MODULO 1 Ingeniería de Requisitos

- Ciclo de Vida del Software

El ciclo de vida del software (SDLC) es un proceso utilizado para construir y evolucionar software, compuesto por diversas fases que aseguran el cumplimiento de los requisitos y procedimientos de desarrollo. La normativa ISO/IEC/IEEE 12207:2017 proporciona un marco común para estos procesos, involucrando a los interesados y garantizando la satisfacción del cliente. Los elementos clave del ciclo de vida incluyen:

1. \*\*Fases\*\*: Conjunto de actividades orientadas a un objetivo dentro del proyecto, que agrupa tareas y establece requisitos temporales para la asignación de recursos.

2. \*\*Entregables\*\*: Productos intermedios, materiales o inmateriales, que permiten evaluar el progreso del proyecto mediante la comprobación de su adecuación a los requisitos establecidos.

#### Fases del Modelo de Ciclo de Vida del Software

1. \*\*Planificación\*\*: Definición del problema, alcances y objetivos del software.

- Objetivos: Estudio de viabilidad y planificación detallada.

2. \*\*Análisis (definición de requisitos)\*\*: Definición de los requisitos que guiarán el desarrollo del proyecto.

- Objetivos: Conocer los requisitos, asegurar que son alcanzables y formalizar un acuerdo con el cliente.

3. \*\*Diseño\*\*: Estudio de opciones de implementación y estructura general del software.

- Objetivos: Identificar soluciones tecnológicas, asignar recursos, establecer métodos de validación y ajustar especificaciones.

4. \*\*Implementación\*\*: Desarrollo del software según los diseños especificados.

- Objetivos: Construcción del software siguiendo las especificaciones y soluciones tecnológicas definidas.

5. \*\*Pruebas\*\*: Detección y corrección de fallos en las etapas anteriores.

- Objetivos: Ajustar errores o inconsistencias para asegurar la funcionalidad correcta del software.

6. \*\*Mantenimiento\*\*: Incluye mantenimiento correctivo, adaptativo y perfectivo.

- Objetivos: Asegurar que el uso del software es el pretendido y realizar ajustes y mejoras continuas.

#### Paradigmas de los Modelos de Ciclo de Vida del Software

1. \*\*Paradigma Tradicional\*\*: Lineal, completando cada fase antes de avanzar a la siguiente.

- \*\*Desventaja\*\*: Pérdida de tiempo al encontrar errores en fases avanzadas, ya que se debe retroceder y reestructurar el trabajo.

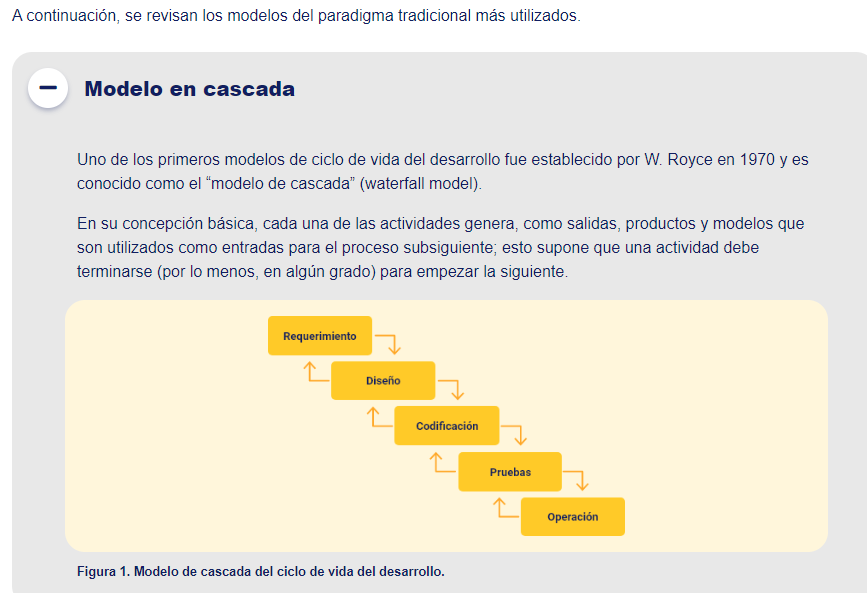
2. \*\*Paradigma Orientado a Objetos\*\*: Enfocado en la creación de clases, análisis de requisitos y diseño.

