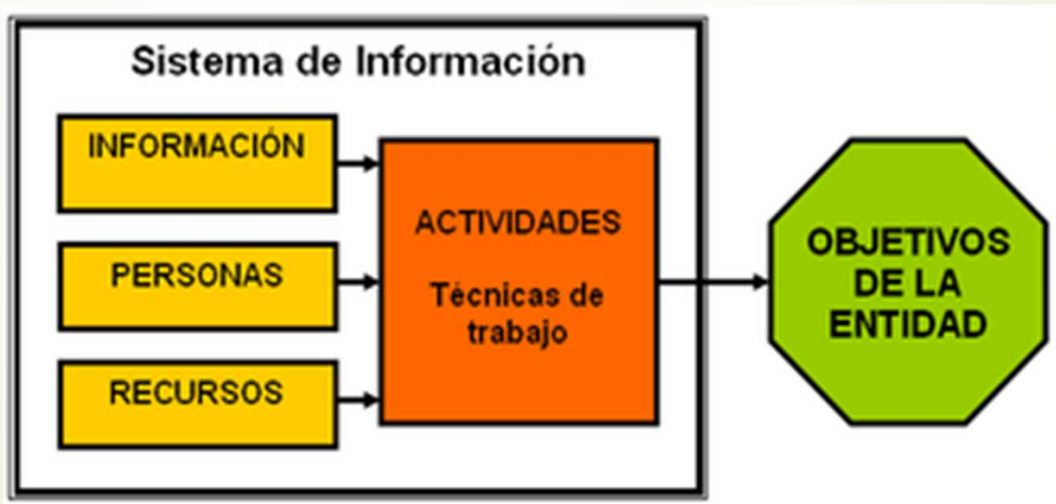
Análisis y Diseño de Sistemas de información

- 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 o un objetivo.**



Análisis y Diseño de Sistemas de información

- Sistema de Información:

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

- Sistema de ventas…

Análisis y Diseño de Sistemas de información

- **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 y diagnostico de problemas para recomendar mejoras al sistema. Se establecen los servicios que se deben brindar y sus restricciones.

- **Diseño:** Es el proceso de planificar e idear una solución en base a necesidades previamente fijadas. Se utilizan estructuras y mecanismos para tal fin.

Análisis y Diseño de Sistemas de información

- **Objetivos:**

Análisis de Sistemas:

Tomar una visión general del sistema.

En esta etapa los hechos relevados deben **documentarse de manera comprensible** para el cliente y los desarrolladores.

Diseño de Sistemas:

Especificar las necesidades de hardware y software. Tiene como finalidad **resolver los problemas** planteados en la etapa anterior para poder luego transformar dichas soluciones a código de computadoras.

Análisis y Diseño de Sistemas de información

- **¿Qué es un sistema?**

  Es un conjunto de componentes que interactúan entre si para lograr un objetivo común.

- **Clasificación de Sistemas:**

  Abiertos: Interactúan con el ambiente.

  Cerrados: No interactúan con el ambiente.

Definiciones en general

- **Sistema de Programación**

  Se compone de un conjunto de programas autónomos dedicados o no a una sola aplicación

- **Subsistema:** Es un sistema de programación que forma parte de otro mas grande.

- **Programa:** Es la especificación de la solución a un problema, que ejecuta una computadora.
- **Proceso:** Es un programa en ejecución.

Definiciones en general

- **Objeto de programa:** cualquier entidad que en un programa puede recibir un nombre. (variables, constantes, procedimientos, funciones)
- **Modulo:** es una "colección de objetos de programa" identificada con un nombre. (Objeto -> Modulo)
- **Procedimiento:** es un objeto ejecutable de un programa.
- **Función:** es un tipo particular de procedimiento que devuelve un solo valor.

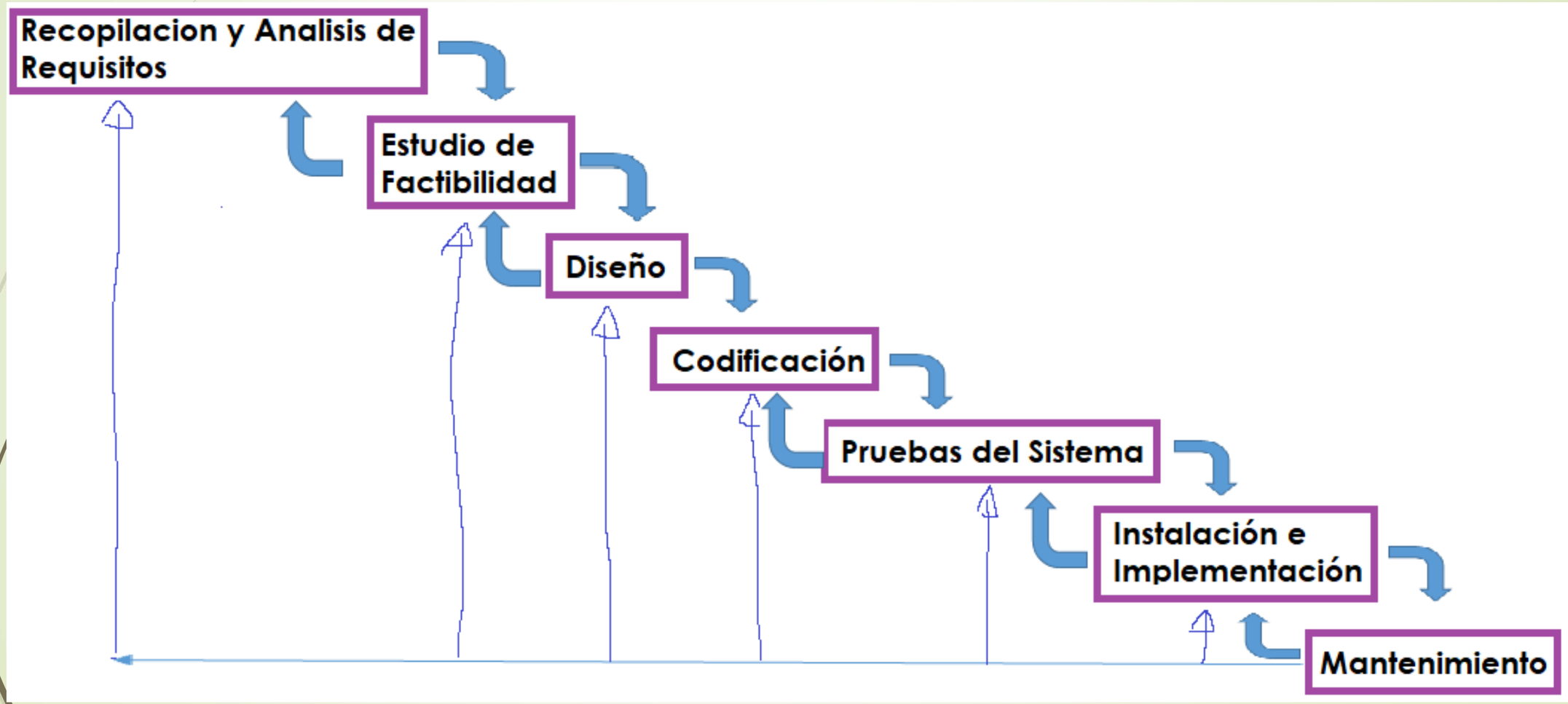
Ciclo de Vida del Desarrollo de Software (SDLC)

Conjunto de actividades que analistas, diseñadores y usuarios realizan para desarrollar e implementar un sistema de software.

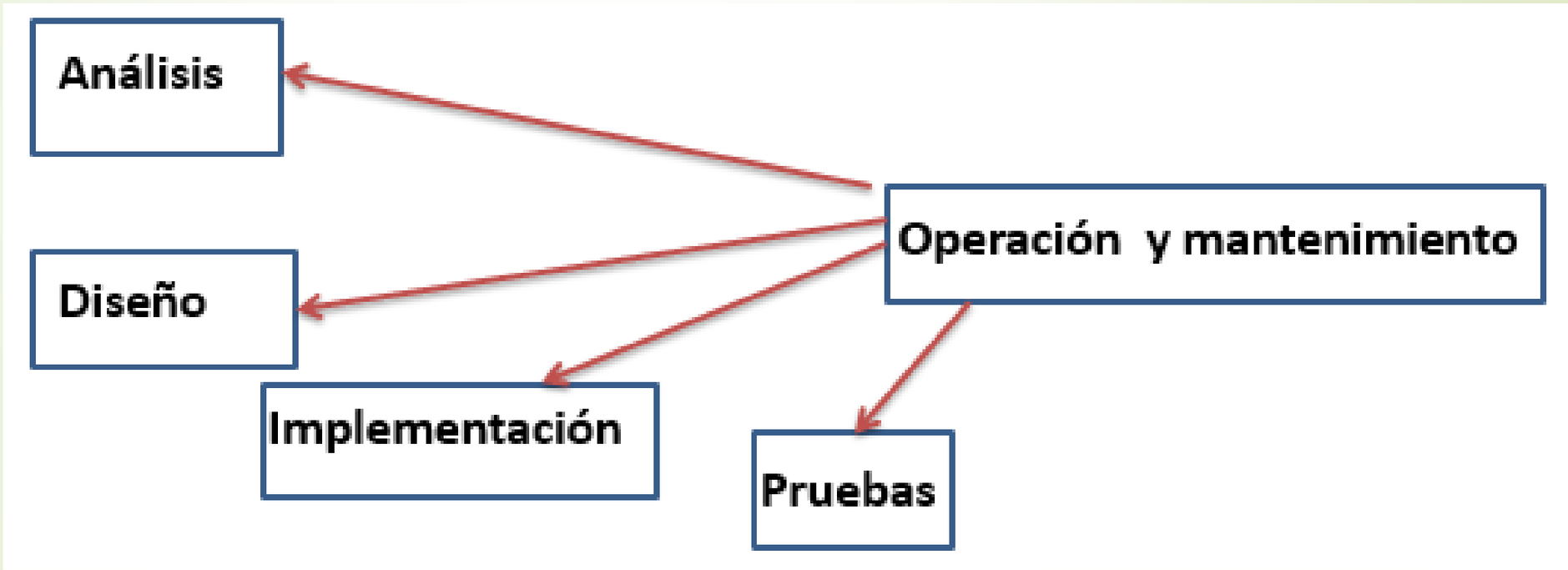
Ciclo de Vida del Desarrollo de Software (SDLC)

1. **Recopilación y análisis de requisitos**
   - **Investigación Preliminar.**
   - **Determinación de Requisitos del sistema.**
2. **Estudio de Factibilidad**
3. **Diseño**
   - **Diseño de alto/bajo nivel.**
4. **Codificación**
   - **Pruebas de Unidades.**
5. **Pruebas del Sistema**
   - **Pruebas de Aceptación.**
6. **Instalación e Implementación**
7. **Mantenimiento**
   - **Evaluación**

Ciclo de Vida del Desarrollo de Software (SDLC)



Ciclo de Vida del Desarrollo de Software (SDLC)



Ciclo de Vida del Desarrollo de Software (SDLC)

1. Recopilación y Análisis de Requisitos

Implica adquirir, reconocer y comprender las facetas importantes del sistema que está bajo estudio

a) ¿Qué es lo que hace?
b) ¿Cómo lo hace?
c) ¿Con qué frecuencia se presenta?
d) ¿Qué tan grande es el volumen de transacciones o de decisiones?
e) ¿Cuál es el grado de eficiencia con que se realizan las tareas?
f) ¿Existe algún problema?
g) Si existe, ¿qué tan serio es?
h) Si existe, ¿cuál es la causa que lo origina?

