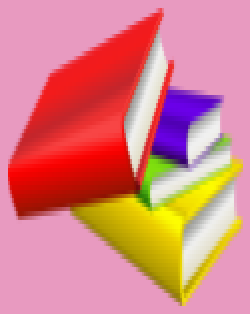
INSTITUTO TECNOLOGICO DE IZTAPALAPA

"RESUMEN DEL LIBRO SWEBOK"

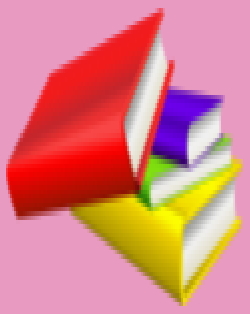
EQUIPO CONFORMADO POR:

TLAPALAMATL ROBLES MAYTE AKETZALY NO.CONTROL:181080167

MARTÍNEZ GÓMEZ DIEGO ANTONIO
NO. CONTROL:181080160



RUIZ BARCO NATALIA
NO. CONTROL:181080149



CARRERA: INGENIERIA SISTEMAS COMPUTACIONALES

GRUPO: ISC-5AV

MATERIA: FUNDAMENTOS DE INGENIERÍA DE SOFTWARE

PROFESOR: M.C. ABIEL TOMÁS PARRA HERNÁNDEZ

CAPÍTULO 1

Se entiende perfectamente que son muy vulnerables los proyectos de software cuando existen relaciones con los requisitos y ellos son muy deficientes. Literalmente cuando se utiliza el término de ingeniería de requisitos es para poder denotar su manejo de los requisitos. Lo que realmente se me hizo muy interesante la parte en donde menciona que solamente por parte de la literatura se ocupa ingeniería de requisitos, pero independientemente se le pone o llama ingeniero de software o en casos muy especiales se le llama requisitos especiales, a mi en lo personal me parece muy curioso que a veces cosas así sean tan complejas. Es interesante que con ayuda de un proceso de requisitos se pueda tener una descripción de alto nivel de un procesado junto con sus restricciones con las que pueda operar el proceso y dentro de ellas las cuales entran para poder operarlos. Por que es posible que se puedan tener complicaciones con nuestro equipo de trabajo.

No olvidemos que los requisitos de un producto son parte fundamental para la ingeniería de software que se va a desarrollar, por que el software garantiza que se cumpla con todas las características.

Es por eso por lo que para que tenga éxito un proceso de requisitos debe verse como un proceso que aplica actividades estrechamente acomodado por que más que como una sola actividad está hecha para realizar un proyecto de software. Lo más padre e increíble de este capítulo es que aclara realmente es que todos los problemas posibles para la ingeniería de software son algo que realmente se puede controlar y tienen todas las soluciones posibles.

En esta parte del capítulo nos habla de la obtención de los requisitos del software y como ingenieros el como recopilarlos con la primera etapa de comprender el problema para poder resolverlo ya que es una fundamental y es donde identificamos y establecemos las relaciones en el equipo de desarrollo con el cliente. Es bueno decir que de los principios fundamentales de un buen proceso de obtención de requisitos es el de una comunicación eficaz, ya que una comunicación continua a través del todo el ciclo de desarrollo de software con partes o diferencias en distinto tiempo ya que antes del desarrollo comienza las especificaciones de los requerimientos que pueden formar. Continuando con esto nos ha barca una extensión sobre formas diferentes obtener requisitos del software los alcances, clasificaciones, límites, si es funcional o no, la prioridad que existe dado que muchos de este capítulo es base a los fundamentos necesarios ya que podemos utilizar distinta herramientas, técnicas aun así tenemos que saber que muchas de estas son requerimientos el cual es fundamental hacer una análisis de los requisito, detectar y solucionar los conflictos que puede haber en los equipos, analizar qué requisito es de un alto nivel, un requisito que puede estar en el proceso o el producto ante todo existe factores que influye que es la naturaleza del problema que exige que

determinemos los aspectos y sean analizados, la experiencia del ingeniero es más a menudo que exista más productividad y adoptar una anotación o un método por la experiencia y los requisitos de proceso del cliente ya que estos imponen su notación, favorecen o prohíben cualquier que no es familiarizados este factor puede entrar en conflicto, teniendo en cuenta que casi todos o mayoría de los casos es útil comenzar un modelo previsto y el entorno externo.

De acuerdo al análisis es que se relaciona con tema 4 también se va relacionando con los temas o subtemas 5.3 y 6.3 ya que los temas se están relacionando con el método formales de la ingeniería de software zona de conocimiento de los modelos y métodos, en el análisis nos informa que hubo un gran impacto de inteligencia, ya que se quiere tener un lenguaje sea formalmente semántica definida, que con el análisis se requiere con un dos beneficios, el primero quiere la expresión de requisitos expresados en el idioma que se debe especificar de manera precisa, en segundo lugar deben ser razonado que permitan las propiedades deseadas del software y se debe especificar que se va utilizar y cómo se va probar y cuanto razonamiento se requiere que las herramientas sean factibles para cualquier cosa que sean sistemas triviales ya que se dividen en dos tipos que son probadores de teoremas o damas de modelo.

En ningún de los casos se puede realizar o plantear una prueba automatizada ya que se debe ver el razonamiento necesario de las herramientas, los análisis se centran por lo general en etapas tardías de análisis de requisitos, cuando se aplican la formalización de hasta objetos de negocios y los requisitos de los usuarios han llegado a un enfoque agudo a través de medios tales, ya que una vez que los requisitos se estabilizan han sido elaborados para que se especifique las propiedades concretas del software, que sea un beneficio formalización al menos los requisitos críticos ya que esto va permitir la validación estática que el software específica.

En este documento su definición del sistema es que los requisitos o conceptos de operaciones de documentos se van a registrar a los requisitos del sistema y esto hace una definición de los requisitos de alto nivel de la perspectiva del dominio y el marketing puede desempeñar estos roles para el software impulsados por el mercado.

