Ingeniería del Software

Introducción

¿Qué son los sistemas informáticos?

Un sistema informático (SI) es un sistema que permite almacenar y procesar información; es un conjunto de partes interrelacionadas: **hardware**, **software, procedimientos** y **personas**.
El hardware incluye computadoras o cualquier tipo de dispositivo electrónico, que consisten en procesadores, memoria, sistemas de almacenamiento externo, etc.
El software incluye al sistema operativo, al sistema de gestión de base de datos y aplicaciones.
Los procedimientos implican una serie de actividades del usuario que debe soportar un sistema informático y que aseguren que los datos correctos llegan a las personas adecuadas en el momento oportuno.
Por último, el soporte humano incluye al personal técnico que crean y mantienen el sistema (analistas, programadores, operarios, etc.) y a los usuarios que lo utilizan.

Ingeniería del software

El software es la suma total de los programas de computadora, la documentación asociada y los datos que pertenecen a un sistema de cómputo" y "un producto de software es un producto diseñado para un usuario". En este contexto, la **Ingeniería de Software** (SE del inglés "Software Engineering") **es un enfoque sistemático del desarrollo, operación, mantenimiento y retiro del software.**

Su origen se debe a que el entorno actual de desarrollo de sistemas software viene adoleciendo de:

- **Retrasos considerables en la planificación.**
- **Poca productividad.**
- **Elevadas cargas de mantenimiento.**
- **Demandas cada vez más desfasadas con las ofertas.**
- **Baja calidad y fiabilidad del producto.**
- **Dependencia de los realizadores.**

Esto es esto es lo que se ha denominado comúnmente "crisis del software", que se ha originado históricamente en los siguientes pasos:

- **Primera Fase**. Los albores (1945-1955). Programar no es una tarea diferenciada del diseño de una máquina. Uso de lenguaje máquina y ensamblador.

- **Segunda Fase**. El florecimiento (1955-1965). Aparecen multitud de lenguajes. Era posible hacer casi todo.

- **Tercera Fase**. La crisis (1965-1970). Desarrollo inacabable de grandes programas. Ineficiencia, errores, costo impredecible. Nada es posible.

- **Cuarta Fase**. Innovación conceptual (1970-1980). Fundamentos de programación. Verificación de programas. Metodologías de diseño.

- **Quinta Fase**. El diseño es el problema (1980-1990). Entornos de programación. Especificación formal. Programación automática.

- **Sexta Fase**. Burbuja .com (1990-?). Aplicaciones web. Y actualmente aplicaciones móviles para casi todo.

¿Cómo se define crisis?

La palabra crisis se define en el diccionario como "un punto decisivo en el curso de algo; momento, etapa, o evento decisivo o crucial". Sin embargo para el software no ha habido ningún punto crucial, sólo una lenta evolución.

La crisis en la industria del software ha permanecido durante muchos años, lo cual parece una contradicción para el término. Lo que si se podría decir es que hay un problema crónico en el desarrollo de software.

Ello ha venido originado por una falta de:

- Formalismo y metodología.
- Herramientas de soporte.
- Administración eficaz.

Actualmente está surgiendo una gran expectativa ante la evolución de la Ingeniería del Software, al ir apareciendo nuevos métodos y herramientas formales que van a permitir en el futuro un planteamiento de ingeniería en el proceso de elaboración de software. Dicho planteamiento vendrá a paliar la demanda creciente por parte de los usuarios, permitiendo dar respuesta a los problemas de:

- Administración
- Calidad
- Productividad
- Facilidad de mantenimiento

Este último es uno de los grandes problemas, pues puede llegar a suponer un importe superior al 60% del total del costo del software.

Las nuevas metodologías suponen un enfoque integral del problema, abarcando todas las fases, que en su mayoría no se consideraba en los desarrollos tradicionales. En particular son fundamentales la reducción de costos y plazos, así como la calidad del producto final. Estas tecnologías constituyen la denominada "**Ingeniería del Software**", que se puede definir como "el tratamiento sistemático de todas las fases del ciclo de vida del software". Hay otras definiciones, pero todas inciden en la importancia de una disciplina de ingeniería para el desarrollo de software.

Seguidamente se dan algunas definiciones ampliamente aceptadas dentro de la informática:

DEFINICIONES DE BOEHM

- Ingeniería es la aplicación de la ciencia y las matemáticas mediante lo cual las propiedades de la materia y las fuentes de energía de la naturaleza se hacen útiles al hombre en estructuras, máquinas, productos, sistemas y procesos.

- Software es el conjunto de programas, procedimientos y documentación asociados a un sistema, y particularmente a un sistema computacional.