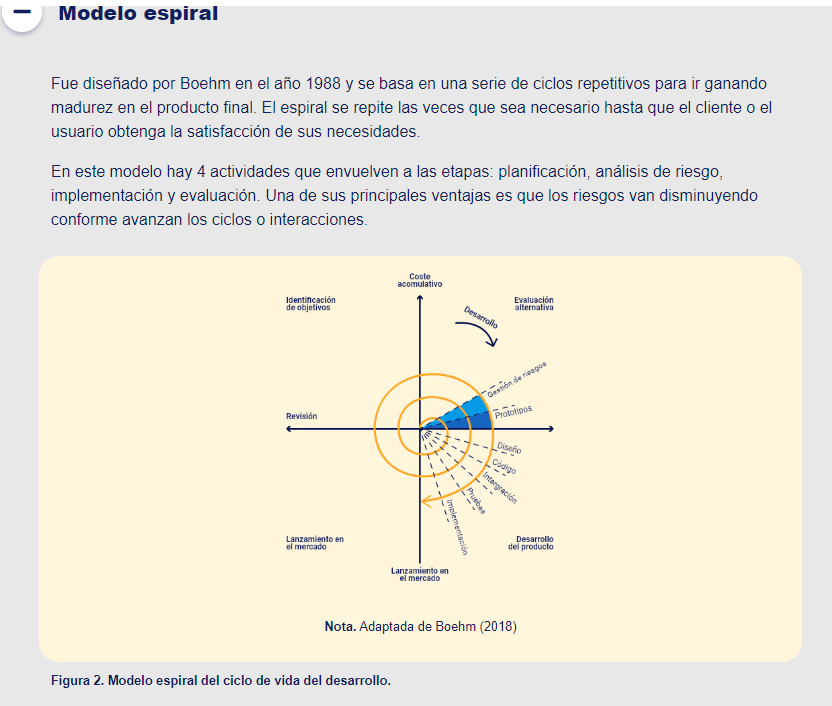
- \*\*Ventaja\*\*: Reutilización del código fuente en otros proyectos.

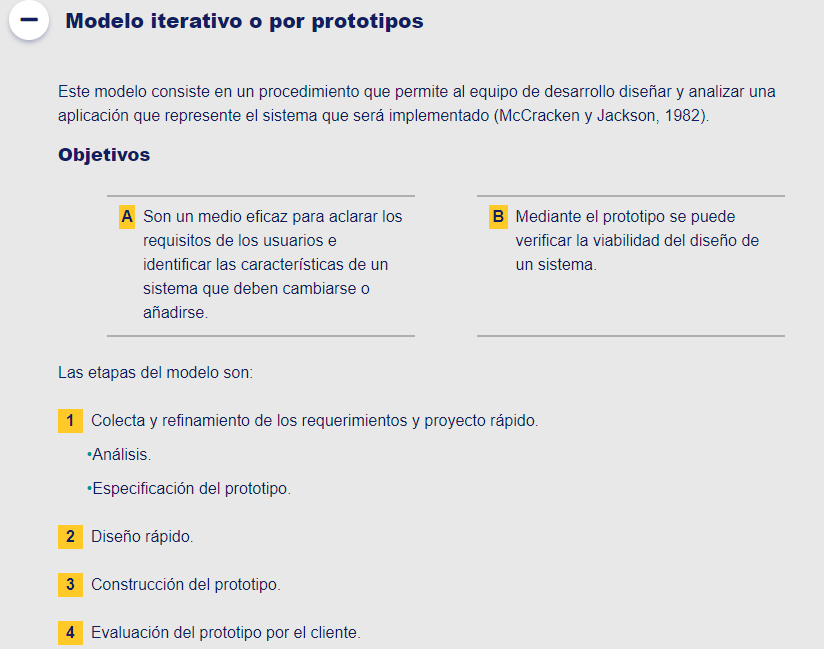
3. \*\*Paradigma de Desarrollo Ágil\*\*: Desarrolla proyectos rápidamente, simplificando procesos y manteniendo al cliente involucrado activamente.

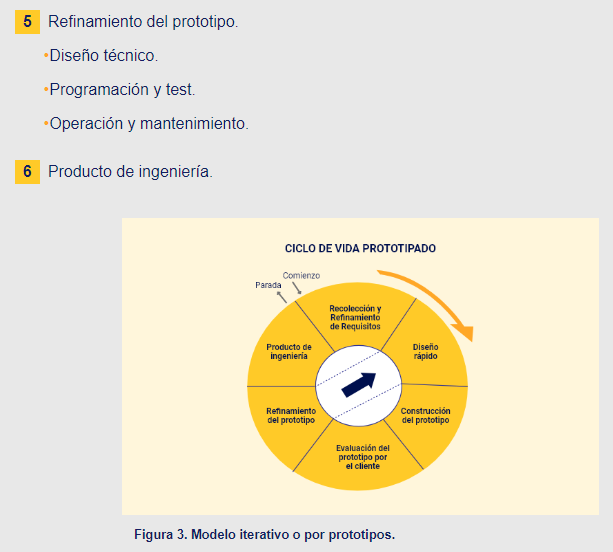
- \*\*Ventaja\*\*: Flexibilidad y adaptabilidad, con mejoras constantes basadas en la retroalimentación del cliente.