CAPÍTULO 2

Lo que entiendo en este capítulo es que lo importante de un componente de software cuente con su descripción necesaria para poder tener un nivel alto de detalle que permita su construcción. Con cualquier modelo que nosotros queramos realizar debemos revisar que realmente cumpla con el modelo de los requisitos, en donde también debemos tener en cuenta la verificación y validación de los sistemas además de tomarlos como insumos y puntos de partida para la construcción y pruebas.

Es muy importante tener en cuenta el diseño arquitectónico del software y el diseño detallado del software. El diseño (KA) solo va enfocado al diseño de descomposición por lo que yo entendí. Y (KA) no aborda el tema de investigación que generalmente se realiza durante el software proceso de requisitos con el objetivo de conceptualizar y especificar software para satisfacer las necesidades y requisitos descubiertos. Lo que realmente me parece muy curioso es en donde evaden el problema por cuestiones en donde se puede solucionar con algo tan simple como lo es el diseño en un software, es ahí en donde posiblemente se pueda evitar o solucionar los problemas. Dentro de todo este lío es importante tener en cuenta que para poder tener o generar un buen diseño para el software primero debemos tener en cuenta que debemos cumplir con todos los requisitos para tener características, análisis y construcción del software. Donde encontramos una vez más la arquitectura del diseño y el diseño detallado.

“una ley, doctrina o asunción integral y fundamental" es una ley muy importante para el diseño del software ya que gracias a ella podemos ver vistos puntos muy importantes que van basados en ellos como lo son:

- acoplamiento y cohesión;
- descomposición y modularización;
- encapsulación/ocultación de información;
- separación de la interfaz y la aplicación;
- suficiencia, integridad, y la prinza;
- y la separación de preocupaciones.

Realmente este capítulo ha sido muy interesante en el aspecto de poder entender más fácilmente todo y la manera de ver que el diseño también va enfocado en otros aspectos importantes en software.

Esto nos habla de bastante que la concurrencia, control y manejo de eventos, manejo de errores, excepciones y fallas entre más ya que se ocupa descomponer el

software en procesos; tareas, subprocesos, abordando cuestiones relacionadas con la eficiencia, sincronización y programación. Aquellos diseños tienen que ver como organizan los datos y controlar el flujo así es como manejan eventos reactivos y temporales, la persistencia de datos que es diseño del manejo de datos a larga duración, una distribución de componentes que va más basado el cómo se distribuye el software a través del hardware, el cómo se comunican los componentes y cómo se pueden utilizar. Como existe el problema de tener que prevenir, tolerar, procesar y tratar con condiciones excepcionales ya que la interacción y presentación es un diseño que se refiere a cómo estructurar y organizar. De igual manera se ve la Seguridad se preocupa como prevenir la divulgación, creación, cambio, etc… de acceso de la información y los recursos, se ocupará para tolerar los ataques o violaciones para que tengamos la confianza. La estructura y la arquitectura de software es un sentido son necesarias para razonar con el sistema y comprender los elementos del software la relaciones entre ello y sus propiedades. También hay estructura para los miradores son diferente facetas de un diseño de software que puede describirse y documentarse, debe de marcar muchos puntos de vista que pertenecen a distintos problemas asociado con el software ya que tendremos que tener en cuenta que buscaremos la satisfacción de los requisitos funcionales frente la vista del proceso como el diseño entre más cosas. Y los estilos que tiene que ver los elementos y relaciones junto a las limitaciones sobre cómo se pueden usar, patrones de diseño y las de decisiones que se toman a cabo que tiene niveles, no solo eso sino tener para todas las personas que puedan verlos cambiarlo, modificarlo y será algo que se verá en cada ocasión posible.

Varias herramientas y técnicas que pueden ayudar a analizar y evaluar la calidad del diseño de software que tiene técnicas informales y formalizadas para determinar la calidad de artefactos por diseño un ejemplo sería la arquitectura revisiones, requisitos de rastreo, las revisiones de diseño de software también pueden evaluar la seguridad ayuda para las instalaciones, operaciones y uso por ejemplo los manuales y archivos de ayuda, y tiene la ayudo de evaluar la seguridad.

Análisis estático: está formado por formal o semiformal que tiene un análisis de diseño que se puede utilizar para la evaluación de un diseño el ejemplo sería el análisis de un árbol de fallas o una verificación cruzada automatizada, al hacer un diseño de análisis de vulnerabilidad se va ir formando un diseño de análisis de vulnerabilidad por ejemplo análisis estático de las debilidades de seguridad al realizarse la seguridad es una preocupación, al formar un análisis de diseño utiliza modelos matemáticos que permiten a los diseñadores predicar el comportamiento y validar el rendimiento del software y tener que dejar de depender de las pruebas.

Ya que en un análisis de pruebas se puede detectar especificación residual y errores de diseño que los puede generar la imprecisión, ambigüedad y a veces otros tipos de errores

CAPÍTULO 3

Cuando hablamos de una construcción de software estamos también hablando de creación detallada de software de trabajo a través de una combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración. La construcción de software normalmente produce el mayor número de elementos de configuración que necesitan gestionar en un proyecto de software (archivos fuente, documentación, casos de prueba).

Esto resulta ser un factor importante influir en la forma en que las personas transmiten la intención a las computadoras y conduce a uno de los impulsos más fuertes en la construcción de software. Los cambios en los entornos en los que opera el software también afectan al software de diversas maneras. En la construcción de software, los activos típicos que se reutilizan incluyen bibliotecas, módulos, componentes, código fuente y activos listos para usar son llamados los (COTS).