- **Ingeniería de software** es la aplicación de la ciencia y las matemáticas mediante la cual la capacidad de los equipos computacionales se hacen útiles al hombre a través de programas de computadora, procedimientos y documentación asociada.

Durante los primeros años de la informática, el software se consideraba como un añadido. La programación era un "arte", para el que no existían metodologías, era un proceso que se realizaba sin ninguna planificación. En esta época toda la programación se desarrollaba a medida para cada aplicación, y en consecuencia tenía muy poca difusión, habitualmente quien lo escribía era porque lo necesitaba, y era quien lo mantenía.

En una segunda época (a partir de mitad de la década de 1960) se estableció el software como producto y aparecieron las empresas dedicadas al desarrollo y distribución masiva del mismo. El origen del término Ingeniería del Software, se atribuye a dos conferencias organizadas por la OTAN en 1967 y 1968.

La tercera era comenzó a mediados de la década de 1970, época en la que los sistemas informáticos aumentaron mucho en su complejidad, y nacieron las redes de ordenadores. Esto supuso mucha presión para los desarrolladores, aunque los ordenadores para uso personal, apenas estaban difundidos. Esta época acabó con la aparición de los microprocesadores.

La cuarta era de la evolución de los sistemas informáticos, comienza hacia 1990 y se dirige al impacto colectivo de los ordenadores y el software, en todos los entornos. La industria del software tiene un gran peso en la economía mundial. Aparecen las técnicas de redes neuronales, junto con la lógica difusa.

Hoy en día el software tiene un doble papel. Es un producto, pero simultáneamente es el vehículo para hacer entrega de un producto. Como producto permite el uso del hardware, ya sea, por ejemplo, un ordenador personal, un teléfono móvil. Como vehículo utilizado para hacer entrega del producto, actúa como base de control, por ejemplo un sistema operativo, o un sistema gestor de redes**. El software hace entrega de lo que se considera como el producto más importante del siglo veintiuno, la información**. El software transforma datos personales para que sean más útiles en un entorno local, gestiona información comercial para mejorar

la competitividad, proporciona el acceso a redes a nivel mundial, y ofrece el medio de adquirir información en todas sus formas.

La ingeniería del software trata áreas muy diversas de la informática y de las ciencias de la computación, aplicables a un amplio espectro de campos, tales como negocios, investigación científica, medicina, producción, logística, banca, meteorología, derecho, redes, entre otras muchas.

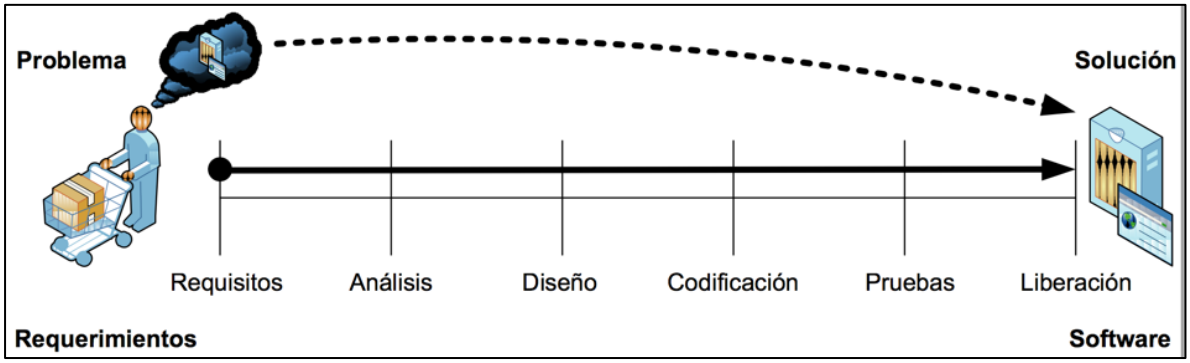
Sin embargo, es frecuente que en la práctica diaria profesional no se incluya prácticamente ninguna de las recomendaciones más elementales de la ingeniería del software. Es habitual que el desarrollo de software se parezca más al descontrol del cuento de «si los programadores fueran albañiles...» que a una idílica y bien organizada "factoría de software" (concepto de gran vigencia a finales de los ochenta).

De hecho, las evaluaciones de los procesos productivos de software realizadas a raíz de los modelos de procesos de software confirman que el desarrollo de software suele estar básicamente en estado caótico. Y no sólo en pequeñas empresas, sino también en grandes proyectos.

Ciclos de vida de desarrollo de software

Es el conjunto completo de actividades de ingeniería de software necesarias para transformar los requerimientos del usuario en software.
El proceso que se sigue para construir, entregar y hacer evolucionar un sistema informático. Desde la concepción de una idea, hasta la entrega de un sistema desarrollado con su posterior retiro.

