Proceso y Elicitación de Requerimientos

Requisito es una necesidad documentada sobre un producto o servicio. **Establece qué debe hacer el sistema y no cómo hacerlo.**

Condición o capacidad que un usuario necesita para resolver un problema o lograr un objetivo. Algo que el sistema debe hacer o capacidad que debe poseer.

Tres tipos de requisitos en Ingeniería en Sistemas:

* **Requisito funcional:** descripción de lo que el sistema debe hacer. Algo que el sistema debe ser capaz de realizar. "el sistema debe permitir a los usuarios iniciar sesión en su cuenta" o "el sistema debe calcular la factura de un cliente y aplicar el impuesto correspondiente"

Se relacionan directamente con el software y se puede probar en su 100% si se cumple.

* **Requisito no funcional:** de rendimiento, de calidad, especifica algo sobre el propio sistema y como debe realizar sus funciones. Ejemplos: disponibilidad, testeo, mantenimiento, facilidad de uso. “el sistema debe ser capaz de manejar 1000 usuarios simultáneamente" o "el sistema debe estar disponible para el uso del usuario durante el horario de oficina".

Se relacionan con la calidad del software y no se pueden probar directamente.

* Otros tipos de limitaciones externas, que afectan de una forma indirecta al producto. Ejemplo: compatibilidad con sistema operativo, cambio de leyes.

Una colección de requisitos describe las características del sistema deseado. Se omite el cómo puede lograrse su implementación.

**Características:**

* No ambiguo: claro y preciso.
* Conciso: lenguaje comprensible, no técnico.
* Consistente: no puede entrar en conflicto con otro requisito.
* Completo: deben tener toda la información necesaria.
* Alcanzable: debe ser realista.
* Verificable: se debe verificar al 100% si fue satisfecho o no.

La etapa de requisitos es Analisis de requisitos.

Metodologías

La ingeniería de software requiere llevar a cabo numerosas tareas agrupadas en etapas. Al conjunto de etapas se denomina ciclo de vida. Las etapas comunes a casi todos los ciclos de vida son las siguientes:

**Obtención de los requisitos:** se identifica sobre que se está trabajando (tema principal), identificar los recursos que se tienen. Se entiende el contexto del negocio para poder identificar los requisitos. Comprender las necesidades y expectativas de los usuarios finales y los interesados en el software.

**Análisis de Requisitos:** en este paso se analiza la información recolectada del paso anterior, se definen los requisitos del sistema en un conjunto de requisitos funcionales y no funcionales, para asegurarse de que el software cumpla con las necesidades.

A veces los clientes creen saber todo el funcionamiento que necesita el software, pero, requiere la habilidad de un especialista para reconocer requisitos incompletos.

**Limitaciones:** el software tiene la capacidad de emular inteligencia, pero solo posee modelos creados que abarcan un conjunto de soluciones, por lo que puede llegar a ser limitado. No es capaz de emular el pensamiento humano.

**Especificación:** la especificación de requisitos es esencial para el éxito del software ya que describe el comportamiento del software una vez desarrollado. La identificación de las necesidades de negocio es fundamental para recolectar, clasificar, identificar, priorizar y especificar los requisitos del software.

Tecnicas de especificación de requisitos: Casos de uso y las historias de usuario.

**Arquitectura:** la Arquitectura de Software se enfoca en el diseño de componentes de una aplicación usando patrones de arquitectura y herramientas CASE (*Computer Aided Software Engineering*). El diseño arquitectonico se documenta utilizando diagramas: diagramades de clase, bases de datos, despliegue y secuencia. Y debe permitir la visualizacion y validacion entre las entidades del negocio.

