudiera desempeñar el caso de uso; las postcondiciones o el estado del sistema después de que el caso de uso se ha terminado, cualquier suposición hecha que pudieran afectar el método de caso de uso, cualquier asuntos excelentes o preguntas que se deben responder antes de la implementación del caso de uso, una declaración opcional de prioridad del caso de uso y la declaración opcional de riesgo involucrada al crear el caso de uso.

DIAGRAMA DE SECUENCIA

Pueden ilustrar una sucesión de interacciones entre clases o instancias de objetos en los escenarios de caso de uso.

En la parte de arriba, se representan los actores y clases o instancias.

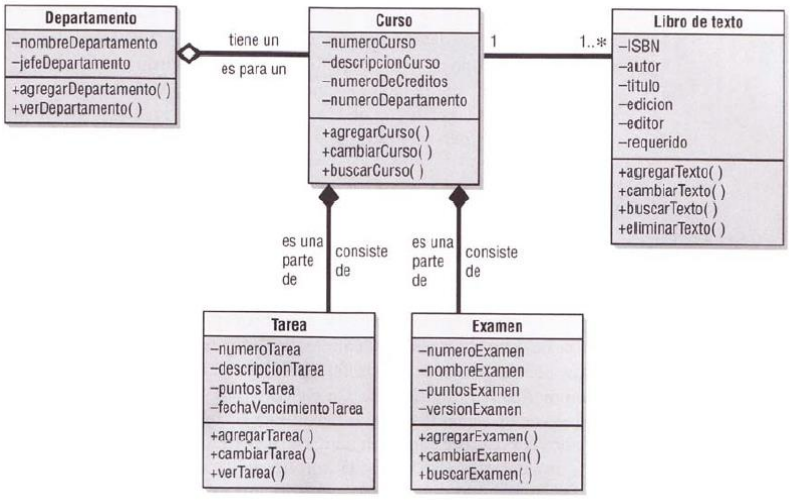


- **nombreDelObjeto:** Un nombre seguido de dos puntos, representa un objeto.
- **:clase** Dos puntos seguidos de un nombre, representan una clase.
- **nombreDelObjeto:clase** Un nombre, seguido de dos puntos y otro nombre, representa un objeto de una clase.
- **Una línea vertical:** representa la trayectoria de la vida de la clase o del objeto. Comienza cuando se crea y finaliza cuando se destruye.
- Una **X** al final de la trayectoria, indica cuando se destruye el objeto.
- Una **barra lateral o rectángulo:** muestra el enfoque de control cuando el objeto se encuentra realizando algo.
- **Flechas:** Muestran mensajes o signos que se envían entre las clases
- Las puntas de **flechas solidadas** representan llamadas síncronas. Se usan cuando la clase emisora espera una respuesta de la clase receptora, y el control se devuelve a la clase emisora cuando la clase que recibe el mensaje termina su ejecución.
- Las **flechas con media punta** (o abiertas) representan llamadas asíncronas, es decir, llamadas que se envían sin esperar a que sean devueltas a la clase que las emite. Ej.: Un menú para ejecutar un programa.
- Las **flechas punteadas** representan el retorno del mensaje.

DIAGRAMAS DE CLASE

Los **diagramas de clase** muestran las características estáticas del sistema y no representan ningún procesamiento en particular. También muestra la naturaleza de las relaciones entre clases.

Un diagrama de clases podría mostrar simplemente el nombre de la clase; o el nombre de la clase y los atributos; o el nombre de la clase, los atributos y los métodos. Mostrar solo el nombre de la clase es útil cuando el diagrama es muy complejo e incluye muchas clases.



Hay cuatro tipos de clases:

Clases de **entidad:** Representan elementos de la vida real, como gente, cosas, etc. Las clases de entidad son las entidades representadas en un diagrama entidad-relación. Cada objeto tiene muchos atributos, pero la clase debe incluir solo aquellos que utiliza la organización.

Clases de **limite o interfaz:** Estas clases ofrecen a los usuarios un medio para trabajar con el sistema. Hay dos categorías: Humana y de sistema. Una interfaz **humana** puede ser una pantalla, un formulario web, un menú, etc. Deben crearse prototipos de las interfaces humanas y a menudo se usa storyband (una ilustración del argumento o secuencia escena post escena) para modelar la secuencia de interacciones. Una interfaz del **sistema** implica el envío o

recepción de datos de otros sistemas. Esto podría incluir a las bases de datos de la organización. Las interfaces externas son las menos estables, por que con frecuencia hay poco o ningún control sobre el socio externo que podría alterar el formato del mensaje o los datos.

Clases **abstractas**: Son las clases que no son posible instanciar directamente. Están vinculadas a clases concretas en una relación generalización/especialización (gen/esp).

Clases de **control**: Las clases de control, o activas, se utilizan para controlar el flujo de actividades, y funcionan como coordinadoras al implementar clases. Para lograr clases reutilizables, un diagrama de clases podría incluir muchas clases pequeñas de control. Con frecuencia, las clases de control se derivan durante el diseño del sistema.