Todos los diferentes ciclos de vida cuentan con al menos estas actividades:



Los ciclos de vida más conocidos y utilizados son:

1) **Modelo Cascada.**
2) **Prototipos.**
3) **Desarrrollo Incremental.**

1) Modelo Cascada

El modelo más antiguo en el desarrollo de software es el modelo de cascada. Tiene su origen en la fabricación y la industria de la construcción. Este modelo fue adoptado en el ámbito de desarrollo de software, ya que no existía un modelo disponible entonces. La descripción formal de este modelo fue hecha por Winston W. Royce. Este modelo sigue un flujo secuencial, donde el progreso es como el de la cascada de una etapa a otra. Las distintas fases del modelo de cascada son los siguientes:

* **Modelado del Negocio y Requerimientos del Software**: En esta fase se modelan los procesos de negocio actuales y se descubren los requerimientos del cliente. Se los documenta mediante una especificación de requisitos. Este documento se utiliza como la base para crear el sistema.

* **Diseño**: Esta es una fase importante, donde se diseña todo el software, teniendo las especificaciones de requisitos de software en consideración. La arquitectura del sistema junto con el hardware y software necesarios son trabajados.

* **Codificación**: Después de la etapa de diseño viene la etapa de codificación, donde se escribe el código del software. Cuando los módulos están listos, las pruebas unitarias se llevan a cabo en ellos, lo que ayuda en el control si hay problemas con el software. En caso de defectos, el código se corrige, antes de proceder a la siguiente etapa.

* **Prueba y depuración:** Después de que el software ha sido desarrollado por completo, se pasa a los testeadores. Se ejecutan las diferentes pruebas del software, y se corrigen los defectos descubiertos.

* **Entrega**: Una vez que el software se considera validado y verificado, después de la depuración, se inicia la implementación del software en el lado del cliente. Al cliente se le da un profundo conocimiento en el software para que sea capaz de utilizar el mismo.

* **Mantenimiento**: Después de que el software ha sido instalado, el cliente utiliza el software y puede pedir cambios. Los cambios y el mantenimiento general de los programas son atendidos en esta fase.

2) Prototipos

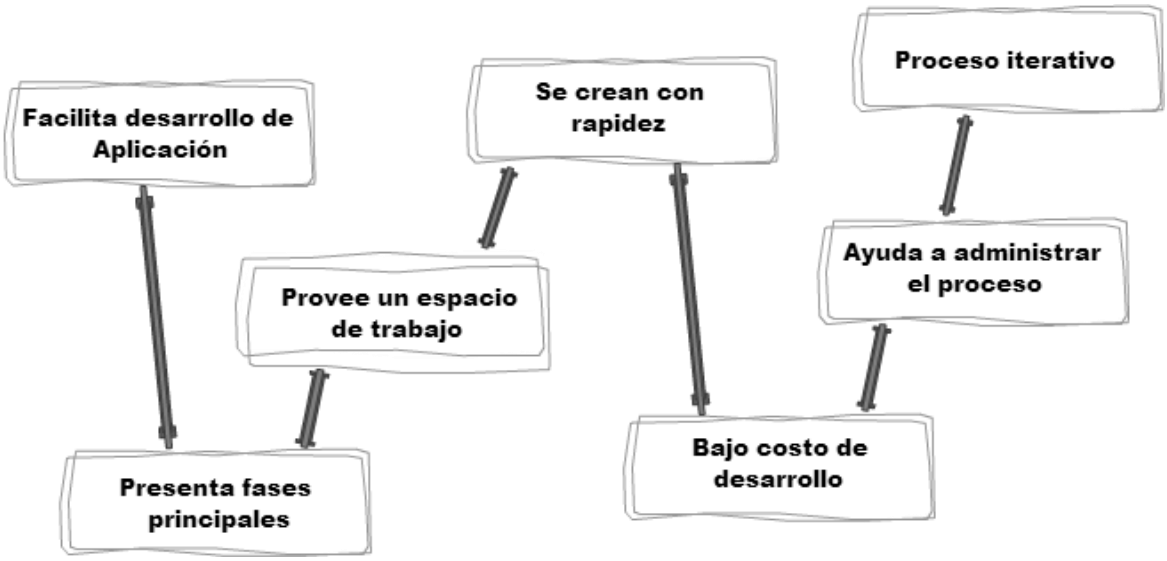
Un prototipo es la representación de un sistema. Su objetivo es lograr un producto intermedio, utilizando los programas adecuados sin gastar tanto "tiempo y dinero", y una vez aceptado se podrá iniciar el verdadero desarrollo de software.

