



CORPORACIÓN
UNIVERSITARIA
REMINGTON

ESCUELA DE CIENCIAS BÁSICAS E INGENIERÍA
PROGRAMA: Ingeniería de Sistemas
ASIGNATURA: Ingeniería del Software II

CORPORACIÓN UNIVERSITARIA REMINGTON
DIRECCIÓN PEDAGÓGICA

Este material es propiedad de la Corporación Universitaria Remington (CUR), para los estudiantes de la CUR en todo el país.

2011

CRÉDITOS



El módulo de estudio de la asignatura Ingeniería del Software II del programa Ingeniería de Sistemas es propiedad de la Corporación Universitaria Remington. Las imágenes fueron tomadas de diferentes fuentes que se relacionan en los derechos de autor y las citas en la bibliografía. El contenido del módulo está protegido por las leyes de derechos de autor que rigen al país.

Este material tiene fines educativos y no puede usarse con propósitos económicos o comerciales.

AUTOR

Yolfaris Naidith Fuertes Arroyo

Ingeniera de Sistemas.

5 años de Experiencia docente

yolfaris.fuertes@remington.edu.co

Rodrigo Alcides Patiño Arango

Ingeniero de Sistemas.

12 años de Experiencia docente

Rodrigo.patino@remington.edu.co

Nota: el autor certificó (de manera verbal o escrita) No haber incurrido en fraude científico, plagio o vicios de autoría; en caso contrario eximió de toda responsabilidad a la Corporación Universitaria Remington, y se declaró como el único responsable.

RESPONSABLES

ESCUELA DE CIENCIAS BÁSICAS E INGENIERÍA

Director Dr. Mauricio Sepúlveda

ingenieria.director@remington.edu.co

Director Pedagógico

Octavio Toro Chica

dirpedagogica.director@remington.edu.co

Coordinadora de Medios y Mediaciones

Angélica Ricaurte Avendaño

mediaciones.coordinador01@remington.edu.co

GRUPO DE APOYO

Personal de la Unidad de Medios y Mediaciones

EDICIÓN Y MONTAJE

Primera versión. Febrero de 2011.

Derechos Reservados



Esta obra es publicada bajo la licencia Creative Commons. Reconocimiento-No Comercial-Compartir Igual 2.5 Colombia.

TABLA DE CONTENIDO

1. MAPA DE LA ASIGNATURA

INGENIERÍA DEL SOFTWARE II

```
graph TD; A[INGENIERÍA DEL SOFTWARE II] --> B[PROPÓSITO GENERAL DEL MÓDULO]; B --> C[OBJETIVO GENERAL]; C --> D[OBJETIVOS ESPECÍFICOS];
```

PROPÓSITO GENERAL DEL MÓDULO

Busca desarrollar destrezas y habilidades en las personas que elijan como profesión esta rama de la ingeniería, explorando cada parte del proceso general de la estructura del conocimiento y sus diferentes factores de riesgo a través de la gestión de la configuración.

OBJETIVO GENERAL

Inducir en el estudiante la exploración de las habilidades y destrezas en cuanto al análisis, diseño, construcción e implementación de proyectos informáticos u otros proyectos, direccionados a la necesidad de la construcción de una aplicación que le permita al cliente la administración, supervisión, control de la información y así la realización de un buen diligenciamiento de la misma y la obtención de excelentes resultados.

OBJETIVOS ESPECÍFICOS

- ◆ Explorar los diferentes modelos prescriptivos de proceso, para el afianzamiento de los conocimientos, evitando equivocaciones en el momento del desarrollo de un producto.
- ◆ Plantear diversos problemas, que permitan la ubicación del estudiante, dentro de un ambiente experimental que le exija la aplicación de diversas estrategias y le permitan la exploración del conocimiento teórico – práctico para el desarrollo de diversas soluciones específicas.
- ◆ Inculcar en el estudiante sobre la responsabilidad que tiene cuando se enfrenta a la construcción de un determinado proyecto y a la aplicación idónea de todos los conocimientos adquiridos en la materia y la recopilación de otras, con el fin de que se garantice la calidad del desarrollo del software y la satisfacción del

Unidades

UNIDAD 1

Capacidad para comprender los diferentes enfoques acerca de la aplicabilidad de los modelos prescriptivos de procesos.

UNIDAD 2

Habilidad para identificar los diferentes enfoques estructurados de la gestión del conocimiento

UNIDAD 3

Capacidad para desarrollar destrezas al momento de desarrollar software enfocado en el Modelo de Desarrollo Rápido.

UNIDAD 1 EL PROCESO: OPERACIÓN Y GESTIÓN

OBJETIVO GENERAL

- Explorar los diferentes modelos prescriptivos de proceso, para el afianzamiento de los conocimientos, evitando equivocaciones en el momento del desarrollo de un producto.

OBJETIVOS ESPECÍFICOS

- ✓ Comprender la estructura de cada uno de los modelos prescriptivos de procesos para su excelente operatividad, analizando paso a paso la forma de utilizarlos para la aplicabilidad en la solución de problemas.

PRUEBA INICIAL

En los siguientes enunciados seleccione la respuesta correcta:

1. Un modelo prescriptivo de procesos definen:
 - a. Un conjunto de normas a tener en cuenta cuando se desarrolla software.
 - b. Un conjunto de operaciones de gestión.
 - c. Un conjunto distinto de actividades, acciones, tareas, fundamentos y producto de trabajo que se requieren al momento de desarrollar software de alta calidad.
 - d. Un conjunto similar de actividades, acciones, tareas, fundamentos y producto de trabajo que se requieren al momento de desarrollar software de alta calidad.
 - e. Todas las anteriores.
 - f. Ninguna de las anteriores.
2. Un modelo prescriptivo de procesos llena el marco de trabajo con:
 - a. Conjunto de tareas implícitas para las acciones de la ingeniería del software.
 - b. Conjunto de tareas explícitas para las acciones de la ingeniería del software.
 - c. Conjunto de tareas virtual para las acciones de la ingeniería del software.
 - d. Conjunto de tareas hipotéticas para las acciones de la ingeniería del software.
 - e. Todas las anteriores.
 - f. Ninguna de las anteriores.

3. Adoptar un modelo prescriptivo de procesos es importante porque proporciona:
- a. Planificación, supervisión y control a una actividad que si no se inspecciona puede volverse caótica.
 - b. Estabilidad, control y organización a una actividad que si no se controla puede volverse caótica.
 - c. Reconocimiento, comprobación y observación a una actividad que si no se controla puede volverse caótica.
 - d. Identificación, Supervisión y planificación a una actividad que si no se controla puede volverse caótica.
 - e. Todas las anteriores.
 - f. Ninguna de las anteriores.

Modelos Operativos Prescriptivos

- **Introducción a los modelos prescriptivos de procesos**

¹Estos modelos están fundamentados en ordenar el caos en el desarrollo de software. Los modelos han traído varias estructuras que son necesarias para las tareas de la ingeniería del software y muestran la ruta efectiva para los equipos de software. Sin lugar a dudas los trabajos analizados y realizados continúan “al borde del caos”.

El borde del caos se define como “un estado natural entre el orden y el caos, una relación estrecha entre la estructura y la sorpresa”. El borde del caos se puede visualizar como un estado inestable, estructurado en forma parcial... es inestable porque es atraído de manera constante hacia el caos o hacia el orden absoluto.

Se tiende a pensar que el orden es el estado ideal de la naturaleza. Esto podría ser un error. La investigación... apoya la teoría de que la operación lejos del equilibrio genera creatividad, procesos organizados por si mismo y retroalimentación creciente. El orden absoluto significa ausencia de la variabilidad, lo cual sería una ventaja en ambientes imprescindibles. El cambio ocurre cuando existe alguna estructura para que pueda organizarse, dicha estructura no debe ser tan rígida como para que evite el cambio. Por otro lado, demasiado caos puede imposibilitar la coordinación y la coherencia. La falta de estructura no siempre significa desorden.

Por otro lado, cada vez que se desarrolla un software siempre se busca que éste sea de buena calidad, para lo cual se hace uso de los modelos prescriptivos los cuales mediante la aplicación adecuada de actividades, tareas y acciones lo hacen posible. Estos modelos serán la base para el trabajo de la ingeniería del software aunque puedan existir dificultades o imperfección en su utilización.

Los siguientes son los tipos de modelos prescriptivos

- **Cascada Pura**

En un modelo en cascada, un proyecto progresa a través de una ordenada de pasos partiendo del concepto inicial del software hasta la prueba del sistema. El proyecto realiza una revisión al final de cada etapa para determinar si está

¹ Roger Pressman

preparado para pasar a la siguiente etapa, por ejemplo, desde el análisis de requerimientos hacia el diseño de la arquitectura. Cuando la revisión determina que el proyecto no está listo para pasar a la siguiente etapa, permanece en la etapa actual hasta que esté preparado.

El modelo en cascada está dirigido por documentos; es decir, los productos principales del trabajo que se pasan de etapa en etapa son documentos. El modelo en cascada pura se utiliza correctamente para ciclos de productos en los que se tienen una definición estable del producto, y también cuando se está trabajando con metodologías técnicas conocidas. En estos casos, el modelo en cascada ayuda a localizar errores en las primeras etapas del proyecto a un bajo coste. Proporciona los requerimientos que los desarrolladores anhelan. Si se está construyendo una versión de mantenimiento bien definida de un producto existente o migrando un producto existente a una nueva plataforma, un ciclo de vida en cascada puede ser una elección correcta para el desarrollo rápido.

El modelo de cascada pura ayuda a minimizar los gastos de la planificación porque permite realizarla sin problemas. No proporciona resultados tangibles en forma de software hasta el final del ciclo de vida, pero alguien familiarizado con el modelo. La documentación que genera proporciona indicaciones significativas del progreso a lo largo del ciclo de vida.

El modelo en cascada funciona bien con proyectos complejos que se entienden correctamente, debido a que se pueden obtener beneficios al enfrentarse a la complejidad de forma ordenada. Funciona correctamente cuando los requerimientos de costes y de planificación. El modelo evita una fuente común de errores importantes, eliminando los cambios que se pueden producir a medio camino.

El modelo en cascada funciona especialmente bien si se dispone de personal poco cualificado o inexperto, porque presenta el proyecto con una estructura que ayuda a minimizar el esfuerzo inútil.

Las desventajas del modelo en cascada se centran en la dificultad para especificar claramente los requerimientos al comienzo del proyecto, antes de que se realice ningún trabajo de diseño y antes de escribir ningún código.

Los desarrolladores se quejan de los usuarios que no saben lo que quieren, pero supongamos que se cambian los papeles. Imagínese intentando explicarle en detalle a un ingeniero de automoción cómo es un coche. Lo explica al ingeniero que necesita un motor, una carrocería, ventanillas, volante, pedal de acelerador, pedal de freno, el freno de emergencia, asientos y demás. Pero, ¿puede recordar

todo lo que el ingeniero de automoción necesitará conocer para construir su coche?

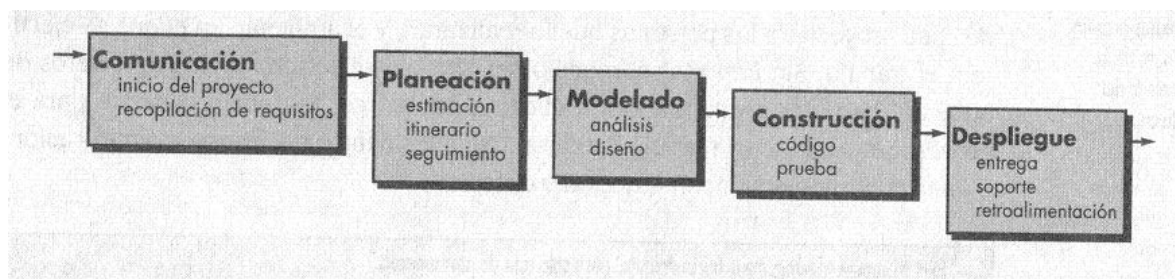
Suponga que se le olvida indicarle que necesita la luz de marcha atrás. El ingeniero se va y vuelve a los seis meses con un coche que no tiene luz de marcha atrás. Le comenta: "Oh, cielos, olvidé decirle que el coche necesita luz de marcha atrás."

- **Cascada con fases solapadas**

Peter DeGrace describe una de las modificaciones del modelo en cascada como el "modelo sashimi". El nombre procede del modelo de desarrollo hardware japonés (de Fuji-Xerox), y se refiere al estilo japonés de presentar el pescado crudo en lonchas solapándose entre sí (el hecho de que este modelo esté relacionado con el pescado no significa que esté relacionado con el modelo de ciclo de vida del salmón).

El modelo en cascada tradicional permite un solapamiento mínimo entre las etapas en la revisión del final de cada etapa. Este modelo sugiere un grado mayor de solapamiento; por ejemplo, sugiere que se debería tener bien hecho el diseño global y quizás a medio hacer el diseño detallado antes de considerar completo el análisis de requerimientos. Creo que esto es una aproximación razonable para muchos proyectos, en los que se aumentan las ideas importantes que se descubren cuando avanzan a través de sus ciclos de desarrollo y que funcionan mal con planes de desarrollo estrictamente secuenciales.

En el modelo de cascada pura, la documentación ideal es aquella documentación que un equipo completamente distinto entre dos etapas cualesquiera. La pregunta es: "¿Por qué?" si puede ofrecer una continuidad personal entre el concepto del software, análisis de requerimientos, diseño global, diseño detallado, codificación y depuración, no necesita tanta documentación. Se puede seguir un modelo de cascada modificada y reducir sustancialmente la documentación necesaria.



Gráfica #1

Ciclo de vida en cascada (secuencial): el esquema aplica para todos los tipos de cascada.

El modelo sashimi no está exento de problemas. Debido al solapamiento entre las etapas, los hitos son más ambiguos, y esto hace más difícil trazar el progreso correctamente. La realización de actividades en paralelo puede suponer una mala comunicación, suposiciones incorrectas e ineficacia. Si está trabajando en un proyecto pequeño y bien definido, algo cercano al modelo en cascada pura puede ser el mejor modelo disponible.

- **Cascada con subproyectos**

Otro problema que aparece en el modelo de cascada pura, desde el punto de vista del desarrollo rápido, es que se supone que se ha terminado completamente el diseño global antes de comenzar con el diseño detallado, y se supone que se ha terminado completamente el diseño detallado antes de comenzar la codificación y la depuración. Los sistemas presentan algunas áreas que incluyen sorpresas del diseño, pero presentan otras áreas que hemos implementado anteriormente muchas veces y no incluyen sorpresas. ¿Por qué retrasar la implementación de las áreas que son fáciles de diseñar solamente por que estamos esperando el diseño de un área difícil? Si la arquitectura ha dividido proyectos separados, cada una de los cuales puede proseguir su propio ritmo.

El riesgo principal de este enfoque es la presencia de interdependencias imprevistas. Se pueden tener parcialmente en cuenta eliminando dependencias durante el desarrollo de la arquitectura, o esperar hasta después del diseño detallado para dividir el proyecto en subproyectos.

- **Cascada con reducción de riesgo**

Otro de los inconvenientes del modelo en cascada es que requiere la definición completa de los requerimientos antes de comenzar el diseño de la arquitectura, algo que parece razonable excepto porque también requiere comprender totalmente los requerimientos antes de comenzar el diseño global. Modificación la cascada (de nuevo, solamente muy poco) puede colocar un espiral para reducir el riesgo en lo alto de la cascada para controlar el riesgo de los requerimientos. Puede desarrollar un prototipo de interfaz de usuario, utilizar cuadernos, tener entrevistas con los usuarios, crear cintas de video donde los usuarios interactúan con un sistema más antiguo, o utilizar otros métodos que considere apropiados para la identificación de los requerimientos.

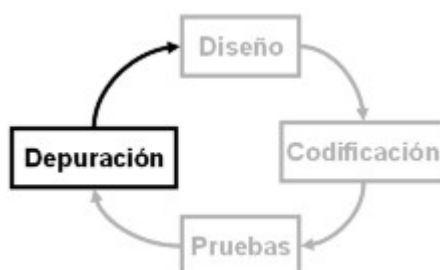
El análisis de requerimientos y el diseño de la arquitectura se muestran en gris, indicando que se deben controlar durante la etapa de reducción de riesgos mejor que durante la etapa de la cascada.

El preámbulo de la reducción de riesgos para el ciclo de vida en cascada no está limitado a los requerimientos. Podría utilizarlo para reducir el riesgo de la arquitectura o en cualquier otro riesgo del proyecto. Cuando el producto depende del desarrollo de un núcleo de alto riesgo para el sistema, se debería utilizar un ciclo de reducción de riesgos para desarrollar totalmente el núcleo de alto riesgo antes de acometer un proyecto a gran escala.

- **Modelo Codificar y Corregir**

El modelo de codificar y corregir (code-and-fix) es un modelo poco útil, pero sin embargo bastante común; por esto, me gustaría explicarlo. Si no ha seleccionado explícitamente otro modelo de ciclo de vida, por omisión estará utilizando probablemente el modelo de codificar y corregir. Si no ha realizado demasiada planificación del proyecto, indudablemente está utilizando el modelo codificar y corregir. Combinando con una planificación corta, el modelo codificar y corregir da paso al enfoque de codificar a destajo descrito anteriormente.