Reutilizar algo a menudo que trasciende el límite de los proyectos, lo que significa que los activos utilizados se pueden construir en otros proyectos u organizaciones. Algunos modelos son más lineales desde el punto de vista de la construcción, como la cascada y modelos de ciclo de vida de entrega por etapas. Por lo que entendí estos modelos tratan la construcción como una actividad que se produce sólo después de que se haya completado un trabajo significativo de requisitos previos, incluido el trabajo detallado de requisitos, extenso trabajo de diseño y planificación detallada.

Hablando de práctica es de lo que nos inicia ya que como un ingeniero de software tenemos que lidiar a veces con cosas caóticas y cambiar las limitaciones, debemos hacerlo con presión, algunos de los proyectos deben de tener una considerable actividad de diseño a la construcción mientras otros realizan un fase explícitamente centrada en el diseño independientemente así como los trabajadores de una construcción deben de haber modificaciones a pequeña escala, como una modificación sería el lenguaje de la construcción que iniciemos un lenguaje más simple de configuración en el software. Las instalaciones son los archivos de configuración basadas en texto utilizando Windows ya que los lenguajes son herramientas para crear aplicaciones a partir de elementos y pueden ser más complejo que los de configuración ya que del kit de herramientas se define a aplicaciones o las que puede esta simplemente implícitas, también de secuencias, y unos que son más flexibles en el tipo de construcción que contiene menor cantidad de información. Hablando de esta técnicas una de las importantes y se considera que se aplican en la actividad de construcción de software es la codificación que son técnicas para crear código fuente, que pueden ser usadas en clase, el manejo de las condiciones el prevenir violaciones a la seguridad de código, la organización , documentación y el ajuste de código, que al final se realizarán pruebas que se

puede aplicar de dos maneras a un así al realizar la integración a las nuevas tecnologías, como parametrizar o define el tipo/clase de todos los otros que utiliza que normalmente se compilan en el código en tiempo de desarrollos y luego a partir del código para que no degraden el rendimiento, es un predicado ejecutable que es colocado en la programación.

La forma en que se está manejando los errores afecta al software capacidad para cumplir con los requisitos relacionados con la corrección, solidez y otro atributos no funcionales las afirmaciones a veces se utilizan para comprobar por errores, ya que otras técnicas de manejo de errores como devolviendo un valor neutral, sustituyendo el siguientes pieza de datos válidos que van registrando un mensaje de advertencia que también se utiliza un código de error o cerrar el software.

Las excepciones se utilizan para detectar y procesar errores o eventos excepcionales, la estructura básica que tiene una excepción que es una rutina usa throw para lanzar un error detectada y un bloqueo de manejo de excepciones detectara la excepción en un try-catch bloquear.

El bloque try-catch puede procesar la condición errónea en la rutina o puede regresar con un control a la rutina de llamada, el manejo de excepciones las políticas deben ser diseñadas cuidadosamente siguiendo principios comunes como la inclusión en la que el mensaje de excepciones toda la información que condujo al excepción, evitando los bloques de captura vacíos, sabiendo las excepciones que arrojan el código de la biblioteca, tal vez creando un reporte de excepciones del programa.

Las estrategias de tolerancia a veces fallan las más comunes son incluir la realización de copias de seguridad y reintentar, utilizando auxiliares códigos, usando algoritmos de votación y reemplazando un valor erróneo con un valor falso que tendrá un efecto benigno.

CAPÍTULO 4

En el software debemos de tener la dinámica de que un programa proporciona los comportamientos esperados en un conjunto finito de casos de prueba, adecuadamente seleccionados del dominio de ejecución generalmente infinito. Muchos de los términos se utilizan en la ingeniería de software literatura para poder describir un mal funcionamiento en particular la culpa, fracaso, y el error, entre otros.

De todas estas se hacen pruebas que se llevan a cabo en vista de objetivos específicos, que se declaran más o menos explícitamente y con diferentes grados de precisión. Indicando los objetivos de las pruebas en términos de la medición y el control del proceso de prueba. Hay muchos casos en donde los casos de prueba se pueden diseñar para que las especificaciones funcionales se implementen correctamente, lo que se conoce de manera diversa en la literatura como pruebas de conformidad, pruebas de corrección o pruebas funcionales.

Este capítulo fue muy interesante por lo mismo que software y todas sus dinámicas, casos, fracasos, errores, y culpas, en donde es muy importante en donde todo eso funciona de manera junto con los requisitos.

Este es uno de los puntos iniciales que tiende hacer muy complicado ya que se realizan las pruebas de aceptación y calificación que determina si un sistema satisface sus criterios que, generalmente verificando los comportamientos deseados, que un así pueda ir en contra de los requisitos del cliente puede especificar o realiza directamente actividad para corroborar. Las pruebas de instalación a menudo el sistema se verifica en el entorno de destino que como una prueba del sistema realiza unas configuraciones de hardware que son las pruebas alfa y beta que antes de que se publiquen el software se proporciona un pequeño grupo a seleccionado usuario potenciales para su uso (alfa) y con mayor usuario son llamados beta. Veremos el logro y la evaluación de confiabilidad que es el identificar y corregir fallas, pruebas de regresión, rendimiento, seguridad, estrés, consecutivas, recuperación, interfaz, configuración y el uso e interacción persona-máquina que sus principales tareas que los usuarios finales aprender y utilicen el software que en general puede implicar probar funcione que respaldan las tareas del usuario. Las técnicas que se utiliza para la prueba son diferentes ya que su labor es detectar tanto fallos como sea posible. La clasificación de las técnicas de la prueba que se presenta se basa en cómo se genera las pruebas desde la intuición y la experiencia como ingeniero de software, especificaciones, estructura del código, fallas reales o imaginarias por descubrir, predecir uso, modelos o la naturaleza de la aplicación. Existen técnicas basadas en dominio de entrada como son; Partición de equivalencia, prueba por pares, análisis de valor límite y pruebas aleatorias. Técnicas basadas en código como son; Flujo de control-criterios basado, criterios basados en el flujo de datos, modelos de referencia para códigos. Técnicas basadas en fallas que son; Erró al adivinar, prueba de mutación. Y así habiendo

más técnicas para su uso. Nos ayuda a dar un enfoque especial en los roles en una especificación del flujo de trabajo que esta se dirige.

