Ingeniería de Software

La **ingeniería de software** es la disciplina que se encarga de planificar, desarrollar y mantener software construido por un equipo o equipos de ingenieros, de forma sustentada, ordenada y cuantificable, ajustándose a costos y tiempo previamente estimados (de manera rentable).

El **software** es:

- Alma y cerebro de una computadora,
- Conocimiento adquirido acerca de un área de aplicación,
- **Conjunto de programas y datos necesarios para convertir una computadora en una máquina de propósito específico para una aplicación particular,**
- Documentación producida durante el desarrollo de un sistema,
- Conocimiento empaquetado: es tanto producto como objeto.

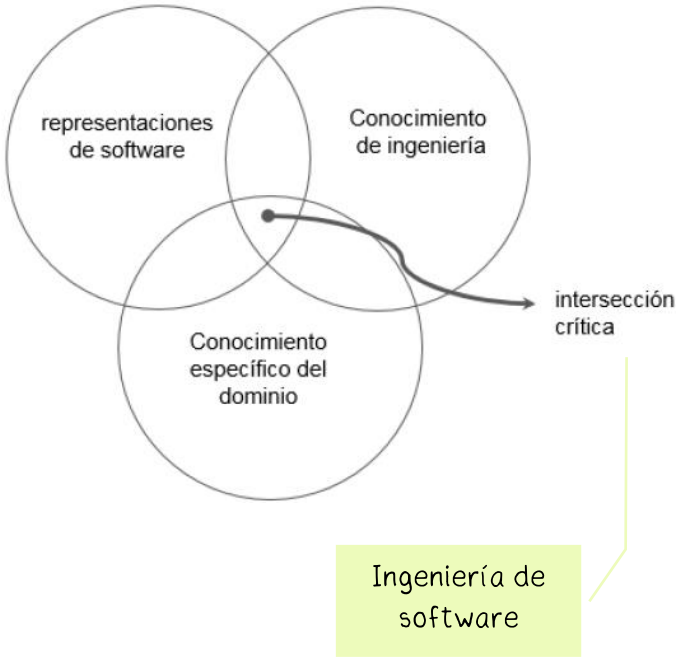
Características del software:

- Maleable
- Producción humano intensiva: se vincula más con el diseño e implementación que con la manufactura.
- El ingeniero dispone de herramientas para describir el producto que no son distintas del producto.

El software no solo son programas ejecutables, abarca otra información relevante:

Clases de información

- Representaciones de software: cualquier información que en forma directa representa un eventual conjunto de programas y datos asociados: programas, diseños, especificaciones escritas en lenguaje formal, requisitos del sistema;
- Conocimiento de la ingeniería de software: toda información relativa al desarrollo en general o particular: información sobre la tecnología de software (métodos, conceptos, técnicas)
- Conocimiento del dominio específico: conocimiento del proceso especifico a ser controlado. Es esencial para la creación del software.

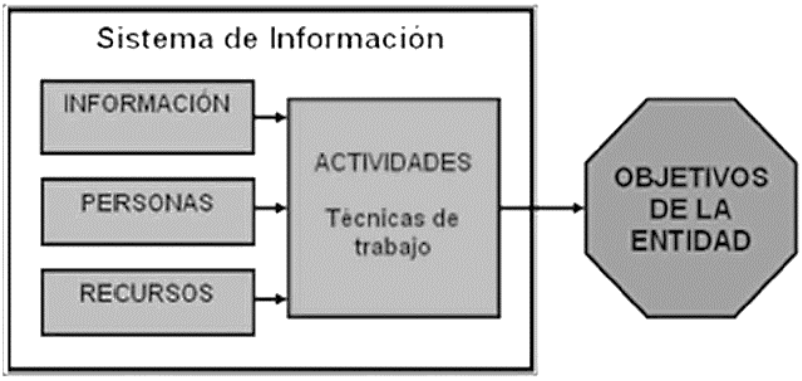


Sistema de información

Un **sistema** es un conjunto de componentes que interactúan entre si para lograr un objetivo en común.

Los sistemas se clasifican en abiertos (interactúan con el ambiente) o cerrados (no interactúan con el ambiente).

Un **sistema de información** es un conjunto de elementos orientados al tratamiento y administración de datos e información, organizados y listos para su uso posterior, generados para cubrir una necesidad u objetivo.



El primordial objetivo de un sistema de información es apoyar la toma de decisiones y controlar todo lo que en ella ocurre.

La importancia de un sistema de información radica en la eficiencia en la correlación de una gran cantidad de datos ingresados a través de procesos diseñados para cada área con el objetivo de producir información válida para la posterior toma de decisiones.

