**LAS ORGANIZACIONES COMO SISTEMAS**

Las organizaciones están compuestas por sistemas más pequeños e interrelacionados (departamentos, unidades, divisiones, etc.) que brindan funciones especializadas. Algunas de las funciones comunes son contabilidad, marketing, producción, procesamiento de datos y administración. Las funciones especializadas (sistemas más pequeños) se reintegran en un momento dado a través de diversos medios para formar un todo organizacional efectivo.

Todos los sistemas y subsistemas están interrelacionados y son interdependientes. Este hecho tiene implicaciones importantes, tanto para las organizaciones como para los analistas de sistemas que buscan ayudar a estas organizaciones a cumplir mejor sus objetivos. Cuando se modifica o elimina algún elemento de un sistema, también se afecta considerablemente al resto de los elementos y subsistemas de ese sistema.

Las organizaciones como sistemas es que todos los sistemas están contenidos por límites

que los separan de sus entornos.

Para poder adaptarse y sobrevivir, las organizaciones primero deben ser capaces de importar personas, materia prima e información a través de sus límites (entradas) y después intercambiar sus productos terminados, servicios o información con el mundo exterior (salidas).

El sistema ideal, en todo caso, es uno que se corrija o regule automáticamente de tal forma que no se requieran decisiones basadas en acontecimientos comunes.

Al tomar una perspectiva de sistemas, los analistas pueden empezar a descifrar y comprender en términos generales las diversas empresas con las que entrarán en contacto.

**DESCRIPCIÓN GRÁFICA DE LOS SISTEMAS**

Los diagramas de flujo de datos se enfocan en los datos que fluyen hacia el sistema y salen de él, además

del procesamiento de estos datos.

El diagrama de flujo de datos a nivel de contexto es una manera de mostrar el alcance del sistema o lo que

se va a incluir en él. Las entidades externas están fuera del alcance y esto es algo sobre lo que el sistema no tiene control.

Otra forma en que un analista de sistemas puede mostrar el alcance del sistema y definir límites apropiados para el mismo es mediante el uso de un modelo entidad-relación.

**MODELADO DE CASOS DE USO**

Aunque en un principio se presentaron como un diagrama para usarlo en el UML orientado a objetos, ahora los casos de uso se utilizan sin importar la metodología para el desarrollo de sistemas. Se pueden utilizar como parte del SDLC o en el modelado ágil. Un modelo de caso de uso describe *qué* hace un sistema sin describir *cómo* lo hace.

El modelo de caso de uso presenta al sistema desde la perspectiva de un usuario fuera del mismo.

Un analista desarrolla casos de uso en un esfuerzo de cooperación con los expertos de negocios que ayudan a definir los requerimientos del sistema.

Desde la perspectiva de un actor (o usuario), un caso de uso debe producir algo de valor. Por lo tanto, el analista debe determinar qué es importante para el usuario y debe recordar incluirlo en el diagrama del caso de uso.

**Símbolos de los casos de uso**

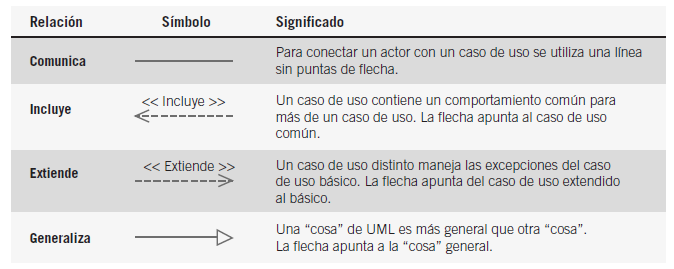
Un diagrama de caso de uso contiene los símbolos del actor y del caso de uso, junto con líneas conectoras. Los actores son similares a las entidades externas; existen fuera del sistema. El término *actor* se refiere a un rol específico de un usuario del sistema.

El actor existe fuera del sistema e interactúa con éste de una manera específica. Un actor puede ser un humano, otro sistema o un dispositivo como un teclado o una conexión Web. Los actores pueden iniciar una instancia de un caso de uso. Un actor puede interactuar con uno o más casos de uso; un caso de uso puede involucrar a uno o más actores.

Un caso de uso provee a los desarrolladores una perspectiva de lo que quieren los usuarios, sin detalles técnicos o implementación.