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Análisis y determinación de Requisitos**

Es el **estudio de un sistema** para conocer como trabaja y donde es necesario efectuar mejoras.

Estudiar un sistema puede permitir comprenderlo, cambiarlo, mejorarlo, adaptarlo, duplicarlo y explicarlo.

Estudiar un sistema sirve para comprender el funcionamiento del sistema, descubrir sus límites/fronteras visibles y/o no visibles, entender el objetivo del sistema y cómo interactúa con otros sistemas externos.

Ciclo de Vida del Desarrollo de Software (SDLC)

- **¿Qué es un Requisito?**

Es una característica que debe incluirse en el nuevo sistema.

- El analista debe encontrarlos y clasificarlos según su tipo:

  Esenciales

  No esenciales

Ciclo de Vida del Desarrollo de Software (SDLC)

- **¿Cómo determinar requisitos?**

  Se deben realizar 3 grandes actividades:

  Anticipación: Preveer características del sistema, investigar aspectos relacionados aunque esto introduzca sesgo propio.

  Investigación: Estudio de los mismos.

  Especificación: Los datos obtenidos durante la recopilación se analizan para determinar las especificaciones de requisitos.

  ¿Es posible hacerlo y cumple con solucionar algún problema?

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Técnicas para encontrar hechos**

Entrevistas

Se recolecta información de personas.

Es un método de contacto directo con personas.

El éxito depende de la preparación del entrevistador.

Cuestionarios:

Se recolecta información de un grupo grande de personas.

Puede asegurar el anonimato

Método indirecto que no da posibilidad de re-preguntar.

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Técnicas para encontrar hechos**

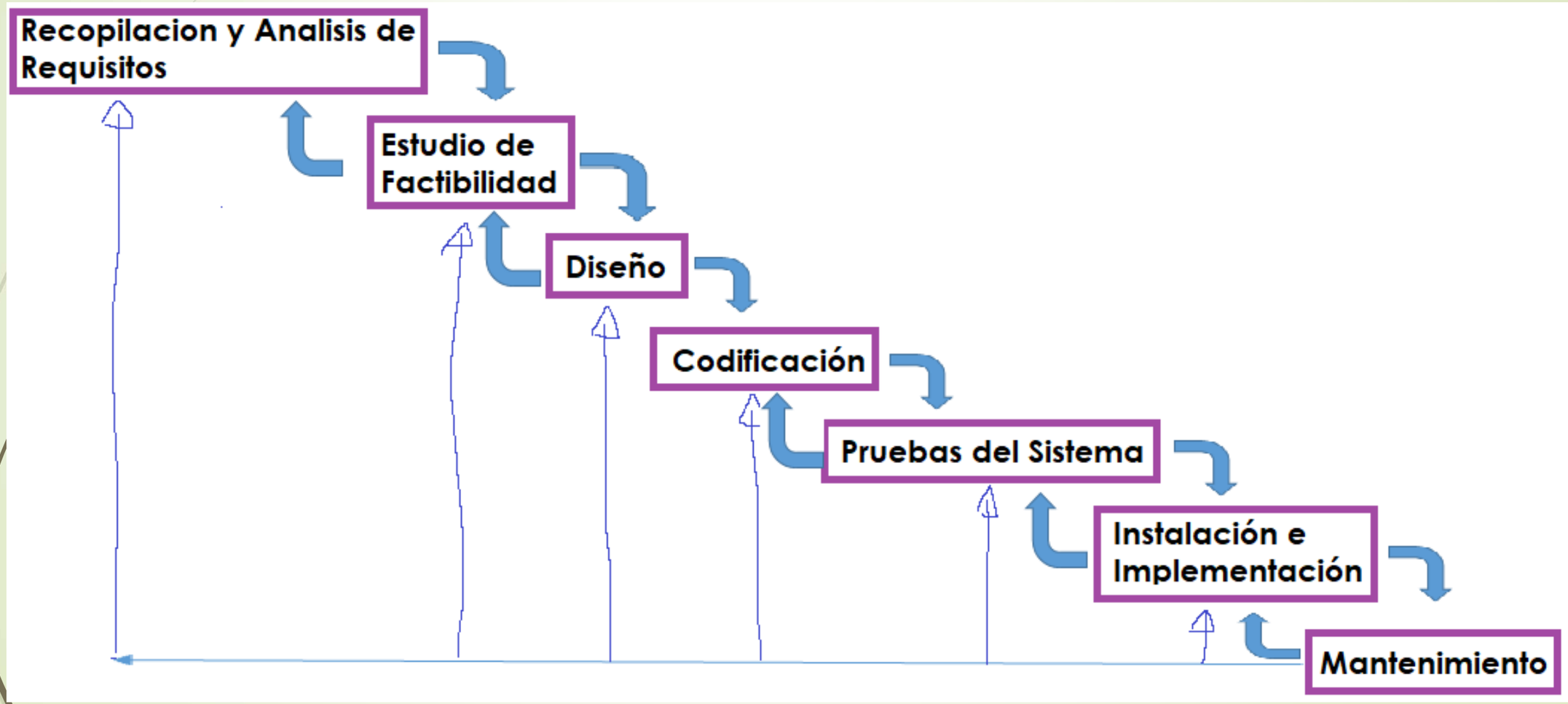
Revisión de los registros:

Examinar documentación gestionada por el cliente.

Examinar entradas y salidas del sistema que se desea reemplazar.

Observación In-situ: permite observar la realidad tal cual es sin ser manipulada.

Ciclo de Vida del Desarrollo de Software (SDLC)



Ciclo de Vida del Desarrollo de Software (SDLC)

- Fundamentos del Análisis de requisitos:
  - Dificultad del cliente de expresar sus necesidades.
  - Distinguir lo necesario de lo que quiere "sin fundamentos".
  - El analista actúa como un **interrogador**, **consultor** y **persona que resuelve problemas**. (Ej. trabajo de migración de datos)

- **1) Reconocimiento de problemas**
  - Identifica la características del problema.

Ciclo de Vida del Desarrollo de Software (SDLC)

- Fundamentos del Análisis de requisitos:
  - **2) Evaluación de problemas y síntesis de la solución**
    - Se concentra en los **datos de salida**.
    - Funciones que debe realizar
    - Interfaces definidas.
    - Restricciones que se deben aplicar.

Ciclo de Vida del Desarrollo de Software (SDLC)

- Fundamentos del Análisis de requisitos:

  - **3) Modelamiento**
    - Se utilizan modelos para representar el sistema a construir.
    - Los modelos se utilizan como base de revisión entre el cliente y los desarrolladores.
    - Se puede utilizar lenguajes formales como UML.

Ciclo de Vida del Desarrollo de Software (SDLC)

- Fundamentos del Análisis de requisitos:
  - **4) Especificación (de los requisitos)**
    - Documentar lo producido en pasos **2 y 3**.
      - Predicción de costos
      - Beneficios
      - Características del sistema
      - Cronograma de desarrollo.
  - **5) Revisión**
    - El "Documento de Requisitos" sirve de contrato.

Ciclo de Vida del Desarrollo de Software (SDLC)

- Los roles:
  - Analista
    - Detecta hechos relevantes en la actividad de la empresa.
    - Reúne información y determina requisitos.
    - Sugiere mejoras. ¿Dónde?
  - Diseñador
    - Diseña el nuevo sistema. Estructura. Resuelve.
    - Es una tarea técnica.
  - Programador
  - Testeador

Ciclo de Vida del Desarrollo de Software (SDLC)

- Los roles:
  - Jefe de Etapa
  - Jefe de Proyecto

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Revisión permanente con el cliente**
  - **La entrevista se divide en 3 momentos**
    - El antes:
      - Preparación previa
    - El durante:
      - Actitud receptiva
      - Demostrar interés por el tema
      - Motivar a usuarios a comentar detalles del sistema.
    - El después
      - Documentar lo charlado y revisión con el cliente.

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Fase 2. Estudio de Factibilidad**
  - Técnica
    - ¿Existe la tecnología de software y hardware necesaria?
  - Económica
    - ¿Se cubren los costos?
    - ¿Representa una ganancia?
  - Operativa
    - Se analiza el recurso humano.
    - ¿Habrá resistencia al cambio de sistema?
  - Legal
    - ¿Es legal hacer lo que me piden?
  - Temporal
    - ¿Puede el proyecto ser completado en el tiempo dado?

Ciclo de Vida del Desarrollo de Software (SDLC)

Fase 3. Diseño

- Se estable la forma en la que el sistema cumplirá los requisitos identificados en la etapa de Análisis.

- **“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 mas programas de computación”.**

Ciclo de Vida del Desarrollo de Software (SDLC)

- Tareas de Diseño:
  - Identificación de Reportes y salidas (con detalles).
  - Datos de entrada, cálculos intermedios y los que se almacenarán.
  - Se describen en detalle los procedimientos de calculo y los datos individuales.
  - Se seleccionan las estructuras de almacenamiento de datos y los dispositivos de almacenamiento.
  - Los procedimientos indican **cómo** procesar datos y producir salidas.