Con la utilización del modelo codificar y corregir, se indica mediante una idea a nivel general que lo que se va a desarrollar o se va a trabajar. En esta etapa se pueden combinar distintos modelos que coadyuven a la entrega de un producto que cumpla con las expectativas del cliente.



Gráfica #2
Modelo codificar y corregir

“El modelo de codificar y corregir tiene dos ventajas. En primer lugar, no conlleva ninguna gestión: no se pierde tiempo en la planificación, en la documentación, en el control de calidad, en el cumplimiento de los estándares, o en cualquier otra actividad que no sea la codificación pura. Como se pasa directamente a codificar, se pueden mostrar inmediatamente indicios de progreso. En segundo lugar, requiere poca experiencia: cualquier persona que haya escrito alguna vez un

programa de ordenador está familiarizada con el modelo de codificar y corregir. Cualquiera puede utilizarlo.

Para proyectos pequeños que se intentan liquidar poco después de ser construidos, este modelo puede ser útil (para programas pequeños de demostración de conceptos, para demostraciones de duración corta, o prototipos).

Este modelo resulta peligroso para otro tipo de proyectos que no sean pequeños. Puede que no suponga gestión alguna, pero tampoco ofrece medios de evaluación de la calidad o de identificación de riesgos. Si al llevar tres cuartas partes de la codificación descubre que el diseño es incorrecto, no hay otra solución que desechar el trabajo y comenzar de nuevo. Otros modelos le permitirán detectar un error tan fundamental mucho antes, cuando hubiera sido menos costoso solucionarlo. En definitiva, este modelo de ciclo de vida no tiene cabida en un proyecto de desarrollo rápido, excepto para los pequeños proyectos señalados”².

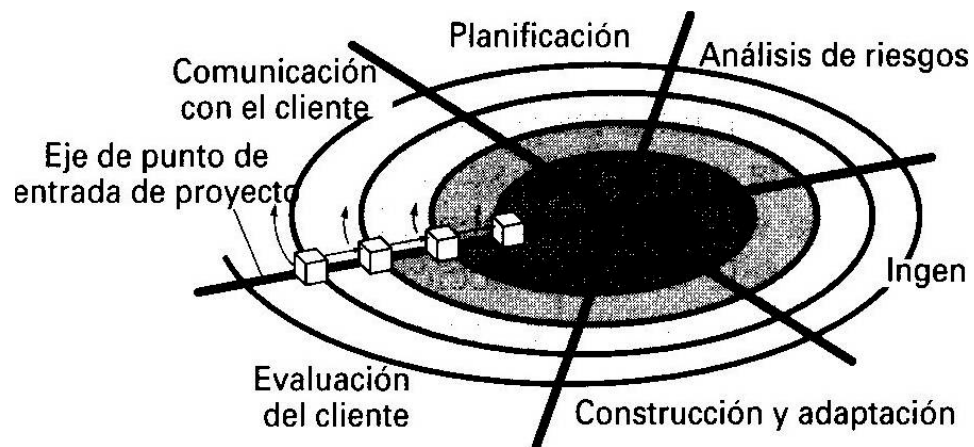
- **Modelo Espiral**

Es un modelo de proceso evolutivo que conjuga la naturaleza iterativa de la construcción de prototipos con los aspectos controlados y sistemáticos del modelo en cascada.

El modelo de desarrollo en espiral es un generador del modelo de proceso guiado por el riesgo que se emplea para conducir sistemas intensivos de ingeniería de software concurrente y con múltiples usuarios. Contiene dos características principales.

Dentro de este modelo se debe entender la manera cíclica e incremental como se va avanzando en cada etapa, buscando en todo momento disminuir el riesgo que se puede presentar en cualquier momento.

² mx.answers.yahoo.com › ... › [Software](#), [Qué metodología se puede usar para un proyecto de software](#) ... , consultado el 23-05-2011



Gráfica #3
Ciclo de vida del modelo espiral

Cuando se aplica el modelo en espiral, el software se desarrolla en una serie de entregas evolutivas. Durante las primeras iteraciones, la entrega tal vez sea un documento del modelo o prototipo.

Dentro del modelo en espiral, está involucrado las actividades del marco de trabajo, las cuales se aplican cada vez que se termina cada etapa dentro del modelo.

Se realizan actividades implicadas en cada circuito alrededor de la espiral que tiene sentido del movimiento de las manecillas del reloj y que se inicia desde el centro.

El modelo en espiral puede adaptarse y aplicarse a lo largo de la vida del software de computadora.

Si el concepto se desarrollara en un producto real, el proceso continúa en la siguiente fase de la espiral y comienza un “proyecto de desarrollo de un producto nuevo”. El nuevo producto evolucionará a través de un número de iteraciones alrededor de la espiral. La espiral permanece operativa hasta que el software se retira.

“El software evoluciona conforme avanza el proceso, el desarrollador y el cliente entienden y reaccionan de la mejor manera ante los riesgos en cada etapa evolutiva.

El modelo en espiral exige una consideración directa de los riesgos técnicos en todas las etapas del proyecto y se aplica en forma apropiada, debe reducir los riesgos antes de que se vuelvan problemáticos.”³

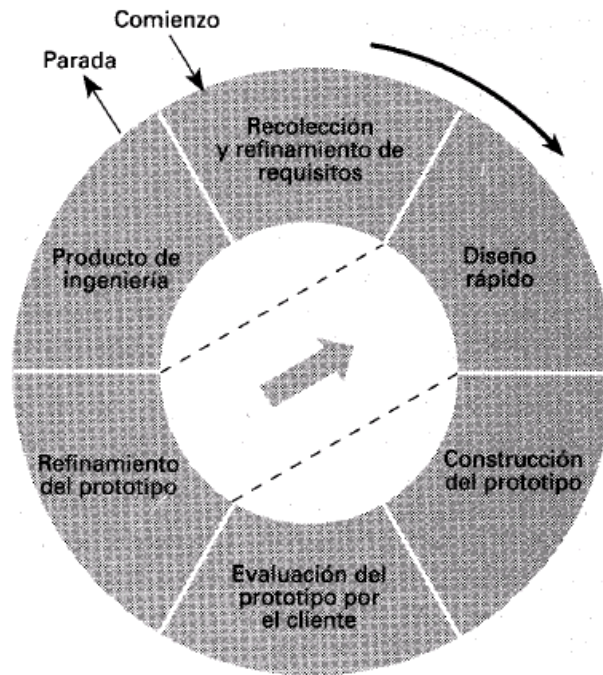
- **Modelo Prototipado**

Dada a la experiencia del cliente y al bagaje de su actividad, tiene la capacidad de definir diversos objetivos que orienten al desarrollador a elaborar una aplicación conforme a sus necesidad, pero la manera detallada como debe trabajar en cuanto a la entrada, proceso y salida no la define tan fácilmente dado a que muchos de sus conocimientos están enfocados a un aprendizaje empírico o rutinario que lo llevan a imaginarse que todo su proceso está bien.

A pesar de que la construcción de prototipos se puede utilizar como un modelo de procesos independiente, se emplea mas comúnmente como una técnica susceptible de implementarse dentro de un contexto de cualquiera de los modelos de procesos.

En lo concerniente a la construcción o desarrollo de prototipos, se debe dar un buen entendimiento, definiendo con claridad el objetivo y los elementos fundamentales que deben tener cada formulario, ya sea de ingreso o de salida los cuales ayudarán a tomar decisiones en los momentos más críticos de la empresa u organización

³ (S.A). (S.F): “Modelos de procesos” -
eisc.univalle.edu.co/cursos/web/material/.../ds1_modelos_de_proceso.pdf



Gráfica #4
Proceso de un modelo prototipado

Con el prototipo y su utilización se debían reconocer cuales son las necesidades puntuales del software a implementar. En muchos casos cuando se hacen prototipos especiales y funcionales, estos pueden ser utilizados parcialmente para la implementación de diversos diagramas que orienten a la solución en el menor tiempo posible.

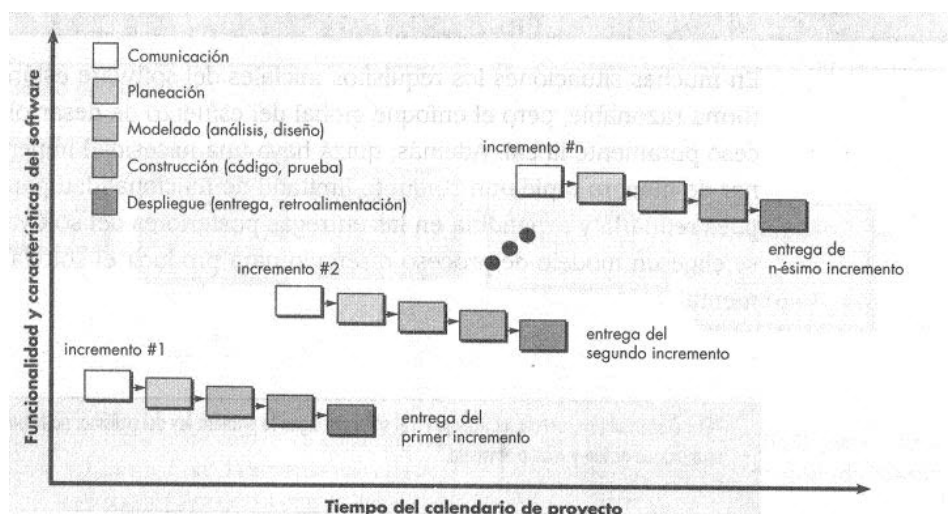
Es verdad que a los clientes y los desarrolladores les gusta el paradigma de construcción de prototipos. A los usuarios les gusta el sistema real y a los desarrolladores les gusta construir algo de inmediato. En la construcción de prototipos pueden existir problemas por lo siguiente:

- a. Cuando al cliente se le hacen las primeras muestras de la solución de su problema, desea la instalación total en el menor tiempo posible, olvidando o desconociendo que los desarrolladores requieren más tiempo para garantizar la calidad y procurar una mayor facilidad de mantenimiento.
- b. El cliente no entiende cuando se observa que el software se debe volver a hacer debido a que no tiene la calidad esperada para lo cual pide algunos avances para ser utilizado mientras que lo vuelven hacer aplicando los prototipos que se tienen.

- c. El desarrollador establece compromisos de implementación para lograr que el prototipo funcione con rapidez. Se puede utilizar sistema operativo y lenguaje de programación inadecuado solo porque está disponible y es conocido. La selección menos ideal ahora se ha convertido en una parte integral del sistema.

- **Entrega por etapas**

El modelo de entrega por etapas es otro modelo de ciclo de vida en el que el software se muestra al cliente en etapas refinadas sucesivamente. A diferencia del modelo de prototipo evolutivo, cuando se utiliza la entrega por etapas, se conoce exactamente qué es lo que se va a construir cuando se procede a construirlo. Lo que diferencia al modelo de entrega por etapas es que el software no se entrega al final del proyecto de una tacada. Se entrega por etapas sucesivas a lo largo del proyecto. (Este modelo se conoce también como “implementación incremental”).



Gráfica #5

Proceso de un modelo incremental (refinar secuencia)

La principal ventaja de la entrega por etapas es que le permite proporcionar una funcionalidad útil en las manos de su cliente antes de entregar el 100 por 100 del proyecto al final del mismo. Si planifica sus etapas cuidadosamente, puede que le sea posible entregar las prestaciones más importantes al principio, y sus clientes pueden comenzar a usar el software en este punto.

La entrega por etapas también proporciona signos tangibles de progreso en el proyecto, y se generan con enfoques menos incrementales. Estos signos de progreso pueden ser un valioso aliado para mantener la presión planificación a un nivel apropiado.

El principal inconveniente de la entrega por etapas es que no funcionaría sin una planificación adecuada tanto para niveles técnicos como para niveles en gestión. En un nivel de gestión, hay que asegurarse de que las etapas que se planifican son significativas para el cliente, y que el trabajo se distribuye entre el personal del proyecto de tal forma que pueden completar su trabajo a tiempo para cada etapa con fecha límite. En un nivel técnico, hay que asegurarse de que se han tenido en cuenta todas las dependencias técnicas entre los diferentes componentes de un producto. Un error común es retrasar el desarrollo de un componente hasta la etapa 4, solamente para descubrir que un componente planificado para la etapa 2 no puede trabajar sin él.

- **Entrega Evolutiva**

Esta se encuentra dentro del prototipado evolutivo y la entrega por etapas, en esta se crea una versión inicial, se le lleva al cliente para que observe y recomiende cambios o mejoras y con esto se va garantizando que el objetivo se cumpla de acuerdo a lo estipulado inicialmente.

El parecido entre la entrega evolutiva y el Prototipado evolutivo depende realmente de hasta qué punto se lleva a cabo una planificación para adaptarse a las solicitudes de los clientes.



Gráfica #6

Proceso de un modelo evolutivo incremental

En el prototipado evolutivo su fundamento está en la parte visible del sistema, se observa los faltantes y hacer la respectiva corrección y en la entrega evolutiva su fundamento está en el núcleo o parte principal del sistema, el cual corresponde a todas las diferentes funciones que en el mismo se aplican, donde el cliente determina si el trabajo va enfocado a la solución de su problema.

- **Diseño de planificación**

El modelo de ciclo de vida de diseño por planificación es similar al modelo de ciclo de vida de entrega por etapas, en el que se planifica desarrollar el producto en etapas sucesivas. La diferencia radica que no siempre se conoce al principio si se tendrá el producto para la última entrega. Se pueden tener cinco etapas planificadas. Pero sólo se llega a la tercera etapa debido a que se tiene una fecha límite inamovible.

Este modelo de ciclo de vida puede ser una estrategia válida para asegurar que se tiene un producto listo a entregar en una fecha determinada. Si se debe tener absolutamente el software funcionando a tiempo para una presentación comercial, o para final de año, o para cualquier otra fecha inamovible, esta estrategia garantiza que se tendrá algo. Esta estrategia es particularmente útil para las partes del producto que no se quieren realizar en el camino crítico. Por ejemplo, el sistema operativo Microsoft Windows incluye bastantes complementos, destacando Word Pad, Paint y la Red de corazones. Microsoft podría utilizar el diseño por planificación para evitar que estos complementos retrasaran Windows en general.

Al no llevar a cabo todas las etapas, se desperdicia tiempo y por ende la entrega no se cumplirá a cabalidad con lo que el cliente desea y espera.

La decisión para el utilizar el modelo de diseño por planificación se somete principalmente a la pregunta de cuánta confianza se tiene en la habilidad para la planificación. Si se tiene mucha confianza para que se puedan alcanzar los objetivos de la planificación, esta aproximación es ineficiente. Si se tiene una menor confianza, esta aproximación podría salvar su vida.

- **Diseño por herramientas**

Se pretende incluir una prestación dentro de un producto solo si las herramientas de software existente la soportan directamente. Si no está soportada se deja. Por herramienta se refiere a librerías de código y clases, generadores de código, lenguaje de desarrolladores rápido, y otras herramientas software que reducen de manera espectacular el tiempo de implementación.

Si hace un buen uso de herramientas para el desarrollo de software, no será posible incluir toda la funcionalidad ideal, pero si se hace una buena selección se pueden implementar un alto porcentaje de la funcionalidad esperada.

Este modelo se puede combinar haciendo uso de una espiral, con la entrega por etapas, la entrega evolutiva y el diseño por planificación.

El diseño por herramientas tiene pocos inconvenientes importantes. Se pierde mucho control sobre el producto, puede que no se posible lleva a cabo la

implementación de todas las prestaciones exactamente de la forma que queremos.

Si desarrollan programas pequeños, estos no tendrán a corto, mediano o largo plazo mucho inconveniente, pero si se piensa en aplicaciones grandes estos se pueden convertir en un posible eslabón débil de la cadena de producción.

Ejercicios tema 1

1. ¿Cuáles son las señales de que el desarrollo de un modelo prescriptivo no se está implementando correctamente?
2. Describir tres situaciones de la vida real en las cuales el cliente y el usuario final son el mismo. Describir tres situaciones en las cuales son diferentes.
3. El desempeño es una consideración importante durante la planificación de un software. Comentar cómo se puede interpretar de manera diferente el desempeño, dependiendo del área de aplicación del software.
4. ¿Cuáles adaptaciones se requieren en el proceso si el prototipo evolucionara hacia un sistema o producto que pueda entregarse?
5. ¿Es posible combinar modelos de procesos? (Justifique su respuesta a través de un ejemplo enfocado al campo empresarial)
6. Para usted, cual es el propósito de la evaluación del proceso de desarrollo de un software. (Explique)
7. Explique con sus propias palabras cual ha sido el impacto del “caos” en la ingeniería del software
8. Como todos los modelos de proceso el DRA tiene inconvenientes o desventajas, mencione alguno de los inconvenientes o desventajas que puede tener este modelo en la construcción de proyectos grandes

Prueba Final (Unidad I)

1. Este modelo también es llamado ciclo de vida clásico:
 - a. Incremental.
 - b. Entrega evolutiva.
 - c. Espiral.
 - d. Cascada.