**Relaciones de los casos de uso**

Las relaciones activas se conocen como relaciones de comportamiento. Hay cuatro tipos básicos de relaciones de comportamiento: comunica, incluye, extiende y generaliza.



**COMUNICACIÓN** Esta relación de comportamiento se utiliza para conectar un actor con un caso de uso. Recuerde que la tarea del caso de uso es proporcionar cierto tipo de resultado que sea benéfico para el actor en el sistema.

**INCLUSIÓN** Esta relación (también conocida como relación de usos) describe la situación en la que un caso de uso contiene comportamiento común para más de un caso de uso.

**EXTENSIÓN** Esta relación 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 a partir del caso de uso básico.

**GENERALIZACIÓN** Esta relación implica que una cosa es más común que otra. Esta relación puede existir entre dos actores o dos casos de uso.

Los casos de uso se documentan mediante una plantilla de documento de caso de uso predeterminada por la organización, que facilita la lectura de los casos de uso y provee información estandarizada para cada caso de uso en el modelo.

**Por qué son útiles los diagramas de casos de uso**

Los diagramas de casos de uso identifican a todos los actores en el dominio del problema, de forma que un analista de sistemas se puede concentrar en lo que los humanos desean y necesitan para usar el sistema, extender sus capacidades y disfrutar su interacción con la tecnología.

Las acciones a completar también se muestran con claridad en el diagrama de caso de uso. Esto no sólo facilita al analista la tarea de identificar los procesos, sino que también ayuda en la comunicación con los demás analistas en el equipo y con los ejecutivos de la empresa.

Los diagramas de caso de uso se están haciendo populares debido a su sencillez y carencia de detalles técnicos. Se utilizan para mostrar el alcance de un sistema, junto con las principales características del mismo y los actores que trabajan con esas características principales.

**CONCEPTOS ORIENTADOS A OBJETOS**

**Objetos**

Los objetos son personas, lugares o cosas relevantes para el sistema a analizar. Los sistemas orientados a objetos describen las entidades como objetos.

**Clases**

Por lo general, los objetos forman parte de un grupo de elementos similares, conocidos como clases.

Los objetos se representan y agrupan mediante clases, las cuales son óptimas para la reutilización y la facilidad de mantenimiento. Una clase define el conjunto de atributos compartidos y comportamientos que se encuentran en cada objeto de la clase.

El término *instanciar* se utiliza cuando se crea un objeto a partir de una clase.

Cada clase debe tener un nombre que la distinga de las demás. Por lo general, los nombres de las clases son sustantivos o frases cortas y empiezan con mayúscula. En UML, una clase se dibuja como un rectángulo. Este rectángulo contiene otras dos características importantes: una lista de atributos y una serie de métodos.

Un atributo describe cierta propiedad que poseen todos los objetos de la clase.

Un método es una acción que se puede solicitar de cualquier objeto de la clase. Los métodos son los procesos que una clase sabe cómo llevar a cabo.

**Herencia**

Las clases pueden tener hijos; es decir, se puede crear una clase a partir de otra. En UML, la clase original (o padre) se conoce como clase base; a la clase hija se le denomina clase derivada. una clase derivada puede tener atributos y comportamientos adicionales.

La herencia reduce la labor de programación al permitir que se utilicen los objetos comunes con facilidad.

**CONCEPTOS Y DIAGRAMAS DEL LENGUAJE UNIFICADO DE MODELADO (UML)**

El UML consiste en cosas, relaciones y diagramas. Los primeros componentes (o elementos primarios) de UML se llaman cosas. Tal vez usted prefiera otra denominación, como objeto, pero en UML se les llama cosas. Las cosas estructurales son las más comunes. Las cosas estructurales son clases, interfaces, casos de uso y muchos otros elementos que proveen la forma de crear modelos. Las cosas estructurales permiten al usuario describir las relaciones. Las cosas de comportamiento describen la forma en que funcionan las cosas. Algunos ejemplos de cosas de comportamiento son las interacciones y las máquinas

de estado. Las cosas de grupo se utilizan para definir límites. El paquete es un ejemplo de cosa de grupo. Por último, tenemos las cosas de anotaciones, para poder agregar notas en los diagramas.

