

# ***Ingeniería y Calidad***

## ***De Software***

### ***1 – Introducción a la Ingeniería del Software***

#### **Software ¿Qué es?**

El software son programas de cómputo y documentación asociada. Los productos de software se desarrollan para un cliente en particular o para un mercado en general.

El buen software debe entregar al usuario la funcionalidad y el desempeño requeridos, y debe ser sustentable, confiable y utilizable.

#### **Ingeniería de Software**

La ingeniería de software es una disciplina de la ingeniería que se interesa por todos los aspectos de la producción del software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación.

La ingeniería de sistemas se interesa por todos los aspectos del desarrollo de sistemas basados en computadoras, incluidos hardware, software e ingeniería de procesos. La ingeniería de software es parte de este proceso más general.

#### **Antecedentes**

El software era desarrollado principalmente para aplicaciones científicas y militares. Los programas eran relativamente simples, escritos por los mismos científicos que usaban las computadoras. A medida que las computadoras se volvieron más accesibles, se empezaron a usar en diversos sectores como los bancos, la aviación, la administración pública y la industria. Esto trajo una necesidad de sistemas más complejos. Las técnicas de programación eran rudimentarias, y no existían metodologías formales para gestionar proyectos grandes. Esto llevó a problemas de fiabilidad, mantenimiento y escalabilidad.

## **Crisis del Software**

Es un término popularizado en 1968 en una conferencia de la OTAN, donde se describen el conjunto de principales problemas del software de aquella época.

- Proyectos que no se completaban o se entregaban tarde
- Costos que se disparaban más allá de lo presupuestado
- Software lleno de errores y difíciles de mantener
- Falta de personal capacitado y metodologías organizadas

## **Estado actual**

La crisis dio origen a la Ingeniería de Software como disciplina formal. Hoy contamos con metodologías como Agile, DevOps, y modelos como Scrum y Kanban. Lo que no significa que no se presenten desafíos en la actualidad:

- Proyectos aún pueden exceder tiempos y presupuestos
- La complejidad sigue creciendo con la inteligencia artificial, la nube y los sistemas distribuidos
- La seguridad y la privacidad son ahora preocupaciones centrales

## **Disciplinas de la Ingeniería de Software**

- **Disciplinas Técnicas**
  - ❖ Requerimientos
  - ❖ Análisis y Diseño
  - ❖ Construcción
  - ❖ Prueba
  - ❖ Despliegue
- **Disciplinas de Gestión**
  - ❖ Planificación de Proyecto
  - ❖ Monitoreo y Control de Proyectos
- **Disciplinas de Soporte**
  - ❖ Gestión de Configuración de Software
  - ❖ Aseguramiento de Calidad
  - ❖ Métricas

## **Proceso de Software**

Serie de actividades, acciones y tareas relacionadas que la gente usa para conducir el desarrollo o mantenimiento de un producto de software y sus productos

asociados. Una **actividad** busca lograr un objetivo amplio y se desarrolla sin importar el dominio de la aplicación, tamaño del proyecto, complejidad del esfuerzo o grado de rigor con el que se usara la ingeniería de software. Una **acción** es un conjunto de tareas que producen un producto importante del trabajo. Una **tarea** se centra en un objetivo pequeño, pero bien definido que produce un resultado tangible.

El Proceso de desarrollo lo define la organización en función de los objetivos y de las características del software.

- **Proceso Definido**

Inspirado en las líneas de producción. Asume que podemos repetir el mismo proceso una y otra vez, indefinidamente, y obtener los mismos resultados. La administración y control provienen de la predictibilidad del proceso definido. Esto es difícilmente aplicable al software, ya que no es tan definible. No podemos simplificar el software a una entrada de inputs que producen el mismo output, es decir, el output no es predecible en el software.

Muchas corrientes y procesos confían en estas premisas para llevar a cabo un proyecto de software.