- e. Desarrollo ágil.
 - f. Todas las anteriores.
 - g. Ninguna de las anteriores.
2. El siguiente modelo es el paradigma más antiguo para la ingeniería del software.
- a. Cascada.
 - b. Desarrollo ágil.
 - c. Espiral.
 - d. Incremental.
 - e. Entrega evolutiva.
 - f. Todas las anteriores.
 - g. Ninguna de las anteriores.
3. El siguiente modelo produce una versión completa en forma incremental con cada iteración.
- a. Entrega evolutiva.
 - b. Incremental.
 - c. Espiral.
 - d. Cascada.
 - e. Desarrollo ágil.
 - f. Todas las anteriores.
 - g. Ninguna de las anteriores.
4. El siguiente modelo se puede adoptar y aplicar a través del ciclo de vida completo de una aplicación, desde el desarrollo del concepto hasta el mantenimiento.
- a. Incremental.
 - b. Espiral.
 - c. Evolutivo.
 - d. Desarrollo ágil.
 - e. Cascada.
 - f. Todas las anteriores.
 - g. Ninguna de las anteriores.

UNIDAD 2 MODELOS DE GESTIÓN

OBJETIVO GENERAL

- Plantear diversos problemas, que permitan la ubicación del estudiante, dentro de un ambiente experimental que le exija la aplicación de diversas estrategias y le permitan la exploración del conocimiento teórico – práctico para el desarrollo de diversas soluciones específicas.

OBJETIVOS ESPECÍFICOS

- ✓ Comprender el concepto de la gestión de proyectos, para una mayor aplicabilidad al momento de desarrollar software de alta calidad.
- ✓ Analizar el nivel de complejidad e importancia de la aplicación de las métricas de proceso y proyecto en la medición, calidad, integración y organización del desarrollo de software.
- ✓ Entender con claridad el concepto sobre estimación de proyectos de software para que el margen de error sea menor.
- ✓ Comprender la manera como se establece un cronograma de actividades analizar la importancia de la planificación, supervisión y control al momento de la realización de un proyecto.
- ✓ Identificar los riesgos que se dan con la realización de un proyecto, teniendo presente los riesgos reactivos y proactivos para la elaboración de un plan de contingencia para contrarrestarlos.
- ✓ Reconocer que la calidad del software debe presentarse desde el levantamiento y durante todas las etapas sobre las cuales se trabajará, para la entrega de un producto que satisfaga las necesidades del cliente.
- ✓ Identificar la conceptualización, que guiará para la aplicación correcta, en la gestión de cambio dentro de un proyecto, teniendo la seguridad en las modificaciones pertinentes.

PRUEBA INICIAL

1. La gestión de proyectos de software involucra:

- a. “La planificación, supervisión y control del personal, los procesos y los eventos que ocurren mientras el software evoluciona desde un concepto preliminar hasta una implementación operativa.
 - b. La planificación, dirección y supervisión del personal, los procesos y los eventos que ocurren mientras el software evoluciona desde un concepto preliminar hasta una implementación operativa”⁴.
 - c. La planificación, supervisión y evaluación del personal, los procesos y los eventos que ocurren mientras el software evoluciona desde un concepto preliminar hasta una implementación operativa.
 - d. Todas las anteriores.
 - e. Ninguna de las anteriores.
2. El proceso del software y las métricas de proyecto son medidas:
- a. Cualitativas que permiten a los ingenieros de software obtener una visión de la eficacia del proceso de software y los proyectos que llevan a cabo utilizando el proceso como marco de trabajo.
 - b. Cuantitativas que permiten a los ingenieros de software obtener una visión de la eficacia del proceso de software y los proyectos que llevan a cabo utilizando el proceso como marco de trabajo.
 - c. Cualitativas y cuantitativas que permiten a los ingenieros de software obtener una visión de la eficacia del proceso de software y los proyectos que llevan a cabo utilizando el proceso como marco de trabajo.
 - d. Todas las anteriores.
 - e. Ninguna de las anteriores.
3. La estimación de proyectos de software determina:
- a. Cuánto dinero, esfuerzo, capital intelectual, tomará construir un sistema o producto específico basado en software.
 - b. Cuánto dinero, esfuerzo, recursos y tiempo tomará construir un sistema o producto específico basado en software.
 - c. Cuánto dinero, recursos, conocimiento tácito y explícito tomará construir un sistema o producto específico basado en software.
 - d. Todas las anteriores.
 - e. Ninguna de las anteriores.

⁴ www.utpl.edu.ec/.../Plan_de_contenidos:_Ingeniería_Software,_Plan_de_contenidos:_Ingeniería_Software_-_Computacion, consultado el 21-5-2011

Gestión de proyectos de software

- **Gestión de proyectos**

Gestionar un proyecto significa (planificar, supervisar y controlar el personal, los procesos y eventos mientras se construye software, garantizando por supuesto la calidad)

Todas las personas de una u otra forma utilizan la gestión: en su vida laboral, personal, familiar, educativa, etcétera. Ya que siempre terminan planificando, supervisando y controlando cada aspecto a realizar en el diario vivir de las actividades antes mencionadas.

La gestión de proyectos (al momento de construir software) se fundamenta en cuatro P: personal, producto, proceso y proyecto.

- **Personal:** Debe ser bien seleccionado (preparado y altamente calificado), motivado e incentivado constantemente por los gestores o parte ejecutiva a cargo del desarrollo del proyecto, ya que de esto depende la construcción del producto. Se debe tener siempre presente que la buena comunicación con el personal es la prioridad principal para un buen entendimiento de tareas o actividades relacionadas. Un gestor que no motiva e incentiva con palabras o reconocimientos a su personal se arriesga a que este se desmotive y trabaje tan solo por cumplir con su responsabilidad, no dando todo de sí mismo, ni esmerándose por obtener lo mejor de la tarea que realiza. El personal es el elemento fundamental y central de todo proyecto.
- **Producto:** Antes de empezar a desarrollar o construir el proyecto, se recomienda tener claro y presente que es lo que se va a construir, cómo se construirá, donde, cuando, quienes participaran, etcétera. Para lo cual el desarrollador del software y el cliente se deben poner en contacto preferiblemente cara a cara para acordar detalles. De estas reuniones deben quedar claro tanto el objetivo general del proyecto como los específicos y además todo lo relacionado con el producto a construir (requisitos exigidos por el cliente)
- **El proceso:** Se recomienda tener presente el marco de trabajo o actividades sombrilla para poder establecer considerablemente toda la planeación del producto que se desarrollará (software), y donde el equipo de trabajo entra a jugar un papel fundamental ya que es el encargado de ejecutar estas tareas, por lo tanto antes del inicio del proyecto deben tener todo bien claro y verificado.
- **El proyecto:** Se debe realizar planificada mente, con una supervisión constante y un control riguroso que asegure la excelente gestión y el

éxito de todo proyecto. De la buena planificación depende el diseño que se realice para la construcción del producto.



Gráfica #7

Proceso de la gestión de un proyecto de software

En **conclusión** se debe tener presente que el personal debe estar bien organizado, distribuido en equipos que trabajen coordinados en donde prevalezca la comunicación, motivados para realizar un buen trabajo al momento de desarrollar software de alta calidad. Para el desarrollo del producto es fundamental la comunicación constante con el cliente. Se debe tener presente que el proceso se debe adaptar al personal y al problema teniendo en cuenta la organización de las actividades o tareas utilizando el marco de trabajo, y finalmente, el proyecto debe estar bien planeado, organizado y estructurado para que le permita al equipo de software realizar un exitoso trabajo.

- **El principio W⁵HH**

⁵Proceso del software, Bany Boehm [BOE96] afirma: «... se necesita un principio de organización que haga una simplificación con el fin de proporcionar planes [de proyectos] sencillos para proyectos pequeños». Boehm sugiere un enfoque que trate los objetivos, hitos y planificación, Responsabilidades, enfoque técnico **y** de gestión, y **los** recursos requeridos del proyecto. Bohem lo llama el principio «WWWWHH», después de una serie de preguntas (7 cuestiones) que conducen ala definición de las características clave del proyecto **y** el plan del proyecto resultante:

¿Por qué se desarrolla el sistema?

La respuesta a esta pregunta permite a todas las partes evaluar la validez de las razones del negocio para el trabajo del **software**. Dicho de otra forma, justifica el propósito del negocio el gasto en personal, tiempo, y dinero?

¿Qué se realizará y cuándo?

La respuesta a estas preguntas ayuda al equipo a establecer la planificación del proyecto identificando las tareas clave del proyecto y los hitos requeridos por el cliente.

¿Qué preguntas necesitan ser respondidas para desarrollar un Plan de Proyecto?

¿Quién es el responsable de una función?

Antes en este capítulo, señalamos que el papel y la responsabilidad de cada miembro del equipo de software deben estar definidos. La respuesta a la pregunta ayuda a cumplir esto.

¿Dónde están situados organizacionalmente?

No todos los roles y responsabilidades residen en el equipo de software. El cliente, los usuarios, y otros directivos también tienen responsabilidades en el plan de Proyecto de Software

¿Cómo estará realizado el trabajo desde el punto de vista técnico y de gestión?

Una vez establecido el ámbito del producto, se debe definir una estrategia técnica y de gestión para el proyecto.

⁵ Roger S. Pressman

¿Qué cantidad de cada recurso se necesita?

La respuesta a esta pregunta se deriva de las estimaciones realizadas (Capítulo 5) basadas en respuestas a las preguntas anteriores.

El principio W5HH de Boehm es aplicable sin tener en cuenta el tamaño o la complejidad del proyecto de software. Las preguntas señaladas proporcionan un perfil de planificación al gestor del proyecto y al equipo de software.

- **Análisis personal**

Al momento de desarrollar software de alta calidad se debe tener presente la realización de una excelente gestión de proyectos, ya que de esto depende la buena administración, supervisión y control de cada uno de los procesos que se realizarán dentro del proyecto de software. Para realizar una buena aplicación de software los ingenieros desarrolladores deben centrarse primero que todo en la etapa de comunicación, porque de ella depende la obtención de requisitos que le permitirán al equipo ejecutor tener claridad de lo que el cliente desea obtener con la aplicación. Esta etapa o actividad del marco de trabajo es la clave principal para un equipo de software. Se debe tener muy claro que para realizar un proyecto se tiene que tener claridad absoluta de lo que se efectuará, de esta forma es mucho más garantizada la gestión que se aplicará a cada uno de los procesos a desarrollar dentro de la planificación de las actividades, acciones o tareas que recomienda el marco de trabajo.

Ejercicios del tema 1

1. Porque cree usted que los proyectos de software necesitan ser gestionados. (Justifique su respuesta con un ejemplo)
2. ¿Qué se busca cuando se elige a alguien para dirigir un proyecto de software?
3. De las cuatro P, cuál es el elemento central de un proyecto de software y porque? (Justifique su respuesta)
4. Porque fracasan los equipos de trabajo que buscan consolidar una meta XY.
5. Qué piensa usted:
 - a) El problema debe adaptarse al proceso y al personal

- b) El proceso debe adaptarse al personal y al problema
 - c) El problema y el personal deben adaptarse al proceso
- Justifique su respuesta con un ejemplo direccionado al campo empresarial.
6. Realice un ejemplo vivencial enfocado al campo organizacional o ente económico, en donde involucre directamente la gestión de proyectos, aplique un marco de trabajo bien organizado para detallar toda la gestión.

Métricas para proyectos de software

- **Métricas de proceso y proyecto**

Son medidas cuantificables que buscan la eficacia del proyecto. La medición permite evaluar el proceso y el proyecto objetivamente. La eficacia del proyecto se busca a través de las mejoras en la calidad y productividad. Estas medidas deben ser analizadas y evaluadas adecuadamente. Las métricas de proceso tienen impacto a largo plazo, ya que el proceso de mediación requiere tiempo, paciencia y concentración para evitar posibles errores en la toma de decisiones. Las métricas del proyecto contribuyen al desarrollo de métricas de proceso.

Gráfico #8

Determinantes para la calidad del software y la eficacia

- **Factores que influyen en la calidad del software**

- ✓ Habilidad
 - ✓ Motivación
- } Del personal

Se hace referencia al personal que trabaja en el desarrollo del software y que está involucrado directamente en todo el proceso del proyecto.

Tengamos en cuenta que la motivación con la que trabaje el equipo desarrollador depende en gran parte del gestor del proyecto (Ingenieros de software) (Líder).

Gráfica #9

Jerarquía organizacional del equipo ejecutor

Las métricas o medidas cuantificables

Requieren:

- ✓ Tiempo
- ✓ Esfuerzo

Estos dos elementos permiten al equipo desarrollador:

- a. Planificar
- b. Realizar un seguimiento
- c. Realizar un control

A todo el proceso que se está realizando con el fin de evaluar la calidad con la que se está construyendo el producto.

Dentro de la jerarquía organizacional se debe tener en cuenta lo siguiente:

- El gestor ejecutivo define los aspectos del negocio.
- El gestor técnico: planifica, motiva y controla a los profesionales
- Los profesionales son los directamente relacionados con la construcción del producto.
- El cliente es quien en cada avance presentado evalúa el producto. El cliente está en constante comunicación con los profesionales y con los gestores del proyecto.

El usuario final es la persona que manipulará el producto. Puede ser el mismo cliente.

UN GESTOR DE PROYECTOS DEBE TENER LAS SIGUIENTES CARACTERISTICAS

- a. Resolución de problemas.
- b. Dotes de gestor
 - Dirigir
 - Encabezar
- c. Tener presente el Incentivo
- d. Fomentar el trabajo en equipo
- e. Claridad en la toma de decisiones.

• **Finalidad de las métricas de proceso**

Las métricas de proceso se emplean con el fin de minimizar el tiempo de desarrollo haciendo los ajustes necesarios para evitar demoras y reducir los problemas y riesgos potenciales.

También se emplean para valorar la calidad del producto sobre una base actual. Conforme la calidad mejora los defectos, estos minimizan, y mientras esto sucede también se reduce la

Cantidad de reelaboración requerida durante el proyecto. Esto conduce a una reducción en el costo global del proyecto.

- ✓ Las métricas del proceso de software se utilizan con propósito estratégico.
- ✓ Las métricas de proyecto son tácticas.

La primera aplicación de las métricas del proyecto en la mayoría de los proyectos de software ocurre durante la estimación (esfuerzo y tiempo) Todas las aplicaciones de métricas tienen un significado conforme el software evoluciona desde los requisitos hasta el diseño

Se recopilan métricas técnicas para valorar la calidad del diseño y mejorar los indicadores que influirán en el enfoque que se adopte para la generación y prueba del código.

- **Medición del software**

Se clasifica en 2 categorías:

1)

- ✓ Medidas directas del proceso de software: (costo y esfuerzo aplicados)

Medidas directas del producto: (líneas de códigos producidas, rapidez de ejecución y defectos reportados en el desarrollo del producto.

2)

- ✓ Medidas indirectas del producto (incluye funcionalidad, calidad, complejidad, eficiencia, confiabilidad, facilidad de mantenimiento, etcétera.

Las métricas del proyecto se consolidan con el fin de crear métricas del proceso que sean para la organización de software como un todo.

- **Medición de la calidad**

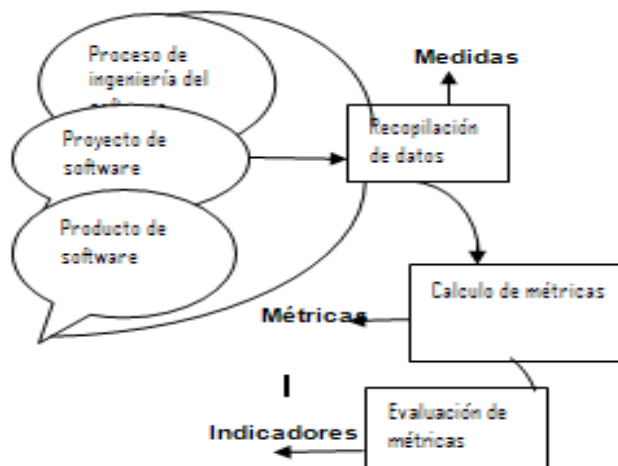
- ✓ Corrección
- ✓ Facilidad de mantenimiento
- ✓ La integridad y facilidad de uso

Estas medidas ofrecen indicadores útiles para el equipo del proyecto. La meta primordial de la ingeniería del software es producir un sistema, aplicación o producto de alta calidad dentro de un marco temporal que satisfaga necesidades en el mercado.

Las métricas del proceso permiten que una organización adopte una visión estratégica al proporcionar información detallada de la eficacia de un proceso de software.

Gráfica #10

Proceso de recopilación de métricas de software



- **Métricas del producto para software**

Por su naturaleza la ingeniería es una disciplina cuantitativa. Los ingenieros usan números como apoyo para el diseño y la evaluación del producto que construirán. Las métricas del producto ayudan a conocer mejor el diseño y la construcción del software que elaboran. Estas métricas se concentran en atributos específicos de los productos de trabajo de la ingeniería del software y se recopilan a medida que se realizan las tareas técnicas (análisis, diseño, codificación y pruebas). El objetivo de la métrica del producto es servir de apoyo para construir software de mayor calidad.