Las relaciones son el pegamento que mantiene las cosas unidas entre sí. Es conveniente pensar en las relaciones de dos formas. Las relaciones estructurales se utilizan para unir las cosas en los diagramas estructurales. Las relaciones estructurales incluyen dependencias, agregaciones, asociaciones y generalizaciones. Por ejemplo, las relaciones estructurales muestran herencia. Las relaciones de comportamiento se utilizan en los diagramas de comportamiento. Los cuatro tipos básicos de relaciones de comportamiento son comunica, incluye, extiende y generaliza.

Hay dos tipos principales de diagramas en UML: diagramas estructurales y diagramas de comportamiento.

Los diagramas estructurales se utilizan, por ejemplo, para describir las relaciones entre las clases. Éstos se dividen en diagramas de clases, diagramas de objetos, diagramas de componentes y diagramas de despliegue. Por otro lado, los diagramas de comportamiento se pueden utilizar para describir la interacción entre las personas (actores en UML) y lo que denominamos caso de uso, o la forma en que los actores utilizan el sistema. Los diagramas de comportamiento se dividen en diagramas de casos de uso, diagramas de secuencia, diagramas de comunicación, diagramas de estados y diagramas de actividad.

**MODELADO DE CASOS DE USO**

Un caso de uso siempre describe tres cosas: un actor que inicia un evento, el evento que desencadena un

caso de uso y el caso de uso que realiza las acciones desencadenadas por el evento.

**DIAGRAMAS DE ACTIVIDAD**

Los diagramas de actividad muestran la secuencia de actividades en un proceso, incluyendo las actividades secuenciales y paralelas, además de las decisiones que se toman. Por lo general se crea un diagrama de actividad para un caso de uso y puede mostrar los distintos escenarios posibles.

Un rectángulo con esquinas redondas representa una actividad.

Una flecha representa a un evento. Los eventos representan cosas que ocurren en cierto momento y lugar.

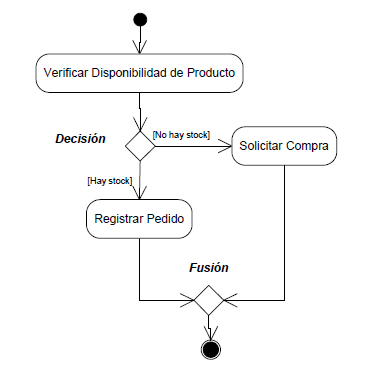
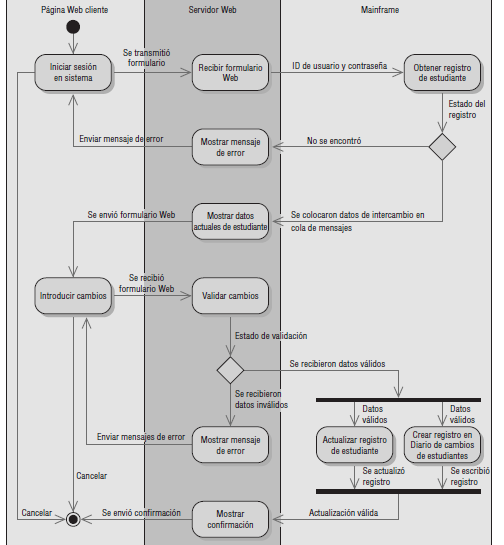
Un diamante representa una decisión (también conocida como ramificación) o una fusión. En las decisiones hay una flecha que entra al diamante y varias que salen de él. Se puede incluir una condición de guardia, que muestra los valores de la condición. Las fusiones muestran varios eventos que se combinan para formar un evento.

Un rectángulo largo y plano representa una barra de sincronización. Estas barras se utilizan para mostrar las actividades paralelas, donde puede haber un evento que entre a la barra de sincronización y varios eventos que salgan de ella, a lo cual se le denomina bifurcación. Una sincronización en la que varios eventos se fusionan en uno solo se denomina unión.

Hay dos símbolos que muestran el inicio y fin del diagrama. El estado inicial se muestra como un círculo

relleno. El estado final se muestra como un círculo negro rodeado por un círculo blanco.

