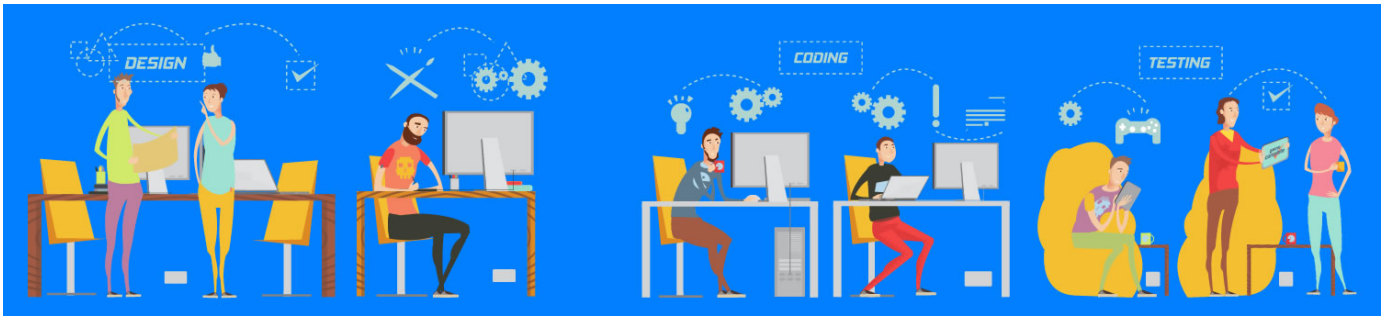
# Guía de estudio Metodología en el Desarrollo de Software


Sitio: Instituto Superior Politécnico Córdoba
Curso: Programador Full Stack - TSDWAD - 2022
Libro: Guía de estudio Metodología en el Desarrollo de Software

Imprimido por: Pablo Matias Jose MONTOYA
Día: martes, 18 octubre 2022, 7:43

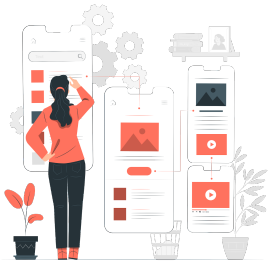
# Descripción




Vector de desarrollo web creado por macrovector - www.freepik.es

# Tabla de contenidos

# 1. Metodología de Desarrollo de Software



Según la IEEE*, el software es el "**conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación**". Es la parte de un sistema que se puede **codificar** para ejecutarse en una computadora como un conjunto de instrucciones, e incluye la **documentación** asociada necesaria para comprender, transformar y usar esa **solución**. Estos documentos describen la **organización del sistema**, explican al usuario **cómo utilizarlo** y obtener eventuales actualizaciones del producto.

*  *Desafío*: Investiga que es la IEEE, si es a nivel nacional y/o internacional, que normas o regulaciones en el software tiene?

# 1.1. Ciclo de Vida en el desarrollo de Software

El ciclo de vida del software, consta de una serie de pasos relevantes los cuales buscan garantizar que los programas creados sean eficientes, fiables, seguros y respondan a las necesidades de los usuarios finales.

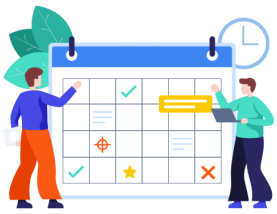
Un ciclo de vida es el conjunto de **fases [o procesos]** por las que pasa el sistema de software desde que se concibe [o inicio], se desarrolla hasta que se retira del servicio finalizando su uso.

Las fases o procesos están estandarizados, es decir que existe un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, explotación y mantenimiento de un producto de software, abarcando la vida del sistema, desde la definición de requerimientos hasta la finalización de su uso.

