Requisito

- Es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio.
- Es una condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo.
- Es algo que el sistema debe hacer o cualidad que debe poseer.
- Se utilizan como datos de entrada en la etapa de diseño del producto.
- Establece **qué** debe hacer el sistema, pero no el **cómo.**

Requisito en ingeniería de software

- Existen tres tipos: Funcionales, no funcionales y limitaciones externas
- Funcionales: es una descripción de lo que el sistema debe hacer. Especifica algo que el sistema debe ser capaz de hacer
- No funcionales: especifica algo propio del sistema y como debe realizar sus funciones. Puede ser de rendimiento, de calidad. Un ejemplo puede ser la disponibilidad, testeo, mantenimiento, facilidad de uso.
- Limitaciones externas: afectan de forma indirecta al producto. Pueden ir desde la compatibilidad con un sistema operativo, hasta lo legal como regulaciones y leyes.
- También existen los pseudorequisitos que se refiere a aquellos referidos al entorno donde será instalado o implementado el sistema y que determinan en gran medida su desarrollo. Pueden ser cuestiones de software y hardware.

Características

- Los requisitos bien formulados deben satisfacer las siguientes características:
- **No ambiguo**: deben ser claros, precisos y tener una única interpretación posible.
- **Conciso**: debe estar en un lenguaje comprensible por el inversor en lugar de uno técnico y especializado. De todas formas, debe referenciar los aspectos importantes.
- **Consistente**: ningún requisito debe entrar en conflicto con otro. El lenguaje entre los requisitos debe ser el mismo.
- **Completo**: toda la información necesaria debe estar en los requisitos, y no pueden referenciar a fuentes externas.
- **Alcanzable**: un requisito debe ser un objetivo realista, alcanzable con dinero, tiempo u otro recurso.
- **Verificable**: se debe poder verificar si el requisito fue satisfecho o no. Se puede hacer a través de inspección, análisis, demostración o testeo
- Estas características son subjetivas, por lo tanto, se suele usar métricas o indicadores que pueden ser calculados de forma automática.

Análisis de requisitos

- En esta etapa se estudian los requisitos para verificar que estén correctamente adecuados a las características.
- Se enfocan e intentan solucionar las deficiencias que los requisitos puedan tener.

Requisitos funcionales

- Define una función del sistema de software o sus componentes.
- Establecen los comportamientos del software
- Pueden ser cálculos, detalles técnicos, manipulación de datos y otras funcionalidades especificas que se supone que el sistema debe cumplir
- Los requisitos de comportamiento para cada requisito funcional se muestran en los casos de uso
- Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación
- Como algunos requisitos son previos al diseño de caso de uso, los requisitos deben ser establecidos antes. Ambos se complementan en un proceso bidireccional.
- Contiene un nombre, numero de serie único y un resumen. Esta información se usa para ayudar al lector a entender por que el requisito es necesario, y para seguir al mismo durante el desarrollo del proyecto
- El núcleo de los requisitos esta en la descripción del comportamiento requerido, que debe ser clara y concisa, y que debe venir de reglas organizacionales o del negocio, o ser descubiertas por interacción con usuarios, inversores y otros expertos en la organización

Requisitos no funcionales

- Especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.
- Se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento
- El requisito no funcional son las restricciones o condiciones que impone el cliente al programa que necesita, por ejemplo el tiempo de entrega del programa, el lenguaje o la cantidad de usuarios
- Categorias:
    - Requisitos de calidad de ejecución, que incluyen seguridad, usabilidad y otros medibles en tiempo de ejecución
    - Requisitos de calidad de evolución, como testeabilidad, escalabilidad, que se evalúan en los elementos estáticos del sistema
- Ej: rendimiento, disponibilidad, durabilidad, estabilidad, accesibilidad, adaptabilidad, capacidad, etc

Casos de uso

- Es la descripción de una acción o actividad
- Un diagrama de caso de uso es una descripción de las actividades que deberá realizar alguien
- Un actor es una entidad que participa en el diagrama, externa al sistema que guarda una relación con este y que le demanda una funcionalidad (Incluye operadores humanos, pero también a otros sistemas externos)
- Un mismo individuo corresponde a uno o mas actores
- Este diagrama representa a un sistema o subsistema como un conjunto de interacciones que se desarrollarán entre casos de uso, y entre caso de uso y actor, en respuesta a un evento que inicia un actor principal (diagrama que muestra la relación entre los actores y casos de uso)
- Se utilizar para ilustrar los requisitos del sistema al mostrar como reacciona a eventos que se producen en su ámbito o en él mismo
- Tipos de relaciones:
    - Comunica: asociación entre actor y caso de uso que denota la participación del actor
    - Usa: relación de dependencia entre dos casos de uso que denota la inclusión del comportamiento de uno en otro
    - Generalización: indica que un caso de uso es una variante de otro. El caso de uso especializado puede varias cualquier aspecto de caso de uso base
    - Extiende: Ofrece una forma de extensión más controlada que la generalización. El caso de uso base declara un conjunto de puntos de extensión, y el caso de uso especializado solo puede alterar el comportamiento de los puntos de extensión marcados
- Se utiliza extends entre casos de usos cuando un caso de uso similar a otro pero que hace algo mas que este. También se usa cuando se presenta una variación del comportamiento normal.
- Se utiliza uses cuando una parte tiene un comportamiento similar en dos casos de uso y no se quiere repetir la descripción
- Se usa include cuando se repite un comportamiento en dos casos de uso y se quiere evitar la repetición
- Pueden existir relaciones de herencia entre casos de uso y actores
- Los casos de uso deben usar la lengua del usuario final o del experto, en colaboración con los analistas de requisitos y los clientes
- Cada caso debe describir una única tarea
- Un caso de uso describe una característica del sistema
- No describen ninguna funcionalidad interna (oculta al exterior) del sistema, ni explican como se implementará, sino que solo muestran lo que el actor hace o debe hacer para realizar la operación
- El caso de uso debe describir la tarea del negocio, tener un nivel apropiado de detalle y ser sencillo para que se pueda lanzar en una sola vez
- El actor primario es el que inicia la comunicación, el actor secundario es el que responde a la comunicación del sistema