Ciclo de Vida del Desarrollo de Software (SDLC)

Fase 4. Codificación

- Tareas de la Codificación:
  - **Transcribir a código ejecutable** las especificaciones de diseño.
  - **Documentar el cómo y porqué** se programan los procedimientos y funciones de cierta manera.
  - Realizar las **pruebas de unidad**.
  - **Depurar**

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Fase 5. Pruebas**

  - Se integran las unidades individuales y se realiza pruebas integradoras para corroborar que todo funcione.
    - **Validar:**
      - ¿Se ha construido el producto correcto?
      - ¿Cumple con lo solicitado por el cliente?
    - **Verificar:**
      - ¿Se ha construido el producto correctamente?
  - Grupos de Prueba:
    - **ALFA**
    - **BETA**

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Fase 6. Instalación e implementación**

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.

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Fase 7. Mantenimiento**
  - Tipos de mantenimiento:
    - **Correctivo:** Corregir errores no descubiertos previamente.
    - **Perfectivo:** Mejorar la funcionalidad del sistema.
    - **Adaptativo:** Modificar el sistema para hacer frente a necesidades del entorno.
    - **Preventivo:** Modificar rutinas de código que son propensas a generar errores futuros.

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Fase 7. Mantenimiento**
  - Dimensiones en la evaluación de un sistema:

    - **Evaluación Operacional:**
      - ¿Cómo funciona el sistema? Facilidad de uso.

    - **Impacto Organizacional:**
  - Medir beneficios a nivel institucional (finanzas, eficiencia operacional, impacto competitivo)

Ciclo de Vida del Desarrollo de Software (SDLC)

- **Fase 7. Mantenimiento**
  - Dimensiones en la evaluación de un sistema:

    - **Opinión de los Administradores:**
      - Evaluación de directivos, administradores y usuarios directos.

    - **Desempeño del desarrollo:**
      - Evaluación del proceso de desarrollo(tiempo, dinero y esfuerzo).
      - Valoración de métodos, modelos y herramientas de desarrollo.

Ciclo de Vida del Desarrollo de Software (SDLC)

- Pequeño análisis de costos

Tipos de Sistemas	Costo Porcentual por Fase		
	Análisis/Diseño	Implementación	Pruebas
Sistemas de Mandato y Control	46%	20%	34%
Sistemas Aéreos	34%	20%	**46%**
Sistemas Operativos	33%	17%	**50%**
Sistemas Científicos	44%	26%	30%
Sistemas Comerciales	44%	28%	**28%**

- ¿Como reducir el costo de desarrollo?

Documento de Requisitos

- La **especificación de requisitos de software** (**ERS**) es una descripción completa del comportamiento del sistema que se va a desarrollar.

En primera instancia cumple la **función de un contrato.**

- Contiene:
  - Requisitos funcionales
  - Requisitos no funcionales (complementarios).
  - También puede contener lo que **no se va a hacer**

- Está dirigida tanto al cliente como al equipo de desarrollo.

Documento de Requisitos

Características de una buena ERS:

1. **Completa**: Deben estar presente todos los requisitos incluso lo que no se hace.
2. **Consistente:** Debe ser coherente con los propios requerimientos y también con otros documentos de especificación.
3. **Inequívoca:** La redacción debe ser clara para evitar malas interpretaciones.
4. **Correcta:** El software debe cumplir con los requisitos de la especificación.
5. **Trazable:** Se refiere a la posibilidad de verificar la historia, ubicación o aplicación de un ítem a través de su identificación almacenada y documentada.
6. **Priorizable:** organizar jerárquicamente los requisitos según su relevancia para el negocio ó para las personas, y clasificándolos en esenciales y opcionales.
7. **Modificable:** Se refiere a que debe ser fácilmente modificable.
8. **Verificable:** Debe existir un método finito sin costo para poder probarlo.

Documento de Requisitos

- **Alcances del sistema**
  - Consiste en dar una descripción precisa y comprensible para determinar las dimensiones del sistema.
    - Funciones esperadas
    - Performance deseada
    - Confiabilidad Requerida
    - Comunicación con otros sistemas…
    - Limitaciones conocidas y dependencias.
  - No debe incluirse descripción técnica.

Documento de Requisitos

- **Beneficios esperados**
  - Es “vender” el sistema.
  - Potenciar sus mejores características.
  - No debemos reducirlo a la expresión “ahorro de tiempo y dinero”
  - Puede disminuir el gasto administrativo.
  - Se deben ofrecer servicios que nos diferencien de la competencia (Requisitos no esenciales).

Documento de Requisitos

- **Definir usuarios**
  - Usuarios primarios
    - Interactúan con el sistema ingresando datos y recibiendo salidas.
  - Usuarios Indirectos
    - Reciben información y reportes del sistema pero no ingresan datos.
  - Usuarios gerentes
    - Monitorean y controlan procesos institucionales.
  - Usuarios directivos
    - Responsables del “sistema de información”
    - Evalúan riesgos por fallas en el sistema de información

Documento de Requisitos

- **Definición de Objetivos**
  - Funcionales y de Eficiencia
  - Seguridad
    - Acceso indebido
    - Protección de los datos almacenados (sensibles)
    - Protección de los datos transferidos.
  - Mantenibilidad
  - Características deseables para los objetivos
    - Que sean evaluables
    - Que sean razonables en numero
    - Que sean coherentes entre si.

Documento de Requisitos

- **Análisis de Factibilidad**
  - Técnica
    - ¿Existe la tecnología de software y hardware necesaria?
  - Económica
    - ¿Se cubren los costos?
    - ¿Representa una ganancia?
  - Operativa
    - Se analiza el recurso humano.
    - ¿Habrá resistencia al cambio de sistema?
  - Legal
    - ¿Es legal hacer lo que me piden?
  - Temporal
    - ¿Puede el proyecto ser completado en el tiempo dado ?

Procesos del Ciclo de Vida

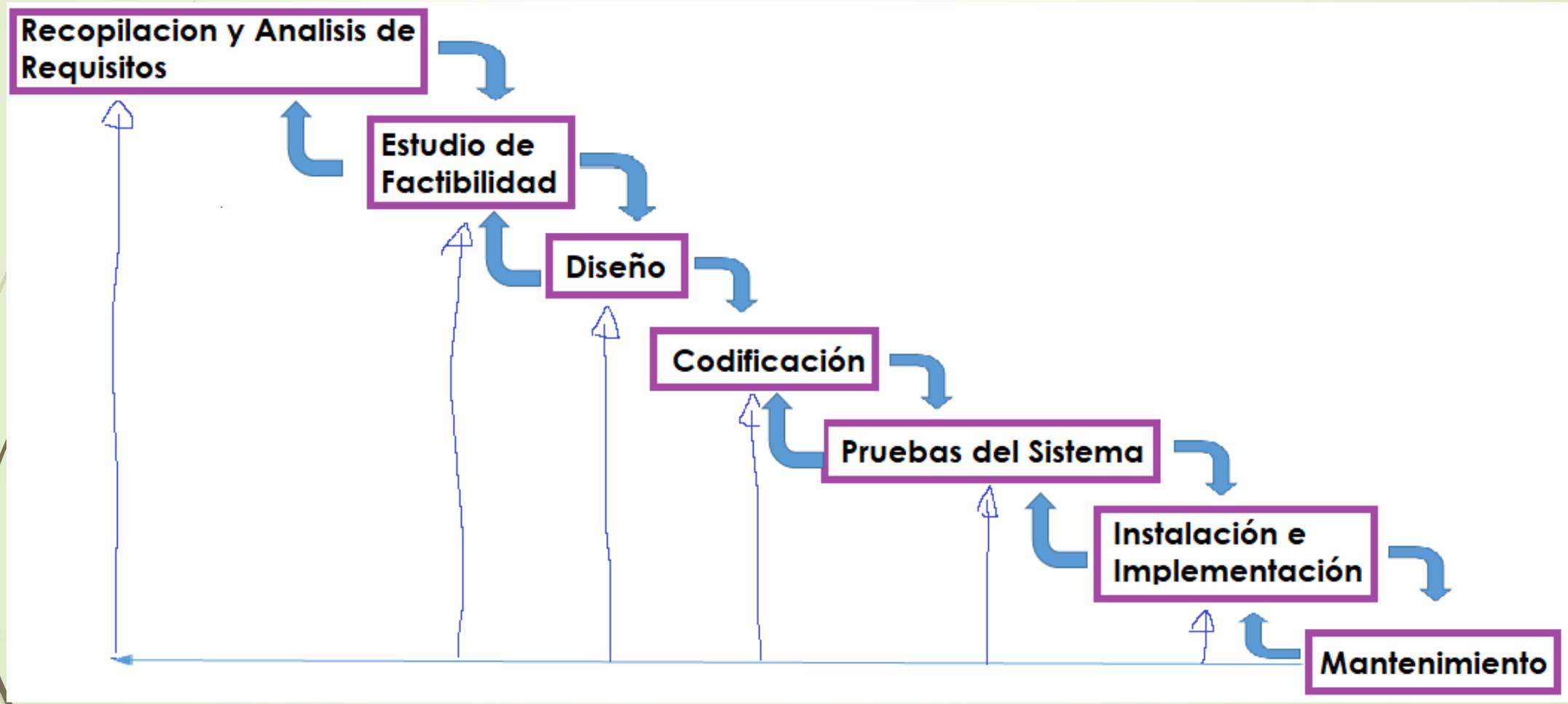
- El proceso de desarrollo de software es un **conjunto de actividades relacionadas** por un modelo o metodología de desarrollo que son útiles para generar productos intermedios de trabajo que permiten desarrollar el producto final de software.
- Es una visión orgánica de la evolución del software durante su desarrollo y uso

El desarrollo de sistemas de software se hace usando un **modelo de proceso.**

Procesos del Ciclo de Vida

- Actividades genéricas de un modelo de proceso
  1. Recopilación y análisis de requisitos
  2. Estudio de Factibilidad
  3. Diseño
  4. Codificación
  5. Pruebas del Sistema
  6. Instalación e Implementación
  7. Mantenimiento