- **Calidad del producto**

Esta se refiere a la satisfacción adecuada de las necesidades en el manejo y control de la información, con estándares que ayuden a reconocer la fiabilidad y manejo adecuado de la solución.

- a. Los requisitos del software son la base de las medidas de calidad.
- b. Los estándares especificados definen un conjunto de criterios de desarrollo que guían la ingeniería del software.
- c. A medida que se soslaya un conjunto de requisitos implícitos (por ejemplo el deseo de alcanzar la facilidad de uso). Si el software cumple con los requisitos explícitos pero no con los implícitos, la calidad del software estará en duda

- **Factores que afectan la calidad del software**

Se dividen en dos grandes grupos:

1. Los que se miden directamente (por ejemplo, defectos descubiertos durante la prueba).
2. Los que sólo se miden indirectamente (por ejemplo, facilidad de uso o de mantenimiento).

Es importante indicar que la calidad se extiende a las características técnicas de los modelos de análisis y diseño, así como a la realización del código fuente de estos.

- **Métricas para el modelo de análisis**

Son varios los aspectos que sirven a este modelado tales como:

Funcionalidad entregada: proporciona una medida indirecta de la funcionalidad que se empaqueta con el software.

Tamaño del sistema: mide el tamaño del sistema de acuerdo a la información existente.

Calidad de la especificación: Proporciona una indicación de la especificidad o hasta donde se ha llevado los requisitos esperados.

- **Métricas para el modelo de diseño**

Estas métricas cuantifican los atributos del diseño de manera tal que le permiten al ingeniero de software evaluar la calidad del diseño. La métrica incluye:

- **Métricas arquitectónicas:** proporcionan un indicio de la calidad del diseño arquitectónico.
- **Métricas a nivel de componentes:** miden la complejidad de los componentes del software y otras características que impactan la calidad.
- **Métricas de diseño de interfaz:** se concentra en la facilidad de uso.
- **Métricas especializadas en diseño orientado a objetos:** miden características de clases, además de las correspondientes a comunicación y colaboración.

- **Métricas para el código fuente**

Miden el código fuente y se usan para evaluar su complejidad, además de la facilidad con que se mantiene y prueba.

- **Métricas de Halstead:** proporcionan medidas únicas de un programa de cómputo.
- **Métricas de complejidad:** miden la complejidad lógica del código fuente.
- **Métricas de longitud:** proporcionan un indicio del tamaño del software.

- **Métricas para pruebas**

Ayudan a diseñar casos de prueba efectivos y a evaluar la eficiencia de las pruebas.

- **Métricas de cobertura de instrucciones y ramas:** lleva al diseño de casos de prueba que proporcionan cobertura del programa.
- **Métricas relacionadas con los defectos:** se concentran en encontrar defectos y no en las propias pruebas.
- **Efectividad de la prueba:** proporcionan un indicio en tiempo real de la efectividad de las pruebas aplicadas.

- **Métricas en el proceso:** métricas relacionadas con el proceso que se determina a medida que se aplican las pruebas.

En muchos casos las métricas de un modelo pueden aplicarse en actividades posteriores de la ingeniería del software. Por ejemplo, las métricas de diseño se utilizan para estimar el esfuerzo requerido para generar código fuente. Además, las métricas de diseño se aprovechan para planear pruebas y el diseño de casos de prueba.

- **Análisis personal**

Las métricas del software proporcionan una manera cuantitativa de evaluar la calidad de los atributos internos de un producto antes de ser construido, proporcionan los conocimientos necesarios para crear modelos efectivos de análisis y diseño, un código sólido y pruebas exhaustivas. Para que resulte útil en la realidad una métrica de software debe ser simple y calculable, persuasiva, consistente y objetiva. Debe ser independiente del lenguaje de programación y proporcionar retroalimentación efectiva al ingeniero del software.

Se debe tener presente que existen medidas cualitativas y cuantitativas dentro de las métricas del software, las cuales permiten garantizar la calidad y productividad del producto. Estas medidas son una buena base para el desarrollo del análisis y diseño, ya que permiten corregir los errores antes de que se conviertan en defectos y por supuesto antes de ser entregada la aplicación al cliente, proporcionando confiabilidad, fiabilidad, integridad y eficiencia tanto al desarrollo como al proyecto en general.

Téngase presente que Los requisitos del software son la base de las medidas de calidad, la cual es garantizada por el cumplimiento de los requisitos de funcionalidad y desempeño explícitamente establecidos, de los estándares de desarrollo explícitamente documentados y de las características implícitas que se esperan de todo software desarrollado profesionalmente.

Ejercicio del tema 2

1. Cree usted que el ámbito de las actividades de gestión varía entre las personas involucradas en un proyecto de software. Si___, No___
(justifique su respuesta)
2. Las decisiones que toman los gestores ejecutivos pueden tener un impacto en el buen desarrollo de un equipo de software. Describa un

ejemplo enfocado al campo organizacional en donde usted demuestre que esto es cierto.

3. Qué perfil cree usted que debe tener:
 - Un líder de equipo
 - Un ingeniero de software
 - Un gestor de proyecto de software.
 - Los participantes de un equipo de software.Defina cada uno de los perfiles.
4. Describir con palabras propias la diferencia entre métricas del proceso y del proyecto.
5. Qué es una medida indirecta y porqué tales medidas son comunes en el trabajo de métricas del software?
- 6.Cuál es el objetivo o fin de las mediciones (métricas).
7. Que tiene que ver la planificación, seguimiento y control de un proyecto de software con las métricas de proceso y proyecto.

Estimación de Proyectos de software

- **Estimación de proyectos**

Comencemos por relacionar las actividades que abarca una planificación de proyectos de software:

- a. Estimación
- b. Programa de trabajo
- c. Análisis de riesgos
- d. Planificación de la gestión de la calidad.
- e. Planificación de la gestión del cambio.

La estimación permite determinar cuanto dinero, esfuerzo, recursos y tiempo tomará construir un sistema o producto específico basado en software.

Siempre que se realizan estimaciones se atisba al futuro y se acepta automáticamente algún grado de incertidumbre. Aunque la estimación es tanto un arte como una ciencia, esta importante actividad no necesita realizarse en forma improvisada. Existen técnicas útiles para la estimación de tiempo y esfuerzo.



Gráfica #11

Planificación de un proyectos de software

Las métricas del proceso y el proyecto ofrecen la perspectiva histórica y la energía para la generación de estimaciones cuantitativas. La experiencia (de toda la gente

involucrada) puede auxiliar enormemente conforme se desarrollan y revisan las estimaciones. Puesto que la estimación coloca los cimientos para las demás actividades de planificación del proyecto, y ésta proporciona la ruta para la ingeniería del software exitosa. La estimación de recursos, costo y programa de trabajo para una tarea de ingeniería del software requiere experiencia, acceso a buena información histórica (métricas) y el valor para comprometerse con predicciones cuantitativas cuando la información cualitativa es todo lo que existe. La estimación implica riesgo inherente, y este conduce a la incertidumbre.

El riesgo de la estimación se mide por el grado de incertidumbre en las estimaciones cuantitativas establecidas para recursos, costo y programa de trabajo. Si el ámbito del proyecto se comprende en forma deficiente o los requisitos del proyecto están sujetos a eventuales cambios, la incertidumbre y el riesgo de la estimación se incrementan peligrosamente. El planificador y, en forma más importante, el cliente deben reconocer que la variabilidad en los requisitos del software significan inestabilidad en costo y programa de trabajo.

La planificación requiere que los gestores técnicos y los miembros del equipo de software establezcan un compromiso inicial, aun cuando sea probable que este “compromiso” pruebe estar equivocado. Sin embargo, un gestor de proyectos no debe obsesionarse con las estimaciones.

El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita al gestor estimar razonablemente recursos, costo y programa de trabajo

Además, las estimaciones deben intentar definir los escenarios de mejor y peor caso de modo que los resultados del proyecto se puedan acotar.

Mientras más conozca, mejor estimará. En consecuencia, actualice sus estimaciones conforme avance el proyecto.

El planificador de proyectos debe estimar tres factores antes de que un proyecto comience: **cuánto tiempo tomará, cuánto esfuerzo requerirá y cuánto personal estará involucrado**. Además, el planificador debe predecir los recursos (hardware y Software) que se requerirán y el riesgo involucrado. La descripción del ámbito ayuda al planificador a desarrollar estimaciones empleando una o más técnicas que se clasifican en dos amplias categorías: descomposición y modelado empírico.

Descomposición: requieren un bosquejo de las principales funciones del software, seguido por estimaciones.

Técnicas empíricas: usan expresiones para esfuerzo y tiempo obtenidas de la ingeniería del software.

Ejercicio del tema 3

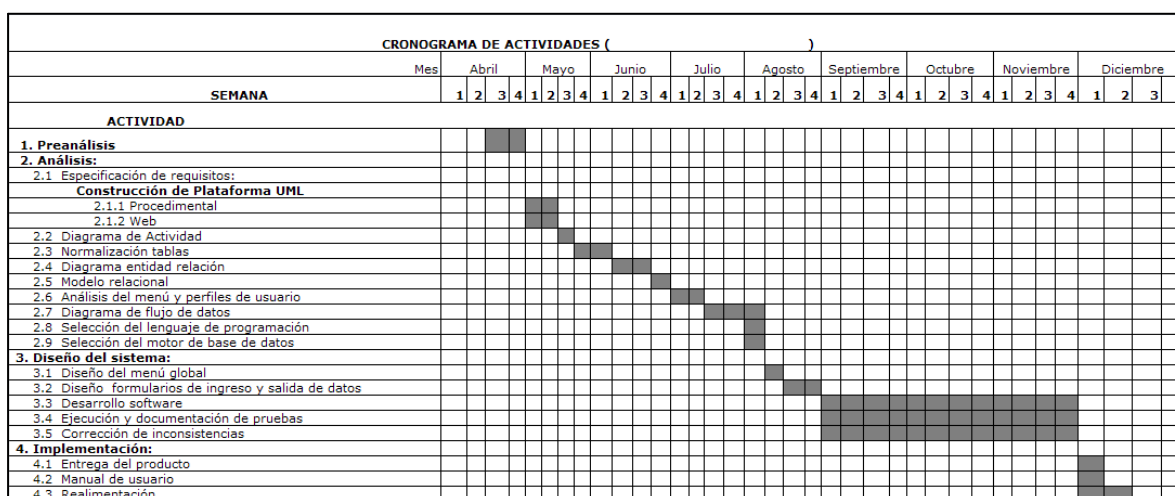
1. Que entiende usted por estimación de proyectos de software
2. Suponga que es el gestor de proyecto de una compañía que construye software para robots caseros. Se le ha contratado para construir el software destinado a un robot que corta el pasto. Describa por escrito el ámbito del software. Asegúrese de que la descripción del ámbito esté acotada. Si no está familiarizado con los robots, investigue un poco antes de comenzar a escribir. Además, establezca sus suposiciones acerca del hardware que se requerirá.
Alternativa: sustituya el robot que corta el pasto por otro problema Robótico que le interese.
3. Como se mide el riesgo de la estimación
4. La complejidad del proyecto de software influye en la precisión de la estimación (medición). Desarrollar una lista de características de software que afecten la complejidad de un proyecto. Establecer prioridades en la lista.
5. Porque es importante dentro de la estimación que se establezcan un compromiso inicial
6. Como explica usted estimar tres factores antes de iniciar un prouyecto

Calendarización de proyectos

- **Definición de calendarización**

Usted seleccionó un modelo de proceso adecuado, identificó las tareas de ingeniería del software que es preciso realizar, estimó la cantidad de trabajo y el número de personas, conoce la fecha límite, incluso consideró los riesgos. Ahora tiene que crear una red de tareas de ingeniería del software que le permitirán tener el trabajo listo a tiempo. Una vez creada la red, tiene que asignar responsabilidades a cada tarea, asegurarse de que se realice y adaptar la red conforme los riesgos se vuelvan realidad. A esto se le llama CALENDARIZACIO y el seguimiento del proyecto de software.

Las tareas de ingeniería del software que dicta el modelo de proceso de software se refinan para la funcionalidad que se construirá. A cada tarea se le asignan esfuerzo y duración y se crea una red de tareas (también llamada “red de actividades”) de tal forma que permita al equipo de software cumplir con la fecha límite de entrega establecida. Cuando se desarrolle una calendarización, compartiméntese el trabajo, anótese las interdependencias de las tareas, asígnese esfuerzo y tiempo a cada tarea, defínase responsabilidades, resultados e hitos.



Calendario de actividades para el desarrollo de un software

- ✓ Una fecha límite irrealizable establecida por alguien externo al grupo de ingeniería del software e impuesta a los gestores y profesionales del equipo.
- ✓ Cambio en los requisitos del cliente que no se reflejan en modificaciones a la calendarización.

- ✓ Una subestimación razonable de la cantidad de esfuerzo o de recursos que se requerirán para realizar el trabajo.
- ✓ Riesgos predecibles o impredecibles que no se consideraron cuando comenzó el proyecto.
- ✓ Dificultades técnicas que no pudieron preverse.
- ✓ Dificultades humanas imprevisibles.

Falta de comunicación entre el personal del proyecto, lo que genera demoras.

Síntesis:

Si se debe agregar personas a un proyecto retrasado, asegúrese de que se les ha asignado trabajo enormemente compartimentado.

- **OBJETIVO DE LA CALENDARIZACION**

Permitir que un gestor defina las tareas de trabajo, establezca sus dependencias, asigne recursos humanos o las tareas y desarrolle una variedad de gráficas, diagramas y tablas (cronograma de actividades que permitirá determinar que tareas se realizarán en un punto dado en el tiempo) que auxilian en el seguimiento y el control del proyecto de software.

Análisis personal

La calendarización es la culminación de una actividad de planificación que es un componente principal de la gestión del proyecto de software. Cuando se combina con métodos de estimación y análisis de riesgos, la calendarización establece un mapa de carreteras para el gestor del proyecto.

La calendarización comienza con el proceso de descomposición. Las características del proyecto se utilizan para adaptar un conjunto de tareas apropiado al trabajo que se realizará. La red de tareas se utiliza para calcular la trayectoria crítica, un cronograma y una variedad de información del proyecto.

Téngase presente que para desarrollar un excelente calendario de actividades se debe haber realizado una excelente gestión de proyectos que permita obtener una buena estimación de cada uno de los procesos a realizar. Recuerde que la estimación permite control absoluto del tiempo, esfuerzo de trabajo y recurso implementado dentro del proyecto, por lo cual, de una acertada estimación depende el modelo prescriptivo a seleccionar y cada actividad a ejecutar.

Ejercicio del tema 4

1. Seleccione un conjunto de tareas apropiadas para la construcción del software que se debe implementar en la elaboración del robot casero que le mencionan en el inciso 2 de los ejercicios propuestos para la estimación de proyectos.
2. Las fechas límites “irracionales” son un hecho de la vida en el negocio del software. ¿Cómo se debe proceder si usted se enfrenta con una de estas fechas en un tiempo ya cumplido?

3. Agregar personal a un proyecto de software retrasado puede retrasarlo más, existen circunstancias en las cuales esto no es cierto. Descríbalas.
4. ¿Qué es lo primero que usted debe tener presente al momento de desarrollar un calendario como plan de trabajo “teniendo presente la fecha límite de tiempo a cumplir” para cualquier tipo de proyecto?

Gestión de riesgo

Riesgos reactivos y proactivos

Reactivo: se presenta cuando ya se está desarrollando el proyecto (dentro de).

Proactivo: comienza mucho antes de que se inicie el trabajo técnico.

Entre estas dos clases de estrategias para el desarrollo de software es fundamental la estrategia de riesgo proactiva, ya que se identifican los riesgos potenciales, se valora su probabilidad e impacto y se les clasifica de acuerdo a la prioridad o importancia. El objetivo principal es evitar el riesgo antes de que suceda, aunque debe estar claro que no todos los riesgos pueden evitarse, pero en lo que se pueda se debe desarrollar un plan de contingencia que responda y controle el riesgo difícil de evitar.

○ Riesgos de software

Se tiene claro que cuando ocurre el riesgo este ocasiona incertidumbre y pérdida indeseables. Los riesgos que se presentan en un proyecto amenazan el desarrollo de la planeación de este, ya que estos riesgos identifican los problemas en la calendarización, estimación, en el personal (riesgo de alta probabilidad), en los recursos, en la parte técnica (los cuales amenazan la calidad y viabilidad del software) porque identifican problemas de interfaz, mantenimiento, implementación y sobre todo en el diseño.

Todo riesgo que se presente en el software afectará directamente al proyecto, por lo cual se debe tener presente una excelente planeación del proceso y sobre todo comunicación constante entre el equipo de trabajo y de este a su vez con el cliente para poder determinar los posibles riesgos y así realizar un plan de contingencia adecuado a su control.

○ Identificación del riesgo dentro de un proyecto de software

Lo primero que se debe hacer en la realización de un proyecto es identificar los riesgos habidos y por haber para poder dar los pasos necesarios para evitarlos o controlarlos. La idea principal es evitar el riesgo antes de que suceda, pero si no

se puede, entonces se debe desarrollar un plan que permita controlar el riesgo cuando se presente.