- **Proceso Empírico**

Viene del empirismo, centra su foco en la experiencia del usuario, de negocio y la experiencia técnica. Asume procesos complicados con variables cambiantes, lo que implica la necesidad de adaptación para garantizar un software de calidad. Cuando se repite el proceso, se pueden llegar a obtener resultados diferentes. La administración y el control es a través de inspecciones frecuentes y adaptaciones. Son procesos que trabajan bien con procesos creativos y complejos.

El proceso empírico no tiene un punto de mejora tan tangible como el definido. Este trabaja sobre distintos aspectos centrándose en las personas, las cuales capitalizan la experiencia. Esto lo hacen a través de un ciclo de retroalimentación que permite la adaptación y mejora, aprendiendo de la experiencia. Los procesos empíricos trabajan con un único tipo de ciclo de vida, el iterativo.

## **Proyecto de Software**

Unidad de gestión del trabajo. Es la instancia específica en la que se aplica el proceso para desarrollar un producto. Implica: Planificación, Gestión de riesgos,

Seguimiento de tareas, y Coordinación de equipo. Dirigidos a obtener resultados y ello se refleja a través de objetivos, los cuales no deben ser ambiguos debido a que guían a dicho proyecto. Las características de un proyecto son:

- Resultado Único
- Orientado a un objetivo
- Fecha de Inicio y Fin
- Elaboración gradual
- Tareas interrelacionadas

## **Producto**

Software desarrollado como resultado del proyecto. Puede ser un sistema a medida o un producto genérico. Su calidad depende de la ejecución del proceso, la gestión del proyecto, y el entendimiento y cumplimiento de los requerimientos.

## **Relación Producto – Proyecto – Proceso en la Gestión de Proyecto**

El **proceso** define el “cómo”, se aplica en el proyecto para crear el producto. El **proyecto** ejecuta el “qué”, usa el proceso para lograr el producto. El **producto** es el resultado, refleja la calidad del proceso y la gestión del proceso.

- **El producto es intangible**

El software es intangible, no se puede ver ni tocar. Los administradores de proyectos de software no pueden constatar el progreso con solo observar el artefacto que se construye. Mas bien, ellos se apoyan en otros para crear la prueba que pueden utilizar al revisar el progreso del trabajo.

- **Los grandes proyectos de Software con frecuencia son excepcionales**

Incluso los administradores que cuentan con vasta experiencia pueden encontrar difícil anticiparse a los problemas. En conjunto con los vertiginosos cambios tecnológicos en computadoras y comunicaciones, pueden volver obsoleta la experiencia de un administrador. Las lecciones aprendidas de proyectos anteriores pueden no ser aplicables a nuevos proyectos.

- **Los procesos de software son variables y específicos de la organización**

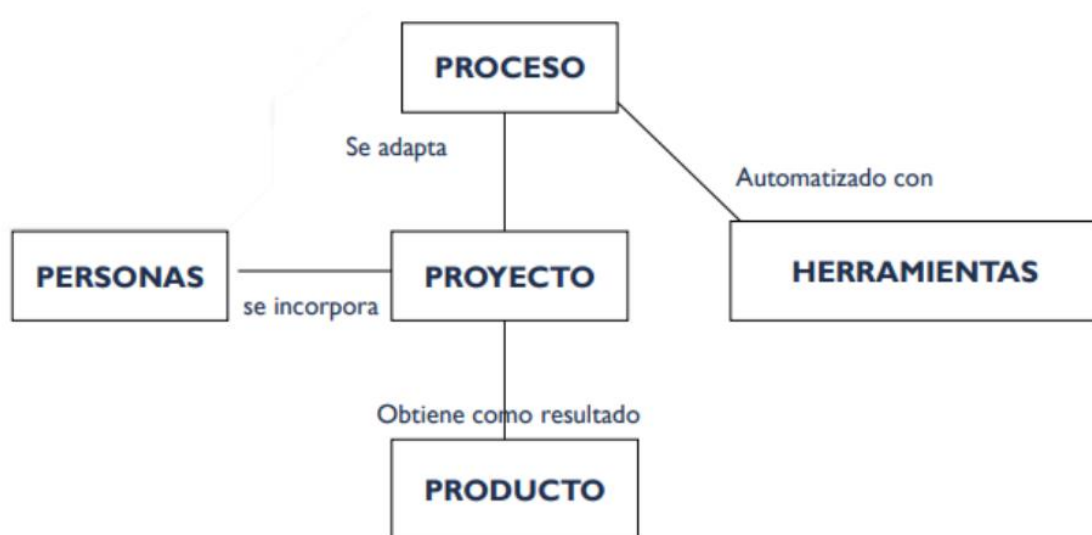
Los procesos de software varían considerablemente de una organización a otra. No es posible predecir de manera confiable cuando un proceso de software particular conducirá a problemas de desarrollo. Esto es especialmente cierto si