Ciclo de Vida del Desarrollo de Software (SDLC)



Procesos del Ciclo de Vida

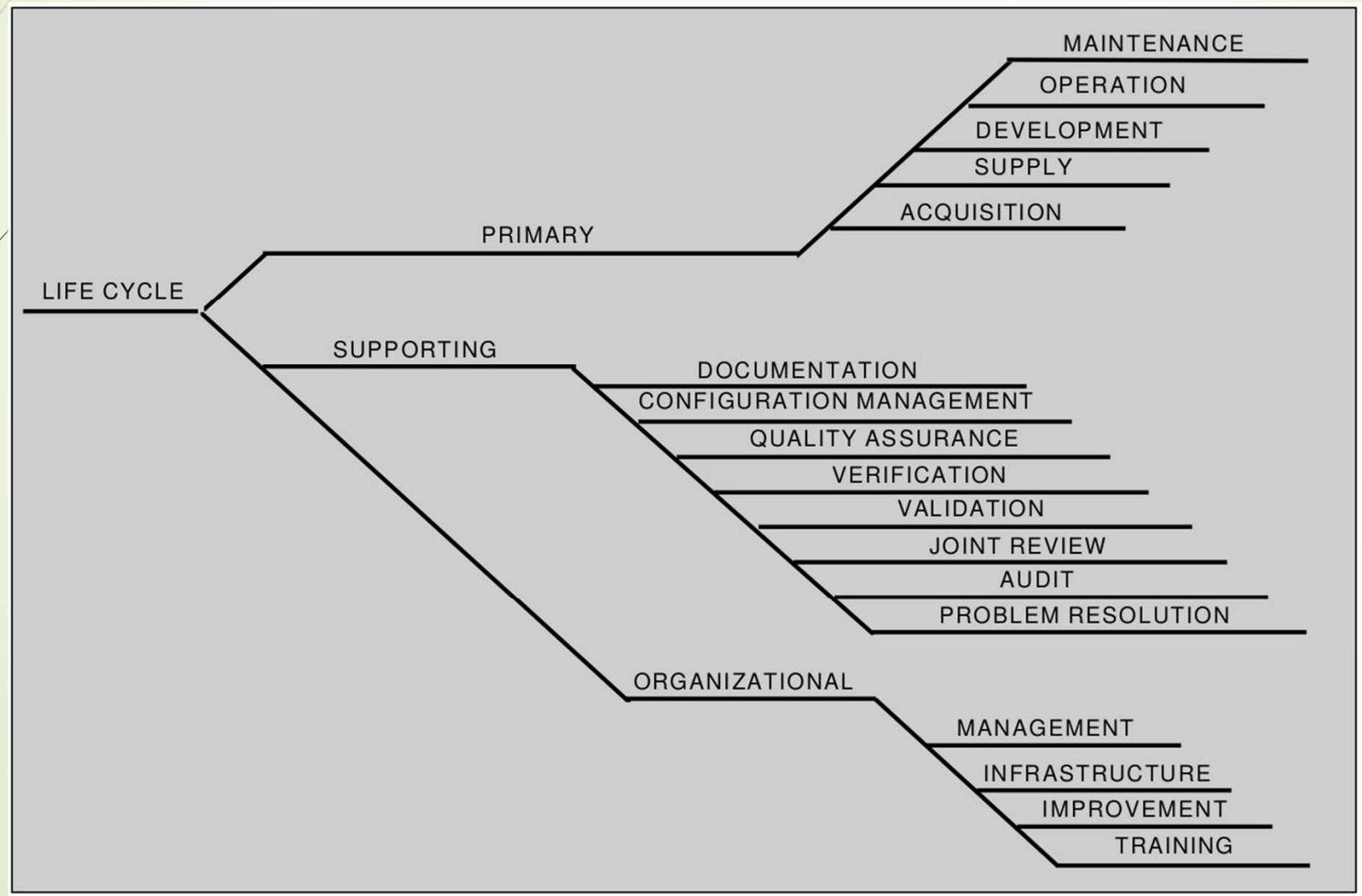
- ¿Que es la ISO/IEC 12207?

El 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 los detalles del proceso.

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

Ciclo de vida del desarrollo de Software (SDLC)

- Procesos del Ciclo de Vida del Software



Ciclo de vida del desarrollo de Software (SDLC)

- Proceso primarios del ciclo de vida

- **Proceso de adquisición:** el que adquiere el software lleva a cabo actividades de selección de necesidades, pedido de propuesta, selección del proveedor, proceso de adquisición y aceptación del sistema.
- **Proceso de provisión:** el que provee el software, desarrollo, o mantenimiento, prepara una propuesta, acuerda un contrato, selecciona procesos, planea, ejecuta, y controla.
- **Proceso de desarrollo:** el/los desarrollador/es de un SW nuevo o modificaciones, revisa requerimientos, usa metodología de desarrollo, y realiza tareas propias para el desarrollo de SW (actividades genéricas).
- **Proceso de operación:** puesta en producción y soporte a usuarios.
- **Proceso de mantenimiento:** por errores, necesidades, o mejoras (correctivas, adaptativas, perfectivas, preventivas) se modifica el código, documentación mediante actividades de análisis, aceptación, migración, hasta el retiro del SW

Ciclo de vida del desarrollo de Software (SDLC)

Proceso de soporte del ciclo de vida

- **Proceso de documentación:** registro de la información producida durante el proceso para la comunicación.
- **Proceso de gestión de configuración:** se identifican los baselines para controlar los cambios (se aplica al código, configuraciones, BD, documentos).
- **Proceso de QA:** se provee un framework de calidad para el cumplimiento de los productos o servicios con sus requerimientos contractuales.
- **Proceso de verificación:** ¿el producto funciona de manera correcta?
- **Proceso de validación:** ¿es el producto correcto?
- **Proceso de revisión conjunta:** se proveen protocolos de revisión.
- **Proceso de auditoría:** se evalúan los productos y actividades con énfasis en los requerimientos y planes.
- **Proceso de resolución de problemas:** se resuelve los problemas tomando acciones correctivas.

Ciclo de vida del desarrollo de Software (SDLC)

Proceso organizacionales del ciclo de vida

- **Proceso gerencial:** define actividades y tareas de administración.
- **Proceso de infraestructura:** La infraestructura puede incluir hardware, software, standards, herramientas, técnicas y medios. Objetivo: definir las actividades para establecer y mantener la infraestructura necesaria.
- **Proceso de mejora:** Objetivos: mejorar el proceso organizacional. Actividades: evaluar, medir, controlar y mejorar el proceso de ciclo de vida.
- **Proceso de entrenamiento:** Actividades: identificar y hacer un plan para incorporar o desarrollar recursos de personal.

Ingeniería de Software

Concepto de Software

- El Software es:
  - Alma y cerebro de una computadora.
  - El **conocimiento adquirido** acerca de un área de aplicación.
  - **Conjunto de programas** y datos necesarios para convertir una computadora en una maquina de propósito especifico para una aplicación particular.
  - **Documentación producida** durante el desarrollo de un sistema

Todo lo que es el software son los aspectos de la información.

Ingeniería de Software

Características del Software

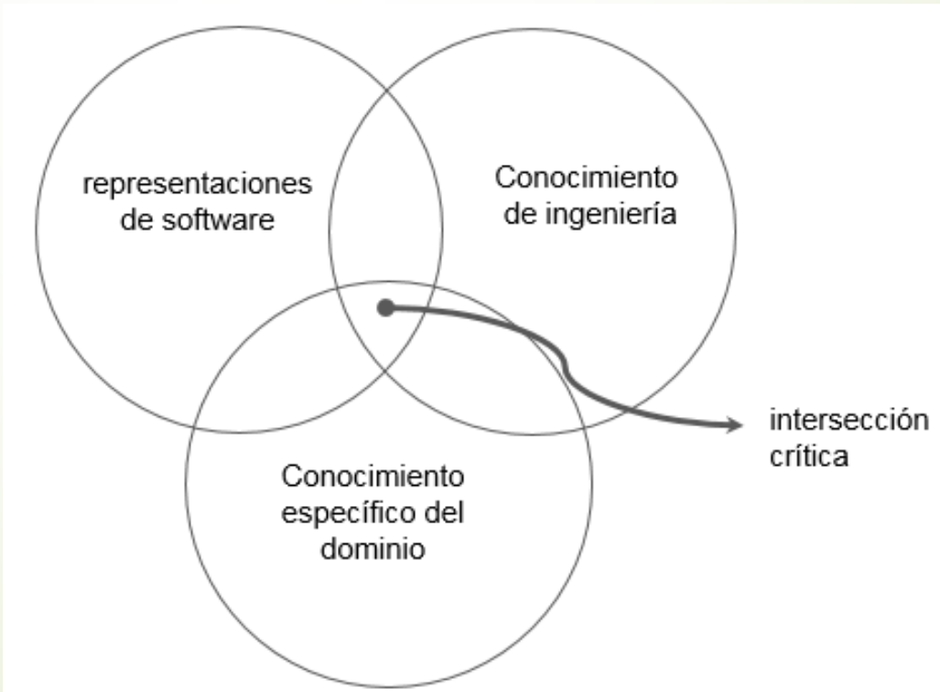
- El software es maleable.
- Se cree que los cambios en software son fáciles.
- Su producción es humano intensiva
- En la IS, a diferencia de lo tradicional, el ingeniero dispone de herramientas para describir el producto que no son distintas del producto.

Ingeniería de Software

La clave de un desarrollo exitoso es no perder o alterar información introduciendo errores.

El software no sólo son programas ejecutables ya que excluye otra información relevante.

Clases de Información



Ingeniería de Software

- Representaciones de Software

Cualquier información que en forma directa representa un eventual conjunto de programas y los datos asociados.

- Programas
- Diseños
- Especificaciones escritas en lenguaje formal
- Requisitos del sistema

Ingeniería de Software

- Conocimiento de la Ingeniería de Software

Toda información relativa al desarrollo en general (como usar método de diseño) ó **relativa a un desarrollo particular**(programa de testeo en un proyecto)

- Información sobre la tecnología de software (métodos, conceptos, técnicas)

- Conocimiento del dominio especifico

Es esencial para la creación del software. Descubrirlo y ponerlo en practica es el objetivo del especialista en el área de aplicación.

