**Crisis de la ingeniería de software. El porqué de la ingeniería de software**

La sofisticación del hardware ha dejado a tras nuestra capacidad de construir software que pueda explotar el potencial del hardware.

Nuestra capacidad de construir nuevos programas no puede descansar a la demanda de nuevos programas.

Nuestra capacidad de mantener los programas existentes está amenazada por el mal diseño y el uso de recursos inadecuados.

**Definición**

“La ingeniería de software es el establecimiento y uso de principios robustos de la ingeniería a fin de obtener económicamente software que sea fiable y que funcione eficientemente sobre máquinas reales” Fritz Bauer

La ingeniería de software se divide en cuatro componentes.

Compromiso con la calidad

Fases

Herramientas

Proceso

**Fases de desarrollo de software (ciclo de vida)**

* Analisis y definición de requerimientos

Entender el problema y capturar los requerimientos funcionales, entidades y actividades

* Diseño del sistema

Proceso creativo de transformar un problema en una solución, se determinan tecnologías de programación y computo

* Codificación del sistema

Proceso ejecutado por el programador de acuerdo al diseño y de manera individual

* Pruebas unitarias

Se examina el código y se compara con las especificaciones y el diseño (se prueban los métodos individualmente)

* Pruebas de integración

Se realizan mediante incorporación de módulos, se evalúa la comunicación que tienen entre ellos y sirven para determinar qué tan eficiente es el diseño

* Pruebas del sistema

Se revisa que el sistema cumpla con las especificaciones iniciales, a veces involucra al cliente

* Implementación del sistema

Instalación, capacitación y plan de implementació

* Mantenimiento

Puede ser correctivo (eliminar errores que se encuentran después de la entrega), adaptativo (En caso de que se implemente una nueva tecnología en el sistema) o perfectivo (hacerlo mas eficiente)

* Fin del ciclo de vida (Port-mortem)

Se cierra el proyecto, se analiza la madurez en el proceso de desarrollo, se establecen áreas de oportunidad, lecciones aprendidas, mejora continua

**Herramientas**

Instrumento o sistema automatizado para realizar de manera mas exacta , eficiente y productiva una tarea.

Ejemplos:

Lenguajes de programación, paradigmas de programación, IDE’s, SGBD,herramientas de modelado, de gestión de proyectos, sistemas de control de versiones, de control de cambios, API’s para pruebas unitarias, sistemas de integración, cloud computing, software de ofimática, de prototipo y maquetado, CASE, etc

**Proceso**

El procedimiento es una receta para lograr el cometido. Establece las actividades principales. Utiliza recursos y está sujeto a una serie de restricciones.

Un proceso es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo. Una actividad busca lograr un objetivo amplio (por ejemplo, comunicación con los participantes) y se desarrolla sin importar el dominio de la aplicación, tamaño del proyecto, complejidad del esfuerzo o grado de rigor con el que se usará la ingeniería de software. Una acción (diseño de la arquitectura) es un conjunto de tareas que producen un producto importante del trabajo (por ejemplo, un modelo del diseño de la arquitectura). Una tarea se centra en un objetivo pequeño, pero bien definido (por ejemplo, realizar una prueba unitaria) que produce un resultado tangible. En el contexto de la ingeniería de software, un proceso no es una prescripción rígida de cómo elaborar software de cómputo. Por el contrario, es un enfoque adaptable que permite que las personas que hacen el trabajo (el equipo de software) busquen y elijan el conjunto apropiado de acciones y tareas para el trabajo. Se busca siempre entregar el software en forma oportuna y con calidad suficiente para satisfacer a quienes patrocinaron su creación y a aquellos que lo usarán.

**Conceptos de calidad. La pieza base de la ingeniería de Software**

Utilidad: que satisfaga las necesidades del usuario, ya que con frecuencia no desempeñan las funciones esperadas debido principalmente a una pobre comunicación con el cliente.

Confiabilidad: capacidad de un programa para desempeñar una función requerida bajo ciertas condiciones durante un tiempo específico. El grado de confiabilidad deseado en un producto depende del costo de fallas.

Claridad: los productos de software deben ser escritos con claridad y ser fáciles de entender tanto internamente como externamente, ya que las pruebas y actividades consumen un grado del presupuesto del proyecto.

Económico: el producto debe ser costeable en su desarrollo, mantenimiento y uso. Un software debe operar normalmente usando menos tiempo o recursos humanos o materiales de los que requerían antes de tenerlo.