Estos paradigmas ayudan a estructurar y documentar cada etapa del desarrollo, asegurando la validación y calidad del producto final.

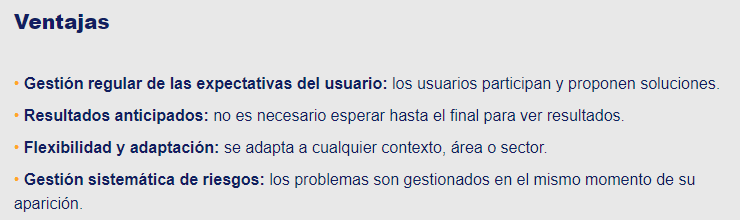


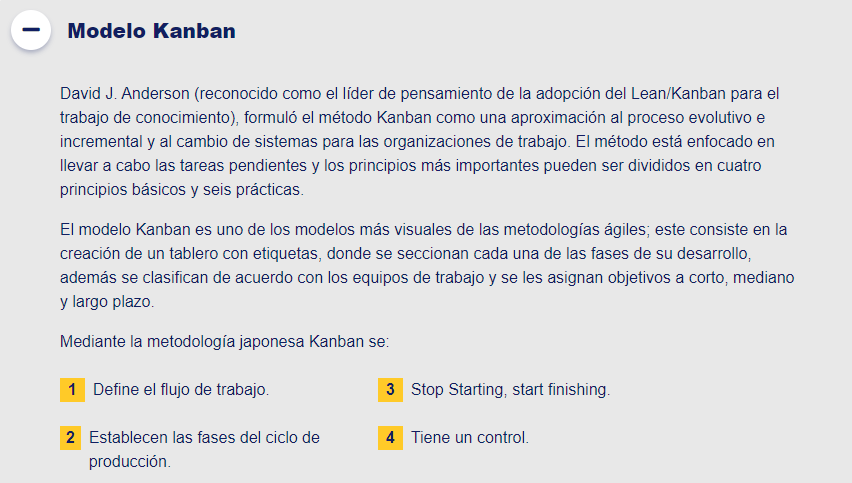


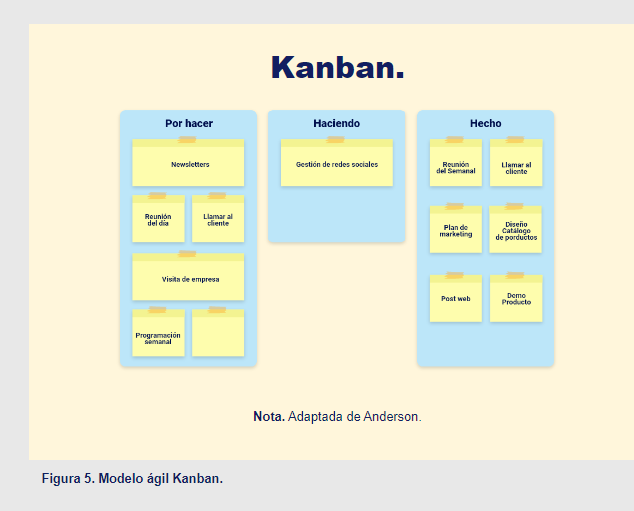


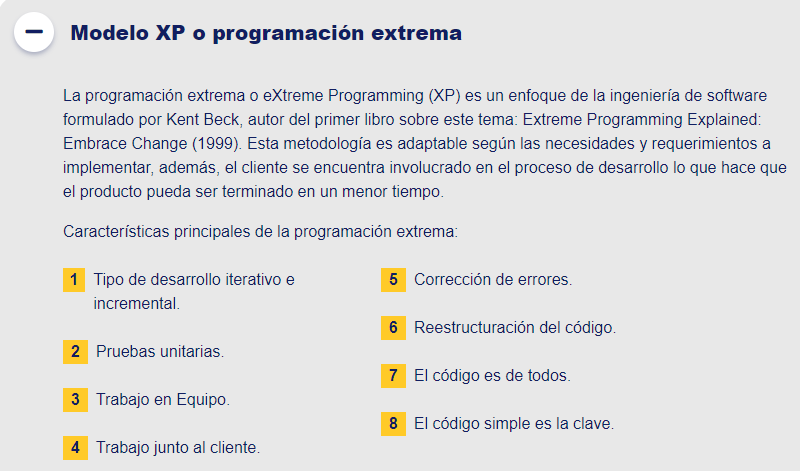




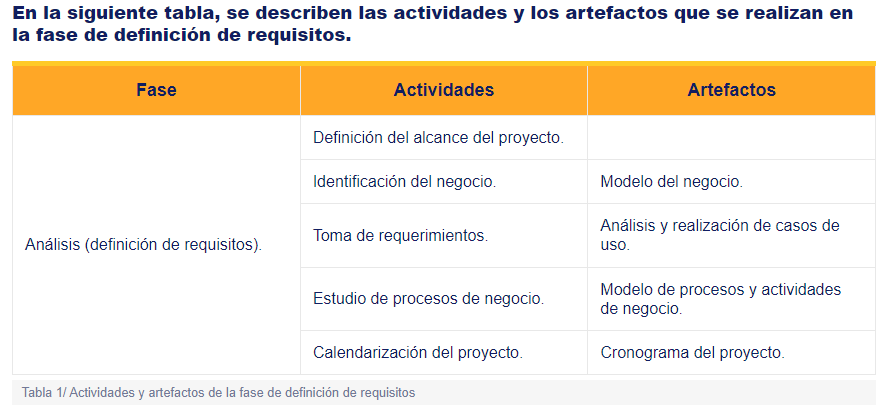






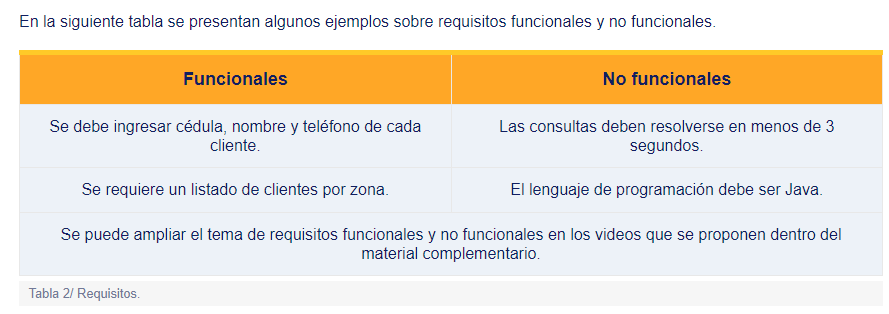




**La fase de definición de requisitos en el ciclo de vida del software es crucial. Aquí se recopilan, examinan y formulan los requisitos del cliente, así como se verifican posibles restricciones. Este análisis es esencial para entender qué se necesita realmente y definir adecuadamente los requerimientos del sistema. Es un desafío común que al principio el cliente no siempre tenga claro lo que necesita, por lo que se utilizan diversas técnicas para clarificarlo. Además, los requerimientos suelen ser inestables, con aproximadamente el 25% cambiando antes de que el sistema se utilice. Por ello, es importante implementar prácticas efectivas para gestionar y controlar la evolución de los requisitos del sistema.