- Conocimiento del proceso especifico a ser controlado
- Reglas de Contabilidad.
- Procedimientos específicos de un área. RRHH. Sueldos. Venta.

Ingeniería de Software

El software es tanto un producto como un objeto, esto es **conocimiento empaquetado**.

Los programas contienen conocimiento. Una de las principales razones de la **reusabilidad** es no perder este conocimiento.

- Software como Conocimiento
- Software como Producto.

Ingeniería de Software

La ingeniería de software es establecer y usar los principios de ingeniería conocidos para obtener software rentable y confiable que funcione eficientemente en máquinas reales. [**Fritz Bauer** 1969]

La ingeniería de Software: (1) es un enfoque sistemático, disciplinado, y cuantificable para el desarrollo, operación, y mantenimiento del Software; esto es, la aplicación ingenieril al software. (2) El estudio de los enfoques en (1) [**IEEE** 1993]

La ingeniería de Software es una disciplina de la ingeniería que se ocupa de todos los aspectos de producción de Software [**Ian Sommerville**]

Campo de la ciencia de la computación que trata la construcción de sistemas de software que son tan grandes o tan complejos que tienen que ser construidos por un equipo o equipos de ingenieros [**Ghezzi**, Fundamentals of Software engineering, 1991]

Ingeniería de Software

La ingeniería de software es establecer y usar los principios de ingeniería conocidos para obtener software rentable y confiable que funcione eficientemente en máquinas reales. [**Fritz Bauer** 1969]

La ingeniería de Software: (1) es un enfoque sistemático, disciplinado, y cuantificable para el desarrollo, operación, y mantenimiento del Software; esto es, la aplicación ingenieril al software. (2) El estudio de los enfoques en (1) [**IEEE** 1993]

La ingeniería de Software es una disciplina de la ingeniería que se ocupa de todos los aspectos de producción de Software [**Ian Sommerville**]

Campo de la ciencia de la computación que trata la construcción de sistemas de software que son tan grandes o tan complejos que tienen que ser construidos por un equipo o equipos de ingenieros [**Ghezzi**, Fundamentals of Software engineering, 1991]

Ingeniería de Software

- Capas de la IS:

  Métodos:

  Indican como construir técnicamente el software

  - Planificación
  - Análisis de requisitos
  - Diseño de estructuras de datos y proc. Algorítmicos.
  - Codificación
  - Testeo
  - Mantenimiento

  Los métodos **proporcionan la experiencia técnica** para elaborar software

Ingeniería de Software

- Capas de la IS:

Herramientas: Proporcionan un apoyo automático o semiautomático para los **métodos** y el **proceso**.

Herramientas **CASE** (**C**omputer **A**ided **S**oftware **E**ngineering)

Ingeniería de Software

- Capas de la IS:

  Proceso: Es el pegamento que une los métodos y las herramientas y facilita el desarrollo racional y oportuno del software. Definen la secuencia en la que se aplican los métodos.

  Define la secuencia en que se aplican:

  - los métodos,
  - las entregas que se requieren
  - Los controles que ayudan a asegurar la calidad.
  - La evaluación del progreso.

Ingeniería de Software

- Capas de la IS:

Ingeniería de Software

- El proceso del software es:

Un conjunto de **actividades**, **acciones** y **tareas** que se ejecutan cuando va a crearse algún producto del trabajo.

- El proceso establece para su conformación:
  - **Actividades Estructurales**:
    - Comunicación
    - Planeación o planificación
    - Modelado
    - Construcción
    - Despliegue

Se realizan en forma iterativa

Ingeniería de Software

- El proceso establece para su conformación:
  - **Actividades Sombrilla**:
    - Seguimiento y Control:

      Progreso vs Planificación
    - Administración del Riesgo
    - Aseguramiento de la Calidad
    - Revisiones Técnicas

      Descubrir y eliminar errores.
    - Medición

Ingeniería de Software

- 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 el tiempo.

- **Flujo del proceso lineal**

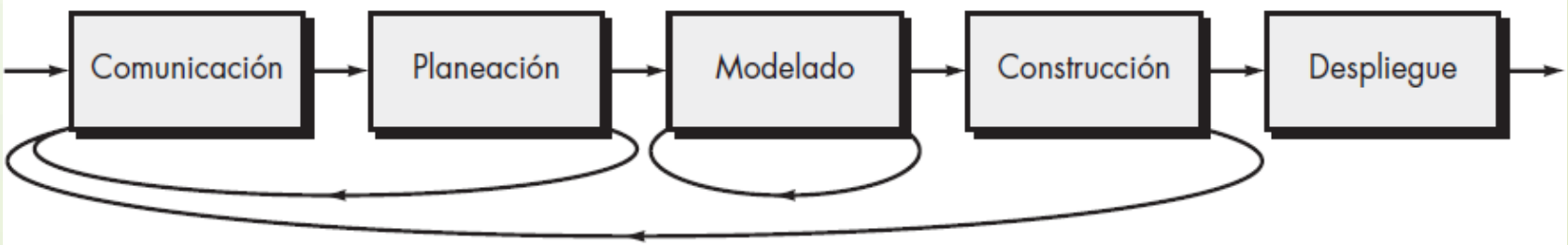
  El resultado (productos intermedios) de una actividad es la entrada de la siguiente

a) Flujo de proceso lineal

Ingeniería de Software

- Flujo de proceso
  - **Flujo de proceso iterativo**

    Las actividades se aplican en forma repetidas para completar diferentes productos

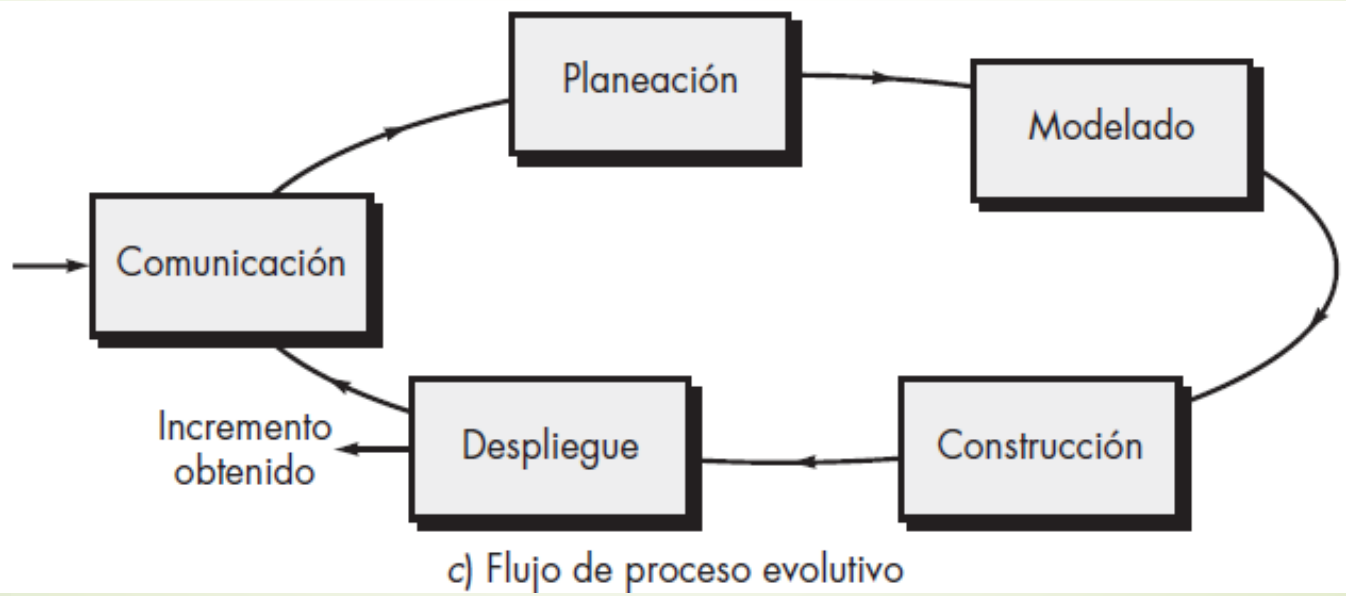


b) Flujo de proceso iterativo

Ingeniería de Software

- Flujo de proceso
  - **Flujo de proceso evolutivo/incremental**

  Genera un incremento en cada iteración.

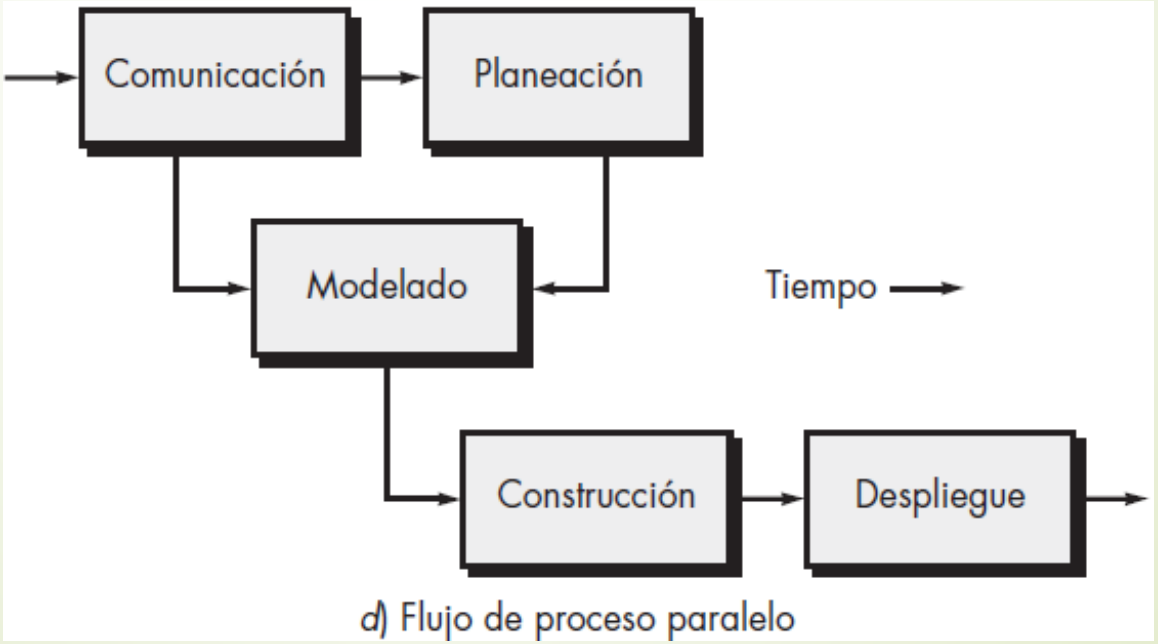


c) Flujo de proceso evolutivo

Ingeniería de Software

- Flujo de proceso
  - **Flujo de proceso paralelo**

    Ejecuta una o mas actividades en paralelo con otras.



