1.3 ingeniería de software

* El software se ha incrustado profundamente en casi todos los aspectos de nuestras vidas y, como consecuencia, el número de personas que tienen interés en las características y funciones que brinda una aplicación específica8 ha crecido en forma notable
* Los requerimientos de la tecnología de la información que demandan los individuos, negocios y gobiernos se hacen más complejos con cada año que pasa.
* Los individuos, negocios y gobiernos dependen cada vez más del software para tomar decisiones estratégicas y tácticas, así como para sus operaciones y control cotidianos.
* A medida que aumenta el valor percibido de una aplicación específica se incrementa la probabilidad de que su base de usuarios y longevidad también crezcan.

La administración total de la calidad, Six Sigma y otras filosofías similares alimentan la cultura de mejora continua, y es esta cultura la que lleva en última instancia al desarrollo de enfoques cada vez más eficaces de la ingeniería de software.

El proceso define una estructura que debe establecerse para la obtención eficaz de tecnología de ingeniería de software

Los métodos de la ingeniería de software proporcionan la experiencia técnica para elaborar software. Incluyen un conjunto amplio de tareas, como comunicación, análisis de los requerimientos, modelación del diseño, construcción del programa, pruebas y apoyo

Herramientas de la ingeniería de software proporcionan un apoyo automatizado o semiautomatizado para el proceso y los métodos

1.4 EL proceso del software

Un proceso es un conjunto de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto del trabajo

Actividad busca lograr un objetivo amplio (por ejemplo, comunicación con los participantes

Una acción (diseño de la arquitectura) es un conjunto de tareas que producen un producto importante del trabajo

La estructura del proceso establece el fundamento para el proceso completo de la ingeniería de software por medio de la identificación de un número pequeño de actividades estructurales que sean aplicables a todos los proyectos de software, sin importar su tamaño o complejidad.

Comunicación. Antes de que comience cualquier trabajo técnico, tiene importancia crítica comunicarse y colaborar con el cliente

Planeación. Cualquier viaje complicado se simplifica si existe un mapa. Un proyecto de software es un viaje difícil, y la actividad de planeación crea un “mapa” que guía al equipo mientras viaja

Modelado. Ya sea usted diseñador de paisaje, constructor de puentes, ingeniero aeronáutico, carpintero o arquitecto, a diario trabaja con modelos. Crea un “bosquejo” del objeto por hacer a fin de entender el panorama general

Construcción. Esta actividad combina la generación de código (ya sea manual o automatizada) y las pruebas que se requieren para descubrir errores en éste. Despliegue. El software (como entidad completa o como un incremento parcialmente terminado) se entrega al consumidor que lo evalúa y que le da retroalimentación, misma que se basa en dicha evaluación.

Seguimiento y control del proyecto de software: permite que el equipo de software evalúe el progreso comparándolo con el plan del proyecto y tome cualquier acción necesaria para apegarse a la programación de actividades. Administración del riesgo: evalúa los riesgos que puedan afectar el resultado del proyecto o la calidad del producto.

Aseguramiento de la calidad del software: define y ejecuta las actividades requeridas para garantizar la calidad del software.

Revisiones técnicas: evalúa los productos del trabajo de la ingeniería de software a fin de descubrir y eliminar errores antes de que se propaguen a la siguiente actividad.

Medición: define y reúne mediciones del proceso, proyecto y producto para ayudar al equipo a entregar el software que satisfaga las necesidades de los participantes; puede usarse junto con todas las demás actividades estructurales y sombrilla.

Administración de la configuración del software: administra los efectos del cambio a lo largo del proceso

del software.

Administración de la reutilización: define criterios para volver a usar el producto del trabajo (incluso los componentes del software) y establece mecanismos para obtener componentes reutilizables.

Preparación y producción del producto del trabajo: agrupa las actividades requeridas para crear productos del trabajo, tales como modelos, documentos, registros, formatos y listas.

1.5 La práctica de la ingeniería de software

Entender el problema. En ocasiones es difícil de admitir, pero la mayor parte de nosotros adoptamos una actitud de orgullo desmedido cuando se nos presenta un problema.

Planear la solución. Ahora entiende el problema (o es lo que piensa) y no puede esperar para escribir el código. Antes de hacerlo, cálmese un poco y haga un pequeño diseño

Ejecutar el plan. El diseño que creó sirve como un mapa de carreteras para el sistema que quiere construir. Puede haber desviaciones inesperadas y es posible que descubra un camino mejor a medida que avanza, pero el “plan” le permitirá proceder sin que se pierda.

Examinar el resultado. No se puede estar seguro de que la solución sea perfecta, pero sí de que se ha diseñado un número suficiente de pruebas para descubrir tantos errores como sea posible.

1.5.2 Principios generales

El diccionario define la palabra principio como “una ley importante o suposición que subyace y se requiere en un sistema de pensamiento”. En este libro se analizarán principios en muchos niveles distintos de abstracción

Primer principio: La razón de que exista todo Un sistema de software existe por una razón: dar valor a sus usuarios. Todas las decisiones deben tomarse teniendo esto en mente.

Segundo principio: MSE (Mantenlo sencillo, estúpido…) El diseño de software no es un proceso caprichoso. Hay muchos factores por considerar en cualquier actividad de diseño

Tercer principio: Mantener la visión Una visión clara es esencial para el éxito de un proyecto de software. Sin ella, casi infaliblemente el proyecto terminará siendo un ser “

Cuarto principio: Otros consumirán lo que usted produce Rara vez se construye en el vacío un sistema de software con fortaleza industrial. En un modo u otro, alguien más lo usará, mantendrá, documentará o, de alguna forma, dependerá de su capacidad para entender el sistema.

Quinto principio: Ábrase al futuro Un sistema con larga vida útil tiene más valor. En los ambientes de cómputo actuales, donde las especificaciones cambian de un momento a otro y las plataformas de hardware

Sexto principio: Planee por anticipado la reutilización La reutilización ahorra tiempo y esfuerzo.15 Al desarrollar un sistema de software, lograr un alto nivel de reutilización es quizá la meta más difícil de lograr.

Séptimo principio: ¡Piense! Este último principio es tal vez el que más se pasa por alto. Pensar en todo con claridad antes de emprender la acción casi siempre produce mejores resultados. Cuando se piensa en algo es más probable que se haga bien