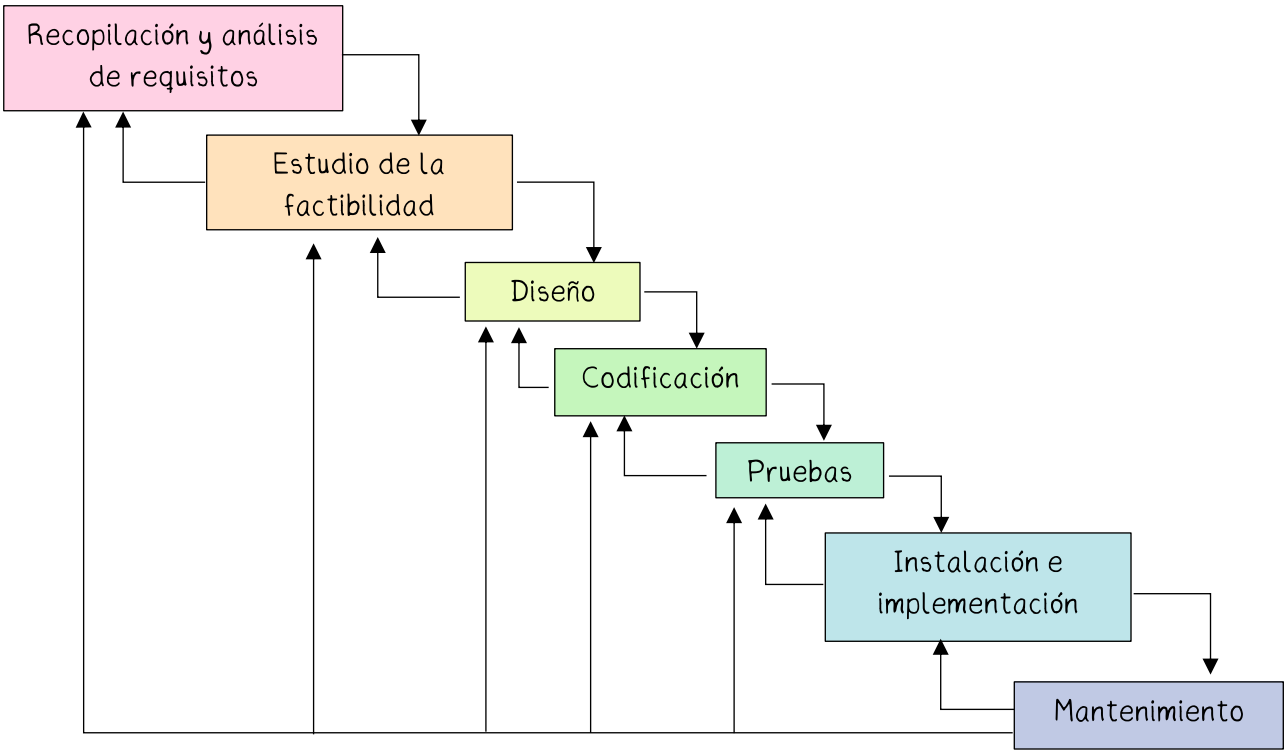
En la actualidad, los recursos materiales de un sistema están constituidos por sistemas informáticos. Los sistemas de información informáticos son una subclase de los sistemas de información en general. Se destacan por ayudar a administrar, recolectar, recuperar, procesar, almacenar y distribuir información relevante para los procesos fundamentales y las particularidades de cada organización.

El **análisis y diseño de sistemas**: se refiere al proceso de examinar la situación de una empresa con el propósito de mejorarla con métodos y procedimientos adecuados.

- Análisis: proceso de clasificación e interpretación de los hechos, diagnostico de problemas y empleo de la información para recomendar mejoras al sistema.
  Especifica que es lo que el sistema debe hacer.
  El objetivo del análisis de sistemas es tomar una visión general del sistema e identificar sus unidades para lo cual es necesario relevar y documentar hechos de modo comprensible para el cliente y los desarrolladores.
- Diseño: proceso de planificar, reemplazar o complementar un sistema existente.
  Especifica cómo alcanzar el objetivo planteado por el análisis.
  El objetivo del diseño de sistemas es resolver los problemas planteados en la etapa de análisis mediante la representación de la solución.

El **ciclo de vida del desarrollo de software** (SDLC) es un conjunto de actividades que analistas, diseñadores y usuarios realizan para desarrollar e implementar un sistema de software. El mismo tiene como objetivo producir sistemas de alta calidad que cumplan o superen las expectativas del cliente mediante la entrega de sistemas que se mueven a través de una fase claramente definida, dentro de los plazos programados y las estimaciones de costos.

El SDLC es un proceso sistemático; se encuentra regido por una serie de pasos:



1- Recopilación y análisis de requisitos

Se trata de tomar una visión general del sistema, identificar sus unidades y los servicios que debe brindar. Entones, se estudia el sistema a fin conocer cómo trabaja y donde es necesario efectuar mejoras.

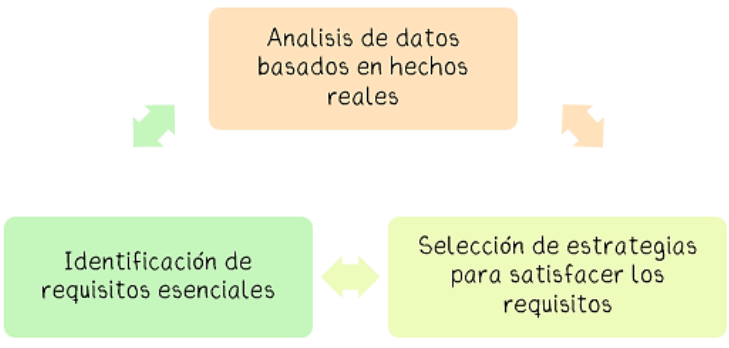
Esto implica de alguna manera responder a las siguientes preguntas: que hace el sistema? Como lo hace? Con que frecuencia se presenta? Que tan grande es el volumen de transacciones? ¿Cuál es el grado de eficiencia con el que se realizan las tareas? Existe algún problema? Si existe que tan serio es? ¿Cuál es la causa que lo origina?

Se definen los requisitos. Un requisito es una característica que debe incluirse en el nuevo sistema. Pueden ser esenciales o no esenciales.

Para determinar requisitos se llevan a cabo 3 actividades:

- Anticipación: prever características del sistema, investigar aspectos relacionados, aunque esto introduzca un sesgo propio.
- Investigación: estudio y documentación del sistema actual, usando técnicas para encontrar hechos, análisis de flujo de datos y análisis de decisión.
- Especificación: análisis de los datos obtenidos durante la recopilación para determinar las especificaciones de requisitos.

  Se compone de:



Los **hechos** se encuentran mediante diversas **técnicas**:

- **Entrevistas**:
  El analista recolecta información proveniente de personas.
  Es un método de contacto directo con personas.
  El éxito depende de la preparación del entrevistador.

  Momentos de una entrevista:
  - Antes: el entrevistador debe tener una preparación previa (cuales son las estructuras formales del sector, flujos de datos, alcance de los cambios propuestos, etc)
  - Durante: el analista debe tener una actitud receptiva, estar alerta a los mensajes incompletos o contradictorios, motivar al usuario a transmitir la información que necesita.
  - Después: se propone hacerle llegar al cliente un informe de las notas que el tomo durante la entrevista para solicitarle que las revise y corrija.