el proyecto de software es parte de un proyecto de ingeniería de sistemas más amplio.

### Componentes de un proyecto

Se suele decir que el proyecto es una instancia del proceso en un ciclo de vida.

Un **proyecto** es llevado a cabo por **personas** que implementan **herramientas** para automatizar los **procesos** y que tiene como resultado un **producto**.



### Administración de Proyectos

Aplicación de conocimientos, habilidades, herramientas y técnicas a las actividades del proyecto para satisfacer los requerimientos del proyecto y poder obtener como resultado el producto esperado.

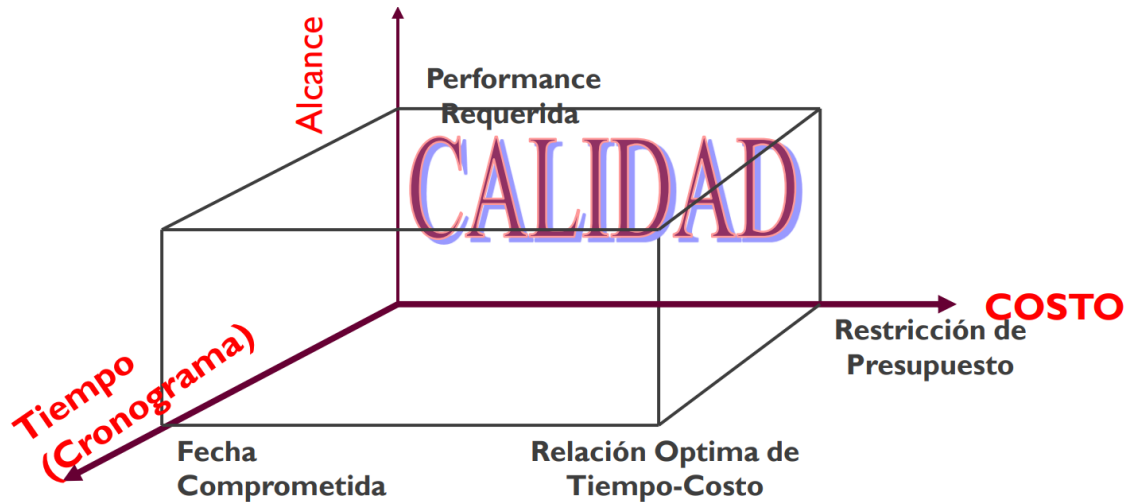
### La triple restricción

Es una de las bases de la gestión tradicional. Plantea que un proyecto va a tener una **restricción de tiempo**, una **restricción de presupuesto** y un **alcance**. Estas tres variables están íntimamente relacionadas de manera tal que, si se modifica una, se deben modificar una de las otras o ambas. El balance de las mismas afecta la calidad del proyecto, y debemos tener en cuenta que, **la calidad del producto no se negocia**.