La combinando funcional y estructural

Técnicas de prueba basadas en modelos y códigos a menudo se contrastan como funcionales o estructurales pruebas estos dos enfoques para probar la selección no deben verse como alternativas sino más bien como complementos de hecho utilizando diferentes fuentes de información y se ha demostrado que destacan diferentes tipos de problemas en los pueden ver ser utilizados una combinación dependiendo del presupuesto consideraciones.

Determinista o aleatorio

Los casos de prueba se deben seleccionar de forma determinista de acuerdo con una de las muchas técnicas o extraídas al azar de alguna distribución de entradas como suele hacer en las pruebas de fiabilidad varias comparaciones analísticas y empíricas han realizado para analizar las condiciones que hacen en un enfoque sea más eficaz que el otro.

Puntuación de mutación

En las pruebas de mutación se deben consultar las pruebas de mutación en la sección 3.4, técnicas basadas en fallas la relación de mutantes muertos al número total de generados.

Comparación y efectividad relativa

De diferentes técnicas se han realizado varios estudios para comparar la efectividad relativa de diferentes pruebas, Técnicas Es importante ser preciso en cuanto a propiedad contra la que se están aplicando las técnicas juzgado; cuál es, por ejemplo, el significado exacto dado al término "eficacia"? Las posibles interpretaciones incluyen el número de pruebas necesarias para encontrar el primer fallo, la relación del número de fallas encontradas a través de pruebas a todas las fallas encontradas durante y después de la prueba, y cuánto se mejoró la confiabilidad. Se han realizado comparaciones analíticas y empíricas entre diferentes técnicas, realizado de acuerdo con cada una de las nociones de eficacia especificada anteriormente.

CAPÍTULO 5

Por lo que vi, los esfuerzos de desarrollo de software dan como resultado la entrega de un producto de software que satisface al usuario Requisitos. En consecuencia, el producto de software debe cambiar o evolucionar. Una vez en funcionamiento, los defectos se descubren, los entornos operativos cambian, y la nueva superficie de requisitos del usuario.

En esta pequeña o grande guía, el mantenimiento del software se define como la totalidad de las actividades necesarias para proporcionar soporte rentable al software. La sesión de Fundamentos de mantenimiento de software es la primera sección presenta los conceptos y terminología que constituyen una base subyacente para entender el papel y el alcance del software Mantenimiento en donde los temas proporcionan definiciones y por qué hay una necesidad de mantenimiento y las categorías de mantenimiento de software son fundamentales para entender su significado este escondido.

La parte más padre del mantenimiento de software es que mantiene el producto de software durante todo su ciclo de vida. Las solicitudes de modificación se registran y rastrean, el impacto de los cambios propuestos es determinado, el código y otros artefactos de software son modificados, se llevan a cabo pruebas, y una nueva versión del producto de software.

Algunos de los problemas claves en el mantenimiento de software como lo es garantizar el mantenimiento eficaz del software tratando de encontrar un fallo en un gran número de líneas de código que otro ingeniero de software desarrolló. Del mismo modo, competir con los desarrolladores de software porque los recursos son una batalla constante.

En este capítulo nos contempla el análisis de impacto que describe cómo realizar de una manera más rentable, así como un cambio al software existente que ese conocimiento para realizar un análisis que identifica todos los sistemas y productos que se vean afectados por una solución de cambio y desarrolla una estimación de los recursos necesarios para lograr el cambio. Mantenibilidad se define como la capacidad del software pueda ser modificado que pueden incluir correcciones, mejoras, o adaptaciones, una de las características principales es la calidad y debe especificarse, revisarse y ser controlado durante las actividades del desarrollo de software con el fin de reducir costos de mantenimientos. La presencia de sistemas sistemáticos y los procesos, técnicas y herramientas ayudan a mejorar la capacidad de mantenimiento del software. Una forma de ver los objetivos es usando una lineación con la organización que describen cómo demostrar el retorno de la inversión de las actividades de mantenimiento de software como el desarrollo inicial que generalmente está basado en proyectos en un tiempo definido y propuesto. La dotación personal que es como atraer y mantener el personal de mantenimiento de

software., El proceso de ciclo de vida del software de un conjunto de actividades, métodos, prácticas y transformaciones que las personas utilizan para desarrollar y mantener software y sus asociados, los aspectos de organizativos del mantenido se describen como identificar que la organización o funcionaria responsable del mantenimiento, el equipo que desarrolla el software no es necesariamente asignado a mantener el software una vez que está operando, dando muchísimos más proceso y actividades de mantenimiento.

Las actividades de planificación de mantenimiento

Una actividad importante para el mantenimiento de software es planificar y deben abordar los problemas asociados con una serie de perspectivas de planificación, que incluye:

- Planificación empresarial (nivel organizativo),
- Planificación de mantenimiento (nivel de transición),
- Lanzamiento / planificación de la versión (nivel de software), y
- Planificación individual de solicitudes de cambio de software (Nivel de solicitud).

A un nivel de solicitud individual, la planificación es llevada a cabo durante el análisis de impacto, el lanzamiento o versión ya que la actividad de planificación requiere que el mantenedor debe tener:

- Recopilar las fechas de disponibilidad del individuo peticiones,
- Estar de acuerdo con los usuarios sobre el contenido de los siguientes lanzamientos / versiones,
- Identificar conflictos potenciales y desarrollar alternativas,
- Evaluar el riesgo de una liberación determinada y desarrollar un plan de retirada en caso de que surjan problemas levantarse
- Informar a todas las partes interesadas.