Podemos definir en un sentido amplio, las siguientes fases:

## Comunicación

Este es el momento en el que un cliente solicita un producto de software determinado. Nos contacta para plasmar sus necesidades concretas y presenta su solicitud de desarrollo de software. (**REQUISITOS**)



## Planificación y análisis

El desarrollo de software comienza con una fase inicial de planificación incluyendo un análisis de **requisitos**. Nos fijamos en los requisitos que piden los clientes para estudiar cuales están poco claros, incompletos, ambiguos o contradictorios. Se indaga en profundidad y se hacen demostraciones prácticas incluyendo a los usuarios clave. Los requisitos se agrupan en requisitos del usuario, requisitos funcionales y requisitos del sistema. La recolección de todos los requisitos se lleva a cabo: estudiando el software actual que tengan, entrevistando a usuarios y desarrolladores, consultando bases de datos o mediante cuestionarios. (**REQUERIMIENTOS**)

## Análisis del sistema

En este paso el equipo del proyecto asigna recursos y planifica el tiempo de duración del proyecto. Se buscan **limitaciones** del producto y se identifican los impactos del proyecto sobre toda la organización en su conjunto. (ALCANCES y LIMITACIONES)

## Diseño

En esta fase ya se comienza a visualizar la solución con la ayuda de las fases previas. Se hace un **diseño lógico** para el modelado *(DER - Modelo Relacional)* y el **diseño físico** *(Base de datos con sus tablas)*. Se crean **metadatos**[*1*], diagramas (en UML son los Diagramas de Clases, Casos de Uso, de Secuencias entre otros), pseudocódigos. La duración de esta fase varía de un proyecto a otro.

## Codificación

Esta fase también denominada 'fase de programación' o 'fase de **desarrollo**' es en la que elige el lenguaje de programación más conveniente, y se desarrollan programas ejecutables y sin errores de manera eficiente. Nuestro enfoque es construir trozos de funcionalidad. Por lo tanto, entregar unidades de funcionalidad concisa. Al final de esta fase se puede obtener un PMV (Producto mínimo viable) o el software completamente desarrollado y listo para implementarse.

## Integración

El Software puede necesitar estar integrado con bibliotecas, bases de datos o con otros programas.

## Pruebas

Esta fase junto con la fase de desarrollo entra en un ciclo continuo hasta que se completan el desarrollo y las pruebas. Probamos, probamos y luego volvemos a probar tanto como sea necesario hasta que la funcionalidad sea del 100%.

Además se hacen evaluaciones para evitar errores, incluyendo la evaluación de módulos, programas, productos, y finalmente evaluación con el cliente final. Encontrar errores y su arreglarlos a tiempo es la clave para conseguir un software confiable y eficiente. (**TESTING**)

## Implementación

Aquí se instala (**DESPLIEGUE**) el software, se evalúa la integración, la adaptabilidad, la portabilidad y se instalan las configuraciones posteriores necesarias.

### Uso y mantenimiento

Esta es una de las fases más importantes del ciclo de vida de desarrollo del software. Puesto que el software ni se rompe ni se desgasta con el uso, su mantenimiento incluye tres puntos diferenciados:

- Eliminar los defectos detectados durante su vida útil (mantenimiento correctivo).
- Adaptarlo a nuevas necesidades (mantenimiento adaptativo).
- Añadirle nuevas funcionalidades (mantenimiento perfectivo).

Si es necesario se dan nuevas funcionalidades, o se brinda mas documentación sobre como operar, además de mantener el software en perfecto estado de funcionamiento (incluso desarrollar nuevas formas que no se habían previsto) y, por ende, habrá más propuestas de mejoras.

Fuente: https://ungoti.com/es/soluciones/desarrollo-de-software/sdlc/

 ***Desafío***: Leer el siguiente material y busquen otros recursos para luego debatir en grupos las etapas mas relevantes:

https://intelequia.com/blog/post/2083/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber

Analizar si es lo mismo Requisitos y Requerimientos. Similitudes y/o diferencias.

