1.- El software

1.1- Características del Software:

Para poder comprender lo que es el software, es importante examinar las características del software que lo diferencian de otras cosas que los hombres pueden construir. Cuando se construye hardware, el proceso creativo humano (análisis, diseño, construcción, prueba) se traduce finalmente en una forma física (chips, tarjetas de circuitos impresos, fuentes de potencia, etc.).
El software es un elemento del sistema que es lógico, en lugar de físico. Por tanto el software tiene unas características considerablemente distintas a las del hardware.

*El **software se desarrolla**, no se fabrica en un sentido clásico.*
Aunque existen similitudes entre el desarrollo del software y la construcción del hardware, ambas actividades son fundamentalmente diferentes.

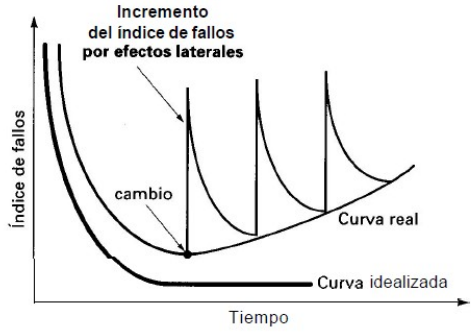
- En ambas actividades la buena calidad se adquiere mediante un buen diseño.
- Ambas actividades dependen de las personas.
- Ambas actividades requieren la construcción de un «producto» pero los enfoques son diferentes.

Los costes del software se encuentran en la ingeniería. Esto significa que los proyectos de software no se pueden gestionar como si fueran proyectos de fabricación.

*El software **no se «estropea»**.*
La Figura describe, para el hardware, la proporción de fallos como una función del tiempo. Esa relación, indica que el hardware exhibe relativamente muchos fallos al principio de su vida (estos fallos son atribuibles normalmente a defectos del diseño o de la fabricación); una vez corregidos los defectos, la tasa de fallos cae hasta un nivel estacionario donde permanece durante un cierto periodo de tiempo. Sin embargo, conforme pasa el tiempo, el hardware empieza a desgastarse y la tasa de fallos se incrementa.

El software no es susceptible a los males del entorno que hacen que el hardware se estropee. Los defectos no detectados harán que falle el programa durante las primeras etapas de su vida. Sin embargo, una vez que se corrigen (suponiendo que no se introducen nuevos errores) la curva se aplana, como se muestra en la figura.

El software ***no se estropea. ¡Pero se deteriora!*** Durante su vida, el software sufre cambios (mantenimiento). Conforme se hacen los cambios, es bastante probable que se introduzcan nuevos defectos, haciendo que la curva de fallos tenga picos. Lentamente, el nivel mínimo de fallos comienza a crecer --*el software se va deteriorando debido a los cambios*--.
Otro aspecto de ese deterioro ilustra la diferencia entre el hardware y el software. Cuando un componente de hardware se estropea se sustituye por una pieza de repuesto.



Curva de fallos real e idealizada del software.

1.2 Tipos de sistemas:

El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales. Algunas veces es difícil establecer categorías genéricas para las aplicaciones del software que sean significativas

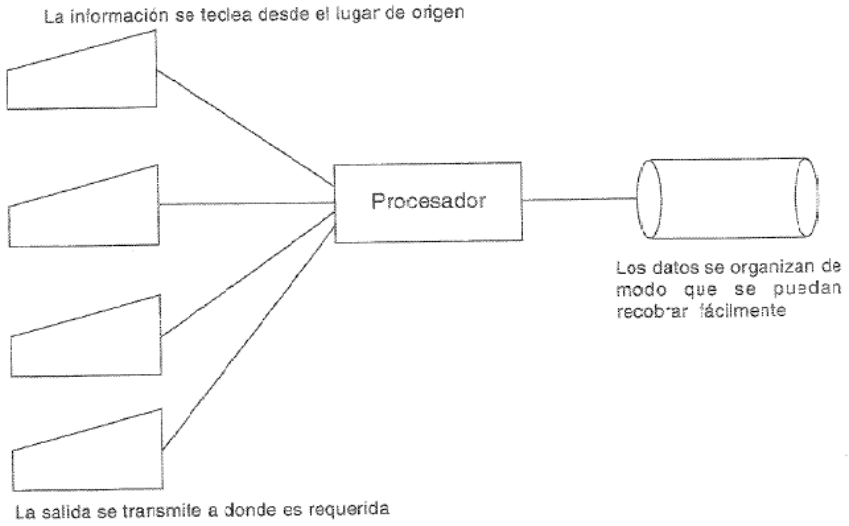
Sistemas batch o por lote:

En los sistemas por lote o *batch, la* ejecución de un programa no tienen control o supervisión directa del usuario, o sea, su ejecución no precisa ningún tipo de interacción con el usuario.
Los primeros computadores eran máquinas enormes (físicamente) que se controlaban desde una consola. Los dispositivos de entrada comunes eran lectores de tarjetas y unidades de cinta. Los dispositivos de salida comunes eran impresoras de líneas, unidades de cinta y perforadoras de tarjetas. Los usuarios de tales sistemas no interactuaban directamente con los computadores; más bien el usuario preparaba un trabajo que consistía en el programa, los datos, y cierta información de control acerca de la naturaleza del trabajo (tarjetas de control)- y lo entregaba al operador del computador. El trabajo casi siempre se presentaba en forma de tarjetas perforadas. En algún momento posterior (minutos, horas o días después), aparecía la salida, que consistía en el resultado del programa junto con un vaciado (o vuelco) de la memoria y los registros en caso de haber un error de programa.
El sistema operativo en estos primeros computadores era sencillo. Su principal obligación era transferir control automáticamente de un trabajo al siguiente. El sistema operativo siempre estaba (residente) en la memoria.
A fin de agilizar el procesamiento, los programas con necesidades similares se agrupaban en Lotes y se introducían en el computador como un grupo. Así, los programadores dejaban sus programas con el operador; éste acomodaba los programas para formar lotes con

necesidades similares y, cuando el computador quedaba disponible, ejecutaba cada lote. La salida de cada trabajo se devolvía al programador apropiado.

Sistemas en línea:

En contraste a los sistemas por lote, los sistemas en línea interactúan con el usuario (por ejemplo un usuario humano con una terminal)
Definición de Yourdon, 1972: un sistema en línea es aquel que acepta materia de entrada directamente del área donde se creó. También es el sistema en el que el material de salida, o el resultado de la computación se devuelve directamente donde es requerido.

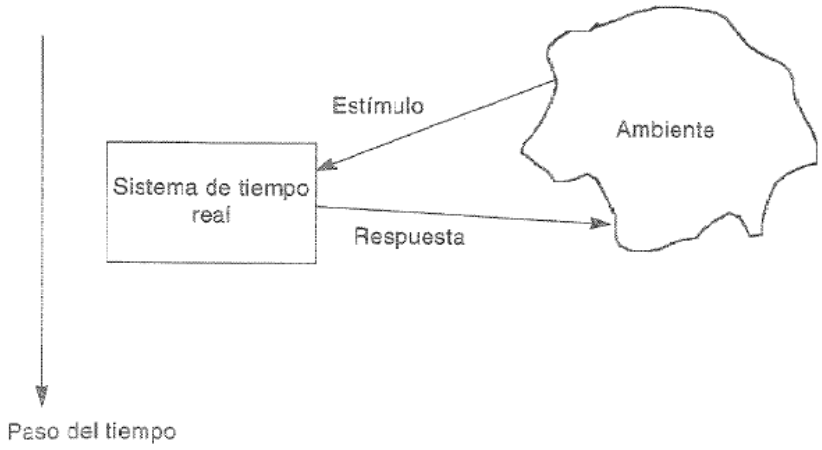


Un ejemplo de esto es: un usuario inserta una tarjeta en un cajero automático y se ha identificado, pero aun no me ha dado su clave secreta. Un cambio de estado es "me ha dado su clave secreta" y ahora puedo proceder a determinar si desea retirar efectivo o desea que le informe el estado de cuenta.
Estos sistemas se caracterizan por:

- Simultáneamente se lleva a cabo el proceso de muchas actividades.
- Se asignan prioridades diferentes a diferentes procesos.
- Se interrumpe una tarea antes de concluirla. Para comenzar otra con mayor prioridad.
- Existe gran comunicación entre tareas.
- Existe acceso simultáneo a datos comunes, tanto en memoria como en almacenamiento secundario.
- Existe un uso y asignación dinámico de memoria RAM

Sistemas de tiempo real:

Un sistema de tiempo real es considerado por muchos como una variante de un sistema en línea, pero es importante distinguirlos.
Definición por Martín, 1967: un sistema de tiempo real pude definirse como aquel que controla un ambiente recibiendo datos, procesándolos y devolviéndolos con la suficiente rapidez como para influir en dicho ambiente en ese momento. Los sistemas en líneas, la información se obtiene de una manera rápida, pero los sistemas de tiempo real deben reaccionar en milisegundos o microsegundos.
Ejemplos: sistemas de control de procesos, sistemas de guía de proyectiles.
Otra diferencia es que los sistemas en línea interactúan directamente con las personas mientra que los sistemas de tiempo real interactúan tanto con las personas como con el ambiente.