  Ventaja: se tiene un feedback directo, posibilidad de repreguntar.
  Desventaja: no siempre los entrevistados están abiertos a brindar información.

- Cuestionarios:
  Permiten al analista reunir información relacionada proveniente a un gran número de personas.
  Ventaja: se asegura el anonimato de los encuestados
  Desventaja: no da posibilidad de repreguntar

- Revisión de registros:
  El analista examina la documentación gestionada por el cliente, se examinan registros y reportes relacionados con el sistema y los usuarios para conocer la organización y sus operaciones.

  Ventaja: obtención de datos reales
  Desventaja: difícil que el cliente proporcione esta información para un presupuesto

- Observación In-situ:
  Permite al analista ganar información sobre la forma en que se llevan a cabo las actividades, permite observar la realidad tal cual es sin ser manipulada.

Fundamentos del análisis de requisitos

Siempre ocurre que el cliente trata de formular un concepto nebuloso de la función y performance que debe desarrollar el sistema. Por lo tanto, el analista actúa como un interrogador, consultor y persona que resuelve problemas.

a. Reconocimiento de problemas: identifica los aspectos de cada problema y sus características particulares.
b. Evaluación de problemas y síntesis de la solución: se concentra en los datos que el sistema produce, funciones, interfaces, restricciones entre otras.

c. Modelamiento: se crea un modelo (simple) gráfico y textual del sistema a construir el cual será usado como base de diseño y revisión entre el cliente y los desarrolladores. Se pueden utilizar lenguajes formales.
d. Especificación: se documenta los pasos anteriores en un *documento de requisitos*. Además, se incluye una predicción de costos, beneficios, características del sistema, cronograma de desarrollo, etc.
e. Revisión: el documento anterior es discutido con el cliente. Sirve de contrato.

Tareas del analisis:

- Estudio de detalles y procedimientos del S.I.
- Documentación de detalles para su revisión y discusión con otros usuarios.
  La documentación debe ser clara para que puedan ser comprendidos los detalles del sistema por los usuarios.
- Evaluación de la eficiencia y efectividad del sistema actual y de sus procedimientos.
- Recomendación, con la debida justificación, mejoras y ampliaciones del sistema actual.
- Fomentación de la participación de gerentes y empleados en todo el proceso de desarrollo del sistema.

2- Estudio de la factibilidad

Se analiza si es posible hacer lo que el cliente pide. Implica estudiar la factibilidad:

- Técnica: si el proyecto puede realizarse con la tecnología de software y hardware existente.
- Económica: si los beneficios esperados cubren los costos
- Operativa: se analiza el recurso humano, si habrá resistencia al cambio de sistema
- Legal: si es legalmente posible hacer lo que se pide
- Temporal: si el proyecto puede ser completado en el tiempo dado

3- Diseño

El diseño de software es el proceso de representar las funciones de cada sistema de software de manera tal que se puedan transformar con facilidad en uno o más programas de computación.

En esta etapa se establece la forma en la que el sistema cumplirá los requisitos identificados en la etapa de análisis.
Se representa la solución al problema de manera sencilla para que después alguien lo pueda implementar.

Tareas del diseño:

- Identificación de reportes y salidas (con detalles)
- Indicación de datos de entrada, cálculos intermedios y donde se almacenarán
- Descripción en detalle de los procedimientos de cálculo y datos individuales.
  Los procedimientos indican como procesar datos y producir salidas.
- Selección de las estructuras de almacenamiento de datos y los dispositivos de almacenamiento.

4- Codificación

Los desarrolladores construyen todo el sistema escribiendo código usando el lenguaje de programación elegido. Deben seguir pautas de codificación predefinidas.

Tareas de la codificación:

- Transcribir a código ejecutable las especificaciones de diseño
- Documentar el cómo y porque se programan los procedimientos y funciones de cierta manera
- Realizar pruebas de unidad, esto implica la comprobación de que cada unidad de programa cumple con su especificación.
- Depurar

5- Pruebas

Se integran las unidades individuales y se realizan pruebas integradoras para corroborar que todo funcione.
Se debe validar y verificar el sistema.

- Validación: se ve si se ha construido el producto correcto, si cumple con la definición de las necesidades del cliente
- Verificación: se controla si se ha construido correctamente el producto, si las funciones del producto son las que se especificaron en el diseño.

El mismo sistema se usa de manera experimental para asegurarse que no tenga fallas y que funcione de acuerdo a las especificaciones y requisitos del usuario.

Se usan distintos grupos de prueba (ALFA y BETA, preproducción) con la intención de descubrir fallas antes de que la organización o empresa instale el sistema y dependa de él. En este sentido, se tratan de encontrar la mayor cantidad de errores antes de la etapa de producción.

Alfa: el sistema se instala en diferentes servidores (sigue siendo un entorno controlado)

Beta: el cliente hace las pruebas (deja de ser un entorno controlado)

6- Instalación e implementación

Se instala el sistema. La instalación es el proceso de verificar e instalar nuevos equipos, entrenar a los usuarios, instalar la aplicación y construir todos los archivos de datos necesarios para usarla.
Una vez instalado, el software deberá sufrir cambios en el tiempo. En este sentido, la implementación es un proceso en constante evolución.
La implementación de múltiples instancias de un mismo sistema genera múltiples versiones.

7- Mantenimiento

Proceso de corregir errores en el sistema y modificarlo para que refleje los cambios en el ambiente.
Tipos de mantenimiento:

- Correctivo: se corrigen errores no descubiertos previamente
- Perfectivo: se mejora la funcionalidad del sistema
- Adaptativo: se modifica el sistema para hacer frente a necesidades del entorno.
- Preventivo: se modifican las rutinas de código que son propensas a generar errores futuros.