d) Flujo de proceso paralelo

Ingeniería de Software

Modelos de Proceso Prescriptivos vs Descriptivos

- **Los modelos prescriptivos:**
  1. Modelo de la Cascada
  2. Modelo Incremental
  3. Modelo de Prototipos
  4. Modelo de Espiral

- **Los modelos descriptivos**
  - Modelos Agiles

Modelos prescriptivos vs descriptivos

- **Los modelos prescriptivos:**
  - Se centran en el control del proceso definiendo las notaciones utilizadas.
  - Orden y consistencia del proyecto son primordiales
  - Prescriben **actividades**, **acciones** ante determinados síntomas, **tareas**, productos intermedios, QA, y control de cambios.
  - El proceso se mejora formalmente. Un modelo fuertemente prescriptivo es de difícil adaptación a los cambios (difícil de personalizar).

¿Existe forma de mejorar éstos modelos?

Modelos prescriptivos vs descriptivos

- **Los modelos descriptivos**

  - Se centran en el factor humano
  - Se le da mayor valor al individuo, a la colaboración con el cliente, y al desarrollo incremental con iteraciones muy cortas.
  - Pone un acento en la disciplina del equipo
  - El proceso se adapta ante los cambios de manera más ágil
  - El proceso se mejora informalmente. Sin disciplina, un modelo sólo descriptivo puede llevar a inconsistencias y caos.
  - Estos modelos está demostrando tener éxito en entornos de requisitos cambiantes.

Modelos prescriptivos vs descriptivos

Si los modelos prescriptivos apuestan por estructurar, ordenar el desarrollo y predecir la variabilidad…

¿son éstos inapropiados para un software **que se basa en el cambio**?

Si rechazamos los modelos de proceso tradicionales, y los reemplazamos con algo menos estructurado.
¿hacemos imposible la coordinación y coherencia en el desarrollo de software?

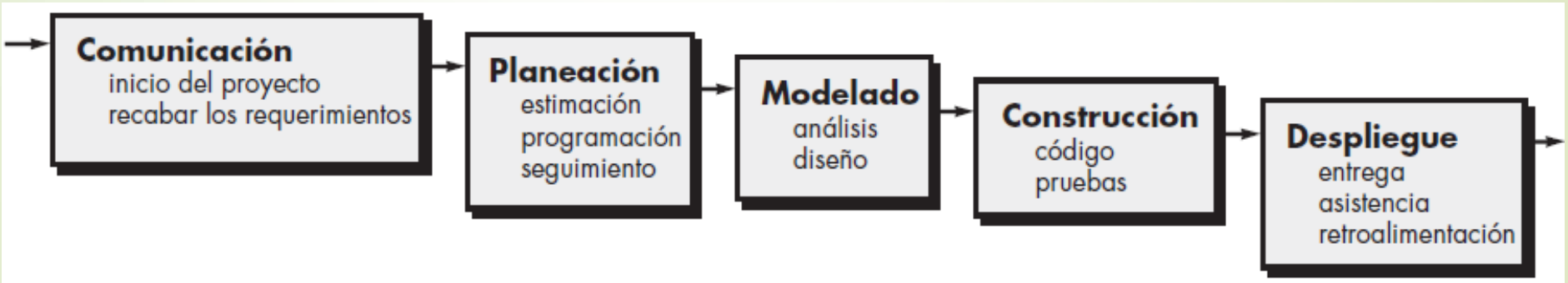
Modelos de Proceso

1. Modelo de la Cascada:

Las actividades se aplican secuencialmente

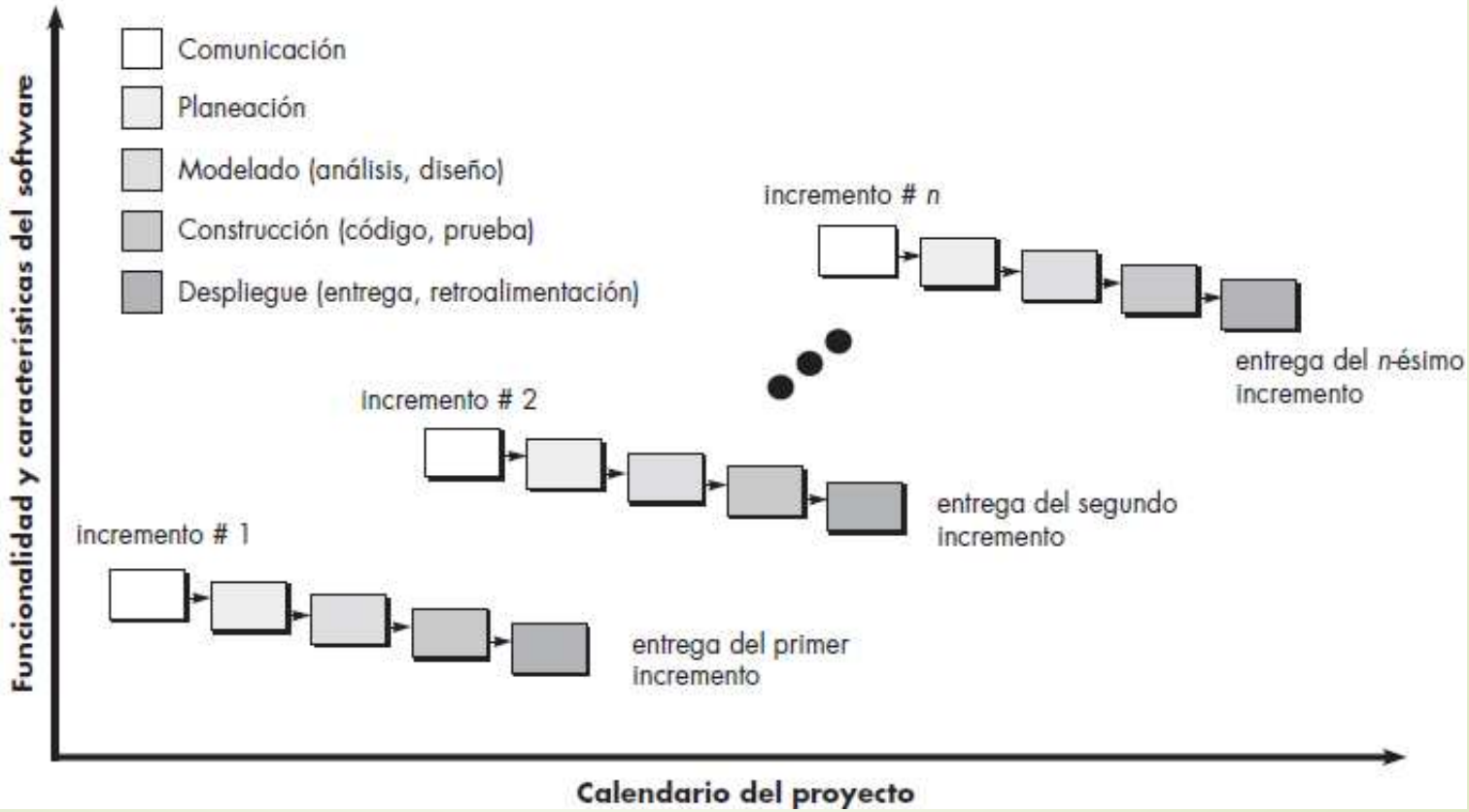
- ✓ resulta natural, sencillo de aplicar
- X particiones poco flexible del proyecto en distintas etapas, lo que representa una dificultad para responder al cambio

Apropiado cuando los requerimientos son bien entendidos y existen restricciones contractuales.



Modelos de Proceso

- **2. Modelo de Proceso Incremental:**



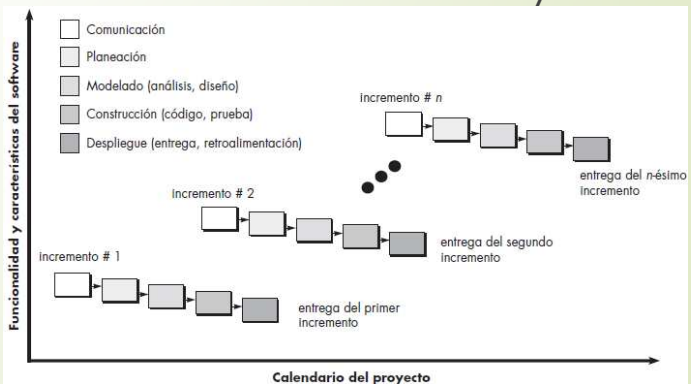
Modelos de Proceso

2. Modelo de Proceso Incremental:

Pasa por todas las actividades en forma escalonada sobre partes manejables de funcionalidad para liberar entregas frecuentes mientras progresa el calendario

- ✓ mantiene un producto estable, trabajo en paralelo, escalable
- X las funcionalidades no siempre pueden ser divididas, dificultad para manejar muchas ramas o versiones

Apropiado cuando los requerimientos no están del todo claro y se esperan cambios en los requerimientos