El modelo de prototipos permite que todo el sistema, o algunos de sus partes, se construyan rápidamente para comprender con facilidad y aclarar ciertos aspectos en los que se aseguren que el desarrollador, el usuario y el cliente estén de acuerdo en lo que se necesita así como también la solución que se propone para dicha necesidad y de esta forma minimizar el riesgo y la incertidumbre en el desarrollo. Este modelo se encarga del desarrollo de diseños para que estos sean analizados y prescindir de ellos a medida que se adhieran nuevas especificaciones. Es ideal para medir el alcance del producto, pero no se asegura su uso real.
Este modelo principalmente se lo aplica cuando un cliente define un conjunto de objetivos generales para el software a desarrollarse sin delimitar detalladamente los requisitos de entrada, procesamiento y salida, es decir, cuando el responsable no está seguro de la eficacia de un algoritmo, de la adaptabilidad del sistema o de la forma en que interactúa el hombre y la máquina. Este modelo se encarga principalmente de ayudar al desarrollador y al cliente a entender de mejor manera cuál será el resultado de la construcción cuando los requisitos estén satisfechos.
El paradigma de construcción de prototipos tiene tres pasos:

- Escuchar al cliente. Recolección de requisitos. Se encuentran y definen los objetivos globales, se identifican los requisitos conocidos y las áreas donde es obligatorio más definición.
- Construir y revisar la maqueta (prototipo).
- El cliente prueba la maqueta (prototipo) y lo utiliza para refinar los requisitos del software.