Se evalúa el sistema para identificar puntos débiles y puntos fuertes. La evaluación ocurre a lo largo de cualquiera de las siguientes dimensiones:

- Evaluación operacional: se evalúa como funciona el sistema, incluyendo su facilidad de uso, tiempo de respuesta, confiabilidad global y nivel de utilización.
- Impacto organizacional: se identifican y miden los beneficios para la organización de áreas tales como las finanzas, eficiencia operacional e impacto competitivo.
- Opinión de los administradores: se evalúa la opinión de directivos, administradores y usuarios directos.
- Desempeño del desarrollo: se evalúa el proceso de desarrollo (tiempo, dinero, esfuerzo, métodos y herramientas utilizadas).

Análisis de costos: los costos del desarrollo de software son mayores al principio y final del ciclo de desarrollo. La mayor parte de los costos de mantenimiento no resultan de errores sino de cambios en las necesidades.

La reducción de costos se logra mediante un buen análisis (encontrando las necesidades reales del usuario) y un diseño efectivo del software.

El proceso del ciclo de vida

El **proceso de desarrollo de software** es un conjunto de actividades relacionadas por un modelo o metodología de desarrollo que son útiles para generar productos intermedios de trabajo que permiten desarrollar el producto final de software. El desarrollo de sistemas de software se hace usando un modelo de proceso.

Estándar ISO/EIC 12207: describe la arquitectura del ciclo de vida del software, pero no especifica los detalles de cómo realizar o llevar a cabo el proceso, da un lineamiento general.

El objetivo más importante de esta norma es proporcionar una estructura común para que los compradores, proveedores, desarrolladores, personal de mantenimiento, operadores, gestores y técnicos involucrados en el desarrollo de software utilicen un lenguaje en común.

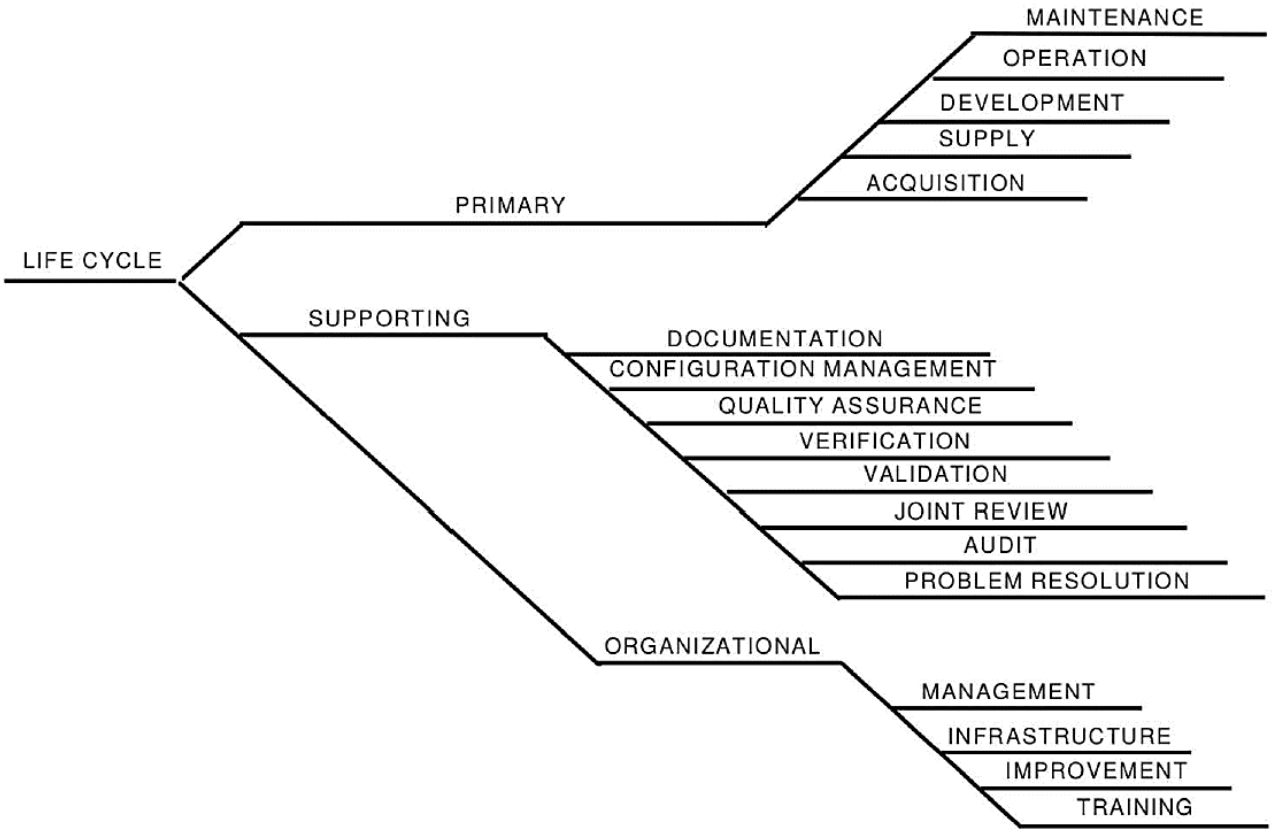
Las características de esta norma son:

- Proceso estructurado utilizando terminología aceptada
- Documento relativamente de alto nivel
- No especifica detalladamente como desarrollar actividades.
- No prescribe el nombre, formato, o el contenido de la documentación.

Esta norma define una serie de procesos que abarcan las diferentes etapas del ciclo de vida del software (7 vistas anteriormente)

Cada proceso está diseñado en términos de sus propias actividades, cada una de las cuales está diseñada además en términos de las tareas que las componen.