Los rectángulos que rodean otros círculos se denominan carriles (*swimlanes*). Estos carriles indican particionamiento y se utilizan para mostrar qué actividades se realizan en cada plataforma, como un navegador, servidor o computadora mainframe; también muestran las actividades que realizan distintos grupos de usuarios. Los carriles son zonas que pueden describir tanto la lógica como la responsabilidad de una clase.



**DIAGRAMAS DE SECUENCIA Y DE COMUNICACIÓN**

**Diagramas de secuencia**

Los diagramas de secuencia pueden ilustrar una sucesión de interacciones entre clases o instancias de objetos a través del tiempo. A menudo, los diagramas de secuencia se utilizan para ilustrar el procesamiento descrito en los escenarios de casos de uso.

**Diagramas de comunicación**

Los diagramas de comunicación se introdujeron en el UML 2.0. Su nombre original en el UML 1.x era diagramas de colaboración. Los diagramas de comunicación describen las interacciones entre dos o más cosas en el sistema que desempeñan un comportamiento mayor a lo que cualesquiera de las dos cosas pueden hacer por su cuenta.

Un diagrama de comunicación consta de tres partes: los objetos (también llamados participantes), los enlaces de comunicación y los mensajes que se pueden pasar a través de esos enlaces. Los diagramas de comunicación muestran la misma información que un diagrama de secuencia, pero pueden ser más difíciles de leer. Para poder mostrar el orden en el tiempo, debemos indicar un número de secuencia y describir el mensaje.

**UN PROCESO DIRIGIO POR CASOS DE USO**

Los casos de uso han sido adoptados casi universalmente para la captura de requisitos de sistemas software en general, pero los casos de uso son mucho mas que una herramienta para capturar requisitos. Dirigen el proceso de desarrollo en su totalidad. Los casos de uso son la entrada fundamental cuando se identifican y especifican clases, subsistemas e interfaces cuando se identifican y especifican casos de prueba, y cuando se planifican las iteraciones del desarrollo y la integración del sistema. Para cada iteración, nos guian a través del conjunto completo de flujos de trabajo, desde la captura de requisitos, pasando por el análisis, diseño e implementación, hasta la prueba, enlazando estos diferentes flujos de trabajo.

**Desarrollo dirigido por casos de uso en pocas palabras**

La captura de requisitos tiene dos objetivos: encontrar los verdaderos requisitos y representarlos de un modo adecuado para los usuarios, clientes y desarrolladores. Entendemos por “verdaderos requisitos” a aquellos que cuando se implementen añadirán el valor esperado para los usuarios.

Normalmente, un sistema tiene muchos tipos de usuarios. Cada tipo de usuario se representa por un actor. Los actores utilizan el sistema interactuando con los casos de uso. Un caso de uso es una secuencia de acciones que el sistema lleva a cabo para ofrecer algún resultado de valor para el actor.

El modelo de análisis es una especificación detallada de los requisitos y funciona como primera aproximación del modelo de diseño, aunque es un modelo con entidad propia. El modelo de análisis también se utiliza para crear un sistema robusto y flexible que emplea la reutilización de componentes de manera considerable. Los desarrolladores crean un modelo de análisis que utiliza el modelo de casos de uso como entrada. Cada caso de uso en el modelo de casos de uso se traducirá en una realización de caso de uso en el modelo de análisis. Durante el flujo de trabajo de prueba los ingenieros de prueba verifican que el sistema implementa de verdad la funcionalidad descrita en los casos de uso y que satisface los requisitos del sistema.

Lo novedoso y diferente del Proceso Unificado es que la prueba puede planificarse al principio del ciclo de desarrollo. Tan pronto como se hayan capturados los casos de uso, es posible especificar los casos de prueba y determinar el orden en el cual realizarlos, integrarlos y probarlos.

**¿Por qué casos de uso?**

-Proporcionan un medio sistemático e intuitivo de capturar requisitos funcionales centrándose en el valor añadido para el usuario.

-Dirigen todo el proceso de desarrollo debido a que la mayoría de las actividades como el análisis, diseño y prueba se llevan a cabo partiendo de los casos de uso. El diseño y la prueba pueden también planificarse y coordinarse en términos de casos de uso.

No todos los actores representan a personas. Pueden ser actores otros sistemas o hardware externo que interactuará con el sistema. Los actores se comunican con el sistema mediante el envio y recepción de mensajes hacia y desde el sistema según este lleva a cabo los casos de uso.