- **Alcance**

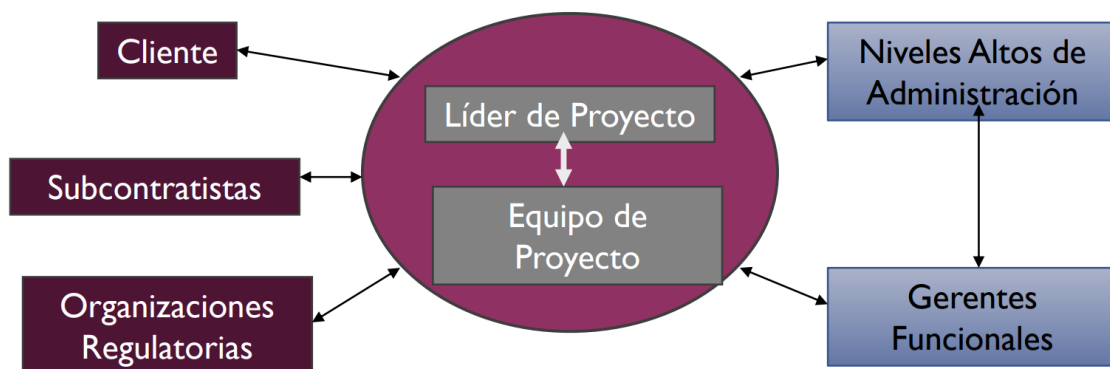
Son los requerimientos que el cliente expresa, el objetivo, lo que quiero alcanzar.

- **Tiempo**  
Tiempo que debería llevar completar el proyecto.
- **Costo**  
Costo en recursos y dinero.



### Roles dentro de un proyecto

En la gestión tradicional, nuestro equipo esta formado por un **equipo de proyecto** y un **líder de proyecto** (PM).



En los procesos empíricos, el PM no es necesario ya que los proyectos se autogestionan. Es un rol propio de la gestión tradicional.

### Plan de Proyecto

La idea es que sea algo vivo, que se vaya nutriendo y corrigiendo a medida que el proyecto se lleva a cabo. Debe estar en permanente actualización, y a que en el entorno de desarrollo del proyecto se encuentran con variaciones no estimadas al determinar el plan debido a la incertidumbre inicial.

En un plan de proyecto se deben definir las siguientes cosas:

- **Objetivo del proyecto**  
Define el ¿Qué? Debe ser claro y alcanzable.
- **Alcance del proyecto**  
Para definir el alcance del proyecto, primero se debe definir el alcance del producto.

Alcance del proyecto	Alcance del producto
Esto todo el <b>trabajo</b> , y solo el trabajo, que debe hacerse para entregar el producto o servicio con todas las características y funciones especificadas en el objetivo. Se mide contra el plan del proyecto	Son todas las <b>características</b> que pueden incluirse en un producto o servicio. Se mide contra la especificación de requerimientos

- **Definir el proceso y el ciclo de vida**  
Cuando inicia el proyecto se debe definir qué proceso usar y con que ciclo de vida. Se define el ¿Cómo?
- **Definir el equipo de proyecto**  
Define el ¿Quién?
- **Estimación**
  - ❖ **Tamaño**  
Define el producto a construir. Casos de Uso x Complejidad.
  - ❖ **Esfuerzo**  
Horas Perona Lineal del proyecto.
  - ❖ **Tiempo**  
Tiempos del proyecto.
  - ❖ **Costos**  
Determinar un presupuesto necesario.
  - ❖ **Recursos críticos**  
Tanto humanos como físicos que van a hacer falta.
- **Calendarización – Programación**  
Determinar días y horas que se van a trabajar, además de cuantas personas.
- **Riesgos**  
Identificar los riesgos mas probables o que mas pueden impactar. Luego, se gestionan, generando acciones para disminuir el impacto o evitar la ocurrencia. Los que no se pueden evitar, se debe encontrar la manera de aliviarlos. A medida que el proyecto avanza, los riesgos pueden aumentar, disminuir, ser mas probables o aparecer nuevos.

- **Métricas**

Permiten identificar si un proyecto está en línea con lo planificado o si está desviado. Se pueden clasificar en:

- ❖ **Métricas de Proceso**

Permiten saber si como organización se trabaja bien o mal, y que pasa en términos organizacionales. Son las mismas métricas del proyecto, pero despersonalizadas de un proyecto en particular.