La norma define los siguientes procesos del ciclo de vida del software:



1. **Procesos primarios del ciclo de vida:**
   1.1. **Proceso de adquisición:** define las actividades y tareas que el cliente lleva a cabo: selección de necesidades, pedido de propuesta, selección del proveedor, proceso de adquisición y aceptación del sistema, entre otras.
   1.2. **Proceso de provisión:** contiene las actividades y tareas que el proveedor efectúa: preparación de propuesta, elaboración de un contrato, planificación, ejecución, control, revisión, evaluación, entrega, control, etc.
   1.3. **Proceso de desarrollo:** tanto para el desarrollo de software nuevo como para modificación de software existente. Define las actividades y tareas del desarrollador de software: tareas de análisis, diseño, codificación, pruebas e instalación.
   El orden de estas actividades no implica un orden cronológico, pueden ser iteradas y solapadas
   1.4. **Proceso de operación:** contiene las actividades y tareas que el operador de un sistema de software lleva a cabo: puesta en producción y soporte a usuarios.
   1.5. **Proceso de mantenimiento:** consta de las actividades y tareas del mantenedor: modificación del código, migración, baja del software, entre otras.

2. **Procesos de soporte del ciclo de vida:**
   2.1. Proceso de documentación: se registra la información producida durante el proceso para la comunicación.
   Define las actividades para planificar, diseñar, desarrollar, editar, distribuir y mantener los documentos necesarios por todos los interesados.
   2.2. Proceso de gestión de configuración: se identifican las bases para controlar los cambios. Se aplica al código, configuraciones, BD, documentos, etc.

2.3. **Proceso de QA:** proporciona un marco de calidad para asegurar la independencia y objetividad de conformidad de los productos o servicios con sus requisitos contractuales y la adhesión a sus planes establecidos.
2.4. **Proceso de verificación:** proporciona las evaluaciones relacionadas con la verificación de un producto. Abarca la verificación del proceso, de los requisitos, del diseño, del código, de la integración y la documentación.
2.5. **Proceso de validación:** determina si el sistema final cumple con su uso específico provisto.
2.6. **Proceso de revisión conjunta:** se evalúan el estado y productos de una actividad de un proyecto a nivel de administración y técnico.
2.7. **Proceso de auditoria:** se evalúan los productos y actividades con énfasis en los requerimientos y planes.
2.8. **Proceso de resolución de problemas:** requiere de la identificación y análisis de las causas de los problemas y proporciona el mecanismo para instituir un proceso para la resolución de los mismos.

3. **Procesos organizacionales del ciclo de vida:**
3.1. **Proceso gerencial:** define las actividades y tareas del gerente: definición de alcance, planificación, ejecución, control, revisión y evaluación, etc.
3.2. **Proceso de infraestructura:** define las actividades necesarias para el establecimiento y mantenimiento de una infraestructura. La infraestructura puede incluir hardware, software, estándares, herramientas, técnicas, y las instalaciones.
3.3. **Proceso de mejora:** proporciona las actividades necesarias para evaluar, medir, controlar y mejorar el proceso del ciclo de vida.
3.4. **Proceso de capacitación:** plantea identificar y hacer un plan para incorporar o desarrollar recursos de personal en los niveles de gestión y técnicos. Actividades: elaboración de un plan de formación, generación de material de capacitación y dictado de la misma al personal.

Capas de la ingeniería de software

Las capas de la ingeniería de software son:

1) **Métodos:**
Indican como construir técnicamente el software.
Proporcionan la experiencia técnica para elaborar software.
2) **Herramientas:**
Proporcionan un apoyo automático o semiautomático para los métodos y el proceso.
Ej: Herramientas CASE (Computer Aided Software Engineering)
3) **Proceso:**
Pegamento que une los métodos y las herramientas. Define la secuencia en la que se aplican los métodos.

Facilita el desarrollo racional y oportuno del software.
Es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto.
El proceso establece para su conformación:

- Actividades estructurales:
  - Comunicación
  - Planeación
  - Modelado
  - Construcción
  - Despliegue
- Actividades sombrilla:
  - Seguimiento y control: progreso vs planificación
  - Administración del riesgo
  - Aseguramiento de la calidad
  - Revisiones técnicas: búsqueda y eliminación de errores.
  - Medición

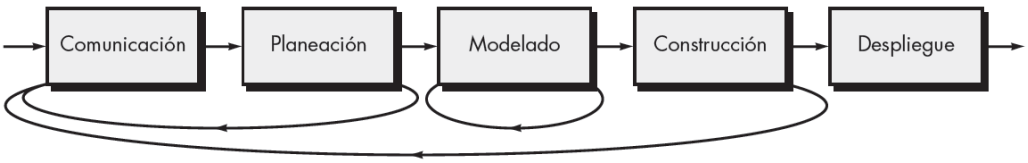
El flujo de proceso describe la manera en que están organizadas las actividades estructurales, las acciones y las tareas del proceso con respecto a la secuencia y tiempo.

Tipos de flujo de proceso:

- **Flujo de proceso lineal**: ejecuta las 5 actividades estructurales en secuencia. El resultado de una actividad es la entrada de la siguiente.

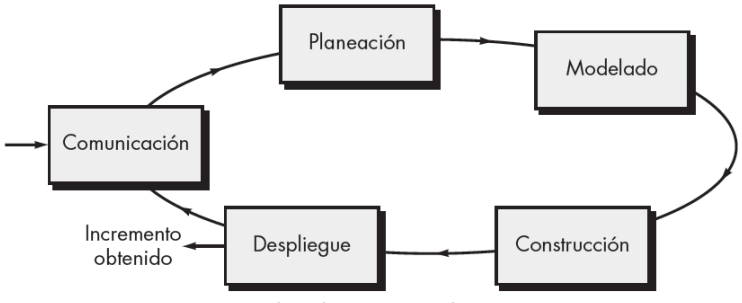
a) Flujo de proceso lineal

- **Flujo de proceso iterativo:** las actividades se aplican en forma repetida para completar diferentes productos.