Los casos de uso tambien se utilizan como contenedores de los requisitos no funcionales, tales como los requisitos de rendimiento, disponibilidad, exactitud y seguridad que son específicos de un caso de uso.

Durante el análisis y el diseño, transformamos el modelo de casos de uso mediante un modelo de análisis en un modelo de diseño, es decir, en una estructura de clasificadores y realizaciones de casos de uso. El objetivo es realizar los casos de uso de una forma económica de manera que el sistema ofrezca un rendimiento adecuado y pueda evolucionar en el futuro.

Un diagrama de colaboración recuerda a un diagrama de clases, pero contiene instancias y enlaces en lugar de clases y asociaciones. Muestra cómo interactúan los objetos secuencialmente o en paralelo, numerando los mensajes que se envían unos a otros.

El modelo de diseño es más físico por naturaleza, mientras que el modelo de análisis es mas conceptual.

**CAPTURA DE REQUISITOS COMO CASOS DE USO**

Los diagramas de estado especifican el ciclo de vida de las instancias de los casos de uso en términos de estados y transiciones entre los estados. Cada transición es una secuencia de acciones. Los diagramas de actividad describen el ciclo de vida con mas detalle describiendo también la secuencia temporal de acciones que tiene lugar dentro de cada transición.

Una instancia de caso de uso es la realización de un caso de uso. Cuando se lleva a cabo una instancia de un caso de uso, ésta interactúa con instancias de actores, y ejecuta una secuencia de acciones según se especifica en el caso de uso. Las instancias de los casos de uso no interactúan con otras instancias de casos de uso. Son atómicas.

El analista de sistemas es responsable de delimitar el sistema, encontrando los actores y los casos de uso y asegurando que el modelo de casos de uso es completo y consistente. El analista de sistemas no es responsable de cada caso de uso en particular. Esto es una responsabilidad aparte, que pertenece al trabajador especificador de casos de uso. El analista también dirige el modelado y el que coordina la captura de requisitos.

**UML**

Es un lenguaje grafico estándar que permite generar modelos en el desarrollo de sistemas durante todas las etapas de su construcción.

Permite:

-Visualizar: proporciona un soporte grafico para la construcción de sus modelos, mejorando la comunicación entre los desarrolladores ya que utiliza una notación bien definida.

-Especifica: permite detalles distintos aspectos de un sistema, tales como, requisitos, funcionalidades, comportamiento, etc.

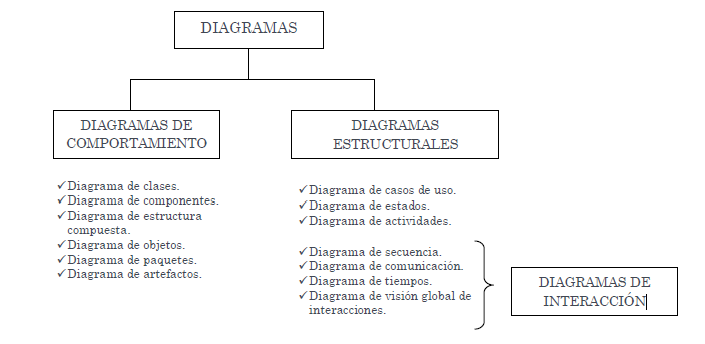
-Construir: permite generar diagramas que ayudan a la construcción de software.

-Documentación: proporciona modelos durante todas las etapas del desarrollo, estos pueden utilizarse como documentación.

**MODELO – PROCESO**

UML no es un proceso, permite crear modelos, define como se realizan, pero no indica cuales o cuántos modelos deben elaborarse durante el desarrollo de un sistema ni en qué momento. Estas son propiedades que define el proceso de desarrollo que se haya seleccionado para la construcción del sistema. Es entonces el proceso de desarrollo quien debe indicar: quién es el responsable de la elaboración de un modelo, qué artefacto se debe construir, cómo construirlo y en qué momento.