Sistemas de apoyo a decisiones y sistemas de planeación estratégica:

Los *sistemas de* ***apoyo a decisiones***, no toman decisiones por si mismo, sino que ayudan a los administradores y a otros usuarios "trabajadores del conocimiento" de una organización a tomar decisiones inteligentes y documentadas acerca de los diversos aspectos de la operación. Estos sistemas son pasivos y se utilizan de manera ad hoc cuando se los necesita, ejemplo: programas de hojas de cálculo, sistemas de análisis estadísticos, sistemas de pronósticos de mercado.
Estos sistemas no solo recuperan datos y exhiben datos, sino que también realiza varios tipos de análisis matemáticos y estadísticos de

los mismos y los presentan en formas fáciles de entender como gráficos y tablas.
Estos sistemas aplican las reglas provistas por el usuario.
Los sistemas de ***plantación estrategia*** son utilizados por los gerentes para evaluar y analizar la misión de la organización. En lugar de dar consejos acerca de alguna decisión aislada, estos sistemas ofrecen consejos más amplios y generales acerca de la naturaleza del mercado, las preferencias de los consumidores, el comportamiento de la competencia, etc.

Sistemas basados en conocimiento:
Estos sistemas también se conocen como sistemas expertos y se asocian con el campo de la inteligencia artificial.
Definición por Rich, 1984: La meta de los científicos de la computación que trabajan en el campo de la inteligencia artificial es producir programas capaces de imitar el desempeño humano en una gran variedad de tareas "inteligentes". Para algunos sistemas expertos la meta está próxima a ser alcanzada; para otros auque aun no sabemos construir programas que funcionen bien por si solos, podemos comenzar a crear programas capaces de auxiliar a las personas en la ejecución de alguna tarea.

1.3.- Crisis del software

La crisis del software, hace referencia a un conjunto de problemas que aparecen en el desarrollo del software para computadoras. Los problemas no están limitados al software que no funciona correctamente, es más, el mal abarca los problemas asociados a

como desarrollarlo, como mantenerlo y como satisfacer las demandas.
La crisis de hoy de la industria del software proviene de las presiones en el mercado de consumo en reducción en el ciclo de vida de productos, personalización de productos, costos, *calidad* y certificación de *procesos*.
A estos fenómenos se suman la necesidad de garantizar seguridad y confiabilidad o la responsabilidad legal por daños ocasionados por un producto.

Problemas
1. La planificación y estimación de costos son frecuentemente muy imprecisas.
2. La productividad no se corresponde con la demanda.
3. La calidad, a veces, no llega a ser ni aceptable.

Tales problemas son solo las manifestaciones visibles de otros:
- ✓ No hay tiempo de recoger datos sobre el proceso de desarrollo sin datos históricos, las estimaciones no han sido buenas.
- ✓ Es frecuente la insatisfacción del cliente, no hay suficiente indagación sobre los requisitos. Poca comunicación con el cliente.
- ✓ La calidad es cuestionable. Recién se empieza a comprender la importancia de las pruebas.
- ✓ El software existente es muy difícil de mantener.

Todos los problemas descriptos pueden corregirse. La clave está en dar un enfoque de ingeniería al desarrollo del software.

2.- Ingeniería de software

2.1.- Introducción:

Cómo podemos garantizar la producción sistemática y controlada de productos de software que satisfagan necesidades de usuarios, a tiempo y dentro de costos establecidos?
Aplicando a la producción de software principios, procedimientos, métricas y herramientas similares a las que se emplean en otras ramas de ingeniería, o sea, aplicando Ingeniería de Software.

2.2.- Definiciones:

Definición por La IEEE (Institute of Electrical and Electronics Engineers):
1. El uso de métodos sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de software.
2. El estudio de técnicas relacionadas con 1

Definición por Fritz Bauer:
La ingeniería del software es el establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales.

Definición por Richard Fairley:
La Ingeniería de Software es la disciplina tecnológica y de administración que se ocupa de la producción y evolución sistemática de productos de software que son desarrollados y modificados dentro de los tiempos y costos estimados.

2.3.- Capas de la Ingeniería de Software

La Ingeniería del software es una tecnología multicapa, donde la primera capa enfatiza que los crecimientos de la ingeniería de software están orientados hacia la calidad.
El fundamento de la ingeniería del software es la capa de *proceso.* El proceso define un marco de trabajo para un conjunto de *áreas clave de proceso* que se deben establecer para la entrega efectiva de la tecnología de la ingeniería del software
Los *métodos* de la ingeniería del software indican «cómo» construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento.
Las *herramientas* de la Ingeniería del software proporcionan un enfoque automático o semi-automático para el proceso y para los métodos. Cuando se integran herramientas para que la información creada por una herramienta la pueda utilizar otra, se establece un sistema de soporte para el desarrollo del software llamado *ingeniería del software asistida por computadora* ***CASE*** (Computer Aided Software Engineering).