**Programación: la duración** está relacionada al diseño previamente realizado.

**Desarrollo de la aplicación:** 5 fases para tener una aplicación eficiente:

* Desarrollo de la infraestructura (desarrollo de elementos que formaran la infraestructura)
* Adaptación del paquete (se examina en detalle el paquete (conjunto de software), sus ventajas y desventajas y se decide si usarlo, modificarlo o cambiarlo).
* Desarrollo de unidades de diseño interactivas (se realizan todas las unidades de diseño que impliquen funcionalidad y se hacen pruebas unitarias y de integración)
* Desarrollo de unidades de diseño batch (se utilizan una combinación de técnicas: diagrama de flujo, diagramas de estructuras, tablas de decisiones, de manera de plasmar de manera precisa las especificaciones asi el equipo de programadores comparte la misma idea)
* Desarrollo de unidades de diseño manuales (se enfoca en la creación de los procesos administrativos que se desarrollaran en torno a la utilización de componentes computarizados de la aplicación).

**Pruebas de software:** el objetivo de las pruebas es detectar y corregir errores y problemas antes de que el software sea puesto en producción.

**Implementación:** es el proceso de convertir las especificaciones en un sistema ejecutable.

**Documentación:** documentación del propio software, modelacion UML, diagramas de casos de uso, pruebas, manuales de usuario, manuales técnicos.

**Mantenimiento:** arreglar errores (bugs) y extender el sistema para agregar funcionalidades. 80% son mejoras de funcionalidad mientras que el otro 20% son solucionar errores.

Caso de uso

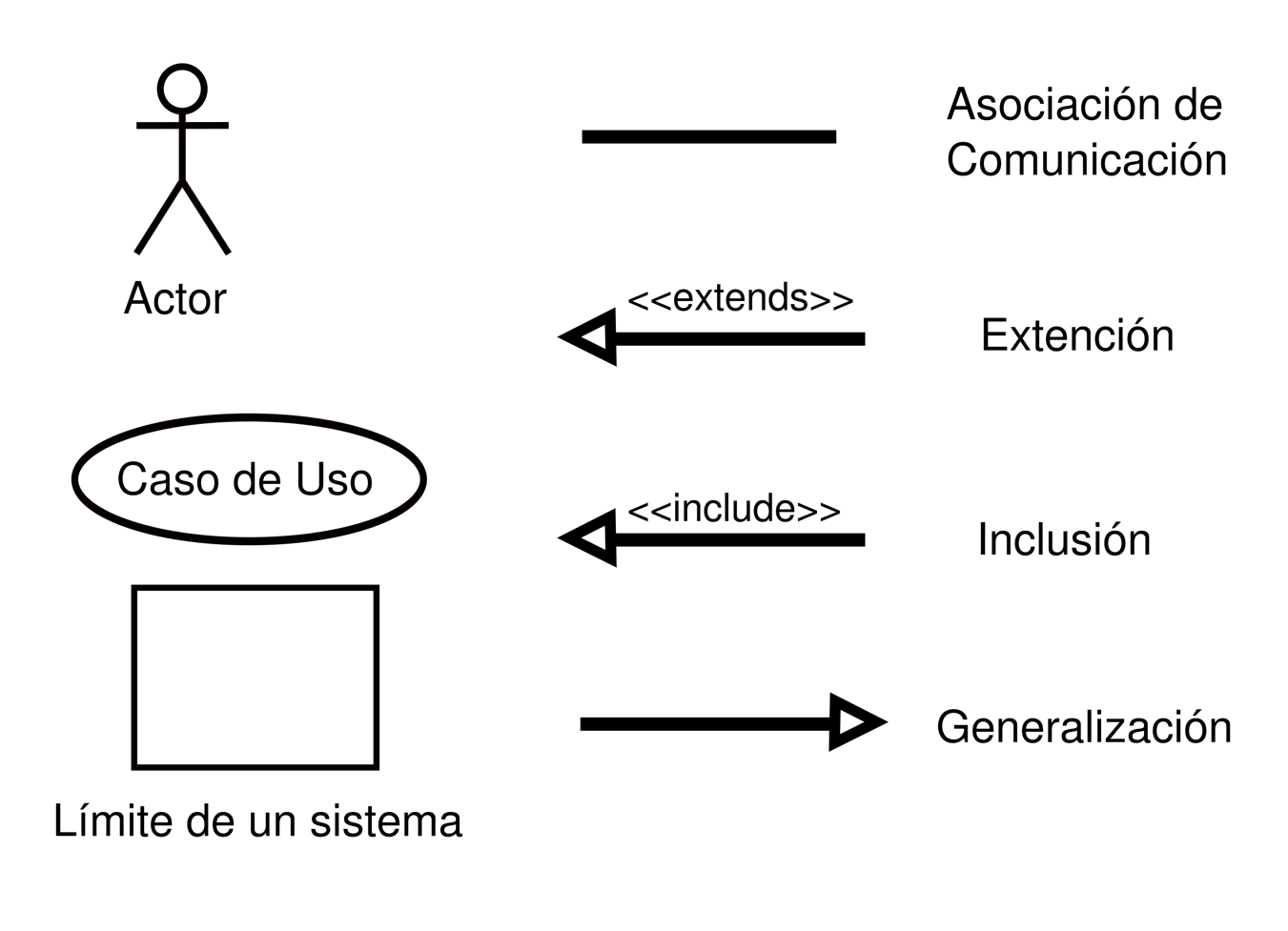
Es una técnica utilizada en ing. Software para describir cómo un usuario interactúa con la aplicación, es una descripción detallada de una acción o tarea de un usuario en un sistemay cómo el sistema responde a esa acción.