Etapas para la elaboración del Modelo de Prototipo

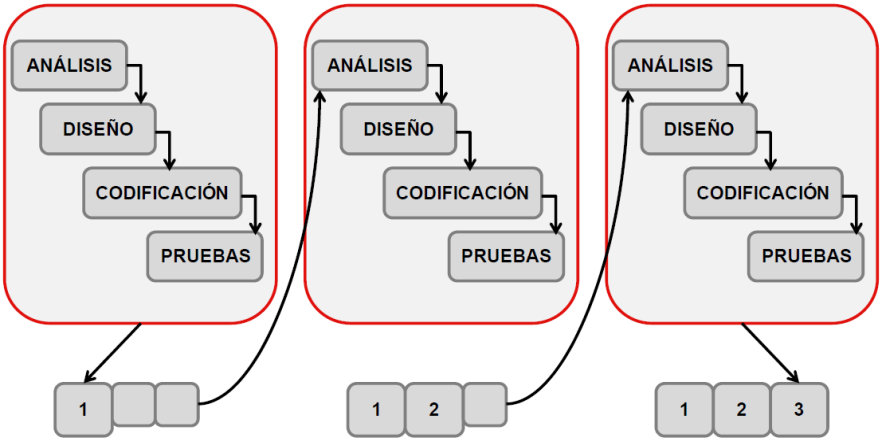
Características del Prototipo



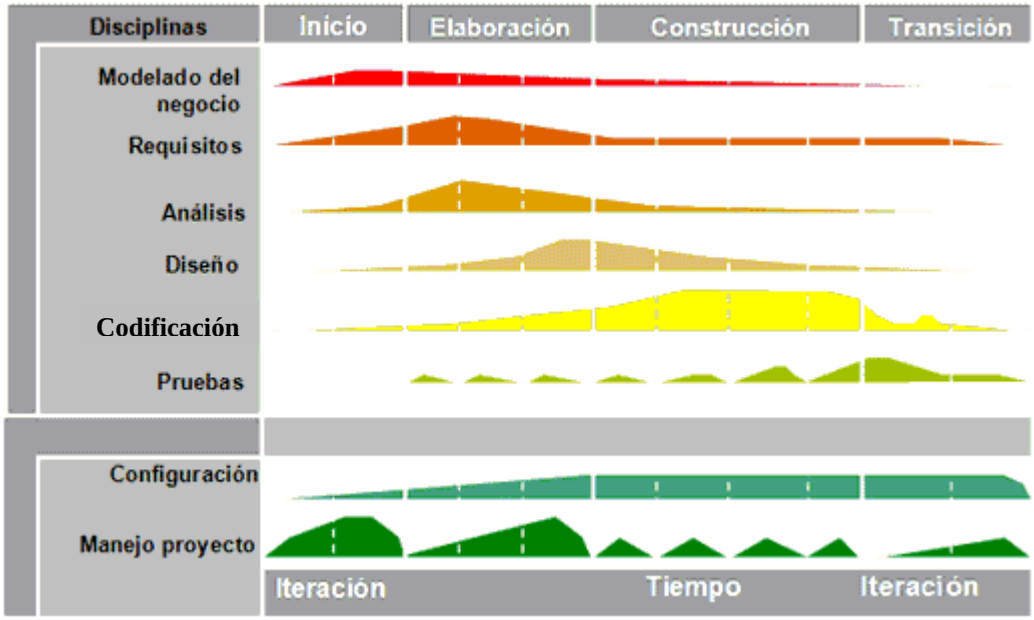
3) Desarrollo Incremental

Se basa en la ampliación y refinamiento del sistema mediante múltiples iteraciones o ciclos.
El sistema crece incrementalmente a lo largo del tiempo, a medida que se realizan las diferentes iteraciones. Al final de cada ciclo se obtiene una versión final del producto, la cual está lista para ser entregada. En caso de ser necesaria una modificación, se deben realizar los mismos pasos a lo largo de otro ciclo.

Se podría decir que es el enfoque más actual para construir software.



El desarrollo se organiza en una serie de iteraciones, cuyo resultado puede ser probado, integrado y ejecutado.



Las fases del desarrollo incremental son:

- **Inicio:** Se comprende el problema.
- **Elaboración:** Se capturan los requisitos del sistema.
- **Construcción:** Se busca obtener una funcionalidad completa.
- **Transición:** Se prepara para entregar el producto.

Al final de cada ciclo se obtiene una versión final entregable del producto.
Cada iteración incluye sus propias actividades de modelado de los procesos de negocio, análisis de requerimientos, diseño, codificación y pruebas, como así también actividades de configuración y gestión del proyecto.
La **intensidad** de cada **actividad** depende de la **fase** en la que se encuentre el proyecto de desarrollo de software.

Cuadro comparativo

Modelo	Ventajas	Desventajas
Cascada	- Simple - Organizado. - Fácil de comprender. - Fácil de utilizar o gestionar. - Sencillo para encontrar problemas antes de la implementación. - Como se establecen todos los requisitos necesarios al principio del proyecto es apropiado para proyectos estables y/o pequeños.	- Difícil que un proyecto se mantenga estrictamente en el modelo. - Genera inseguridad en el cliente. - Incertidumbre sobre el cumplimiento de los requerimientos o del tiempo planificado. - Complicado regresar a etapas anteriores. - Dificultad para incorporar nuevos requerimientos luego de iniciado el proyecto. - Costos exponenciales si se detectan problemas en etapas avanzadas del proyecto.
Prototipo	- Se reduce el riesgo y la incertidumbre sobre el desarrollo. - Genera signos visibles de progreso. - Útil cuando el cliente no identifica los requisitos detallados de entrada, procesamiento o salida. - Ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo o de la forma que debería tomar la interacción humano-máquina. - Permite entender bien el problema antes de la implementación final. - Costos reducidos, ya que es posible realizar cambios en etapas tempranas del desarrollo.	- Requiere trabajo del cliente para evaluar los distintos prototipos y traducirlo en nuevos requisitos. - No se sabe exactamente cuánto será el tiempo de desarrollo ni cuantos prototipos se tienen que desarrollar. - Una vez que el cliente ha dado su aprobación final al prototipo y cree que está a punto de recibir el proyecto final, se encuentra con que es necesario reescribir buena parte del prototipo para hacerlo funcional. El cliente ve funcionando lo que para él es la primera versión del prototipo pero que ha sido construido con "plastilina y alambres", y puede desilusionarse al decirle que el sistema aún no ha sido construido.
Desarrollo Incremental	- Se evitan proyectos de larga duración y se le entrega "algo de valor" al cliente con frecuencia. - Proporciona todas las ventajas del modelo en cascada realimentado, reduciendo sus desventajas sólo al ámbito de cada incremento. - Se pueden gestionar las expectativas del cliente de manera de regular permitiéndole tomar decisiones en cada iteración. - Permite conocer el progreso real del proyecto. Esto le permite al cliente decidir priorizar aspectos, añadir nuevos equipos o cancelar el proyecto. - Minimiza el número de errores en el desarrollo y permite aumentar su calidad.	- La entrega temprana de los proyectos produce la creación de sistemas demasiados simples que a veces se ven un poco monótonos a los ojos del cliente que lo recibe. - Requiere de un cliente involucrado durante todo el curso del proyecto. Hay clientes que simplemente no estarán dispuestos a invertir el tiempo necesario. - Puede sufrir penalizaciones en proyectos en los cuales los requerimientos están previamente definidos, o para proyectos "todo/nada" en los cuales se requiere que se completen en un 100% el producto para ser implementado (por ej., licitaciones de productos software llave en mano).