Facilidades

- Los casos de uso tienen éxitos en sistemas interactivos porque logran expresar la intención que tiene el actor
- Permite que el analista se centre en las necesidades del usuario
- El analista describe los casos de uso que mayor valor aportan al negocio

Limitaciones

- No establecen completamente los requisitos funcionales
- No permiten determinar los requisitos no funcionales
- Se deben complementar con información adicional como reglas de negocio, requisitos no funcionales
- Cada caso crítico del uso debe tener un requisito no funcional centrado en el funcionamiento asociado.

Etapas del proceso

- Al conjunto de estas etapas se le denomina ciclo de vida

Obtención de requisitos

- Identificar el tema principal que motiva el inicio del estudio y creación o modificación del software
- Identificar los recursos que se tienen, que pueden ser recursos humanos o materiales
- Es importante entender el contexto del negocio para identificar adecuadamente los requisitos
- Se debe tener información sobre el problema, lo que incluye a áreas fuera de software como usuarios finales, otros sistemas, dispositivos externos, a los datos que salen del sistema (por la interfaz de usuario, interfaz de red, reportes, graficas y otros medios) y los almacenamientos de datos
- Hay que ver también los puntos críticos, lo que significa tener claro los aspectos que entorpecen y limitan el buen funcionamiento de los procedimientos actuales, los problemas más comunes, los motivos que crean insatisfacción y aquellos que deben ser cubiertos a plenitud.

Análisis de requisitos

- Extraer los requisitos del software es la primera etapa para crearlo.
- El cliente plantea las necesidades e intenta explicar lo que debería hacer el software
- El desarrollador actúa como interrogador
- El análisis de requisitos queda registrado en la Especificación de Requisitos del Sistema
- Se define un diagrama entidad/relación en el que se plasman las principales entidades que participarán en el desarrollo del software
- Esta etapa tiene como finalidad

    - Brindar al usuario todo lo necesario para que pueda trabajar en conjunto con el software desarrollado
    - Tener un control más completo en la etapa de creación de software, en cuanto a tiempo y costos
    - Utilización de métodos más eficientes que permitan el mejor aprovechamiento del software según sea la finalidad de uso del mismo
    - Aumentar la calidad del software desarrollado al disminuir los riesgos de mal funcionamiento

Especificación de requisitos

- Describe el comportamiento esperado en el software una vez desarrollado.
- El éxito del proyecto está en identificar correctamente las necesidades del negocio, así como la interacción con los usuarios funcionales para la recolección, clasificación, identificación, priorización y especificación de los requisitos del software

Arquitectura

- El arquitecto de software el que añade valor a los procesos de negocios gracias a su valioso aporte de soluciones tecnológicas
- La arquitectura de sistemas es una actividad de planeación, ya sea a nivel de infraestructura de red y hardware, o de software
- Consiste en diseño de componentes de una aplicación (entidades de negocio), generalmente utilizando patrones de arquitectura.
- Debe permitir visualizar la interacción entre las entidades del negocio y además poder ser validado, para ello se usan:
    - Diagramas de clases
    - Diagrama de base de datos
    - Diagrama de despliegue
    - Diagrama de secuencia

Desarrollo de la aplicación

- Se deben considerar cinco fases:
    - Desarrollo de la infraestructura
    - Adaptación del paquete
    - Desarrollo de unidades de diseño de interactivas
        - Tiene como objetivo principal
            - Establecer específicamente las acciones que debe efectuar la unidad de diseño
            - La creación de componentes para sus procedimientos

        - Ejecutar pruebas unitarias y de integración en la unidad de diseño
    - Desarrollo de unidades de diseño de batch
    - Desarrollo de unidades de diseño manuales

Pruebas de software

- Comprobar que el software realice correctamente las tareas indicadas en la especificación del problema.
- Una técnica es probar por separado cada módulo del software (prueba unitaria)
- Probarlo de manera integral se le llama prueba de integración
- Se considera una buena practica que las pruebas sean efectuadas por alguien distinto al desarrollador que la programó

Implementación

- Es la realización de una especificación técnica o algoritmos con un programa
- Es el proceso de convertir una especificación del sistema en un sistema ejecutable
- Es una descripción de la estructura del software que se va a implementar, los datos que son parte del sistema, las interfaces entre los componentes del sistema y algunas veces los algoritmos utilizados.

Documentación

- Es tolo lo que concierne la documentación del software y de la gestión del proyecto, pasando por modelaciones UML, diagramas de casos de uso, pruebas, manuales, entre otros

Mantenimiento

- Se dedica a mantener y mejorar el software para corregir errores descubiertos e incorporar nuevos requisitos

Ventajas

- Desde el punto de vista de gestión
    - Facilitar la tarea de seguimiento del proyecto
    - Optimizar el uso de recursos
- Desde el punto de vista de los ingenieros de software
    - Ayudar a comprender el problema
    - Permitir la reutilización
    - Facilitar el mantenimiento del producto final
- Desde el punto de vista de cliente o usuario final

- Garantizar el nivel de calidad del producto final
- Obtener el ciclo de vida adecuado para el proyecto