Existen dos tipos distintos de riesgos para cada una de las categorías anteriormente relacionadas: los riesgos genéricos y los riesgos específicos del producto.

Los riesgos genéricos: son una amenaza demasiado grande que involucra a todo el proyecto de software.

Los riesgos específicos del producto: se pueden identificar solamente si se tiene el conocimiento específico de la tecnología utilizada en el desarrollo del proyecto, del personal y sobre todo de todo el entorno que rodea la construcción del software. Estos riesgos se identifican realizando un plan detallado de análisis sobre el producto que se construirá y toda actividad o tarea requerida que se involucre a su desarrollo.

- **Proyección de riesgo**

Es conocido también como estimación de riesgos. En esta proyección se busca identificar la probabilidad de que el riesgo ocurra realmente y su alto, medio o bajo grado de consecuencias imprevistas. Para esta estimación se debe desarrollar una tabla de riesgos y clasificar esto de acuerdo a su grado de probabilidad de ocurrencia, por su puesto con su respectivo plan de contingencia y el impacto de este al momento de controlar el riesgo. Se debe tener presente que un riesgo se mide por:

- Alta probabilidad de que ocurra
- Probabilidad media de que ocurra
- Baja probabilidad de ocurrencia del problema o situación. Por lo cual, se debe realizar un estudio profundo de identificación del riesgo, su impacto y factores (tiempo, ámbito y su naturaleza de ocurrencia).

- **Refinamiento de riesgo**

⁶Durante las primeras etapas de la planificación del proyecto, un riesgo puede ser declarado de un modo muy general. Con el paso del tiempo y con el aprendizaje sobre el proyecto y sobre el riesgo, es posible refinar el riesgo en un conjunto de riesgos más detallados, cada uno algo más fácil de reducir, supervisar y gestionar.

¿Cuál es una buena forma de describir un riesgo?

Una forma de hacer esto es presentar el riesgo de la forma **condición-transición-consecuencia** (Es decir, el riesgo se presenta de la siguiente

Forma:

Dada esta <condición> entonces existe preocupación por (posiblemente) <consecuencia>.

“Dado que todos los componentes reutilizables del software deben ajustarse a los estándares específicos del diseño y que algunos no lo hacen, es entonces

⁶ Roger S. Pressman

preocupante que (posiblemente) solo el 70 por 100 de los módulos planificados para reutilizar puedan realmente integrarse en el sistema que se está construyendo, teniendo como resultado la necesidad de que el ingeniero tenga que construir el 30 por 100 de los componentes restantes. La condición general que acabamos de destacar puede ser refinada de la siguiente manera:

Subcondición 1: Ciertos componentes reutilizables fueron desarrollados por terceras personas sin el conocimiento de los estándares internos de diseño.

Subcondición 2: El estándar de diseño para interfaces de componentes no ha sido asentado y puede no ajustarse a ciertos componentes reutilizables existentes.

Subcondición 3: Ciertos componentes reutilizables han sido implementados en un lenguaje no soportado por el entorno para el que el sistema ha sido construido.

Las consecuencias relacionadas con estas subcondiciones refinadas siguen siendo las mismas (por ejemplo, el 30 por 100 de los componentes del software deben ser construidos de un modo personalizado), pero el refinamiento ayuda a aislar los riesgos señalados y puede conducir a un análisis y respuesta más sencilla⁷.

- **Reducción, supervisión y gestión de riesgo**

Cuando se empieza a desarrollar un proyecto se deben tener en cuenta tres aspectos importantes:

- ✓ Evitar el riesgo.
- ✓ Supervisar el riesgo, y
- ✓ Gestionar el riesgo y planes de contingencia

Se debe tener presente que en el desarrollo de cualquier proyecto, evitar el riesgo es lo recomendable, lo cual se consigue desarrollando un minucioso plan de reducción de riesgos. El riesgo se reduce si el gestor del proyecto y su equipo colaborador desarrollan estrategias eficaces que controlen la movilidad de dicha situación problemática.

Es importante anotar que los pasos de reducción, supervisión y gestión del riesgo (RSGR) generan gastos o costos adicionales no presupuestados dentro del proyecto. Pero es también importante resaltar que estos extras de costos evitan otros costos más altos que se puedan presentar por problemas ocasionados por los riesgos no controlados. Téngase presente que los riesgos también pueden ocurrir después de que el producto se ha

⁷ www.scribd.com › ... › [Study Guides, Notes, & Quizzes](#), [Administracion de Riesgos](#), consultado 05-21-2011

desarrollado exitosamente y entregado al cliente. ⁸Estos riesgos están típicamente asociados con las consecuencias de la falla de software en el campo. (RSGR)

- **El plan RSGR**

La pretensión principal de este plan es tener un control para disminuir en un alto porcentaje los riesgos que se pueden presentar en la construcción de software o en la elaboración de proyectos empresariales.

Una vez que se ha desarrollado el plan RSGR y el proyecto ha comenzado, empiezan los procedimientos de reducción y supervisión del riesgo. Como ya hemos dicho antes, la reducción del riesgo es una actividad para evitar problemas. La supervisión del riesgo es una actividad de seguimiento del proyecto con tres objetivos principales:

- a. Evaluar cuando un riesgo ocurra
- b. Observar que los procedimientos se están aplicando correctamente
- c. Tener presente información de proyectos que se han desarrollado los cuales sirven como base para solucionar otros proyectos.

Ejercicios del tema 5

1. Ilustre un ejemplo en donde usted sea el responsable de la construcción de un producto de software XY.
 - Utilice modelo de desarrollo ágil para la construcción del producto.
 - Describa la calidad bajo la cual se desarrollará de su producto.
 - Describa posibles riesgos reactivos que afectarían el desarrollo de su producto.
 - Describa posibles riesgos proactivos que afectarían el desarrollo de su producto.
 - Estime el tiempo, esfuerzo y costo de su producto.
 - Priorice las actividades que usted realizará en la planeación del proceso de su producto.
2. Ofrezca 3 ejemplos de otros campos externos a la ingeniería del software que ilustren los problemas asociados con una estrategia de riesgo reactiva.
3. Describir la diferencia entre “riesgos conocidos” y “riesgos predecibles”

⁸ Roger S. Pressman.

4. Piense en una situación en la que un riesgo de alta probabilidad y alto impacto no sería considerado como parte de su plan de reducción, supervisión y gestión de riesgos. (RSGR), dentro de un modelo de desarrollo rápido de procesos.
5. Describa cinco áreas de aplicación de software en las que el análisis de la seguridad y los peligros del software serían una preocupación principal, teniendo en cuenta el modelo de desarrollo ágil.

Gestión de la calidad

- **Calidad de proyectos de software**

Se determina como crear un conjunto de tareas o actividades que ayuden a planificar, supervisar y controlar el desarrollo de un producto determinando su excelente calidad.

⁹El *American Heritage Dictionary*, define la calidad como «una característica o atributo de algo». Como un atributo de un elemento, la calidad se refiere a las características mensurables -cosas que se pueden comparar con estándares conocidos como longitud, color, propiedades eléctricas, maleabilidad, etc.-. Sin embargo, el software en su gran extensión, como entidad intelectual, es más difícil de caracterizar que los objetos físicos. No obstante, sí existen las medidas de características de un programa. Entre estas propiedades se incluyen complejidad

cíclica, cohesión, número de puntos de función, líneas de código, etcétera. Cuando se examina un elemento según sus características mensurables, se pueden encontrar dos tipos de calidad: calidad del diseño y calidad de concordancia.

La *calidad de diseño* se refiere a las características que especifican los ingenieros de software para un elemento. El grado de materiales, tolerancias y las especificaciones del rendimiento contribuyen a la calidad del diseño. Cuando se utilizan materiales de alto grado y se de rendimiento, la calidad de diseño de un producto aumenta, si el producto se fabrica de acuerdo con las especificaciones.

La *calidad de concordancia* es el grado de cumplimiento de las especificaciones de diseño durante su realización. Una vez más, cuanto mayor sea el grado de cumplimiento, más alto será el nivel de calidad de concordancia. En el desarrollo del software, la calidad de diseño comprende los requisitos, especificaciones y el diseño del sistema. La calidad de concordancia es un aspecto centrado principalmente en la implementación. Si la implementación sigue el diseño, y el

⁹ Roger S. Pressman

sistema resultante cumple los objetivos de requisitos y de rendimiento, la calidad de concordancia es alta.

¹⁰Philip Crosby ofrece una respuesta irónica con respecto a la calidad.

El problema de la gestión de calidad no es lo que la gente ignora acerca de ella. El problema es lo que creen saber...

A este respecto, la calidad tiene mucho en común con el sexo. Todo el mundo lo quiere. (En ciertas condiciones, desde luego.) Todos sienten lo que entienden. (Aun cuando no quieran explicarlo.) Todos piensan que su ejecución solo es cuestión de seguir inclinaciones naturales. (Después de todo, la gente se las arregla de alguna forma.) Y, desde luego, la mayoría de las personas piensan que los problemas en estas áreas los provocan otras personas. (Si solo se tomaran el tiempo para hacer las cosas bien.)

Muchos desarrolladores aun piensan que la calidad solo importa cuando se halla terminado en su totalidad el aplicativo o software, esta se debe aplicar durante todo el desarrollo del proyecto.

- **La gestión de la calidad abarca**
 - ✓ Garantía de la calidad del software (SQA)
 - ✓ Tareas específicas de aseguramiento y control de calidad
 - ✓ Prácticas efectivas de ingeniería del software
 - ✓ Control de todos los productos de trabajo del software y los cambios que generan
 - ✓ Procedimiento para garantizar la concordancia con los estándares de desarrollo del software
 - ✓ Mecanismos de medición e informe
- **Garantía de calidad y revisiones del software**

Consiste en evaluar la efectividad, viabilidad y fiabilidad del software o producto a través de un conjunto de funciones de auditorías o revisiones frecuentes que garantizaran el respectivo control del desarrollo del producto. Cuando se habla de revisiones se está direccionando específicamente a la utilización de métricas que garantizan que está funcionando bien y que partes del proyecto presentan error para proceder a posibles estrategias de solución. Estos controles son llevados por los ingenieros de software, cuyo objetivo es descubrir errores en la función lógica o en la implementación cuando se está desarrollando el producto (a este proceso se le llama **Revisiones técnicas formales**), las cuales son las encargadas de

¹⁰ Roger S. Pressman

verificar que el software en revisión satisfaga y garantice completamente su funcionamiento de acuerdo con los estándares predefinidos.

- **Garantía de la calidad**

¹¹La *garantía de calidad* consiste en la auditoría y las funciones de información de la gestión. El objetivo de la garantía de calidad es proporcionar la gestión para informar de los datos necesarios sobre la calidad del producto, por lo que se va adquiriendo una visión más profunda y segura de que la calidad del producto está cumpliendo sus objetivos. Por supuesto, si los datos proporcionados mediante la garantía de calidad identifican problemas, es responsabilidad de la gestión afrontar los problemas y aplicar los recursos necesarios para resolver aspectos de calidad.

La garantía de la calidad la da el buen funcionamiento del producto y con ella la definitiva satisfacción del cliente. (Un cliente satisfecho abre las puertas de la excelencia para el equipo desarrollador)

La meta del aseguramiento de la calidad es brindarle al gestor los datos necesarios para que esté informado acerca de la calidad del producto, y por consiguiente que comprenda y confíe en que la calidad del producto está satisfaciendo sus metas. Si los datos que ofrece el aseguramiento de la calidad identifican problemas, es responsabilidad del gestor abordarlos y aplicar los recursos necesarios para resolver los conflictos de calidad.

- **Control de calidad**

El control de calidad involucra inspección, revisión y pruebas durante todo el proceso de desarrollo para garantizar los requisitos esperados. El control de calidad debe tener presente la retroalimentación del proceso con el que se creó el producto.

Cada producto desarrollado tiene especificaciones bien definidas con las cuales puede compararse la salida de los procesos de cada parte de la solución.

- **Costos de calidad**

La base de normalización casi siempre es monetaria. Una vez que se han normalizado los costos de la calidad sobre una base monetaria, se tienen los datos necesarios para evaluar dónde se encuentran las oportunidades para mejorar los procesos.

Los costos de calidad se dividen en costos de prevención, evaluación y fallas

¹¹ Roger S. Pressman

Prevención: incluyen planificación de calidad, revisiones técnicas formales, equipo de prueba y entrenamiento.

Evaluación: incluyen inspección en el proceso, calibración y mantenimiento de equipo y pruebas.

Fallas: Estos se dividen en fallas internas y externas

Internas: Cuando se dan fallas antes del envío del producto

Externas: Estos se detectan después de la entrega del producto al cliente.

Aun los desarrolladores con alta experiencia están de acuerdo en que el software de alta calidad es una meta importante. Pero ¿cómo se define calidad? Un bromista dijo una vez: “Todo programa hace algo, solo que puede ser la cosa que no queremos que haga”.

Algunos antecedentes:

Se dice que la garantía de la calidad en el desarrollo de software avanza de forma paralela a la de la calidad en la fabricación de hardware. Anteriormente la calidad era responsabilidad exclusiva del programador, pero en la actualidad la calidad se aplica a todas las actividades o tareas desarrolladas en todo medio.

Se dice que la calidad de software es un “patrón de acciones sistemático y planificado” que se requiere para garantizar alta calidad. Para tener la calidad que es necesario la participación de los ingenieros de software, gestores de proyectos, clientes, vendedores y todos lo demás integrantes del equipo de trabajo.

Los integrantes del desarrollo de aplicaciones funcionan como representantes de la casa del cliente, es decir, las personas que están al tanto de la calidad del software deben observar el trabajo desde el punto de vista del cliente que es en última instancia la razón de ser de la empresa desarrolladora.

- **Revisiones del software**

En el proceso de la ingeniería del software es muy importante la revisión del software ya que esta se convierte como un filtro el cual sirve para descubrir los errores y defectos que se pueden eliminar en cualquier momento. La revisión del software “purifican” las actividades de análisis, diseño y codificación. Fredman y Weinberg abordan la revisión del siguiente modo:

El trabajo técnico necesita revisarse por la misma razón que los lápices necesitan gomas. la segunda razón por la que se necesitan las revisiones técnicas es que,

aunque la gente sea buena para captar algunos de sus propios errores, las grandes clases de errores escapan de su creador con mas facilidad de lo que se le escapa a alguien mas.

- **Impacto de los defectos de software en el costo**

Es parte fundamental dentro del desarrollo de software la revisión técnica la cual sirve para encontrar todos los errores antes de liberar el software. Aquí se descubren todos los errores para que no se extienda y afecten el proceso del software.

Cuando se han aplicado correctamente la revisión formal, se han dado a conocer más de un 70% de efectividad al descubrir los fallos en el diseño

Gestión de calidad (Conceptos de calidad / Garantía de la calidad del software SQA / Revisión del software / Revisiones técnicas formales / Enfoques formales acerca del SQA / Garantía de la calidad estadística del software / Fiabilidad del software / Los estándares de calidad ISO 9000 / El plan de SQA.

Ejercicio del tema 6

1. La calidad y la fiabilidad son conceptos relacionados pero fundamentalmente diferentes en varias formas. ¿Cuáles son estas formas?
2. ¿Puede un programa ser correcto y aun así no ser fiable? Explique.
3. ¿Puede un programa ser correcto y aun así no mostrar buena calidad? Explique a través de un ejemplo enfocado al campo empresarial.
4. ¿Por qué con frecuencia existe tensión entre un grupo de ingeniería del software y un grupo independiente de aseguramiento de la calidad del software? ¿Esto es saludable?
5. A usted se le ha dado la responsabilidad de mejorar la calidad del software por medio de su organización. ¿Qué es lo primero que debe hacer? ¿Qué sería lo siguiente? ¿Si construyó el producto utilizando el desarrollo ágil, que es lo primero y último que usted debe hacer?

Gestión del cambio

- **El depósito de elementos de configuración de software**

Anteriormente los elementos de configuración se guardaban como documentos de papel, el cual se dificultaba por las siguientes razones: Dificultad para encontrar un documento, no se sabía con exactitud los cambios realizados y sus razones y largo tiempo en la construcción de un nuevo software.

Actualmente los elementos de configuración se guardan en una base de datos la cual es más fácil de encontrar ya que anteriormente el depósito era una persona la cual debía saber con exactitud dónde y cómo encontrar la información que necesitaba y reconstruir todo cuando la información se perdía.

Lo que se pretende con el almacenamiento es la facilidad para interactuar con las herramientas que integran un determinado software.

- **El depósito de elementos de configuración de software**

Lo que se busca con depósito de los elementos de configuración del software es facilitar la gestión de las bases de datos, pero además impulsa las siguientes funciones. La integridad de los datos: Permite la validación de la entrada, modificación y salida de información cuando se requiera.