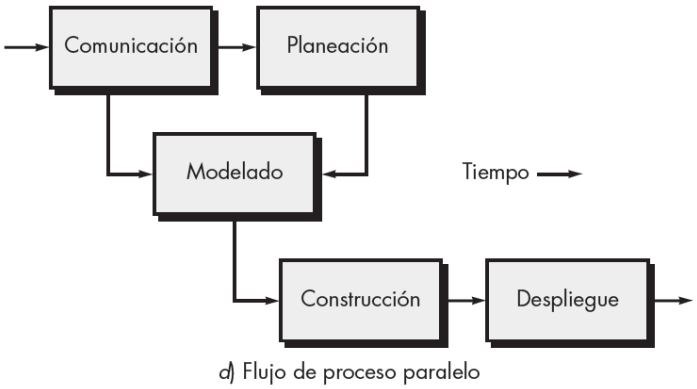
b) Flujo de proceso iterativo

- **Flujo de proceso evolutivo**: realiza las actividades en forma circular, se genera un incremento en cada iteración.



c) Flujo de proceso evolutivo

- **Flujo de proceso paralelo**: ejecuta una o más actividades en paralelo con otras.



d) Flujo de proceso paralelo

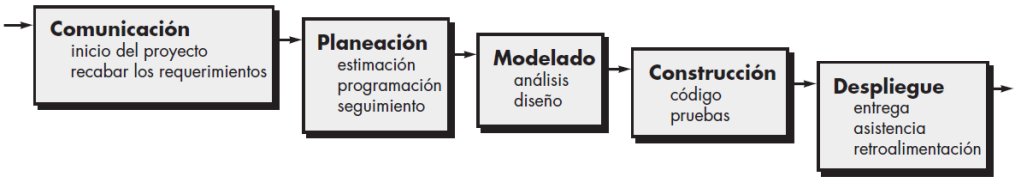
Modelos de procesos

Prescriptivos	Descriptivos
Se centran en el control del proceso.	Se centran en el factor humano
Prescriben actividades, tareas y acciones ante determinados síntomas. También productos intermedios, QA y control de cambios.	Acentúan la disciplina del equipo, se da > valor al individuo, a la colaboración con el cliente y al desarrollo incremental con iteraciones muy cortas.
Proceso se mejora formalmente	Proceso se mejora informalmente
Difícil adaptación a los cambios	Ágil adaptación a los cambios
Ignorancia del contexto	Mayor comprensión del contexto
Limita la creatividad	Fomenta la creatividad

Modelos prescriptivos:

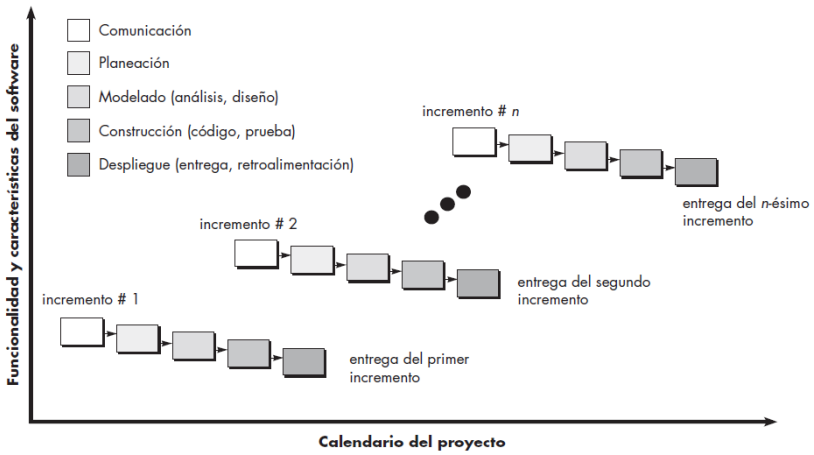
1. Modelo cascada:

- Funcionamiento: Actividades (comunicación, planeación, modelado, construcción, despliegue) se aplican secuencialmente, cada etapa se completa antes de pasar a la siguiente.
- Basado en el flujo: lineal
- Ventajas:
  - Natural
  - Sencillo de aplicar
- Desventajas:
  - Proyectos no suelen seguir un flujo lineal.
  - Dificultad de parte del cliente de enunciar los requerimientos del sistema.
  - Cliente no cuenta con una versión funcional del programa hasta que el proyecto está muy avanzado.
  - Errores detectados en etapas posteriores suelen ser costosos de corregir
- Contexto de aplicación:
  Conviene aplicarse cuando deben hacerse adaptaciones o mejoras bien definidas al sistema.
  - Requisitos bien definidos
  - Proyectos chicos, existen restricciones contractuales
  - Requisitos no cambiantes



2. Modelo incremental:

- Funcionamiento: Se basa en la construcción y entrega de incrementos o versiones funcionales del sistema a lo largo del tiempo. Se dejan funcionando los requisitos principales y se van añadiendo nuevos conforme se avanza.
- Basado en el flujo: paralelo y lineal
- Ventajas:
    - Da la posibilidad de recibir retroalimentación temprana de los usuarios.
    - Permite el trabajo en paralelo
    - Escalable.
- Desventajas:
    - Funcionalidades no siempre divisibles
    - Dificultad para manejar muchas ramas o versiones.
- Contexto de aplicación:
    - Requisitos no del todo definidos
    - Proyectos grandes.
    - Requisitos cambiantes

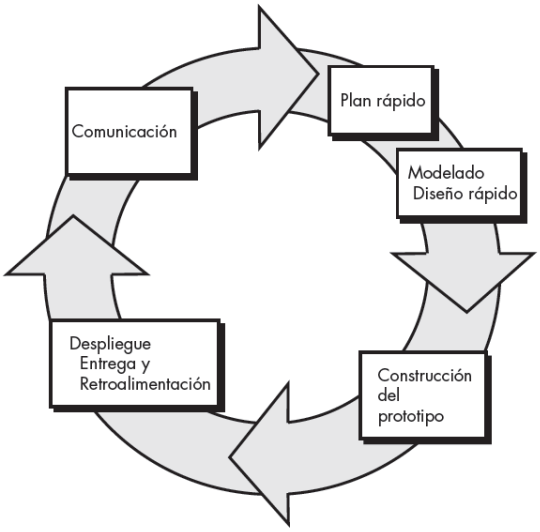


3. Modelo prototipo:

Un prototipo es un sistema que funciona, desarrollado con la finalidad de probar ideas y suposiciones realizadas con el nuevo sistema. Formado por software que acepta entradas, realiza cálculos, produce información y lleva a cabo otras actividades significativas. Es la primera versión de un sistema de información.

