**Unidad 2 - Ciclo de Vida para el desarrollo de Sistemas (SDLC)**

**Definición de análisis de sistemas**

* El análisis de sistemas tiene como propósito analizar sistemáticamente las entradas, el procesamiento, el almacenamiento de datos y la posterior salida de información en el contexto de una empresa en particular.
* Estudio de un sistema y sus componentes
* El **análisis** especifica qué es lo que el sistema debe hacer y el **diseño** establece cómo alcanzar el objetivo.

**Definición de Ingeniería de Software (Según Bauer)**

* Establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales.

**Definición y alcance de Ingeniería de Software (IEEE)**

* La ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la Ingeniería al Software

**Ingeniería de Software**

* Incluye procesos, métodos y herramientas que permiten elaborar a tiempo y con calidad sistemas complejos.
* Es necesario realizar un esfuerzo conjunto para entender el problema antes de desarrollar una aplicación de software.
* Debe existir un compromiso organizacional con la calidad.
* Se caracteriza como “una tecnología multicapa”.
* El objetivo de la IS es lograr productos de software de calidad (tanto en su forma final como durante su elaboración), mediante un proceso apoyado por métodos y herramientas.

**Procesos de desarrollo de software**

* Un proceso de software es una serie de actividades relacionadas que conduce a la elaboración de un producto de software.
* Puede ser:
  + Desde cero.
  + Extendiendo un sistema existente.
  + Modificando un sistema existente.
* Incluye cuatro actividades fundamentales:
  + Especificación: Se define la funcionalidad del software y las restricciones de su operación.
  + Diseño e implementación: Se desarrolla el software para cumplir con las especificaciones.
  + Validación: se verifica que cumple con lo requerido por el cliente.
  + Evolución: Para satisfacer necesidades cambiantes.
* No hay proceso ideal.
* Muchas organizaciones diseñan sus propios métodos.
* Los procesos tienen en cuenta las características específicas de los sistemas (sistemas críticos, empresariales, etc).
* Se busca siempre entregar el software a tiempo y con suficiente calidad para satisfacer a quienes patrocinaron su creación y a aquellos que lo usarán.

**Modelos de procesos de desarrollo de software**

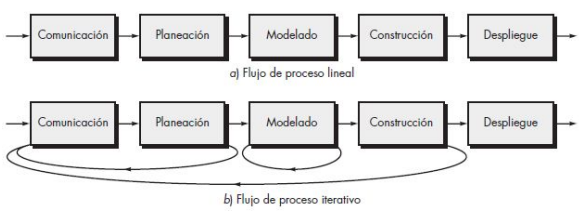
* Representación simplificada de este proceso.
* Denominados “Paradigmas de proceso”.
* Se muestran desde una perspectiva arquitectónica.
* Representan el marco (framework) del proceso, pero no los detalles de las actividades específicas.
* Son modelos genéricos.
* No son descripciones definitivas de los procesos de software.
* Son marcos del proceso (framework), que pueden extenderse y adaptarse para crear procesos más específicos de ingeniería de software.
* La existencia de un proceso del software no es garantía de que el software se entregue a tiempo o que satisfaga las necesidades de los clientes.

**Modelos de proceso**

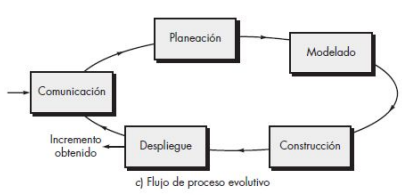
* Dirigidos por un plan: Todas las actividades del proceso se planean por anticipado y el avance se mide contra dicho plan.
* Procesos ágiles:
  + Planeación incremental.
  + Mas fácil modificar el proceso para reflejar los requerimientos cambiantes del cliente.
* Cada enfoque es adecuado para diferentes tipos de software. Por lo general, uno necesita encontrar un equilibrio entre procesos dirigidos por un plan y procesos ágiles.