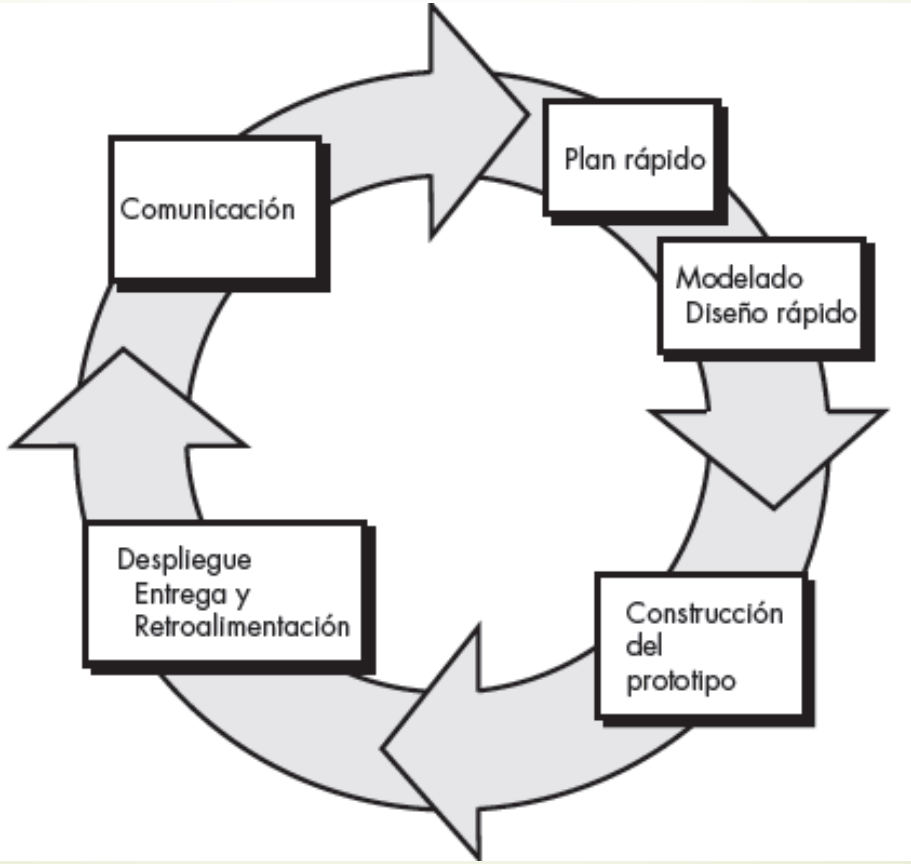


3. Modelo de Prototipo

- Definición
  - El prototipo es un sistema que funciona, desarrollado con la finalidad de probar ideas y suposiciones realizadas con el nuevo sistema. Esta 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.
    - Lehman(1984), considera que es un **modelo evolutivo** donde el concepto de aplicación se transforma paulatinamente en una especificación formal del software.

3. Modelo de Prototipo

- **Modelo de Proceso Prototipado:**



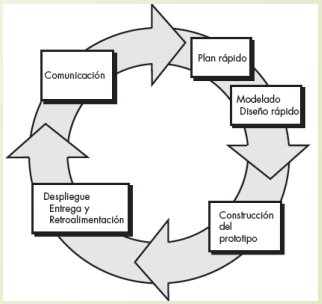
3. Modelo de Prototipo

- **Modelo de Proceso Prototipado:**

Construye rápidamente una versión tangible para el cliente para ganar conocimiento

- ✓ genera un feedback inmediato del cliente **para asentar requerimientos**
- X muchas veces el diseño pobre de un prototipo queda en versiones posteriores, difícil de escalar, y puede destruir las expectativas del cliente

Apropiado cuando se desconocen los requerimientos y el proyecto tiene riesgo moderado a alto



3. Modelo de Prototipo

- 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**.

El **mantenimiento del software** es el proceso de corregir errores en el sistema y modificarlo para que refleje los cambios en el ambiente.

Implementación de **múltiples instancias** de un mismo sistema genera múltiples versiones.

3. Modelo de Prototipo

- 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.

3. Modelo de Prototipo

- Evolución del Software

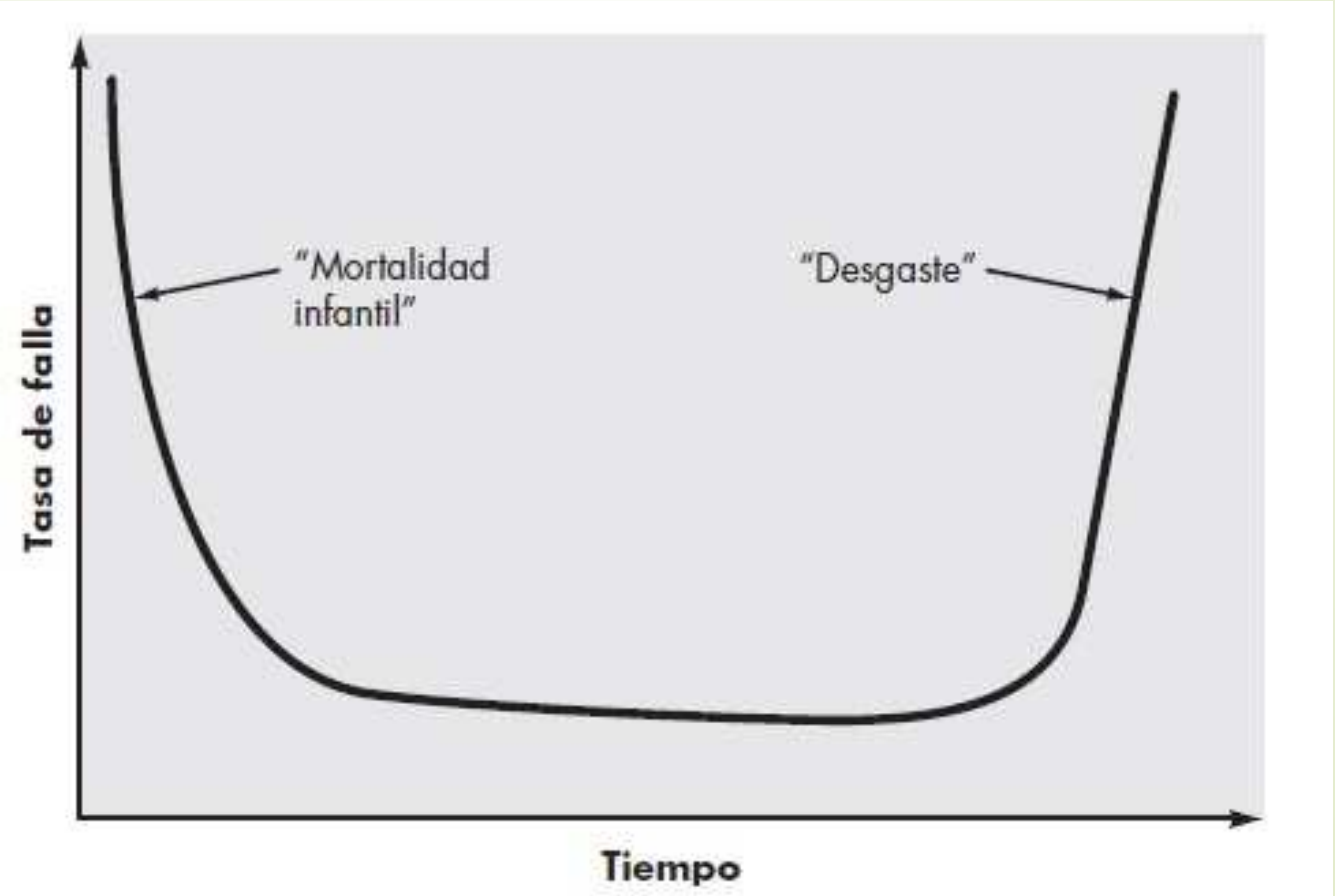
  Según Lehman, existen 5 leyes de evolución de los programas:

  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.

3. Modelo de Prototipo

- Evolución del Software



3. Modelo de Prototipo

- Confiablidad del Software

  Es la característica muy importante en el software.

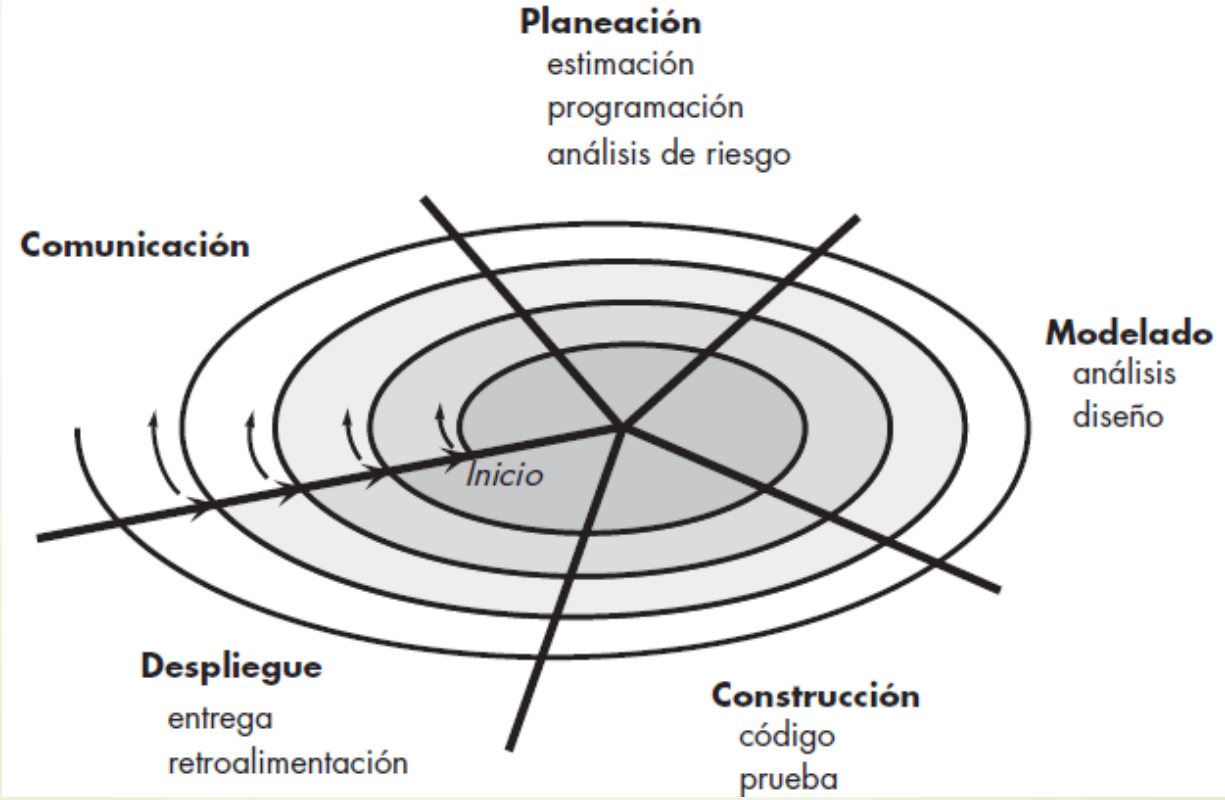
  La confiablidad **depende de:**

  - La correctitud de su diseño.
  - Cuanto se corresponde con la aplicación.
  - La confiabilidad de sus componentes.