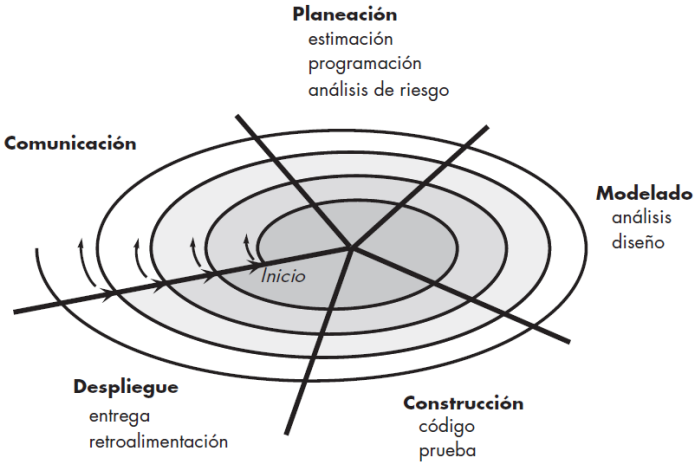
- Funcionamiento: se basa en la creación rápida de prototipos que son mostrados al cliente a fin de obtener los requisitos del sistema para poder construir el mismo.
  Las etapas de análisis y diseño se realizan rápidamente.
- Basado en el flujo: evolutivo

- Ventajas:
  - Genera un feedback inmediato para asentar requerimientos
- Desventajas:
  - Difícil de escalar.
  - Se corre riesgo de quedarse con un diseño pobre.
  - Puede destruir expectativas del cliente
- Contexto de aplicación:
  - Requerimientos no conocidos
  - Proyectos de riesgo moderado a alto
  - Requisitos cambiantes

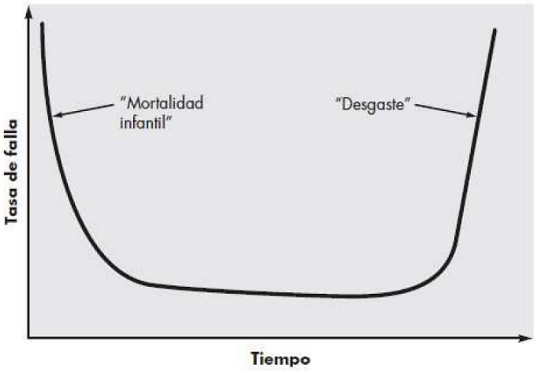


4. Modelo espiral:

- Funcionamiento: es un modelo evolutivo que se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada.
  El software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, lo que se entrega puede ser un modelo o prototipo. En las iteraciones posteriores se producen versiones cada vez más completas del sistema.
  Cada vuelta a la espiral representa una iteración del proceso de desarrollo.
  Se diferencia de los otros modelos porque se incluye un análisis de riesgo, tanto del proyecto como de los requisitos.
- Basado en el flujo: lineal y evolutivo
- Ventajas:
  - Refleja al mundo real de forma más realista
  - Obtención temprana de retroalimentación
  - Gestión proactiva de riesgos
- Desventajas:
  - Complejo de gestionar
  - Si algún riesgo importante no se descubre se presentan problemas.
- Contexto de aplicación:
  - Requerimientos no necesariamente definidos
  - Requisitos cambiantes



Evolución del software



Los grandes sistemas de software no son objetos estáticos, existen en un ambiente sujeto a constantes cambios. El sistema deberá adaptarse cada vez que éste ambiente cambie o irá perdiendo utilidad, hasta quedar desechado. Este proceso de cambio recibe el nombre de **evolución del software**.

Según Lehman, existen 5 leyes de evolución de los programas:

1) Cambio Continuo: Un programa que se usa en un ambiente del mundo debe cambiar o será cada vez menos útil en ese ambiente.
2) Complejidad Creciente: A medida que un programa en evolución cambia, su estructura se hace mas compleja.
3) Evolución del Programa: Es un proceso autoregulador y una medición de atributos del sistema como tamaño, tiempo entre versiones, número de errores.
4) Conservación de la Estabilidad Organizativa: Durante el tiempo de vida de un programa, su rapidez de desarrollo es casi constante.
5) Conservación de la Familiaridad: Durante el tiempo de vida de un sistema, la evolución del cambio del sistema en cada versión es casi constante.