**DIAGRAMAS EN UML**

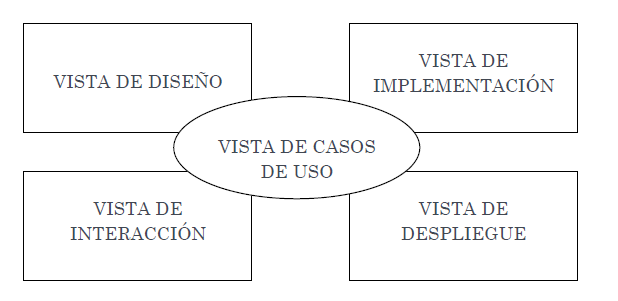
****

**Diagramas Estructurales:** Permiten modelar aspectos estáticos del sistema. Son diagramas que modelan propiedades que se mantienen estables, no muestran flujos de mensajes entre sus elementos ni secuencia.

**Diagramas de Comportamiento:** Permiten modelar aspectos dinámicos del sistema. Estos diagramas permiten ver flujo de mensaje entre elementos, secuencia, actividades, y el movimiento físico de componentes.

**Diagramas de Interacción:** Son un tipo especial de diagramas de comportamiento. Se destacan por mostrar “interacciones”. Una interacción es flujo de mensajes que existe entre dos o más objetos.

La arquitectura es la forma de organizar las distintas partes de software, generalmente se define a rincipio del diseño y se utiliza como guía durante toda la construcción del sistema. El desarrollo de un sistema se puede ver desde distintos puntos de vistas. Con UML se pueden expresar cada una de estas vistas:



**Vista de Casos de Uso**

En esta vista se incluyen los casos de uso que describen la funcionalidad y el comportamiento del sistema tal como lo ven los usuarios, analistas y los encargados de las pruebas del sistema. Se utilizarán los diagramas de casos de uso, de estados, de actividades y diagramas de interacción.

**Vista de Diseño**

En esta vista se incluyen las clases, interfaces y colaboraciones que intervienen en el desarrollo. Para su definición se utilizarán los diagramas de clases, objetos, estructura interna, estados, actividades y diagramas de interacción.

**Vista de Interacción**

En esta vista se incluye todo lo referido al flujo de control entre elementos, teniendo en cuenta sincronización y concurrencia. Se utilizaran los diagramas de clases, objetos, estructura interna, estados, actividades y diagramas de interacción. En este caso se deberá hacer hincapié en las clases activas.

**Vista de Implementación**

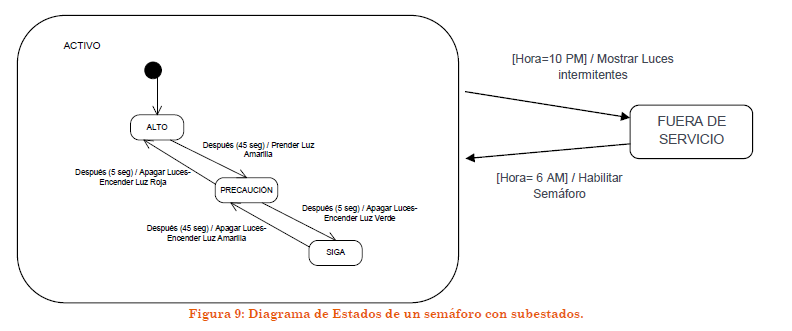
En esta vista se incluyen los artefactos necesarios para poner en funcionamiento el sistema software. Se muestra la relación entre las clases, los componentes y los archivos físicos. Se utilizarán los diagramas de secuencia, actividades y estados.

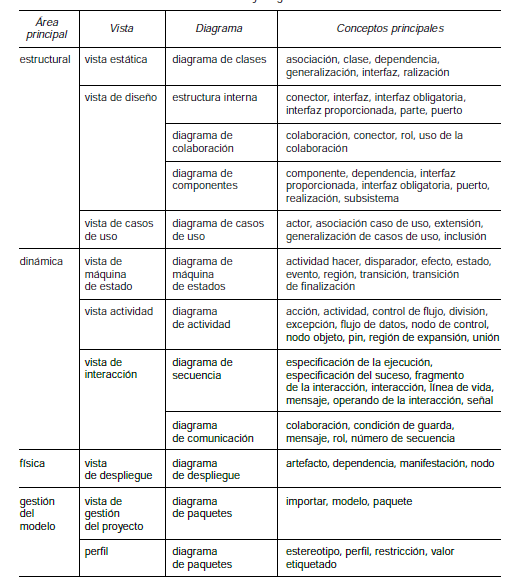
**Vista de Despliegue**

En esta vista se incluyen los nodos, componentes de red y hardware donde se instalará el sistema físico y su distribución. Se utilizarán diagramas de despliegue, estado, actividades y diagramas de interacción.