Mientras los proyectos de desarrollo de software pueden normalmente durar desde algunos meses hasta algunos años, la fase de mantenimiento suele durar muchos años, hacer estimaciones de recursos es un elemento clave de la planificación del mantenimiento.

CAPÍTULO 6

En este capítulo en donde la configuración de un es la característica funcional y física del hardware o software, tal como se establece en la documentación técnica o se logra en un producto, eso también puede ser considerado como una colección de versiones de hardware o elementos de software combinados de acuerdo con procedimientos de construcción específicos para servir a un propósito en particular.

Lo que me parece curioso de SCM controla la evolución y la integridad de un mediante la identificación de sus elementos de gestión y controlar el cambio de y verificar, grabar e información de configuración. En lo que cabe el software se desarrolla con frecuencia como parte de un sistema más grande que contiene hardware.

Lo más interesante son las restricciones que afectan y orientan para el SCM proceso provienen de una serie de fuentes en donde entran las políticas y procedimientos establecidos en empresas u otros niveles organizativos en donde pueden influir o prescribir el diseño y la implementación del proceso SMC para un proyecto determinado. La planificación de un proceso SCM para un proyecto aparte de que es la más importante debe ser coherente con el contexto organizativo, las limitaciones aplicables, la orientación comúnmente aceptada y la naturaleza del proyecto.

En el software debemos de tener un control de interfaz que este interactúa con otro elemento de software o hardware, un cambio a cualquiera del elemento que puede afectar al otro, se identificara los elemento y cómo se realizará los cambio de los artículos gestionados y comunicados la ro puede ser parte de un nivel de sistema más amplio, un proceso de especificación y control de interfaces este involucra. El plan SCM para un proyecto determinado se registra en un plan de gestión de configuración de software (SCMP) que sirve como referencia para el proceso, es mantenido durante el ciclo.

También define y describe seis categorías de información SCM que se incluirán en un SCMP:

• Introducción (propósito, alcance, términos utilizados)

• Gestión de SCM (organización, responsabilidades, autoridades, políticas aplicables, directivas y procedimientos)

• Actividades SCM (identificación de configuración, control de configuración, etc.)

• Horarios SCM (coordinación con otras actividades del proyecto)

• Recursos SCM (herramientas, recursos físicos, y recursos humanos)

• Mantenimiento SCMP.

Hablando se refiere demasiado que es la configuración del software solicitando el cambio, consultando las bibliotecas, evaluando y aprobar el cambio si es necesario ya que antes de eso es adquirir lo requisitos o elementos para la integración del cambio, esto nos brinda una oportunidad para rastrear defectos y recopilación de medidas de actividades de cambio por tipo los aspectos técnicos y gestión de la solicitud de cambio y aceptar, modificar, rechazar o aplazar el cambio propuesto.

Tablero de control de configuración de software la autoridad para aceptar o rechazar propuestas los cambios recaen en una entidad conocida normalmente como, Tablero de control de configuración (CCB). En menor medida, esta autoridad en realidad puede residir con el líder o un individuo asignado en lugar de un tablero de varias personas. Puede haber varios niveles de autoridad de cambio dependiendo de una variedad de criterios, como la criticidad del elemento involucrado, la naturaleza del cambio (por ejemplo, impacto en presupuesto y cronograma), o el proyecto actual punto en el ciclo de vida. La composición de la los CCB utilizados para un sistema determinado varían según estos criterios (un representante de SCM estará siempre presente). Todas las partes interesadas, apropiadas al nivel del CCB, están representados. Cuando el ámbito de autoridad de un CCB es estrictamente software, se conoce como configuración de software Tablero de control (SCCB). Las actividades del CCB suelen estar sujetas a auditorías de calidad de software o revisión.

Contabilidad del estado de la configuración del software (SCSA) es un elemento de la gestión de la configuración que consiste en el registro y la generación de informes de la información necesaria para gestionar una configuración de forma eficaz.

Información de estado de configuración del software

La actividad de la SCSA diseña y opera un sistema para la captura y reporte de información a medida que avanza el ciclo de vida. Como en cualquier sistema de información, se debe identificar, recopilar y mantener la información del estado de la configuración que se administra para las configuraciones en evolución. Se necesitan diversas informaciones y medidas para apoyar el proceso de SCM y para satisfacer las necesidades de gestión de informes de estado de configuración, ingeniería de software y otras actividades relacionadas. Los tipos de información disponibles incluyen identificación de configuración aprobada, así como la identificación y el estado actual de implementación de cambios, desviaciones y exenciones.

Es necesaria alguna forma de soporte de herramientas automatizado para lograr la recopilación de datos de SCSA e informes de tareas; esto podría ser una capacidad de base de datos, una herramienta independiente o una capacidad de una entorno de herramientas integrado.

CAPÍTULO 7

Lo interesante aquí es que la gestión de la ingeniería de software se puede definir como la aplicación de actividades de gestión, casi parece idéntico, pero realmente son términos diferentes y puede ocuparse para diferentes formas de expresarse. Como por ejemplo lo son los clientes a menudo no saben lo que se necesita o lo que es factible.

• Los clientes a menudo carecen de apreciación por las complejidades inherentes a la ingeniería de software, especialmente en lo que respecta al impacto de los requisitos cambiantes.

• Los clientes a menudo no saben lo que se necesita o lo que es factible.

Es increíble cómo a veces un cliente puede hacer de la forma bonita tu trabajo, me refiero a que con cosas a veces tan simples el mismo cliente puede dejar de entenderlas y seguido de eso no sabe cómo o qué tan complejo puede ser la complejidad de algunos proyectos, pero el cliente solo pide y a los demás como equipo les toca entregar con todos los requisitos que necesita la ingeniería de software. Del mismo modo, es también útil si los gerentes de proyectos complejos y programas en los que el software es un componente de la arquitectura del sistema son conscientes de las diferencias que los procesos de software introducen en la gestión de proyectos y la medición de proyectos.