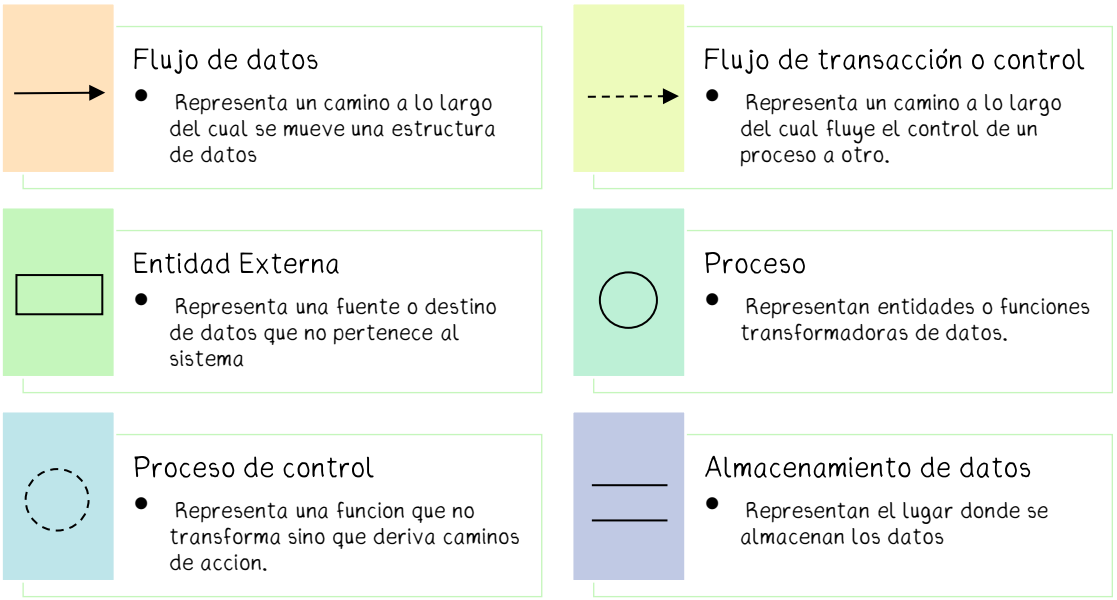
Análisis estructurado

El análisis estructurado es un método de desarrollo de sistemas que consiste en la división del sistema en componentes (separa procesos de datos) y la construcción de un modelo para el mismo, incorporando elementos de análisis y diseño para favorecer el entendimiento de sistemas grandes y complejos.

Este método permite distinguir los elementos lógicos de los elementos físicos.

Los **elementos del análisis y diseño** estructurado son:

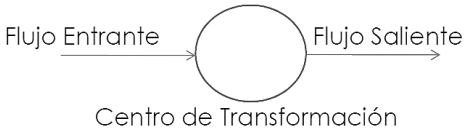
1) **Símbolos gráficos:** son los elementos gráficos del proceso.



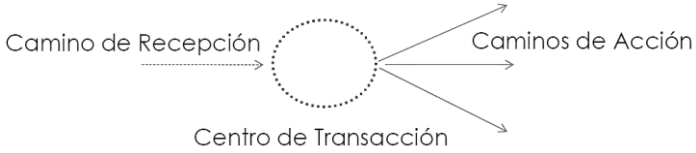
2) **Diagrama de flujo de datos (DFD):** describen el sistema, se diagrama la estructura de la información.
Utilizan un método TOP - DOWN (es decir se va de lo más general a lo más específico)

Tipos de flujo de información:

- Flujo de transformación:



- Flujo de transacción:



Los DFDs se desarrollan en **niveles** diferentes donde, a medida que se profundiza en un nivel se agregan detalles que no fueron considerados en un nivel anterior, siempre manteniendo la consistencia con lo especificado niveles antes (se deben mantener los flujos que entran y salen de cada proceso).

En la práctica no sería necesario usar más de 3 niveles.

- **Nivel 0:** diagrama de contexto, muestra un único proceso como TODO el sistema interactuando con el entorno (entidades externas, almacenamientos)
- **Nivel 1:** muestra el sistema descompuesto en sus principales funciones. (Se sugiere no más de 6 subprocesos)
- **Nivel 2:** para cada proceso se re-explota la burbuja de forma que ya no se tiene una visión general del sistema sino una visión individual del proceso.
- Desde el nivel 2 en adelante: para cada burbuja existen 2 opciones:
  - Se re-explota en sub-funciones
  - Diseño preliminar en castellano estructurado sin detalles de implementación, solo se dan detalles de diseño.

  Todos los procesos deben quedar finalmente descriptos con castellano estructurado

Pautas de desarrollo:

- Todo elemento lleva un nombre distintivo a otro, si dos elementos tienen el mismo nombre se los deberá considerar como el mismo elemento.
- No se indica un orden, es decir no existe flujo de control.
- No se indica la secuencia de ejecución de un proceso,

Ventajas asociadas al uso de los DFDs:

- Se identifican y describen hasta el final del proceso todos los datos que fluyen por todo el sistema
- Se explica porque los datos entran y salen del proceso y cuál es el procesamiento que se realiza sobre ellos.
- Son tan fáciles de leer que permiten que analistas trabajen junto a usuarios para discutir sobre procesos.

3) **Diccionario de datos (DD):** Un DD es un listado organizado de todos los datos pertinentes al sistema, con definiciones rigurosas de manera que el usuario y el analista del sistema comprendan los inputs, los outputs, los componentes de almacenamiento y hasta cálculos intermedios.

   Se trata de las estructuras de datos por lo tanto se especifican los tipos de datos.

   Cada elemento de datos contiene:
   - Nombre: nombre principal de cada dato (flujo)
   - Alias: otros nombres usados para la entrada
   - Donde y como se usa: un listado de los procesos que usan el elemento de datos o de control y como se los usa. Ej:

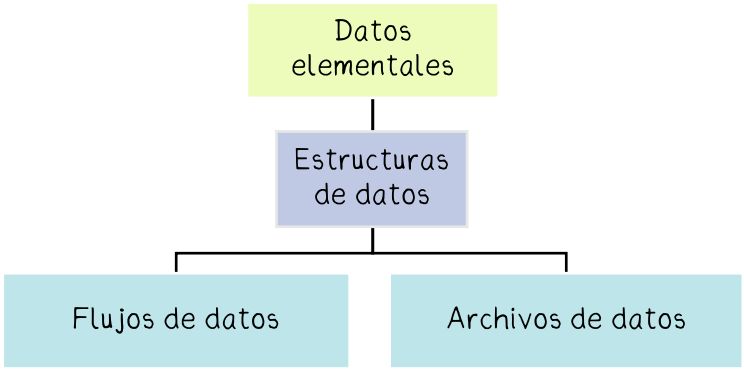
Flujo de datos 1:

B (Entrada)

A (Salida)

   - Descripción de contenido: se describe el contenido elemental o estructurado con una notación especifica. (tipos de datos)
   - Información adicional: valor inicial, restricciones y limitaciones, etc.

   Los datos tienen la siguiente jerarquía:



4) **Castellano estructurado (Pseudo-codigo):** es una descripción algorítmica del proceso. No contiene detalles de implementación.

El diseño estructurado tiene por objetivo crear programas formados por modulos independientes desde el punto de vista funcional. Las decisiones que se toman son de naturaleza estructural.

El diseño es conducido por la información