**Flujos de proceso**

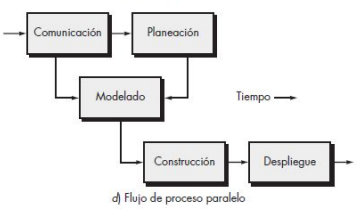
* Un flujo de proceso lineal ejecuta cada una de las actividades en forma secuencial.
* Un flujo de proceso iterativo repite una o mas de las actividades antes de pasar a la siguiente.



* Un flujo de proceso evolutivo realiza las actividades en forma “circular”. Cada circuito lleva a una versión más completa del software.

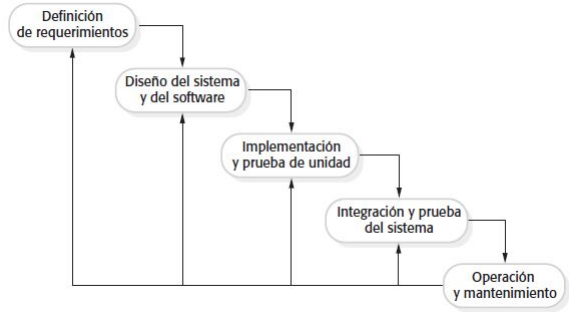


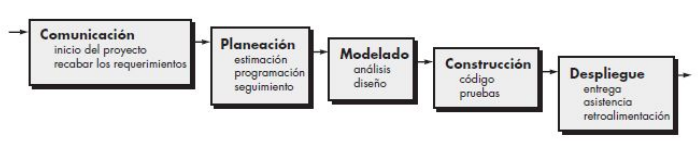
* Un flujo de proceso paralelo ejecuta una o más actividades en paralelo con otras.



**Modelo en Cascada**

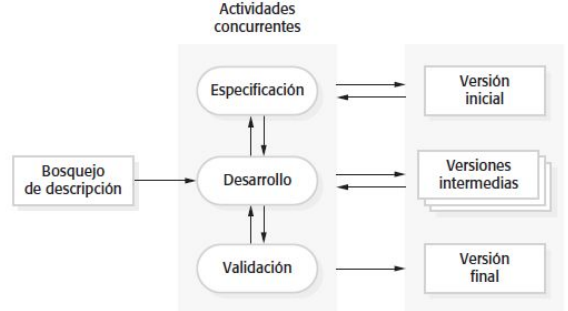
* Primer modelo publicado sobre el proceso de desarrollo de software.
* Debido al paso de una fase a la otra, este modelo se conoce como “modelo en cascada” o ciclo de vida del software.
* Se planifican todas las actividades del proceso con anticipación.
* Cada fase produce documentación que permite monitorear el progreso contra el plan de desarrollo.
* Se recomienda cuando todos los requerimientos han sido establecidos claramente desde un principio.

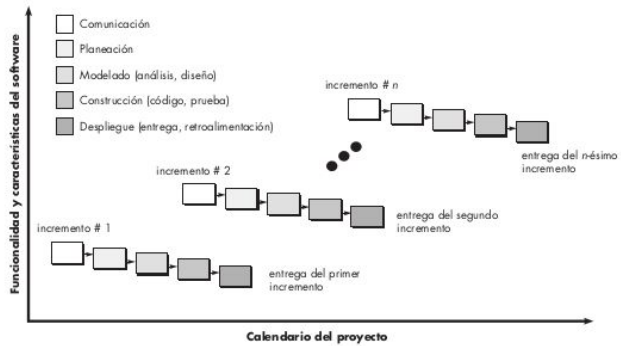




**Modelo incremental**

* Ejecuta una serie de avances (incrementos) que en forma progresiva dan más funcionalidad al cliente.
* El primer incremento suele ser el producto fundamental. Se abordan requerimientos básicos:
  + El cliente usa o evaluá el producto fundamental.
  + Se desarrolla un plan para el incremento que sigue.
* Este proceso se repite después de entregar cada incremento, hasta terminar el producto final.