**

**Los requisitos son condiciones o capacidades necesarias para resolver un problema o alcanzar un objetivo específico. Son fundamentales en el ciclo de vida del software porque establecen el alcance del trabajo, informan sobre las necesidades de los usuarios y afectan el éxito del proyecto. Deben ser necesarios, completos, consistentes, correctos, factibles, modificables, priorizados, verificables, rastreables y claros.**

**Se clasifican en requerimientos de usuario, que describen los servicios esperados del sistema, y requerimientos de sistema, que detallan las funciones y restricciones del sistema. A su vez, los requerimientos de sistema se dividen en funcionales y no funcionales, que abarcan restricciones sobre servicios, desarrollo y estándares.**



**La ingeniería de requisitos se enfoca en desarrollar una especificación completa y consistente para un sistema, asegurando un entendimiento común entre todas las partes. Se compone de cuatro etapas: elicitación, análisis, especificación y validación.**

**En la etapa de elicitación, se descubren los requisitos del sistema, se comprende el problema y se consensúan los requisitos con los clientes. En el análisis, se detectan conflictos y se profundiza en el entendimiento del problema. La especificación documenta los requisitos con detalle, utilizando la notación UML. La validación asegura que los requisitos correspondan a las necesidades de los clientes y usuarios, evitando errores en el desarrollo del producto.**