RELACIONES: Son conexiones entre las clases, similares a aquellas que se encuentran en un diagrama de entidad-relación. Se muestran como líneas que se conectan las clases en un diagrama de clases. Hay dos categorías de relaciones:

- **ASOCIACIONES:** El tipo mas simple de relación, o una conexión estructural entre clases y objetos. Las asociaciones se muestran como una línea simple en un diagrama de clases. Los puntos finales de la línea se etiquetan con un símbolo que indica **la multiplicidad.** El **0** representa **ninguno**, un **1** representa **uno y solo uno**, y un asterisco ***** representa **muchos.** 0…1 representa de 0 a 1, y la notación 1…* **representa uno a muchos**. Las clases de asociación son aquellas que se usan para dividir una asociación muchos a muchos.
- **RELACIONES TODO/PARTE:** Surgen cuando una clase representa el objeto total y otras clases representan parte del mismo. El todo actúa como contenedor de las partes. Estas relaciones se representan en un diagrama de clases mediante una línea con un diamante en un extremo. Una relación todo/parte podrían ser un objeto entidad que tiene partes distintas, como un sistema de cómputo que incluye computadora, copiadora, monitor, etc. La relación todo/parte tienen varias categorías: agregación, colección y composición.

AGREGACIÓN: Se describe como una relación "tiene un". La agregación proporciona un medio para mostrar que el objeto total se compone de la suma de sus partes (otros objetos). Ej.: el departamento tiene un curso y el curso es para un departamento. La relación es más débil, por que el departamento podría cambiarse o eliminarse y el curso todavía existiría. **EL DIAMANTE NO APARECE SOLIDO**

COLECCIÓN: Una colección consta de un todo y sus miembros. Éste podría ser un distrito electoral con votantes o una biblioteca con libros. Los votantes o libros podrían cambiar, pero el todo conserva su identidad. Ésta es una asociación débil.

COMPOSICIÓN: Una relación todo/parte en la cual el todo tiene una responsabilidad por la parte, es una relación aun más fuerte. **SE MUESTRA CON UN DIAMANTE SOLIDO.** Si el todo se elimina, todas las partes se eliminan. Ej.: E n una universidad hay una relación de composición entre un curso y una tarea, así como entre un curso y un examen. Si el curso se elimina, las tareas y exámenes también se eliminan.

HERENCIA: Varias clases podrían tener los mismos atributos y/o métodos. Cuando esto ocurre, se crea una **clase general** que contiene los atributos y métodos comunes. La clase especializada hereda o recibe los atributos y métodos de la clase general. La clase especializada tiene atributos y métodos que son únicos y solo están definidos en la clase especializada. La creación de clases generalizadas y el hecho de permitir que la clase especializada herede sus atributos y métodos fomenta la reutilización, porque el código se usa muchas veces.

GERENALIZACIÓN: Describe una relación entre un tipo general de cosa y un tipo más especifico de cosa. Este tipo de relación se describe a menudo como una relación "es un". Ej.: un automóvil es un vehículo y un camión es un vehículo. El vehículo es el caso general u el automóvil y el camión son cosas más específicas. Se utilizan para modelar la herencia de clases y la especialización. Una **clase general** a veces se conoce como superclase, clase base o clase madre; una **clase especializada** se denomina subclase, clase derivada o clase hija.

POLIMORFISMO (Muchas formas): Propiedad que tiene una subclase de redefinir cualquier método de la superclase. Es la capacidad de un programa orientado a objetos para tener varias versiones del mismo método con el mismo nombre dentro de una relación superclase/subclase. La subclase hereda un método de su clase madre, pero podría

agregarle comportamiento o modificarlo. La subclase podría cambiar el tipo de datos, o cambiar la forma en que trabaja el método.

CLASES ABSTRACTAS: Son clases generales y se utilizan cuando se incluye gen/esp en el diseño. La clase general se convierte en la clase abstracta. La clase abstracta no tiene objetos directos o instancias de clase, y solo se usa con clases especializadas. Por lo general, las clases abstractas tienen atributos y podrían incluir algunos métodos. LA FLECHA APUNTA A LA CLASE GENERAL O SUPERCLASE

MENSAJES: Sirven para enviar información entre las clases. Un mensaje también actúa como un comando, que le indica a la clase receptora que realice alguna tarea. Un mensaje consiste del nombre del método de la clase receptora, así como los atributos que se pasan con el nombre del método. La clase receptora debe tener un método que corresponda con el nombre del mensaje.

PAQUETES: Son los contenedores para otros elementos de UML, como los casos de uso o las clases. Los paquetes pueden mostrar el particionamiento del sistema, indicando cuales clases o casos de uso se agrupan en un subsistema, y se conocen como paquetes lógicos. También pueden ser paquetes componentes (los cuales contienen los componentes físicos del sistema) o paquetes de casos de uso (que contienen un grupo de caso de uso). Los paquetes usan un símbolo de carpeta con el nombre del paquete en la pestaña de la carpeta o centrado en esta última.