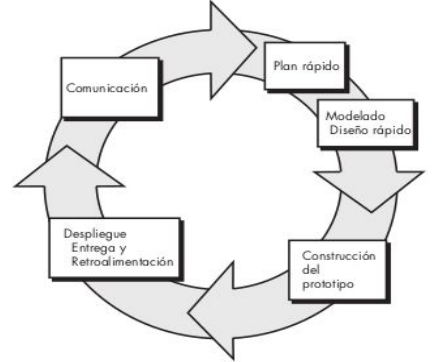




* Resulta más barato y fácil realizar cambios en el software conforme éste se diseña.
* Es más sencillo obtener retroalimentación del cliente sobre el trabajo de desarrollo que se realizó.
* Los clientes tienen posibilidad de usar y ganar valor del software más temprano de lo que sería posible con un proceso en cascada.
* Es útil en particular cuando no se dispone de personal para la implementación completa del proyecto en el plazo establecido por el negocio.

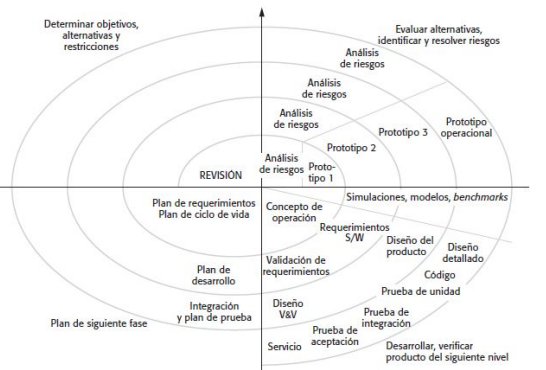
**Modelo de Construcción de Prototipos**

* Los prototipos sirven como mecanismo para identificar los requerimientos del software cuando no están claros.
* Los usuarios adquieren la sensación del sistema real y los desarrolladores logran construir algo de inmediato (“primer sistema”).
* En general, se reduce el número de propuestas de cambio de requerimientos posterior a la entrega.
* Todos los participantes deben estar de acuerdo en que el prototipo sirva como el mecanismo para definir los requerimientos.



**Modelo en espiral**

* Boehm (1988) propuso un marco del proceso de software dirigido por el riesgo.
* El proceso de software se representa como una espiral y no como una secuencia de actividades con cierto retroceso de una actividad a otra.
* Cada ciclo en la espiral representa una fase del proceso de software.



* Cada ciclo se divide en 4 sectores:
  + Establecimiento de objetivos: Se definen objetivos específicos del proyecto. Se identifican restricciones en el proceso y el producto, y se traza un plan de gestión detallado.
  + Valoración y reducción del riesgo: En cada uno de los riesgos identificados del proyecto, se realiza un análisis minucioso. Se dan acciones para reducir el riesgo.
  + Desarrollo y validación: Después de una evaluación del riesgo, se elige un modelo de desarrollo para el sistema.
  + Planeación: se decide si continuar o no con otro ciclo de la espiral. Si se opta por continuar, se trazan los planes para la siguiente fase del proyecto.

**Estudios de factibilidad**

* Factibilidad técnica:
  + Es una medida del éxito de la puesta en práctica de la solución técnica específica y de la disponibilidad de los recursos y los conocimientos técnicos necesarios.
  + Ej: ¿Existe o se puede adquirir la tecnología necesaria para realizar lo que se pide?
* Factibilidad financiera y económica:
  + Es una medida de la eficacia de los costos asociados a un proyecto o una solución, recibe el nombre de análisis costo-beneficio.
  + Ej: El costo de llevar a cabo la investigación completa de sistemas
* Factibilidad operacional:
  + Los proyectos propuestos únicamente tienen beneficio cuando logran ingresar al grupo de sistemas de información que satisfacen los requerimientos de la organización.
  + Ej: ¿Se utilizará el sistema cuando esté terminado e instalado?
* Factibilidad de fechas:
  + Es una medida del éxito que indica si un proyecto es razonable en el cumplimiento de su calendario.
  + Ej: ¿Puede la solución implantarse en un plazo razonable de tiempo?.