  Es una medida de lo bien que proporciona los servicios que los usuarios esperan de él.

Modelos de Proceso

- **4. Modelo de Proceso Espiral:**



Modelos de Proceso

Conclusiones

¿Cuál, de los vistos en clase, es el **mejor** modelo de proceso?

- Modelo Cascada
- Modelo Incremental
- Modelo Prototipo
- Modelo Espiral

Análisis Estructurado

- Fundamentos
  - Los analistas deben:
    - Aprender detalles y procedimientos del S.I.
    - Documentar 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.
    - Evaluar la eficiencia y efectividad del sistema actual y de sus procedimientos.
    - Recomendar, con la debida justificación, mejoras y ampliaciones del sistema actual.
    - Fomentar la participación de gerentes y empleados en todo el proceso de desarrollo del sistema.

Análisis Estructurado

- ¿Cómo funciona?

  - El método de desarrollo de sistemas por análisis estructurado **divide el sistema en componentes y construye un modelo** para el mismo, **incorporando elementos de análisis y diseño** para favorecer una comprensión completa de los sistemas grandes y complejos.

Permite observar los **elementos lógicos** (lo que el sistema hará) separados de los **elementos físicos**(terminales, sistemas de almacenamiento).

Análisis Estructurado

- Elementos:
  - Símbolos Gráficos
    - Elementos gráficos del proceso.
    - El flujo de datos.
    - Sitio o lugar de almacenamiento.
  - Diagrama de Flujo de Datos (DFD)
    - La descripción completa del sistema esta formada por un conjunto de DFDs. Método TOP-DOWN.

Análisis Estructurado

- Elementos:
  - Diccionario de Datos (DD)
    - **TODOS los elementos de datos** que fluyen por el sistema son descriptos en el DD.
  - Castellano Estructurado
    - También llamado Pseudo-código.
    - Descripción algorítmica del proceso.
    - Es una descripción preliminar, sin detalles de implementación.

Diseño Estructurado

- Utiliza los mismos elementos del Análisis Estructurado.
- Tiene por objetivo crear programas formados por **módulos independientes desde el punto de vista funcional.**
- Se toman decisiones importantes de naturaliza estructural.
- **El diseño es conducido por la información.**

Análisis Estructurado

- Símbolos Gráficos:
  - **Flujo de Datos:**

    Representa un camino a lo largo del cual se mueve una estructura de datos.

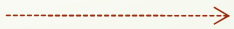
    Yourdon

    Gane y Sarson

  - **Flujo de Transacción o Control:**

    Representa un camino a lo largo del cual fluye el control de un proceso a otro.

    Ward y Mellor

Análisis Estructurado

- Símbolos Gráficos:
  - **Entidad Externa:**

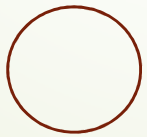
    Representa una fuente o destino de datos que no pertenece al sistema.

Yourdon

Gane y Sarson

  - **Proceso:**

    Representan entidades o funciones transformadoras de datos.



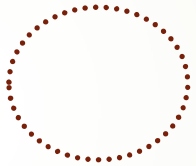
Yourdon

Gane y Sarson

Análisis Estructurado

- Símbolos Gráficos:
  - **Proceso de Control**

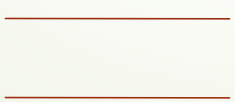
    Representa una función que no transforma sino que deriva caminos de acción.



Ward y Mellor

  - **Almacenamiento de Datos:**

    Representan el lugar donde se almacenan datos.

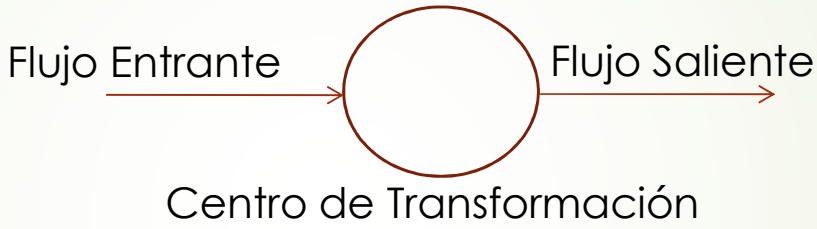


Yourdon

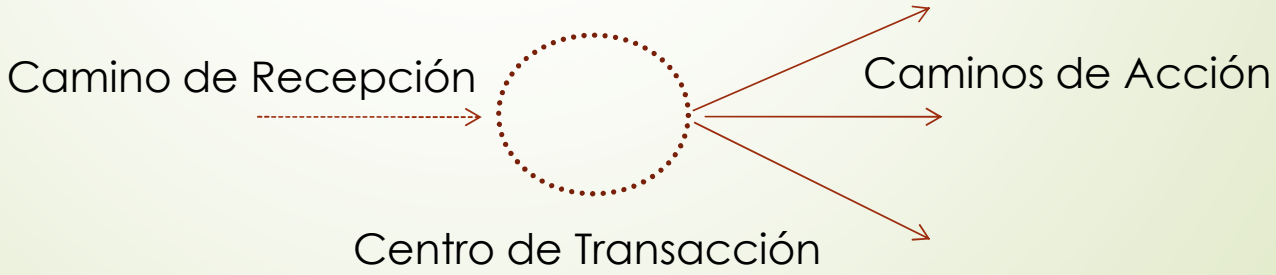
Gane y Sarson

Análisis Estructurado

- Tipos de Flujo de Información:
  - **Flujo de Transformación:**



  - **Flujo de Transacción:**



Análisis Estructurado

- Ventajas asociadas al uso de los DFDs

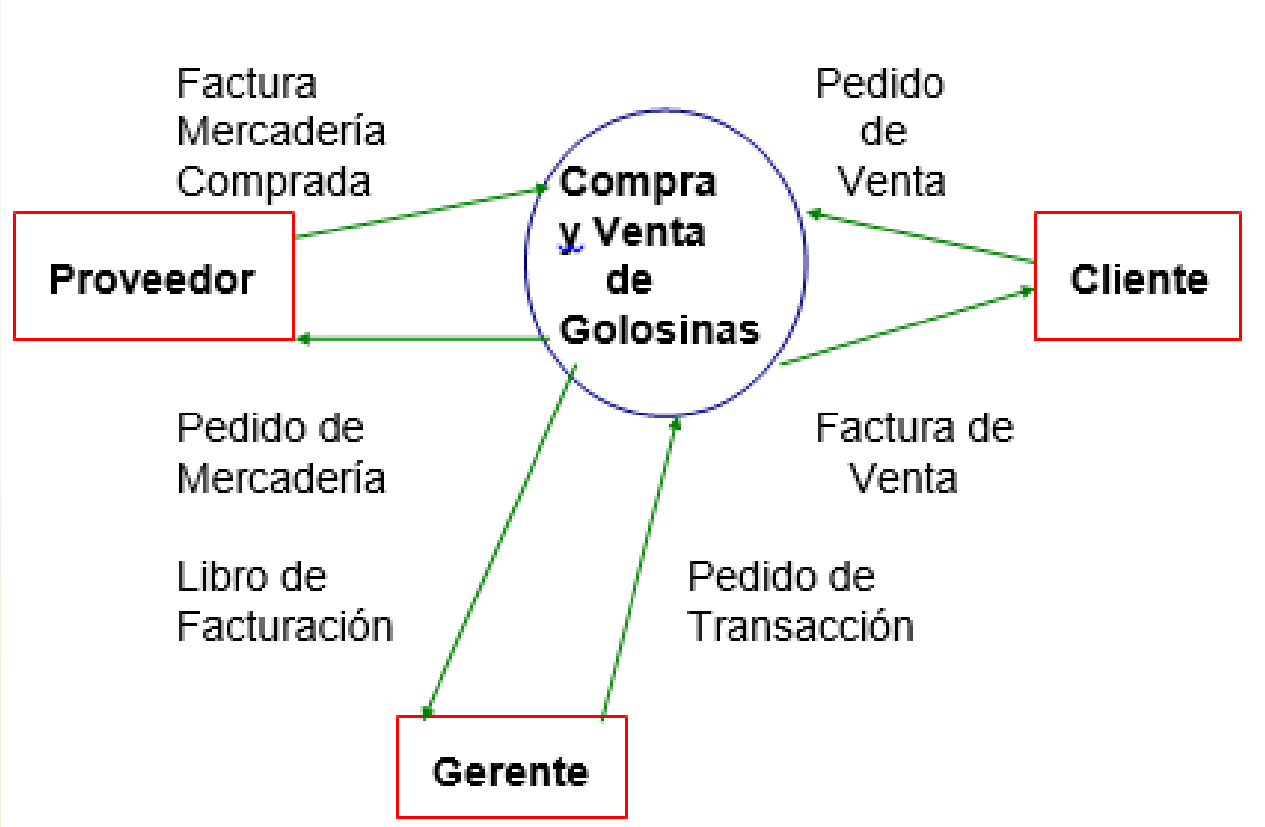
- Se identifican y describen hasta el final del proceso, **TODOS** los datos que fluyen por **TODO** el sistema.

- Se explica por qué los datos entran y salen del proceso y cual es el procesamiento que se realiza sobre ellos.

- Son tan fáciles de leer que permiten que analistas trabajen junto a usuarios para discutir sobre los procesos.

Análisis Estructurado

- Ejemplo de Diagrama de Flujo de datos (DFD)



Análisis Estructurado

- Niveles en 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.
- En cada nuevo nivel se agregan detalles, pero siempre manteniendo la consistencia con lo especificado en el nivel anterior.
- El diagrama de Nivel 0 se llama diagrama de Contexto.