- ✓ Compartir información: Permite la distribución de la información, controlando el acceso a los datos por múltiples usuarios.
- ✓ Integra herramientas: Asignar un modelo de datos para asegurar a la información fácilmente de una manera controlada.
- ✓ Integración de los datos: Proporcionar funciones que permitan realizar varias tareas de la gestión de cambio del software.
- ✓ Fortalecer la metodología: Definir un buen modelo entidad-relación para la ingeniería del software para el manejo de los contenidos del depósito.
- ✓ Estandarización de los documentos: Se refiere a la estandarización en la creación de los objetos para crear los documentos.

- **El progreso de la gestión de cambio del software**

Para este progreso es necesario que el equipo de desarrollo de software de respuesta objetiva a los siguientes interrogantes:

- ¿Cómo identifica un equipo de software los elementos discretos de una configuración de software?
- ¿Cómo gestiona una organización las numerosas versiones existentes de un programa, permitiendo que el cambio se acomode eficientemente?
- ¿Cómo controla una organización los cambios antes y después de que el software se libere al cliente?

- ¿Quién tiene la responsabilidad de aprobar y clasificar los cambios?
- ¿Cómo garantizar que los cambios se hayan realizado adecuadamente?
- ¿Con qué mecanismo se valoran otros cambios que se realizan?
- **Gestión de la configuración para ingeniería web**

La ingeniería web utiliza un proceso incremental iterativo, la cual aplica principios de desarrollo ágil.

Dentro del desarrollo de aplicaciones web se busca que la aplicación tenga una mejor facilidad de uso, buena presentación, excelente manera de navegar y mayor seguridad. Aquí se pretende satisfacer necesidades en todo lo concerniente al desarrollo web. A continuación se enunciarán algunos puntos a tener presente a la hora de realizar planificación, supervisión y control del cambio en una aplicación web de alta calidad:

- **Problemas en la gestión de la configuración para WebApps**

Una parte importante a tener presente es la gestión de la configuración ya que estas harán que cada día el crecimiento de los negocios aumente, porque si no se tiene un control eficiente, la información se puede difundir hacia distintos lugares o presentar errores que hagan que los usuarios no utilicen este sitio web.

Para la gestión de la configuración de la web se deben considerar las siguientes tácticas:

- Contenido: La idea principal es organizar toda la información que se desea mostrar y establecer los mecanismos de control para así garantizar el buen uso, cuya información puede ser texto, video, tablas, entre otros.
- Personal: Muchos creadores de contenidos web realizan el trabajo sin conocimiento alguno de lo es la ingeniería del software e ignorando en su totalidad la gestión de configuración. Por lo tanto esas aplicaciones crecen o sufren cambios sin un análisis adecuado.
- Escalabilidad: Se debe tener presente que el control de las aplicaciones web van directamente proporcional a la escala de la aplicación ya que los cambios pequeños y complejos son los que en cualquier momento pueden presentar grandes problemas. Las aplicaciones bien planeadas van creciendo significativamente de acuerdo a la implementación de sistemas de información, bases de datos, depósitos y portales.
- Políticas: Para este punto es importante resolver los siguientes interrogantes:
 - ¿Quién asume la responsabilidad de la precisión de la información en el sitio web?
 - ¿Quién asegura que se han seguido los procesos de control de calidad antes de que la información se publique en el sitio?

- ¿Quién es el responsable de realizar los cambios?
- ¿Quién asume el costo del cambio?

Con la solución a estos interrogantes se determina la persona idónea para la gestión de la configuración para la webApps.

Objetos de configuración del WebApps

Estos abarcan textos, gráficos, imágenes, videos, audio entre otros. Con estos objetos el desarrollador define la presentación y define la forma en que los objetos se organizan para que sea de buena utilidad para el usuario final.

○ Gestión de contenido

Es mas utilizado cuando se desarrolla una página dinámica en donde se requiere que funcione en el menor tiempo posible, permitiendo la consulta rápida a una base de datos, organiza la información de coincide y luego la muestra al usuario para la toma decisiones.

○ Es subsistema de colección

Son todas las acciones que se requieren para crear o adquirir contenido así como las funciones técnicas para convertir el texto en un formato específico que puede ser HTML o XML y organizar el contenido de paquetes.

○ El subsistema de gestión

Luego de recopilar la información, esta debe ser guardada en un depósito, catalogarse y etiquetarse para definir su estado actual, la versión, los objetos relacionados. Para lo cual el subsistema de gestión debe tener presentes los siguientes elementos:

- Base de datos de contenido: La estructura para almacenar los datos
- Capacidades de la base de datos: Funciones que permiten buscar objetos de contenido, almacenar y recuperar objetos y gestionar la estructura de archivos.
- Funciones de gestión de configuración: Elementos que identifican el control de la versión, gestión de cambio, entre otros.
- **Análisis personal**

Para obtener la calidad de su producto o software se deben realizar constantes revisiones o verificaciones del desarrollo en proceso. Tenga presente que el

control de la variación es la clave para obtener un producto de excelente calidad. La gestión de la calidad del software incorpora tanto el control como el aseguramiento de la excelente calidad del producto, la cual se debe aplicar desde el comienzo del desarrollo del software, lo que quiere decir que la calidad del producto depende de la calidad con la que se desarrolle el proyecto, y este último depende de la buena gestión, estimación y medición que se realice al proyecto en general, ya que este incluye el proceso y el producto, para lo cual se requiere de una buena recopilación y evaluación de requisitos, que permitan una excelente realización y ejecución tanto del proceso que se sigue al momento de desarrollar el producto. Pero si se encuentran errores o inconsistencias, se debe proceder de inmediato a plantear los cambios necesarios que garanticen la viabilidad del software.

El control de cambio es una actividad de procedimiento que asegura la calidad y consistencia del software, conforme los cambios se realizan se aproxima una etapa decisiva del desarrollo, esto siempre y cuando los cambios a realizar sean correctos y permitan mejorar la configuración y implementación de los diferentes procesos propuestos que permitan garantizar la actualización del producto.

Ejercicios del tema 7

1. ¿Por qué cree usted que se debe implementar un cambio en el desarrollo de un proceso que ya fue documentado?
2. Con palabras propias defina que es la gestión de contenido a través de un ejemplo direccionado al campo empresarial.
3. Describa brevemente una diferencia entre la gestión de la configuración del software para un software convencional y la gestión de la configuración del software para aplicaciones web.
4. Cree usted que la gestión de la configuración del software (direccionándolo a la gestión de cambio), es una actividad protectora que se aplica a lo largo de todo el proceso de software. Justifique su respuesta.
5. ¿Cuál es el origen de los cambios que se requieren para el software?
6. ¿Cuáles son las metas y las actividades realizadas por cada uno de los participantes involucrados en la gestión del cambio?

Responda falso o verdadero

1. ¿La gestión de cambio es también llamada gestión de la configuración del software? Justifique.
2. Una meta primordial de la ingeniería del software es mejorar la facilidad con la que los cambios se pueden acomodar y reducir el esfuerzo cuando los cambios se deben realizar. Justifique
3. Un producto de trabajo de ingeniería del software se convierte en línea base sólo después de que se ha revisado y aprobado. Justifique.

Prueba Final (dd II)

En los siguientes enunciados seleccione la respuesta correcta:

1. La gestión eficaz de la gestión de proyectos de software se enfoca sobre las cuatro **P**. Estas son:
 - a. Proyecto, planificación, proceso y productividad.
 - b. Productividad, personal, planificación, y proyecto.
 - c. Planeación, proceso, proyecto y producto.
 - d. Personal, producto, proceso y proyecto.
 - e. Todas las anteriores.
 - f. Ninguna de las anteriores.
2. Los factores más importantes que influyen en la calidad del software son:
 - a. Motivación y seguridad.
 - b. Habilidad y tolerancia.
 - c. Motivación y habilidad.
 - d. Habilidad y conocimiento.
 - e. Todas las anteriores.
 - f. Ninguna de las anteriores.
- 3.Cuál de las siguientes métricas son tácticas:
 - a. Métricas del proceso.
 - b. Métricas del proyecto.
 - c. Métricas del personal.
 - d. Métricas del producto.
 - e. Todas las anteriores.
 - f. Ninguna de las anteriores.
- 4.Cuál de las siguientes métricas son estratégicas:

- a. Métricas del producto.
 - b. Métricas del proceso.
 - c. Métricas del personal.
 - d. Métricas del proyecto.
 - e. Todas las anteriores.
 - f. Ninguna de las anteriores.
5. Que se debe hacer cuando la gestión demanda que se cumpla con una fecha límite que es imposible.
- a. Realizar una estimación detallada empleando datos históricos de proyectos previos.
 - b. Reunirse con el cliente y con la estimación detallada, explicarle porque la fecha límite impuesto es irrealizable.
 - c. Ofrecer una estrategia de desarrollo incremental como alternativa.
 - d. Todas las anteriores.
 - e. Ninguna de las anteriores.

UNIDAD 3 MODELO DE DESARROLLO RÁPIDO DE APLICACIONES

OBJETIVO GENERAL

- Inculcar en el estudiante sobre la responsabilidad que tiene cuando se enfrenta a la construcción de un determinado proyecto y a la aplicación idónea de todos los conocimientos adquiridos en la materia y la recopilación de otras, con el fin de que se garantice la calidad del desarrollo del software y la satisfacción del cliente.

OBJETIVOS ESPECÍFICOS DE LA UNIDAD

- ✓ Identificar cuando se requiere la aplicación desarrollo rápido, los requisitos necesarios para su aplicación, prioridades y actividades necesarias que lleven a la construcción de un producto de alta calidad por medio de este modelo.

PRUEBA INICIAL

En los siguientes enunciados seleccione la respuesta correcta:

1. Un modelo de desarrollo ágil busca:
 - a. Desarrollar software tempranamente.
 - b. La satisfacción del cliente y la entrega temprana de software incremental.
 - c. Equipos de proyectos pequeños y con alta motivación.
 - d. Todas las anteriores.
 - e. Ninguna de las anteriores.
2. Agilidad es:
 - a. Una propuesta efectiva al cambio.
 - b. Ajustarse a cambios rápidos (en el software a construir, entre los miembros del equipo de trabajo, debido a las nuevas tecnologías, etc.)
 - c. El trabajo de individuos que trabajan en equipo y que sus aptitudes y capacidad para colaborar, son esenciales para el éxito del proyecto.
 - d. Todas las anteriores.
 - e. Ninguna de las anteriores.
3. Los tratados clave que deben existir entre la gente que integra un equipo de software efectivo son:
 - a. Competencia y enfoque común.
 - b. Colaboración y habilidad para la toma de decisiones.

- c. Capacidad de resolución de problemas confusos y confianza.
 - d. Respeto mutuo y organización propia.
 - e. Todas las anteriores.
 - f. Ninguna de las anteriores.
4. Los modelos de proceso ágil fueron diseñados para cumplir con los siguientes procesos:
- a. La importancia de la organización propia de los equipos de trabajo.
 - b. Comunicación y colaboración entre los miembros del equipo de trabajo, entre los profesionales y sus clientes.
 - c. Reconocimiento de que el cambio representa una oportunidad y cuidado en la entrega rápida del software diseñado.
 - d. Todas las anteriores.
 - e. Ninguna de las anteriores

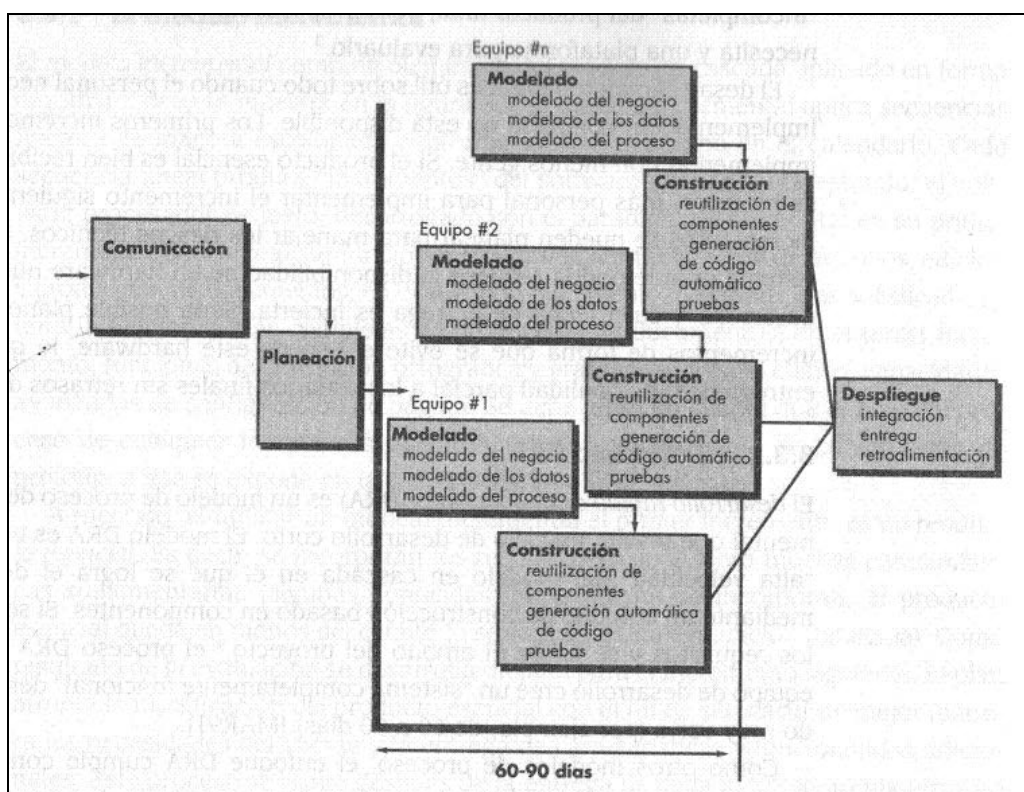
Modelos de desarrollo rápido

- El desarrollo rápido

Para determinadas personas, el desarrollo rápido es la aplicación de un método o una herramienta, para otros es codificar un tiempo determinado completamente, para el programador consiste en usar Prototipado rápido con la última versión de Visual BASIC o Delphi, entre otras posibilidades.

Se dice que el desarrollo rápido es una frase opuesta a desarrollo lento. También se dice que el desarrollo rápido es realizar un software a una gran velocidad mayor a la que se puede alcanzar actualmente.

Actualmente en el desarrollo de proyectos se aplica esta forma de desarrollo rápido debido a la gran necesidad que tienen las empresas en controlar y tener un buen manejo de la información.



Gráfica # 13

Modelo de desarrollo rápido de aplicaciones

○ **COMO LOGRAR EL DESARROLLO RÁPIDO**

Se deben tener presente métodos orientados a la planificación como parte fundamental para el desarrollo rápido.

- Aumentar la velocidad de la programación para entregar el software antes de tiempo o en el tiempo menor estipulado.
- Disminución del riesgo en la planificación para reducir los tiempos
- Mostrar el progreso para disminuir la incertidumbre de un trabajo lento

Se debe tener presente que de acuerdo a la gran necesidad que tiene el cliente para que se le entregue el trabajo lo antes posible, de esto depende el método a seleccionar, no olvidando que seleccionar el método y evitar dificultades es fácil de decir pero difícil de hacer.

○ **Estrategias para el desarrollo rápido**

Si cogiésemos 100 músicos de prestigio a nivel mundial y los pusiésemos en una orquesta sin director, no sonaría como una orquesta de calidad. El tiempo de la selección de cuerda no coincidiría con el de la sección de madera y viento o con la sección de metal. Indicar a los músicos que lo hagan “Lo mejor que puedan” no les ayuda a saber si deber ir más rápido o mas lento. Semejante evento musical sería un desperdicio de talento.

En el desarrollo de software se produce un desperdicio similar de talento. Equipos de desarrolladores inteligentes y dedicados utilizan las técnicas mejores y más recientes, y siguen sin poder alcanzar sus objetivos de planificación.

¹²Una de las trampas más tentadoras en las que cae la gente que quiere desarrollar más rápido, consiste en centrarse solo en métodos de desarrollo orientados a planificación. Podríamos ejecutar un Prototipado rápido, pero si cometemos un error no obtendremos un desarrollo rápido.

Para garantizar que el desarrollo rápido sea eficiente se debe tener presente lo siguiente.

1. Evitar los errores clásicos
2. Aplicar las bases de desarrollo
3. Gestionar los riesgos
4. Aplicar métodos orientados a la planificación.

¹² Ph. D. Roger S. Pressman

- **Errores clásicos**

Muchos desarrolladores de software, clientes y directivos creen tener buenas razones para las decisiones que toman en ciertos momentos, pero son tan repetitivas estas decisiones que siempre creen que tienen buenas razones. Pero son tantos los errores que se han cometido, que predicen las consecuencias con anterioridad y los errores rara vez producen las respuestas que las personas esperan.

Ejemplo de errores clásicos

- Si se necesita salvar un proyecto, se piensa en ingresar mas personal
- Alguien del grupo de trabajo está incomodando, se espera a que termine el proyecto para retirarlo
- Hay que acelerar el proyecto, se observan desarrollos con conocimientos y que inicien el trabajo.