- ❖ **Métricas de Proyecto**

Permiten saber si un proyecto de software en ejecución se está cumpliendo de acuerdo con lo planificado o no.

- ❖ **Métricas de Producto**

Miden cuestiones que tiene una relación directa con el producto que se está construyendo.

Métricas básicas:

- ❖ **Tamaño**

- ❖ **Esfuerzo**

- ❖ **Tiempo**

- ❖ **Defectos**

- **Seguimiento y Control**

El PM se encarga de trabajar sobre lo definido en el plan del proyecto y determinar si se está cumpliendo o no. Para hacerlo se basa en el plan de proyecto y las métricas. Si se pueden corregir los desvíos de un proyecto en el tiempo adecuado, se está a tiempo de poder cumplir el objetivo.

## 2 – Ciclos de Vida

Serie de pasos a través de los cuales el producto o proyecto progresa. Define cuales son las etapas y el orden de cada una. No define que, quien, con que ni como hacer, sino las fases y en que orden se ejecutan.

El **proceso** es una implementación del ciclo de vida que tiene un objetivo que lo guía, que es la creación de un **producto**.

### Ciclos de Vida de Proyecto de Software

Representación de un proceso. Grafica una descripción desde una perspectiva particular. Los modelos especifican las fases de proceso y el orden en el cual se llevan a cabo.

### Ciclos de Vida de Producto de Software

El ciclo de vida de un proyecto termina cuando genera un producto, cuyo ciclo de vida va a necesitar mantenimiento una vez que “termine” de desarrollarse, y se va a mantener bajo este proceso de mantenimiento hasta que se deseché. Ahí podemos decir que el ciclo de vida del producto finaliza.

### Tipos de Ciclos de vida

- **Secuencial**

Se basan en ejecutar una etapa después de otra sin retorno generalmente. Hay algunos que pueden llegar a devolver información (los cascada, por ejemplo).

- **Iterativo**

Permiten la adaptación y mejora del proceso, a través de iteraciones. Además, permite el aprendizaje, mejorando la experiencia del equipo y realizando otra iteración para incrementar el producto.

- **Recurso**

Se utiliza para casos particulares como proyectos de alto riesgo. Se basa en tomar una característica específica y en una iteración me centro en esa funcionalidad, en la siguiente en otra, y así sucesivamente. Estos tipos de

ciclos de vida están en desuso debido a que se comprobó que no son útiles y no funcionan.

### Criterios de elección

Para elegir un ciclo de vida, lo único importante no es la velocidad del mismo, ya que esta varía dependiendo del contexto. Para elegir el mejor ciclo de vida para un proyecto, se deben evaluar las necesidades del mismo y las características del producto. Las necesidades y características más importantes, las que más debemos tener en cuenta son:

Criterio	Cascada puro	Code-and-Fix	Espiral	Cascada modificado	Prototipado evolutivo
Requerimientos poco entendidos	Pobre	Pobre	Excelente	Regular a excelente	Excelente
Arquitectura poco entendida	Pobre	Pobre	Excelente	Regular a excelente	Regular
Alta confiabilidad del sistema	Excelente	Pobre	Excelente	Excelente	Regular
Gran capacidad de crecimiento	Excelente	Regular	Excelente	Excelente	Excelente
Gestión de riesgos	Pobre	Pobre	Excelente	Regular	Regular
Ajuste a cronograma	Regular	Pobre	Regular	Regular	Pobre
Carga administrativa	Pobre	Excelente	Regular	Excelente	Regular
Correcciones en el curso del proyecto	Pobre	Regular	Regular	Regular	Excelente
Visibilidad del progreso para el cliente	Pobre	Regular	Excelente	Regular	Excelente
Visibilidad del progreso para la gerencia	Regular	Pobre	Excelente	Regular a excelente	Regular
Baja sofisticación requerida	Regular	Excelente	Pobre	Pobre a regular	Pobre