***1*** Metadatos

# 1.2. Procesos/Modelos

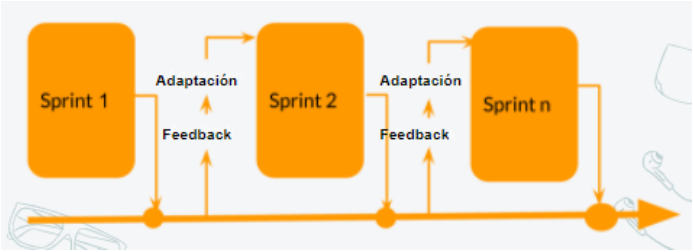
**Proceso para el desarrollo de software**

Define un conjunto completo de actividades necesarias para transformar los requerimientos  de un usuario en un producto.


Existen varios procesos o modelos de desarrollo y podríamos catalogarlos en **Secuenciales o Definidos**, y en **Empíricos**. Los primeros responden siempre de la misma forma ante una determinada entrada, mientras que los segundos dependen de la evolución y adaptación de etapas anteriores.

**Definidos o secuenciales**

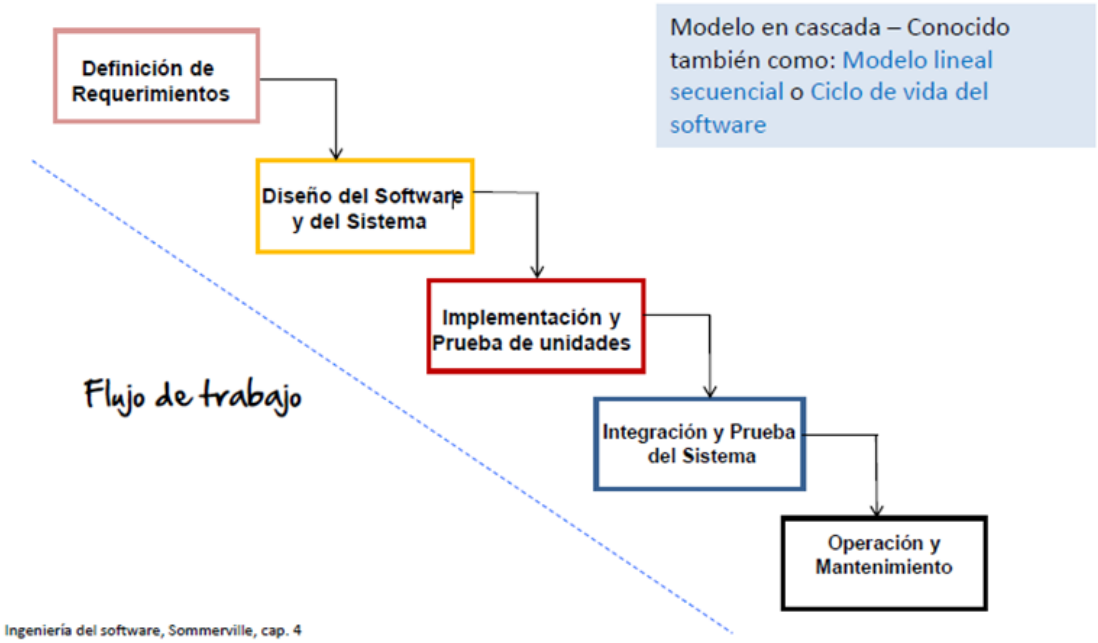

**Empíricos**




**Modelo en Cascada**
Su principal característica es que cada fase del desarrollo está bien definida y separada, siguiendo un orden secuencial en el que cada etapa depende de la finalización de la anterior y que esta última haya pasado un proceso de validación que apruebe el paso a la siguiente.