La Definición de iniciación y alcance en estas actividades se centran en la determinación eficaz de los requisitos de software utilizando diversos métodos de viabilidad del proyecto desde diversos puntos de vista. Una vez establecida la viabilidad del proyecto, las tareas restantes dentro de esta sección son la especificación de los requisitos y la selección de los procesos para la revisión y revisión de los requisitos. El proceso de revisión y revisión de los requisitos es inevitablemente del cambio de las partes deben acordar los medios por los cuales los requisitos y el alcance deben ser revisados, claramente implica que los requisitos no son estables, pero puede y revisarse en determinados a medida que se desarrolla el proyecto. La planificación de proyectos es el primer paso en la planificación de uno debe ser la selección de un modelo de ciclo de vida de desarrollo de software apropiado y tal vez adaptarlo basado en el alcance del proyecto, los requisitos de software y una evaluación de riesgos, así como la planificación de proceso que son modelos de ciclo de vida de desarrollo de software que abarcan un continuo predictivo hasta adaptarlo.

Está la asignación de recursos que es el equipo, instalaciones y las personas deben asignarse a las tareas identificadas, tener en cuenta la gestión de riesgos son conceptos relacionados pero distintos, Gestión de calidad, gestión del plan, etc. El progreso debe ser evaluado sobre el logro de los principales hitos del proyecto o al

finalizar un ciclo de desarrollo iterativo que dé como resultado un incremento de producto, se deben identificar los requisitos y se deben tomar las acciones apropiadas. En momentos preestablecidos y según sea necesario, el progreso general hacia el logro de los objetivos establecidos y satisfacción de las partes interesadas (usuario y cliente) deben evaluarse los requisitos.

Cierre

Un proyecto completo, una fase importante de un proyecto, o un ciclo de desarrollo iterativo llegan al cierre cuando todos los planes y procesos han sido promulgados y completado. Los criterios para el proyecto, debe evaluarse el éxito de la fase o iteración. Una vez que se establece el cierre, las actividades de archivo, retrospectiva y mejora de procesos pueden ser realizadas.

Determinación del cierre

El cierre ocurre cuando las tareas especificadas para un se ha completado un proyecto, una fase o una iteración y se ha confirmado el cumplimiento satisfactorio de los criterios de finalización. Software los requisitos se pueden confirmar como satisfechos o no, y el grado de consecución de los objetivos puede ser determinado. Los procesos de cierre deben involucrar partes interesadas relevantes y dar como resultado la documentación de la aceptación de las partes interesadas relevantes; cualquiera conocido los problemas deben documentarse.

Actividades de cierre

Una vez confirmado el cierre, el archivo de los materiales del proyecto deben realizarse de acuerdo con los métodos acordados por las partes interesadas, la ubicación y la duración, posiblemente incluyendo destrucción de información sensible, software, y el medio en el que residen las copias. La base de datos de medición de la organización debe actualizarse con los datos relevantes del proyecto. Un proyecto, el análisis retrospectivo de fase o iteración debe emprenderse de modo que cuestiones, problemas, riesgos, y las oportunidades encontradas se pueden analizar (ver tema 4, Revisión y evaluación). La lección aprendida debe extraerse del proyecto y alimentarse en el aprendizaje y la mejora organizacional esfuerzos.

Planifique el proceso de medición

• Caracterizar la unidad organizativa. La unidad organizativa proporciona el contexto para medición, por lo que el contexto organizacional debe hacerse explícito, incluyendo las limitaciones que la organización impone a el proceso de medición. La caracterización se puede establecer en términos de organización procesos, dominios de aplicación, tecnología, interfaces organizacionales y estructura organizacionales.

• Identificar necesidades de información. Información las necesidades se basan en los objetivos, limitaciones, riesgos y problemas de la organización unidad. Pueden derivarse de negocios, organizacional, regulatorio y / o producto objetivos. Deben identificarse.

CAPÍTULO 8

Lo más interesante que habla en este capítulo es de un proceso de ingeniería consiste en un conjunto de actividades interrelacionadas que transforman uno o más insumos en productos mientras se consumen recursos para lograr la transformación. En donde muchos de los procesos de disciplinas tradicionales de ingeniería como lo son la, mecánica, civil y química se ocupan para transformar las entidades energéticas y físicas de una forma en otra, como en una presa hidroeléctrica que transforma la energía potencial en energía eléctrica en energía o una refinería de petróleo que utiliza procesos para transformar el petróleo crudo en gasolina. Todo esto me parece algo impresionante, el como se pueden hacer muchas cosas con el petróleo crudo en gasolina, y con ayuda de cuentos.

En donde el proceso de requisitos de software y sus subprocesos pueden ser introducidos y salidos varias veces durante el desarrollo o modificación de software. En donde la definición completa de un proceso de software puede también incluyen las funciones y competencias, el soporte de TI, las técnicas y herramientas de ingeniería de software, y el entorno de trabajo necesario para llevar a cabo la proceso, así como los enfoques y medidas que son como los indicadores clave de rendimiento utilizados para determinar la eficiencia y eficacia de realizarla en un Proceso.

Categorías de procesos de software distintos se ha definido para su uso de las diversas partes del desarrollo de software y la vida útil del mantenimiento. Estos procesos se pueden categorizar como sigue:

1.Los procesos primarios incluyen procesos de software para el desarrollo, operación y mantenimiento de software.

2. Los procesos de apoyo se aplican de forma intermitente o continua en todo un software. ciclo de vida del producto para apoyar los procesos primarios; Incluyen procesos de software como como gestión de la configuración, aseguramiento de la calidad y verificación y validación.

3. Los procesos organizacionales brindan soporte a la ingeniería de software; Incluyen formación, análisis de medición de procesos, gestión de infraestructura, cartera y gestión de reutilización, proceso organizativo mejora y gestión de software modelos de ciclo de vida.