**DIAGRAMA DE ESTADOS**

Un diagrama de estados es una máquina de estados donde se destaca el flujo de control entre los estados por los que pasa un objeto durante su vida útil.





**EL PROCESO UNIFICADO**

El proceso unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software.

El proceso unificado está basado en componentes, lo cual quiere decir que el sistema software es construcción está formado por componentes software interconectados a través de interfaces bien definidas.

El proceso unificado utiliza el Lenguaje Unificado de Modelado para preparar todos los esquemas de un sistema software. Lo que lo hace único al proceso unificado es que esta dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

**EL PROCESO UNIFICADO ESTA DIRIGIDO POR CASOS DE USO**

Un sistema software ve la luz para dar servicio a sus usuarios. Por tanto, para construir un sistema con éxito debemos conocer lo que sus futuros usuarios necesitan y desean.

El termino usuario no solo hace referencia a usuarios humanos sino a otros sistemas. En este sentido, el termino usuario representa alguien o algo que interactúa con el sistema que estamos desarrollando.

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso no solo inician el proceso de desarrollo, sino que le proporcionan un hilo conductor.

**EL PROCESO UNIFICADO ESTÁ CENTRADO EN LA ARQUITECTURA**

La arquitectura en un sistema software se describe mediante diferentes vistas del sistema en construcción.

La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado.

Debe haber interacción entre los casos de uso y la arquitectura. Por un lado, los casos de uso deben encajar en la arquitectura cuando se llevan a cabo. Por otro lado, la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos, ahora y el futuro.

**EL PROCESO UNIFICADO ES ITERATIVO E INCREMENTAL**

El desarrollo de un producto software comercial supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es practico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las iteraciones deben estar controladas; esto es, deben seleccionarse y ejecutarse de una forma planificada. Es por esto por lo que son miniproyectos.

La arquitectura proporciona la estructura sobre la cual guiar las iteraciones, mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración.

**LA VIDA DEL PROCESO UNIFICADO**

El proceso unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye. Cada ciclo concluye con una versión del producto para los clientes.

Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición. Cada fase a su vez se divide en iteraciones.

**FASES DENTRO DE UN CICLO**

Cada ciclo se desarrolla a lo largo del tiempo. Este tiempo, a su vez, se divide en cuatro fases. Dentro de cada fase, los directores o los desarrolladores pueden descomponer adicionalmente el trabajo -en iteraciones con sus incrementos resultantes. Cada fase termina con un hito.

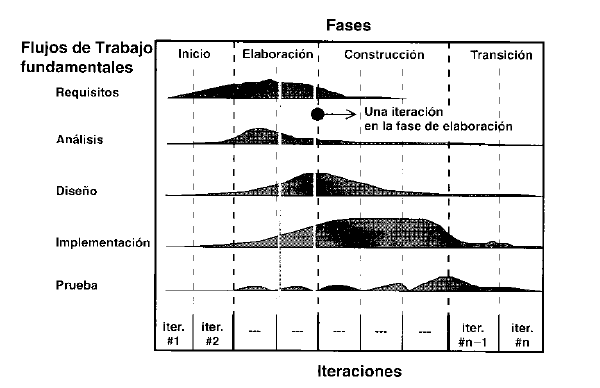
Los hitos tienen muchos objetivos. El más critico es que los directores deben tomar ciertas decisiones cruciales antes de que el trabajo pueda continuar con la siguiente fase. Los hitos también permiten a la dirección, y a los mismos desarrolladores, controlar el progreso del trabajo según pasa por esos cuatro puntos clave. Una iteración típica para por los cincos flujos de trabajo.

Durante la fase de inicio, se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto.