- **Errores clásicos con personas**

- a. **Motivación débil:** Según estudios se dice que esta tiene mayor efecto sobre la productividad y la calidad.
- b. **Personal mediocre:** la capacidad individual y sus relaciones tienen mejor influencia en la productividad. Contratar de forma rápida sin analizar será una amenaza para el desarrollo rápido.
- c. **Empleados problemáticos incontrolados:** Este amenaza la velocidad de desarrollo y es la mas común que se encuentre dentro del un grupo de trabajo
- d. **Hazañas:** Es la satisfacción por el plan que se va a desarrollar que por el progreso de una actividad o desarrollo.
- e. **Añadir más personal a un proyecto retrasado:** Hacer esto es retrasar la productividad de las personas del equipo, se dice que es como echar gasolina al fuego.
- f. **Oficinas repletas o ruidosas:** Trabajar en lugares silenciosos y libres aumenta la productividad.
- g. **Fricciones entre los cliente y los desarrolladores:** Este se presenta cuando no hay claridad entre lo que el cliente quiere o desea y lo que el desarrollador entiende o hace debido a la mala comunicación entre las partes y esto hace que el proyecto se pueda cancelar.
- h. **Expectativas poco realistas:** Se refiere al retraso que se da cuando se compromete a entregar un desarrollo en un tiempo determinado y no se logra. Los directores de proyectos en algunos casos hacen estimaciones o muy altas o muy bajas que ocasionan un desfase para el desarrollo de dicho proyecto.
- i. **Falta de un promotor efectivo del proyecto:** Si no se tiene un promotor idóneo no se podrá garantizar la efectividad en el desarrollo de proyectos.

- j. **Falta de participación de los implicados:** Todos los implicados en el desarrollo de los proyectos deben estar involucrados durante toda la ejecución de cada etapa.
- k. **Falta de participación del usuario:** Sino está involucrado desde el inicio el usuario se puede dar que se retrase la ejecución del proyecto
- l. **Ilusiones:** Se refiere a la incertidumbre que se presenta cuando se está llevando a cabo la solución de un proyecto en donde en algunos casos y de acuerdo al planteamiento de la solución se duda de la forma como se llega al éxito.

También se da confianza plenamente en lo que estamos haciendo y pensamos que para el cliente es normal todo y que le va a gustar el trabajo que se está haciendo.

En otros casos cuando se debe entregar la solución en un tiempo determinado y a veces se tranquilizan porque se piensa que con un poco de esfuerzo al finalizar el tiempo pues se puede entregar el trabajo.

- **Proceso**

- a. **Planificación excesivamente optimista:** Se refiere a la mala planeación que se puede hacer para el desarrollo de un determinado proyecto en donde se planea un proyecto que requiere de 10 meses como otro que lleva 3 meses.
- b. **Gestión de riesgo insuficiente:** Es tener presente que no solo los errores clásicos afectan el desarrollo de un proyecto sino todos aquellos riesgos que no están previstos en el momento pero que a medida que se avanza se pueden presentar y así saber como manejarlos.
- c. **Fallo de los contratados:** Se refiere a la entrega tarde de los avances de un desarrollo por parte de los integrantes y al revisar su trabajo se pueden encontrar dificultades en las tareas asignadas para luego iniciar su corrección.
- d. **Planificación insuficiente:** Como es un desarrollo rápido siempre se debe tener presente lo que está haciendo.
- e. **Abandono de la planificación bajo presión:** Esta se da cuando se falla en una planeación y no tener el plan alternativo para contrarrestar la dificultad y con esto se llega al plan de codificar y corregir.
- f. **Pérdida de tiempo en el inicio difuso:** Es el tiempo que ocurre antes que inicie el proyecto debido a la planeación de sus actividades y presupuestos.
- g. **Diseño inadecuado:** Debido a la presión y la rapidez con la que se hace el desarrollo no se tiene presente el requerido y esperado por el cliente.
- h. **Control insuficiente de la directiva:** Es la forma de analizar si el proyecto va por buen camino desde el inicio hasta su final.

- i. **Omitir tareas necesarias en la estimación:** Es no dejar perder los datos de proyectos anteriores ya que servirán como base en el desarrollo rápido.
- **Producto**
 - a. **Exceso de requerimientos:** Es cuando los proyectos tienen mas requerimientos de los que realmente necesitan, generándose en algunos casos una planificación de software innecesaria.
 - b. **Desarrolladores meticulosos:** Cuando los desarrollares encuentran otras herramientas para el desarrollo se aplicaciones distintas a las planteadas al inicio del proyecto y empiezan a hacer cambios que obstaculizan el trabajo de desarrollo rápido.
 - c. **Tiras y aflojas en la negociación:** Cuando el directivo aprueba el retraso de un plan y añade tareas complementarias después que se realice el cambio y es elaborado por la persona que realizó el retraso.
 - d. **Desarrollo orientado a la investigación:** Se da cuando se necesita investigar nuevos algoritmos o nuevas técnicas de computación, en este momento se está investigando en software y seria un poco difícil hablar de desarrollo rápido.
- **Tecnología**
 - a. **Síndrome de la panacea:** Cuando se confía demasiado en las ventajas de una tecnología que no se había usado antes. También se da cuando se aferra a una técnica o una nueva tecnología y espera solucionar sus problemas de planificación, siendo esto un gran error.
 - b. **Sobreestimación de las ventajas del empleo de nuevas herramientas o métodos:** Las nuevas técnicas también suponen nuevos riesgos que solo se detectan usándolas. También se da sobrestimación cuando se reutiliza código de desarrollos anteriores ya que no se está seguro si se gana o no tiempo.
 - c. **Cambiar de herramientas a mitad del proyecto:** Cuando se hace este cambio será un poco difícil garantizar efectivamente el buen funcionamiento de la aplicación ya que no están previstos completamente los errores que se puedan cometer.
 - d. **Falta de control automático del código fuente:** Si no se tiene presente este punto en el desarrollo de una aplicación y mas aun cuando son aplicaciones grandes en donde trabajan varias personas que deben entregar avances en un tiempo determinado para un proyecto específico, se tiende a dañar el trabajo de uno con el del otro. En algunos casos por falta de control se hace uso en la prueba de un avance que no tiene completamente la última actualización y se pierde gran tiempo al tratar de corregir algo que ya está listo en otra versión.
- **Bases del desarrollo de software**

Si no se utilizan adecuadamente los principios orientados a la planificación y los conceptos principales para el desarrollo de software, el riesgo es muy alto para alcanzar las metas que se ha propuesto y para esto se requiere tener presente las siguientes bases:

- **Bases de gestión:** Si no se implementa principalmente la gestión de proyectos en la solución de problemas, se tiende siempre al fracaso. Dentro de la gestión se controlan principalmente tres elementos como son la planeación, el costo y el producto.

La gestión determina el tamaño del producto, asigna recursos y crea un plan para aplicar los recursos y evitar que se desvíe el proyecto

- Estimación: Este incluye el tamaño del producto, estimación del esfuerzo y por último una planificación.
- Planeación: Poca planeación, contrato erróneo, error en la definición del problema, poca experiencia, presión, poco control de cambio, falsos plazos
- Seguimiento: Este incluye una lista de tareas, planificación de reuniones, informes, presupuestos y todo lo correspondiente a gestión. Es recomendable hacer una medición y seguimiento del proyecto en ejecución para así garantizar el éxito del mismo.
Si no se hace seguimiento a un proyecto, pues no se puede hacer gestión, ya que si no se sabe como va el proyecto menos se harán recomendaciones pertinentes que ayuden a disminuir en un alto porcentaje los riesgos que puede ocasionar.

- **Bases técnicas:**

- Gestión de requerimientos: Permite reunir requerimientos, luego plasmarlos en documentos, cuaderno u otro formato; hacerle el seguimiento del diseño y el código y gestionar los cambios para el resto del proyecto. Según estudios realizados existen tres razones por las que los proyectos son entregados tarde los cuales son: falta de información por parte del usuario, requerimientos incompletos y los cambios en los requerimientos.
Bases para la gestión de requerimientos
 - Metodología de análisis: análisis estructurado, análisis estructurado de datos y orientado a objetos
 - Métodos para crear el modelo del sistema: diagramas de clases, flujos de datos, entidad relación, diccionario de datos.
 - Métodos de comunicación: aplicaciones, prototipos de interfaz del usuario y métodos.

- Relaciones entre la gestión de requerimientos: modelos del ciclo de vida incluyendo Prototipado evolutivo, entrega por etapas, espiral, cascada, codificar y corregir.

- **Diseño:** Es de suma importancia la elaboración de un diseño antes de construir un sistema de software. Dentro del diseño de software se habla del concepto de modularidad y ocultamiento de información como base de dicho diseño
- **FUNDAMENTOS PARA EL DESARROLLO RÁPIDO**

Dentro del desarrollo rápido se pueden presentar varios interrogantes tales como necesita aumento de velocidad?, mejor visibilidad?, menor coste?, velocidad a cualquier costo?, etc.

Luego de un análisis realizado se descubrió que lo que le importaba dentro del desarrollo rápido era menor costo y evitar fallos catastróficos.

Apariencias del desarrollo rápido: Antes de realizar el proyecto hacia la planificación más corta se debe entender:

- **Evitar retrasos:** El cliente requiere que el desarrollo de software deseado solo se demore el plazo estipulado con los presupuestos previstos adecuadamente, y esto se logra alcanzando la máxima velocidad.
- **Predecibilidad:** En este punto se pretende realizar un desarrollo eficiente y utilizar métodos que reduzcan el riesgo de planificación.
- **Menor coste:** La planificación más corta de desarrollo es también la más barata pero esto no garantiza que el trabajo realizado cumpla con la calidad esperada.
- **Fecha fija de pérdida de valor:** El valor de un producto disminuye a lo largo del tiempo y llega hasta cierto tiempo, esto depende la velocidad con la que se está trabajando. Se debe tener presente que la velocidad de desarrollo, disminuye el costo, pero también aumenta la incertidumbre de la planificación.
- **Deseo de horas extras gratuitas:** Las necesidades que el cliente tiene obliga a que los desarrolladores directivos hagan que sus empleados tengan que trabajar más tiempo del normal y esto a ocasiones no garantiza la confiabilidad de la aplicación pero si el empleador da incentivo a sus empleados, pues ellos se comprometen más y la calidad puede mejorar en un alto porcentaje.
- **EXPECTATIVAS POCO REALISTAS DE LOS CLIENTES**
El periodo entre la fecha estimada y la fecha real de finalización cuenta mucho en la percepción de que los proyectos de software van lentos.

- **Distribución del tiempo empleado**

Se debe analizar el tiempo utilizado en un proyecto típico e intentar de reducir dicho tiempo. Se debe analizar los distintos puntos de vista que permitan ver y analizar la posibilidad de disminuir el empleo del tiempo.

- **PLANIFICACIÓN**

Planificación excesivamente optimista: Se puede pensar que la planificación imposible es un problema actual y que la planificación optimista en el desarrollo de software es una tradición. Se ha dicho que todos los problemas de programación se han vuelto urgentes.

También se ha dicho que la excesiva presión en la planificación es el más común de los problemas de ingeniería de software.

Ejemplo de planificación excesiva

El desarrollo de Microsoft Word para Windows 1.0 da una lección clara sobre los efectos de planificación excesiva. WinWord necesitó 5 años para su desarrollo, consumió 660 personas-mes de esfuerzo de desarrollo y generó un sistema de 249.000 líneas de código.

La planificación inicial para este sistema era de 395 días la inicial y la segunda planificación era de 460 días.

Aquí se observa la gran diferencia entre el tiempo estimado y el tiempo real que se llevó el desarrollo esperado.

Luego de terminar el desarrollo, WinWord tardó 12 meses en estabilizarse y se había proyectado solo tres meses.

WinWord no se consideró un desarrollo rápido debido al gran tiempo que se utilizó.

- **Causas fundamentales de la planificación demasiado optimistas.**

- Hay un plazo límite extremo e inmutable
- Los clientes se niegan a aceptar un rango de estimación adecuada y optan por otro.
- Los desarrolladores subestiman el proyecto ya que desean un incentivo o les gusta trabajar bajo presión

- El cliente espera el trabajo en un corto tiempo y el directivo no puede contradecir
- El proyecto comienza con una planificación realista, pero en poco tiempo el proyecto se realiza bajo una planificación demasiado optimista.
- El proyecto siempre se estima mal

- **Presión excesiva en la planificación**

La reacción de los responsables y clientes cuando se enteran que no existe una planificación optimista es cargar más de presión en la planificación sobre los desarrolladores e insistir que trabajen más horas extras.

La planificación excesivamente optimista perjudica la planificación de desarrollo real, pero la presión la perjudica más.

Calidad: El 40% de todos los errores de todo el software son causa de la tensión, se evitarían si se haría una planificación apropiada.

Azar: En el desarrollo de software se debe hacer lo posible por reducir los riesgos. Se necesita corregir riesgos calculados, pero no solo riesgos que podamos gestionar cerrando los ojos y esperando a que funcionen.

Motivación: A los desarrolladores de software les gusta trabajar. Un poco de presión en la planificación resultante de una planificación ligeramente optimista pero factible puede motivar.

Creatividad: En el desarrollo de software debe existir buena creatividad para pensar mucho y persistir cuando la solución deseada no aparece rápidamente.

Agotamiento: Si se hace uso excesivo del personal de desarrollo en un proyecto determinado, pues no estarán con la misma disposición y capacidad para llevar a cabo otro proyecto o harán uso del tiempo disponible en cosas de poca importancia para la empresa debido a la no existencia de otros proyectos.

Cambio de personal: Es recomendable hacer uso del personal que está completamente comprometido y con sentido de pertenencia ya que de esa manera se puede garantizar el trabajo y para esto se debe incentivar muy bien a los empleados para comprometerlos más con la empresa.

- **METODOS ORIENTADO AL CLIENTE**

Planificación: Para este método se debe tener en cuenta lo siguiente:

- **Selecciones un modelo del ciclo de vida:** Ofrezcale a su cliente signo tangible de progreso.
- **Identifique al cliente real:** A la persona que se debe tener presente en todo los proyectos es al jefe que es uno de los que toma las decisiones y manténgalo contento.
- **Establezca un método eficiente para interactuar con el cliente:** Trata al máximo de llegar a un acuerdo con una sola persona porque es mas fácil tomar decisiones y en el mejor tiempo posible y no con seis o mas personas ya que siempre se presentan dificultades entre las partes.
- **Cree un proyecto satisfactorio para todos:** Cree un plan para alcanzar las condiciones de victoria y luego controle el proyecto para ver los riesgos que se pueden presentar y así evitarlos.
- **Gestión de riesgo:** Tenga cuidado con los riesgos que se puedan presentar en su planificación hacia los clientes para evitar dificultades.

Análisis de requerimientos: Los requerimientos reales algunas veces están en conflicto con los requerimientos recopilados.

Los clientes tienden a interpretar los requerimientos de forma amplia mientras los desarrolladores de forma reducida.

Cuando se invierte tiempo prudente en recopilar los requisitos reales tendrá luego grandes ventajas sobre requerimientos extraños y así podrá entregar más rápido el software.

Diseño: Se debe utilizar métodos de diseño que permitan que el cliente pueda cambiar de opinión algunas veces.

El diseño es de gran importancia ya que orienta al cliente en la transformación que se hará para la recolección de la información que necesita controlar dentro de su empresa.

Construcción: Los clientes deben estar involucrados en el proceso de desarrollo del software ya que ellos son los más interesados en saber como va a quedar su sistema. En este punto se debe tener en cuenta:

- Creación de código legible y modificable para responder adecuadamente a las necesidades de los clientes.
- Utilizar métodos que permitan mostrar progreso a los clientes
- Utilice modelos de ciclo de vida que ofrezca al cliente progresos tangibles.

La elección del modelo como por ejemplo el incremental el cual permite entregas frecuentes a los clientes, esto proporciona confianza y una comunicación mas efectiva.

- **MOTIVACIÓN**

La motivación es la mayor influencia individual sobre cómo trabajan las personas, se dice que la motivación tiene mayor influencia en la productividad que en cualquier otro factor.

Factores principales de la motivación

- **Realización:** Facilitarles un entorno de trabajo para centrarse en su desarrollo
 - Las personas trabajan más duro para alcanzar sus propios objetivos que para alcanzar el de otras personas.
 - La planificación que los desarrolladores realizan son muy ambiciosas
- **Posibilidad de superación:** Para el desarrollador es muy importante trabajar en un campo en donde cada día resulten cosas nuevas o nuevos métodos de desarrollo ir creciendo y estar a la vanguardia de las nuevas invenciones y las tecnologías. Las organizaciones deben de dar a sus empleados esa oportunidad de crecer en sus proyectos y no oponerse como lo pueden hacer algunas en la actualidad las cuales siguen con principios fuertemente opuestos
- **El trabajo en sí:** La motivación de las personas debe partir de tres fuentes: El trabajo debe tener sentido, tener responsabilidades por el resultado y conocer los resultados reales de su trabajo.
- Se han definido cinco dimensiones del trabajo en si
 - La diversidad de técnicas
 - La identidad de la tarea
 - La importancia de las tareas
 - Autonomía
 - Realimentación del trabajo
- **Vida personal:** Para un desarrollador, la responsabilidad extra sería mas un engaño que un placer, y la disminución de la vida personal es una fuerte pérdida. Para los directivos no hay tiempo para analizar la vida personal de sus empleados, solo respetan el tiempo de vacaciones y fiestas.