4. Procesos de proyectos cruzados, como reutilización, línea de productos de software e ingeniería de dominio; involucran más de un proyecto en una organización.

En comparación con el proceso anterior o ejemplar resultados y costos; haciendo más modificaciones, Planificar-Hacer-Verificar-Actuar se puede aplicar el modelo de mejora de procesos, por ejemplo, para mejorar los procesos de software que mejorar la prevención de defectos. Antes de que se implemente un nuevo proceso o se modifique uno actual, por los resultados de la medición para la situación actual de obtener si para proporcionar una línea de base para la comparación entre la situación actual y la nueva. Antes de introducir la inspección, el esfuerzo necesario para corregir los defectos descubiertos mediante pruebas se debe medir después de todo un periodo en marcha inicial después del proceso de inspección se introduce el esfuerzo. Represa continua y por etapas las estaciones pueden ser utilizadas para determinar el orden en el que el software los procesos deben mejorarse. En la continua representación, los diferentes niveles de capacidad para diferentes procesos de software proporcionan una guía para determinar el orden en el que se mejorarán los procesos de software dentro de un nivel de madurez se logra para ese nivel de madurez, que proporciona una base para mejorar todos los procesos de software en el siguiente nivel superior.

La medición de productos y procesos de software son preocupados por determinar la eficiencia y efectividad de un proceso, actividad o tarea. La eficiencia de un proceso de software, actividad, o tarea es la proporción de recursos realmente consumidos a los recursos que se espera o se desea consumir en la realización de un proceso, actividad o tarea (consulte Eficiencia en la ingeniería de software

Economía KA). El esfuerzo (o costo equivalente) es el medida principal de recursos para la mayoría del software procesos, actividades y tareas; se mide en

Unidades tales como horas-persona, días-persona, semanas-personal o meses-personal de esfuerzo o su equivalente unidades monetarias, como euros o dólares.

La eficacia es la relación entre la producción real y resultado esperado producido por un proceso de software, actividad o tarea; por ejemplo, el número real de defectos detectados y corregidos durante el software pruebas al número esperado de defectos a ser detectado y corregido, quizás basado en datos históricos para proyectos similares (ver Efectividad en la Ingeniería de Software Economía KA).

Tenga en cuenta que la medición de la eficacia del proceso de software requiere la medición de los atributos del producto; por ejemplo, medición de defectos de software descubiertos y corregidos durante pruebas de software.

Hay que tener cuidado al medir el producto.

Atributos con el propósito de determinar el proceso. Eficacia. Por ejemplo, el número de defectos detectado y corregido mediante pruebas puede no lograr el número esperado de defectos y así proporcionar una medida de eficacia engañosamente baja, ya sea porque el software que se está probando tiene una calidad mejor que la habitual o tal vez porque se ha introducido una inspección previa recién introducida proceso ha reducido el número restante de defectos en el software.

Medidas del producto que pueden ser importantes en determinar la efectividad de los procesos de software incluyen la complejidad del producto, defectos totales, densidad de defectos y calidad de los requisitos, documentación de diseño y otros trabajos relacionados productos.

La calidad del proceso y la medición del producto. los resultados se determinan principalmente por la confiabilidad y validez de los resultados medidos. Medidas que no cumplen estos criterios de calidad puede resultar en interpretaciones incorrectas y defectuosas iniciativas de mejora de procesos de software. Otra propiedad deseable de las mediciones de software incluir la facilidad de recopilación, análisis y presentación, además de una fuerte correlación entre la causa y efecto.

El tema de medición de ingeniería de software en el Software Engineering Management KA describe un proceso para implementar un programa de medición.

CAPÍTULO 9

Modelos y métodos de ingeniería de software imponer estructura en la ingeniería de software con el objetivo de hacer que esa actividad sea sistemática, repetible, y en última instancia más orientado al éxito. El uso de modelos proporciona un enfoque para el problema resolución, una notación y procedimientos para el modelo construcción y análisis. Los métodos proporcionan un enfoque sistemático de la especificación, el diseño, construcción, prueba y verificación del punto final software y productos de trabajo asociados.

Y el poder comunicar aspectos del software a las partes interesadas apropiadas. Los participantes son aquellas personas o partes que tienen un interés implícito en el software, por ejemplo, usuario, comprador, proveedor, arquitecto, autoridad certificadora, evaluador, desarrollador, ingeniero de software y tal vez otros. Si bien hay muchos lenguajes de modelado, notaciones, técnicas y herramientas en la literatura y en la práctica, hay conceptos generales unificadores que se aplican de alguna manera a todos ellos.

También puede resultar difícil comprender los significados precisos de los constructos de modelado los lenguajes se definen por sintáctica y semántica reglas, para los lenguajes textuales, la sintaxis está definida utilizando una gramática de

notación que define construcciones de lenguaje válidas, para lenguajes gráficos, la sintaxis es definidos mediante modelos gráficos denominados metamodelos. En la práctica, suele haber una buena comprensión de la semántica de un modelo de software debido al lenguaje de modelado seleccionado, cómo se utiliza ese lenguaje de modelado para expresar entidades y relaciones dentro de ese modelo, precondiciones, postcondiciones e invariantes un conjunto de suposiciones sobre el estado del software antes para, durante y después de que se ejecute la función o el método. como un conjunto de condiciones previas, postcondiciones e invariantes.

• Condiciones previas: un conjunto de condiciones que deben estar satisfecho antes de la ejecución de la función o método. Si estas condiciones previas no se cumplen antes de la ejecución de la función o método, la función o el método pueden producir resultados erróneos.

• Postcondiciones: un conjunto de condiciones que se garantiza para ser cierto después de la función o El método se ha ejecutado correctamente. Típicamente, las postcondiciones representan cómo el estado del software ha cambiado, cómo los parámetros pasados a la función o método han cambiado, cómo han cambiado los valores de los datos, o cómo se ha visto afectado el valor de retorno.