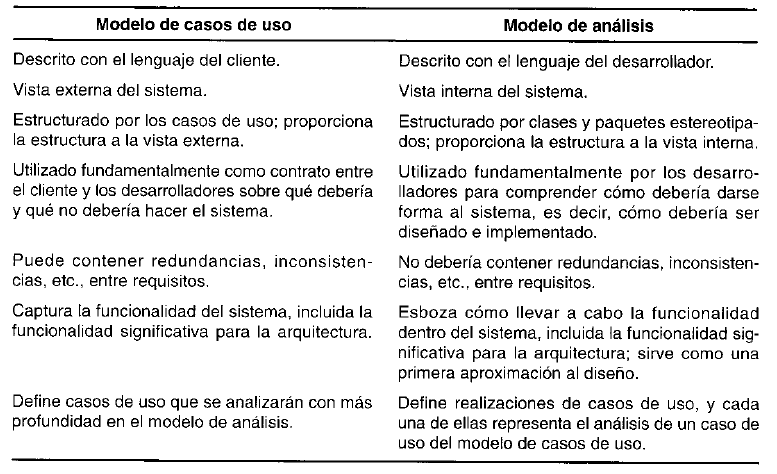
Durante la fase de elaboración, se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. Al final de la fase de elaboración, el director de proyecto está en disposición de planificar las actividades y estimar los recursos necesarios para terminar el proyecto.

Durante la fase de construcción se crea el producto. En esta fase, la línea base de la arquitectura crece hasta convertirse en el sistema completo. El grueso de los recursos requeridos se emplea durante esta fase del desarrollo.

La fase de transición cubre el periodo durante el cual el producto se convierte en versión beta. En la versión eta un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. Los desarrolladores corrigen los problemas e incorporan algunas de las mejoras sugeridas en una versión general dirigida a la totalidad de la comunidad de usuarios.



**ANALISIS**



La estructura de los requisitos que proporciona el análisis también sirve como entrada fundamental para dar formal sistema en su totalidad; esto es debido a que queremos construir el sistema como un todo mantenible, y no sólo describir sus requisitos.

El modelo de análisis nos ayuda a refinar los requisitos y nos permite razonar sobre los aspectos internos del sistema, incluidos sus recursos compartidos internos; nos proporciona una estructura centrada en el mantenimiento, en aspectos tales como la flexibilidad ante los cambios y la reutilización. Esta estructura también se utiliza como entrada en las actividades de diseño e implementación.

**POR QUE EL ANALISIS NO ES DISEÑO NI IMPLEMENTACION**

Nuestra respuesta es que le diseño y la implementación son mucho más que le análisis (refinamiento y estructuración de los requisitos), por lo que se requiere una separación de intereses. En el diseño, debemos moldear el sistema y encontrar su forma, incluyendo su arquitectura. El diseño y la implementación se preocupan en realidad de dar forma al sistema de manera que dé vida a todos los requisitos que incorpora.

**EL PAPEL DEL ANÁLISIS EN EL CICLO DE VIDA DEL SOFTWARE**

Las iteraciones iniciales de la elaboración se centran en el análisis. Eso contribuye a obtener una arquitectura estable y sólida y facilita una comprensión en profundidad de los requisitos.

Una diferencia fundamental entre clases de entidad y clases de entidad del negocio es que las primeras representan objetos manejados por el sistema en consideración, mientras que las ultimas representan objetos presentes en el negocio (y en el dominio del problema) en general. Las clases de entidad reflejan la información de un modo que beneficia a los desarrolladores al diseñar e implementar el sistema, incluyendo su soporte de persistencia. Las clases del dominio describen el contexto del sistema, y por tanto pueden incluir información que el sistema no maneja en absoluto.

El arquitecto es responsable de la integridad del modelo de análisis. El modelo de análisis es correcto cuando realiza la funcionalidad descrita en el modelo de caso, y solo es funcionalidad.

|  |  |
| --- | --- |
| CASOS DE USOS | HISTORIAS DE USUARIOS |
| Proceso Unificado | XP |
| Escrito por el analista | Escrito por cliente con ayuda del analista |
| Lenguaje Coloquial | Escrito en lenguaje del cliente |
| No se utilizan para estimar el tiempo | Sirve como estimador de tiempo para completar la historia de usuarios |
| En los Cu las pruebas se hacen después delas especificación | Se escribe la prueba de aceptación en el reverso de la historia de usuario |
| Etapa de análisis | Etapa de análisis |
| El analista le asigna prioridad a cada cu de acuerdo a los riesgos según PU | El cliente determina la prioridad con asesoramiento del analista |