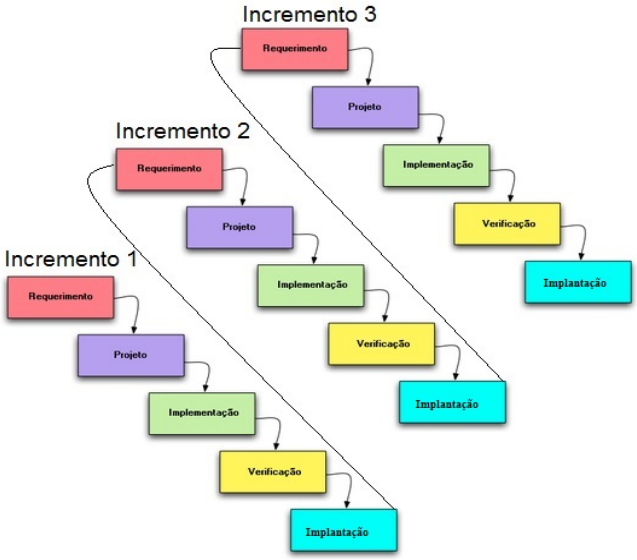
Comúnmente las etapas del modelo son:

- Análisis y definición de los requerimientos.
- Diseño del sistema y del software.
- Implementación y pruebas unitarias.
- Integración y pruebas de sistema.
- Funcionamiento y mantenimiento.




Mas explicación en este video

**Metodología Iterativa e Incremental**
La metodología iterativa e incremental deriva del proceso de desarrollo en cascada pero, con la diferencia de que aquí se admite que las etapas se solapan en tiempo con la finalidad de flexibilizar el tiempo de desarrollo total y así poder alcanzar resultados funcionales de manera temprana.




*Explicado en videos*

- Modelo Iterativo e Incremental de un proyecto

- Breve descripción de Incremental

# 1.3. Resolución de problemas a través de la computadora

**Análisis: ¿Qué ...?**

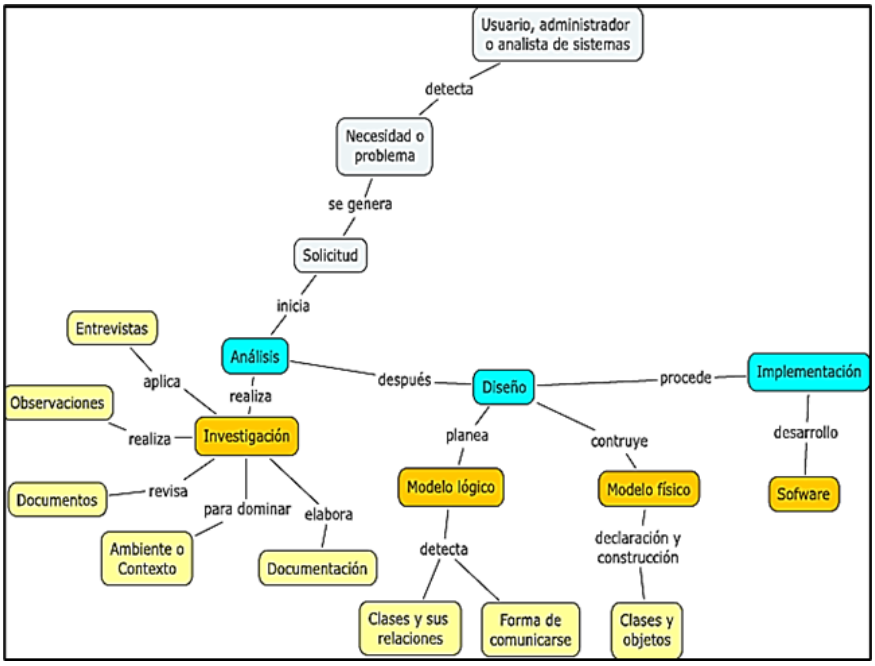
- ¿Qué problema debe resolverse?
- ¿Qué datos se requieren?
- ¿Qué resultados debe arrojar el Sistema?