- **Oportunidades de supervisión técnica:** Es la capacidad de supervisión que adquiere una persona para dirigir a otros. Muchos desarrolladores se comprometen a trabajar con más entusiasmo, motivados por la oportunidad de supervisión.
Dentro de este punto se busca que las personas que han adquirido cierta experiencia se conviertan en monitores de las otras personas ya que tienen la capacidad para orientar en los trabajos y proyectos que se estén llevando a cabo.

i. EQUIPO DE TRABAJO

Es algo más que un conjunto de personas que desean trabajar juntas formando en si el llamado equipo de trabajo, los cuales tienen objetivos y propósitos comunes en donde existe una responsabilidad entre todos.

En muchos casos solo una persona se beneficia del trabajo realizado por un equipo, por lo tanto si las mentes trabajan en común acuerdo el resultado será exitoso de lo contrario podrá ser un fracaso o un trabajo con un éxito incierto.

Sentido de identidad del equipo

Cuando un equipo de trabajo realiza las tareas con una visión y objetivo único, se va presentando una oportunidad de identidad la cual sirve como base para diferenciarlos de las demás personas del montón.

Por ejemplo cuando se tienen claridad en el equipo de futbol que mejor juega, pues se opta por apoyarlo mucho más y son muchas las personas que se adhieren sea de forma directa o indirecta, porque han demostrado la destreza y el gran trabajo en equipo que los diferencia de otros que hacen las cosas por un cumplir o por satisfacer algunas necesidades desde la parte económica.

Mezcla de funciones

Es importante que un equipo tenga una mezcla de habilidades para las diferentes funciones encomendadas, por ejemplo que una que algunas personas tengan la habilidad para desarrollar en el lenguaje x mientras que otros en el lenguaje y así sucesivamente.

El Dr., Meredith Belbin ha identificado las siguientes funciones de dirección:

- Conductor
- Coordinador

- Creador
- Monitor
- Programador
- Soporte
- Investigador
- Responsable del acabado.

Cuando se conforma un buen equipo de trabajo, las personas trabajan distintos proyectos en distintos periodos de tiempo con distintas funciones y luego estos pueden intercambiar actividades para que todo el equipo sea muy competitivo.

Existen otros puntos importantes dentro del equipo de trabajo tales como:

- Compromiso con el equipo
- Confianza mutua
- Interdependencia entre miembros
- Comunicación efectiva
- Sensación de autonomía
- Sentido de refuerzo
- Tamaño reducido del equipo
- Alto nivel de disfrute
- Dirección de un equipo de alto rendimiento

También existen algunas causas de fallo de los equipos de trabajo tales como

- Falta de visión común
- Falta de identidad
- Falta de reconocimiento
- Barreras de productividad
- Comunicación ineficiente
- Falta de confianza
- Personas problemáticas

Modelos de equipos

Se busca que los modelos de equipo que se desarrollen se adapten a cualquier trabajo que se desea implementar, solo basta con algunas modificaciones mínimas y su funcionalidad sería de gran éxito.

Equipo de negocios

A partir del jefe técnico, todas las demás personas tienen el mismo estatus dentro de una empresa u organización, diferenciándose en según la experiencia de cada uno.

La estructura de un equipo de negocios se parece a una estructura jerárquica ya que se concentra la comunicación con la directiva, identificando una persona como responsable del proyecto y de acuerdo a la experiencia de cada persona asigna la tarea específica a desarrollar.

Equipo en la sombra

Es un grupo de personas interdisciplinaria a los cuales se les da la libertad de desarrollar e innovar, en muchos casos los directivos no desean saber como realizan las personas el trabajo sino que les interesa que realmente están trabajando en la marcha.

Los equipos a la sombra son muy importantes ya que su creatividad seria cada vez mejor y más importante para cualquier empresa.

Equipos de búsqueda y rescate

Este se encarga de resolver problemas específicos, necesita conocer con claridad lo que se va a hacer, en que se va a hacer y donde se va a hacer, lo cual le permite identificar la forma de actuar en un momento determinado.

Combina un conocimiento específico en el manejo herramientas de hardware y software y su entorno.

Equipo profesional de atletismo

El encargado de un proyecto puede contratar los especialistas que desee según su especialidad para que realice los trabajos específicos tales como manejo de base de datos, manejo de métricas, interfaz del usuario, etc, pero tampoco con esto se pretende que la persona esté en capacidad de realizar todas o muchas de las tareas o actividades.

CONTROL DEL CONJUNTO DE PRESTACIONES

Según estudios realizados se ha dicho que los cambios de requerimientos dentro del proceso de desarrollos de proyectos son los que incrementan los costos y la planificación.

Cuando se tienen proyectos muy retrasados se pueden presentar grandes interrogantes como puede ser ¿Por qué el software no se entrega a tiempo? y existen diversas respuestas para esta pregunta pero de igual manera retrasa algunos procesos planeados dentro de la empresa o cliente.

Existen tres tipos de controles para el conjunto de prestaciones

- Al principio del proyecto: para planificación y cálculo de presupuesto
- En la mitad del proyecto: para analizar cambios de requerimientos
- Al final del proyecto: recortando para alcanzar objetivos.

Proyecto iniciado: Reducción del conjunto de prestaciones

Es la parte inicial de un proyecto determinado de mostrarse de una forma sencilla y coherente y para lograr esto se sugieren tres métodos:

- **Especificación mínima:** Se da a conocer toda la información detallada del proyecto que se va a llevar a cabo y esto hace que si se desea implementar algún nuevo requerimiento este se adapta sin obstaculizar el proceso que está desarrollando; por lo tanto esta información bien especificada ayudará a entender mejor el diseño y la codificación que se hará.
- **Filtrado de requerimientos:** Cuando se hace el filtro de toda la información que se tiene recolectada como requerimientos para el proyecto en el que se está trabajando, esto ayuda de una manera directa a tener más seguridad en lo que se está haciendo y disminuir en un alto porcentaje el riesgo del proyecto.
- **Desarrollo por versiones:** En el desarrollo de proyectos es recomendable por ejemplo el método de entrega evolutiva y entrega por etapas, ya que periódicamente se hacen versiones en donde al avanzar se van corrigiendo anomalías que se presentan en versiones anteriores y al finalizar el proyecto se puede tener un buen éxito siempre y cuando se tengan presente los aspectos fundamentales para cada versión.

Proyecto avanzado: control de cambios de las prestaciones

Los cambios se van presentando continuamente sea porque faltó algo dentro del análisis o recolección de requisitos o porque existan nuevas necesidades dentro del entorno en que se usa el sistema.

En ocasiones los desarrolladores realizan cambios porque han hecho una gran invención y quieren implementarlo u otras veces hacen modificación porque crean nuevas versiones hacen actualizaciones en las versiones anteriores en caso de las falencias que se puedan presentar.

Efectos de los cambios

Los cambios en los desarrollos de software en muchas ocasiones tienen un alto costo y esto se debe a que no se hizo un buen análisis de requisitos o porque el cliente no dio a conocer con claridad todas sus necesidades.

- **HERRAMIENTAS PARA AUMENTAR LA PRODUCTIVIDAD**

Hoy en día se busca trabajar siempre en máquinas automáticas y es por eso que los avances de invención tanto de hardware o software toman mayor fuerza, ya que no queremos los humanos repetir cosas sino que existan elementos o máquinas que solo se programen y hagan el resto del trabajo con mayor seguridad y al menor costo.

Áreas de especial aplicación

Aplicaciones orientadas a SGBD: Estas aplicaciones están aptas para trabajar en muchas plataformas, lo cual permite extraer con estos contenidos gran cantidad de informes y mediante estos tomar las mejores decisiones.

Aplicación personalizada: Con estas herramientas que se tienen actualmente se pueden desarrollar aplicaciones con diseños que se ajusten a la necesidad de los clientes.

Estrategia de las herramientas para la productividad

El uso de las herramientas es un proceso que requiere de tiempo dinero para lograr el mejor rendimiento de ellas.

Las ventajas que se pueden dar entre las herramientas es al pasar el tiempo desde el inicio en donde se usa y el momento en que los competidores hacen uso de ese mismo tipo de herramientas.

Criterios de selección de herramientas

- **Beneficios estimados:** Es la eficiencia que se espera con la nueva herramienta que se va a usar.
- **Estabilidad del vendedor:** ¿Tiempo de desempeño?, ¿existen respaldo por parte de otra empresa si esta se termina?
- **Calidad:** Estado y calidad de la herramienta que se va adquirir
- **Madurez:** Aquí se analiza la cantidad de versiones que se tienen de esta herramienta.
- **Tiempo de formación:** Se analiza que la persona que va a utilizar la herramienta tenga experiencia en el uso de ella.
- **Aplicabilidad:** Se analiza si la herramienta se adapta a los procesos o viceversa para tomar la mejor decisión.
- **Compatibilidad:** La herramienta funciona sin problema con las demás herramientas que se tienen instaladas?

- **Ámbito de crecimiento:** De acuerdo al proyecto que se está solucionando se debe analizar si de este se pueden sacar nuevas versiones para que la empresa tenga un manejo de información acorde a sus necesidades.
- **Compromiso:** Cuando se adquiere una herramienta debe estar muy seguro de lo que se va a recibir para que luego no se especule sobre otra de mejor calidad.
- **RECUPERACIÓN DE PROYECTOS**

Algunos proyectos según su necesidad se pueden hacer por medio del desarrollo rápido pero realmente pueden tener sus implicaciones debido a que el levantamiento de la información es muy apresurado y se pueden presentar falencias al desarrollar dicho proyecto.

Filosofía

Cuando se trabaja a grandes velocidades los proyectos, no se alcanza a analizar correctamente toda la información provocando esto una pérdida de tiempo perdiéndose el horizonte real de lo que se espera como solución.

Cuando se pierde el control de lo que se desea hacer, no es fácil acomodarse a esa línea real, por lo cual se debe definir nuevamente las coordenadas que debe seguir y que conduzca a la solución esperada por el cliente.

Plan de recuperación

- Evalúe la situación
- Prepárese para corregir el proyecto
- Pregúntele al equipo sobre lo que se necesita hacer
- Rea realista

Personal

- Haga todo lo que sea necesario para recuperar la moral del grupo
- Elimine los problemas de personal mas importante
- Elimine los problemas principales con los responsables
- Si tiene que hacerlo, aumente el número de personas con cuidado
- Céntrese en el tiempo de las personas
- Permitir que los miembros del equipo sean diferentes
- Permitir que los desarrolladores marquen su propio ritmo

Proceso

- Identifique y resuelva los errores clásicos

- Detecte y corrija los procesos de desarrollo destrozados
- Creación de hitos miniatura detallados
- Establecer una terminación vinculada a la terminación de hitos
- Siga el proceso de la planificación
- Razones por las que no se han alcanzado los hitos
- Re calibrar después de un corto periodo de tiempo
- No se comprometa con nueva planificación hasta no terminar la que tiene
- Gestione los riesgos con esmero

Producto

- Estabilizar los requerimientos
- Recorte del conjunto de prestaciones
- Evalúe su posición política
- Eliminar la basura
- Reducir el número de defectos
- Alcanzar un estado correcto conocido.

EJERCICIOS DEL TEMA 1

1. ¿Qué es la agilidad?
2. ¿Cuáles son los pasos a seguir para desarrollar software utilizando el modelo ágil?
3. ¿Qué es un proceso ágil?
4. ¿Por qué cambian tanto los requisitos del cliente en un proceso de desarrollo ágil?
5. De ejemplo de 3 errores clásicos que se puedan presentar en el modelo de desarrollo ágil.
6. Direccione sus ejemplos al campo empresarial del mundo del software.
7. ¿Por qué es importante la elaboración de un diseño del sistema antes de construirlo?
8. Idear un principio de agilidad que pudiera ayudar a un equipo de ingeniería del software a volverse aún más manejable. Justifique
9. ¿Por qué el desarrollo ágil recomienda la comunicación cara a cara con el cliente?

10. ¿Podría cada uno de los procesos ágiles describirse recurriendo a las actividades genéricas del marco de trabajo?

11. ¿Cree usted que la disciplina es esencial para construir un sistema utilizando el desarrollo ágil?

12. **Responda falso o verdadero: (Justifique sus respuestas)**

¿La mayor prioridad del desarrollo ágil es satisfacer al cliente mediante la entrega temprana y continua del software?

¿La atención continua a la excelencia técnica y al buen diseño mejora la agilidad?

¿El software en funcionamiento es la medida primaria de progreso del desarrollo ágil?

¿En el desarrollo ágil es prioridad la habilidad del equipo de trabajo para la toma de decisiones?

Prueba final

1. Mencione por lo menos 3 características del proceso de desarrollo ágil.
¿Qué puede retrasar un desarrollo de software que se construye a través del modelo ágil?
Específicamente en que se centra el proceso de desarrollo ágil.

2. Relacione 3 circunstancias que involucren al factor humano como principal involucrado en el retraso de un software que se desarrolla a través del modelo ágil.

3. ¿Puede usted apoyarse a la vez en el modelo de desarrollo ágil y el modelo cascada para construir un software que involucre una planeación extrema?

4. ¿Por qué las pruebas continuas son un requisito indispensable cuando se construye software a través del modelo de desarrollo ágil?

5. ¿Agilidad es lo mismo que rapidez?

6. ¿Por qué la colaboración continua del cliente es primordial en el proceso de desarrollo ágil?

7. ¿Cree usted que la programación extrema es el proceso ágil que más se utiliza?

8. ¿Por qué la comunicación y colaboración entre los miembros del equipo de software y entre los profesionales y sus clientes es fundamental al momento de desarrollar software amparados en cualquier tipo de modelo prescriptivo de proceso?
9. Haga una síntesis con respecto a este tema de desarrollo ágil y usted recomendaría aplicarlo?

PISTAS DE APRENDIZAJE

Tener en cuenta: Que si se comprende con claridad conceptos sobre los modelos prescriptivos, será más fácil llevar a cabo el desarrollo de un proyecto

Tener en cuenta: El modelo prescriptivo en espiral es el que se utiliza para grandes proyectos pero que se debe tener el personal idóneo para llevarlo a cabo y requiere de buen tiempo

Tener presente: Que dentro de la ejecución de un proyecto es importante tener al personal como elemento fundamental

Tenga presente: Que dentro del proceso se deben utilizar las actividades del marco de trabajo y por ende las actividades, acciones y tareas.

Tener en cuenta: Que las 5wh no son caprichos del autor y luego del tutor sino que son cosas que debe primar en todo proyecto.

Tenga presente: Que las métricas de proceso y proyecto buscan la eficacia del proyecto a través de las mejoras de calidad.

Tener en cuenta: Que el gestor de proyecto debe ser una persona con excelentes conocimientos y personalidad muy definida para que sepa orientar o guiar a su personal.

Traer a memoria: Que la estimación de proyectos no es algo que se lleva a cabo en pocos días, esto requiere de un análisis profundo para determinar con exactitud lo que realmente cuesta el proyecto y el tiempo que se demora.

Traer a memoria: Que el listado de riesgos que se deben tener presentes en la ejecución de un proyecto, en lo posible es recomendable documentarlos para dar a conocer cuál es plan de contingencia a tener presente o cual es la actitud frente a problemas.

Tenga presente: Que todos los cambios que se realicen durante la ejecución del proyectos, deben estar siempre por escrito para evitar malos entendidos al finalizar dicho proyecto.

GLOSARIO

Gestión de proyectos: Planificación, supervisión y control de los pasos a seguir en la construcción de un producto.

Métricas: Medición.

Estimación: Atisbar al futuro.

Proactivo: Antes de.

Reactivo: Durante o después de.

Prototipado: Es la representación simbólica de un producto con el fin de analizar cómo va a quedar dicho producto al finalizar el proceso

W5hh: Corresponde a llevar a cabo lo siguiente: Why, what, when, who, where, how, how.

Escalabilidad: Es la capacidad que se tiene para mejorar adecuadamente la ejecución de una tarea o de un proceso

Riesgo: Es la probabilidad que ocurra algo inesperado existe o que suceda algo que puede afectar una serie de lineamientos o procesos

Fricciones: Hacer que algo sea muy visible a una persona para comentarlo o explicarlo

BIBLIOGRAFÍA

- Ph.D Roger S. Pressman, University of Connecticut, Ingeniería del Software un Enfoque Práctico, sexta edición, Mc Graw Hill, 2002, 2005
- Alfredo Weitzenfeld, Sur de California (Estados Unidos). Ingeniería del Software Orientado a Objetos con UML, Java e Internet, Thomson, 2004.

2. TITULO DE UNIDAD 1 (ESTILO TITULO 1)

2.1. Título de tema 1 de unidad (estilo Titulo 2)

2.1.1. Sub titulo de tema 1 (estilo Titulo 3)

2.1.1.1 Sub tema del tema 1 (estilo, Titulo 4)

Contenido

Viñetas

- ◆ Ddgfhg
- ◆ Ggghh
- ◆ Vvbv

PÁGINA EJEMPLO PARA HORIZONTALES