En un caso de uso se describe quién es el usuario que realiza la acción, qué acción realiza y qué respuesta espera del sistema. Por lo general, se presenta en forma de un diagrama de casos de uso que muestra los diferentes usuarios del sistema, las acciones que pueden realizar y las respuestas esperadas.

Los casos de uso son una herramienta valiosa para la definición de los requisitos del sistema y la identificación de escenarios de uso. Se usa principalmente para POO pero puede ser para otro paradigma.

Actores => personajes que aparecen en el caso de uso. Entidad externa al sistema que le demanda una funcionalidad (humanos, el tiempo, sistemas externos, cosas abstractas).

Relación => conexión entre los elementos del modelo, especialización y generalización son relaciones.



**Tipos de relaciones:**

* Comunica (<<comunicates>>): relación (asociación) entre un actor y un caso de uso, y representa la participación de un actor en un caso de uso.
* Usa (<<uses>> o <<include>>): es una relación de dependencia entre dos casos de uso. Indica que un caso de uso utiliza el comportamiento definido de otro caso de uso. Por ejemplo, caso de uso “realizar compra” puede incluir/usar el caso de uso “procesar pago”.
* Generalización: indica que un caso de uso es una variante de otro. El caso de uso especializado puede variar cualquier aspecto del caso de uso base. Por ejemplo “comprar boleto de avión” el caso especializado sería “comprar boleto de avión con ventanilla”, pero mantienen un parentesco.
* Extiende (<<extend>>): es un estereotipo de dependencia que ofrece una forma más controlada de extender un caso de uso base. Extend permite que el caso de uso especializado altere unos puntos de extensión marcados en el caso de uso base.

Los casos de uso son una herramienta de ingeniería de software que se utilizan para describir cómo un sistema debe funcionar desde la perspectiva del usuario final o del experto del campo de aplicación. Son elaborados en colaboración por analistas de requisitos y clientes y se centran en describir una única meta o tarea. Un caso de uso describe una característica del sistema y puede ser necesario especificar decenas o centenares de casos de uso para definir completamente el nuevo sistema. El grado de formalidad del proyecto y la etapa del proyecto influirán en el nivel de detalle requerido en cada caso de uso.

Los casos de uso no describen ninguna funcionalidad interna oculta del sistema, sino que muestran lo que el actor hace o debe hacer para realizar una operación. Un caso de uso debe describir una tarea del negocio que sirva a una meta de negocio, tener un nivel apropiado de detalle y ser lo suficientemente sencillo como para que un desarrollador lo elabore en un único lanzamiento.

Las situaciones que pueden darse incluyen que un actor se comunique con un caso de uso (si se trata de un actor primario la comunicación la iniciará el actor, en cambio si es secundario, el sistema será el que inicie la comunicación), un caso de uso extienda otro caso de uso y un caso de uso utilice otro caso de uso.