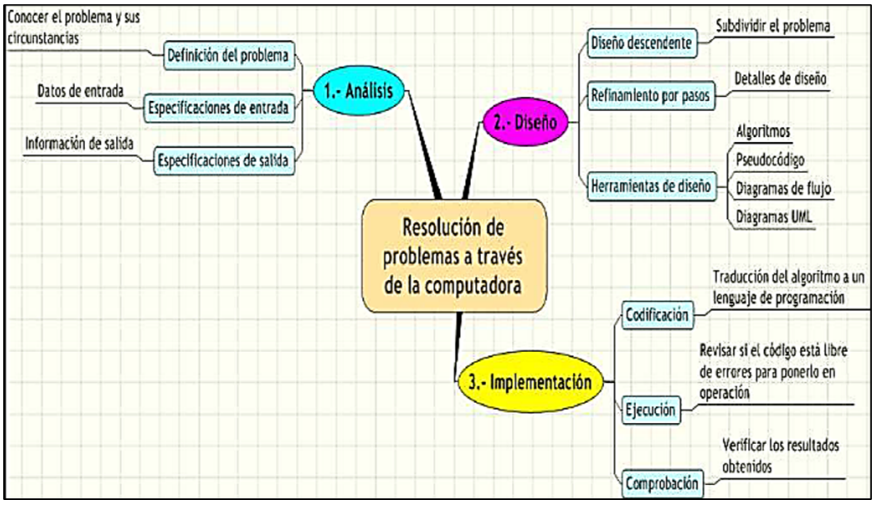
**Diseño: ¿Cómo ...?**

- ¿Cómo atacar el problema?
- ¿Cómo plantear el modelo de solución?
- ¿Cómo aplicar el modelo de solución?

**Implementación: ¿Con qué ...?**

- ¿Con qué lenguaje se desarrolla el modelo?
- ¿Con qué plataforma de desarrollo?
- ¿Con qué recursos de hardware y software?



# 1.4. Metodología Tradicional vs Metodologías Ágiles

La gestión ágil, es cada vez más popular como una alternativa a las metodologías tradicionales de gestión de proyectos, especialmente entre los equipos de desarrollo de software y TI. Cuando intenta decidir qué metodología de gestión de proyectos utilizar, es importante comprender algunas de las diferencias clave entre los enfoques de gestión de proyectos ágiles y tradicionales, ya que cada uno tiene sus propias ventajas y desventajas.

**Metodologías tradicionales**

La gestión de proyectos tradicional se centra en el enfoque lineal siendo la más usada es el modelo en **cascada** o **waterfall**,

Se caracteriza porque todas las fases se realizan en forma secuencial, es decir, que las etapas se llevan a cabo una detrás de otra, en este enfoque, los requisitos son fijos, y el presupuesto y el tiempo se acuerdan antes. Por esta razón, los equipos a menudo enfrentan problemas de presupuesto y cronograma con este enfoque. No puede utilizar la gestión de proyectos tradicional para desarrollar productos complejos, ya que este enfoque no deja espacio para cambiar los requisitos.

*Video Metodología Tradicional*

**Metodologías ágiles**

En la gestión ágil de proyectos, los proyectos se dividen en iteraciones cortas. La iteración dura un máximo de un mes calendario. Y después de cada iteración, obtendrá un nuevo incremento de producto liberable. La gestión ágil de proyectos se centra más en implementar los comentarios del cliente y revisar el producto periódicamente. La colaboración con el cliente es un factor vital en la agilidad. No sigue un plan a ciegas y responde a los cambios rápidamente, estableciendo un marco que permite acortar los tiempos de desarrollo, eliminar la incertidumbre, mejorar la eficiencia del equipo y obtener el resultado deseado. Entre los avances promovidos por Agile se encuentran la reducción de la burocracia y la microgestión en los proyectos, la rápida adaptación a los cambios de orientación, la reducción del tiempo de comercialización de una funcionalidad, producto o servicio, así como también la eliminación de residuos mediante ciclos frecuentes de pruebas y validación

Por ejemplo, **Scrum, Kanban, LeSS, SAFe y Scrumban** son excelentes ejemplos de métodos populares de gestión de proyectos ágiles.