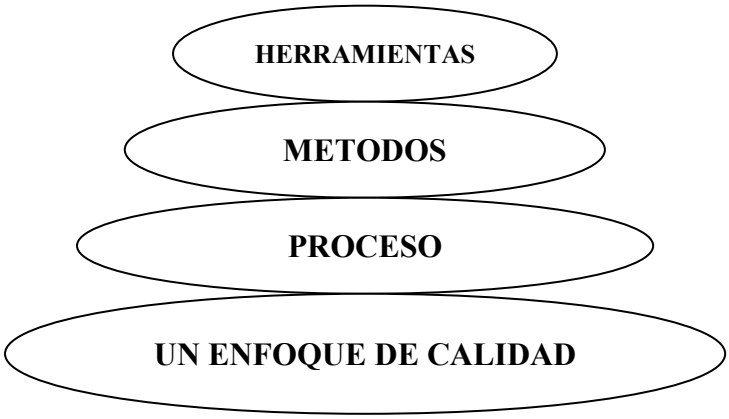


Fig. Capas de la ingeniería de software

2.4.- Fases genéricas de la ingeniería de software

El trabajo que se asocia a la ingeniería del software se puede dividir en tres fases genéricas (*definición, desarrollo, mantenimiento)*, con independencia del área de aplicación, tamaño o complejidad del proyecto:

La fase de definición se centra sobre el ***qué****.* Es decir, durante la definición, el que desarrolla el software intenta identificar qué información ha de ser procesada, qué función y rendimiento se desea, qué comportamiento del sistema, qué interfaces van a ser establecidas, qué restricciones de diseño existen, y qué criterios de validación se necesitan para definir un sistema correcto. Por tanto, han de identificarse los requisitos clave del sistema y del software. Aunque los métodos aplicados durante la fase de definición variarán dependiendo del paradigma de ingeniería del software (o combinación de paradigmas) que se aplique, de alguna manera tendrán lugar tres tareas principales: ingeniería de sistemas o de información, planificación del proyecto del software y análisis de los requisitos.

La *fase de desarrollo* se centra en el ***cómo****.* Es decir, durante el desarrollo un ingeniero del software intenta definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función dentro de una arquitectura de software, cómo han de implementarse los detalles procedimentales, cómo han de caracterizarse interfaces, cómo ha de traducirse el diseño en un lenguaje de programación (o lenguaje no procedimental) y cómo ha de realizarse la prueba. Los métodos aplicados durante la fase de desarrollo variarán, aunque las tres tareas específicas técnicas

deberían ocurrir siempre: diseño del software, generación de código y prueba del software.

La *fase de mantenimiento* se centra en el ***cambio*** que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente.

3.- Los distintos roles en el desarrollo un proyecto de software

Usuarios:
Es el individuo que utiliza el sistema, puede ser un usuario registrado (nombre de usuario y contraseña) o un usuario anónimo.

Cliente:
Es para quien se construye el sistema (persona, organización, etc.)

Administradores:

- Administradores usuarios**:** administradores que están a cargo de varias personas en el área operacional.
- Administradores de informática: son las personas encargadas del proyecto de software en si.
- Administración general: son los administradores de nivel superior que no están directamente involucrados con la organización de informática no son de la organización usuaria.

El administrador querrá que el proyecto se mantenga dentro de los márgenes de tiempo y costo.

Auditores, control de calidad y departamento de normas o estándares:
El objetivo de estos equipos de personas es que su sistema se desarrolle de acuerdo con diversos estándares o normas externos (es decir, externos a su proyecto), por ejemplo estándares del contabilidad desarrollados por su agencia contable o estándares impuestos por el gobierno etc.

Analista de sistema:
El analista de sistemas desempeña varios papeles:

- Arqueólogo y escribano. Una de las principales labores es descubrir detalles y documentar la política de un negocio.
- Innovador. Con el conocimiento de la tecnología de las computadoras el analista debe ayudar al usuario a explorar

aplicaciones novedosas así como formas nuevas de hacer negocio.
- Mediador. Trata de obtener un consenso entre los usuarios administradores, desarrolladores, auditores.

Diseñadores de sistemas:
Es quien recibe los resultados de su trabajo de análisis: la labor de él es transformar la petición, libre de consideraciones de tecnología emanada de los requerimientos del usuario, un diseño arqueológico de alto nivel que servirá de base para los programadores.

Los Programadores:
Reciben los resultados de los diseñadores y los transforman en el código fuente.

Personal de operaciones: Es el responsable del centro de computo, la red, la seguridad del hardware y software además de la ejecución de los programas.

Los testers: Son los que realizan y/o ejecutan las distintas pruebas del sistema**.**

Mantenimiento: Depuran el sistema, ya sea por causa de errores, por mejoras de código fuente, adaptan al sistema a los cambios del entorno, agregan funcionalidad para mejorar el sistema