**Video Metodología Ágil**

**Video síntesis uso de Scrum**

## Aplicando Scrum en nuestros Proyectos de Software

El **ciclo de Scrum** empieza con una reunión de *stakeholders*, durante la cual se crea la visión del proyecto. Después, el *Product Owner* desarrolla un Backlog Priorizado del Producto (*Prioritized Product Backlog*) que contiene una **lista requerimientos del negocio** y del proyecto por orden de importancia en forma de una historia de usuario. Cada **sprint** empieza con una reunión de planificación del sprint (*Sprint Planning Meeting*) durante la cual se consideran las *historias de usuario de alta prioridad* para su inclusión en el sprint. Un sprint generalmente tiene una duración de una a tres semanas durante las cuales el Equipo Scrum trabaja en la creación de entregables (del inglés deliverables) en incrementos del producto.
Durante el sprint, se llevan cabo *Daily Standups* muy breves y concretos, donde los miembros del equipo discuten el *progreso diario*. Hacia el final del sprint, se lleva a cabo una Reunión de Revisión del Sprint (*Sprint Review Meeting)* en la cual se proporciona una *demostración* de los entregables al Product Owner y a los stakeholders relevantes.
El Product Owner acepta los entregables sólo si cumplen con los criterios de aceptación predefinidos. El ciclo del sprint termina con una Reunión de Retrospectiva del Sprint (*Retrospect Sprint Meeting*), donde el equipo analiza las formas de mejorar los

procesos y el rendimiento a medida que avanzan al siguiente sprint.

Ref. Metodologías Agiles y Tradicionales

# 2. ANALISIS - Requerimientos de Software

## ¿Qué son los requerimientos funcionales?

La guía del **Business Analysis Body of Knowledge (BABOK)**, proporciona la siguiente definición para los requerimientos funcionales de una solución:

> *"Los requerimientos funcionales son las descripciones explícitas del comportamiento que debe tener una solución de software y que información debe manejar."*

Por lo tanto, los requerimientos funcionales:

- Expresan las capacidades o cualidades que debe tener la solución para satisfacer los requerimientos de los **interesados de proyecto (Stacke Holders)**
- Se expresan en términos de cuál debe ser el comportamiento de la solución y que información debe manejar.
- Deben proporcionar una descripción lo suficientemente detallada para permitir el desarrollo e implementación de la solución.
- Son los que más influyen en si la solución será aceptada o no por los usuarios.

## Importancia de definir los requerimientos funcionales

Los problemas y errores en la gestión de requerimientos funcionales son citados como una de las causas más frecuentes que ocasionan insatisfacción de las expectativas de los interesados en proyectos de software.

Profundizando en las causas de estos problemas, las situaciones observadas con mayor frecuencia son:

- **Requerimientos funcionales** con descripciones muy ambiguas, produciendo interpretaciones inadecuadas por parte del equipo de desarrollo.
- El requerimiento funcional no fue entendido adecuadamente cuando fue levantado con el interesado, pasando información incorrecta al equipo de desarrollo.
- En su forma original, el requerimiento funcional no era **factible técnicamente** y el equipo de desarrollo realizó modificaciones que no fueron aprobadas por los interesados.

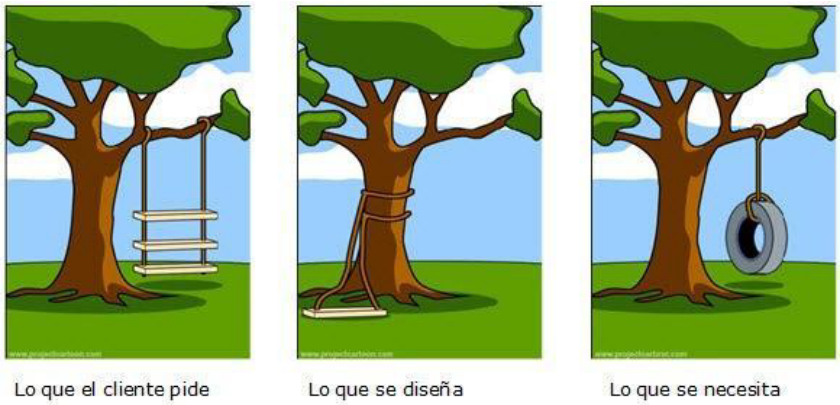
Es fundamental la aplicación de metodologías probadas de gestión de requerimientos funcionales que eviten que estos problemas sucedan, algunas de las prácticas más recomendadas son:

- *Ante la presencia de ambigüedades,* solicitar información adicional, mesas de trabajo o reuniones con los interesados. Es menos costoso esperar a obtener una descripción que aclare las dudas antes que asumir y avanzar en el desarrollo.
- *Validar* las descripciones escritas de los requerimientos funcionales antes de comenzar su desarrollo.
- Cualquier duda que presente el equipo de desarrollo buscar comunicación con los interesados para su resolución.
- Si durante el desarrollo se presentará la necesidad de **modificar algún requerimiento funcional** debido a razones técnicas, solicitar mesas de trabajo o reunión, no proceder con un desarrollo e implementación que no ha sido validado por los interesados.

Siguiendo estas prácticas podemos asegurar una mejor aceptación de los requerimientos funcionales cuando llegue el momento de la implementación.

# 2.1. Especificación de Requerimientos

## Requerimientos de Software




- Representan el conjunto completo de resultados a ser obtenidos utilizando el sistema.
- Deben mostrar todo lo que el sistema debe hacer más todas las restricciones sobre la funcionalidad.
- Como resultado de esta actividad se obtiene un documento. **Template -> IEEE Std. 830-1998**
- La especificación de requerimientos es muy importante porque:
  - Define claramente el alcance.
  - Puede ser utilizado como "**contrato**" con el cliente.
- Los requerimientos definen el "**que**" del sistema.
- El diseño define el "**como**" del sistema.
- Durante el análisis de requerimientos no se consideran descripciones específicas de la implementación (a menos que el cliente lo pida). Ej: lenguajes, base de datos, etc.
- Los requerimientos deben centrarse en el cliente/usuario y el problema.

### Requerimientos funcionales

- Enunciados acerca de servicios que el SE debe proveer, cómo debería reaccionar el sistema a entradas particulares y cómo debería comportarse el sistema en situaciones específicas
- En algunos casos, también explican lo que no debe hacer el sistema.

### Requerimientos no funcionales

- Limitaciones sobre servicios o funciones que ofrece el sistema
- Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares
- Se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema

### Obtención de requerimientos

Existen diferentes metodologías para realizar el análisis de requerimientos y el posterior diseño de la solución.

- Una metodología utilizada en la actualidad es:

  - *- Object Oriented Analysis*
  - *- Object Oriented Design*

- Es una metodología basada en teoría de objetos:

- Clases
- Atributos
- Operaciones
- Relaciones
  - Asociación
  - Agregación y Composición
  - Generalización

**Cuando se utiliza la metodología "Object Oriented Analysis",** un procedimiento muy utilizado es el siguiente:
- Obtener la historia del cliente, donde se explica la necesidad a resolver.
- Identificar todos los sustantivos relevantes del texto y transformarlos en clases.
- Identificar todos los adjetivos relevantes del texto y transformarlos en atributos de las clases.

- Identificar todos los verbos relevantes del texto y transformarlos en métodos de las clases.
- Identificar las relaciones entre las clases.


### Validación de requerimientos

Proceso por el cual se determina si la especificación es consistente con las necesidades del cliente. Incluye verificar trazabilidad entre la especificación y el documento de requerimientos.

Se deben hacer las siguientes verificaciones:

○ **Validez**: que el usuario valide que es lo que quiere.

○ **Consistencia**: que no haya contradicciones.

○ **Realismo**: que pueda implementarse.

○ **Verificabilidad**: diseñar un conjunto de pruebas para demostrar que el sistema cumple con los requerimientos.