Otra técnica para gestionar y controlar el riesgo de software es la creación de casos de garantía. Un caso de aseguramiento es un artefacto razonado y auditable creado para respaldar el argumento de que su reclamo no satisface las reclamaciones.

Los métodos ágiles nacieron en la década de 1990 a partir de la necesidad de reducir la aparente gran sobrecarga asociada con los métodos pesados basados en planes utilizados en proyectos de desarrollo de software a gran escala.

Los métodos ágiles se consideran métodos ligeros en el sentido de que se caracterizan por ciclos de desarrollo cortos e iterativos, equipos auto organizados, diseños más simples, refactorización de código, basado en pruebas desarrollo, participación frecuente del cliente y un énfasis en la creación de un trabajo demostrable producto con cada ciclo de desarrollo.

En la literatura se encuentran disponibles muchos métodos ágiles; algunos de los enfoques más populares, que se discuten aquí brevemente, incluyen Rapid

Desarrollo de aplicaciones (RAD), eXtreme Programming (XP), Scrum y Feature-Driven

Desarrollo (FDD).

• RAD: métodos rápidos de desarrollo de software se utilizan principalmente en el desarrollo de aplicaciones de sistemas empresariales con uso intensivo de datos. El RAD El método está habilitado con herramientas de desarrollo de bases de datos de

propósito especial utilizadas por software ingenieros para desarrollar, probar e implementar rápidamente aplicaciones comerciales nuevas o modificadas.

• XP: este enfoque utiliza historias o escenarios para requisitos, primero desarrolla pruebas, tiene participación directa del cliente en el equipo (que normalmente define las pruebas de aceptación), utiliza programación por pares, y proporciona una refactorización e integración continuas del código. Los cuentos se descomponen en tareas, se priorizan, estiman, desarrollan y prueban. Cada incremento de software se prueba con y pruebas manuales; un incremento puede ser lanzado con frecuencia, como cada par de semanas más o menos.

• Scrum: este enfoque ágil es más proyecto amigable con la administración que los demás. Los scrum master gestiona las actividades dentro el incremento del proyecto; cada incremento es llamado sprint y no dura más de 30 días. Una lista de elementos de la cartera de productos (PBI) es desarrollada a partir del cual se identifican tareas definidas, priorizadas y estimadas. Se prueba una versión funcional del software y liberado en cada incremento. Scrum diario

Las reuniones garantizan que el trabajo se gestione según la planificación.

• FDD: este es un enfoque de desarrollo de software iterativo, corto y basado en modelos que utiliza un proceso de cinco fases: (1) desarrollar un producto modelo para abarcar la amplitud del dominio, (2) crear la lista de necesidades o características, (3) construir el plan de desarrollo de funciones, (4) desarrollar diseños para características específicas de iteración, y (5) codifique, pruebe y luego integre las funciones.

CAPÍTULO 10

En este capítulo se habla perfectamente de la calidad del software y el por qué es importante que se tomen algunas áreas va bien explicado que la calidad del software ya que puede referirse a las características deseables de los productos de software, en donde a la medida un producto de software en particular posee esas características, y a los procesos, y las técnicas utilizadas para lograr esas características.

Más recientemente, la calidad del software se define como la "capacidad del producto de software para satisfacer y necesidades implícitas en condiciones especificadas” y como el grado en que un producto de software cumple con los requisitos establecidos, sin embargo, la calidad depende del grado en que esos requisitos establecidos representen con precisión las necesidades, deseos y expectativas de las partes interesadas. Respecto a esto es muy cierto porque hasta para armar una aplicación se necesita observar esa característica y si realmente

cumplirá con lo que un usuario o cliente desea, entonces, para esto es necesario completarlo con un software de requisitos para tener en cuenta todas estas necesidades.

Algo que a mi perspectiva no es algo tan simple como se mira en el libro es la parte de lograr la calidad del software, por cuestiones en donde se vean de qué tamaño son los requisitos, bueno su grado, en donde puede ser o más grande o muy pequeño, por ejemplo, las mediciones de CoSQ son ejemplos de procesos medidas que pueden utilizarse para inferir en las características de un producto. La premisa subyacente a la CoSQ es que el nivel de calidad en un producto de software puede ser inferido del costo de las actividades relacionadas con el tratamiento de las consecuencias de la mala calidad. Es por eso que debemos de tener más cuidado con el producto y su calidad.

La gestión de la calidad del software es la colección de todos los procesos que garantizan que los productos de software, servicios e implementaciones de procesos de ciclo de vida cumplir con los objetivos de calidad, SQM define procesos, propietarios de procesos, requisitos para los procesos, mediciones del proceso y sus resultados, canales de retroalimentación a lo largo de todo el ciclo de vida del software. La gestión de riesgos también puede jugar un papel importante papel en la entrega de software de calidad, incorporando técnicas disciplinadas de análisis y gestión de riesgos en los procesos del ciclo de vida del software pueden ayudar a mejorar la calidad del producto. Garantía de calidad del software para sofocar un malentendido generalizado, la garantía de calidad del software no está probando. Software aseguramiento de la calidad (SQA) es un conjunto de actividades que definen y evaluar la idoneidad de los procesos de software para proporcionar evidencia que establezca la confianza de que los procesos de software son apropiados y producir productos de software de calidad adecuada para los fines previstos. Verificación validación V es ayudar a la organización de desarrollo a incorporar calidad en el sistema durante el ciclo de vida. Los procesos de V&V proporcionan una evaluación objetiva de productos y procesos en todo el ciclo vital. Esta evaluación demuestra si los requisitos son correctos, completos, precisos, consistentes y comprobables. La validación es un intento de garantizar que el derecho el producto está construido, es decir, el producto cumple su propósito específico previsto. Tanto la verificación como el proceso de validación comienzan temprano en fase de desarrollo o mantenimiento. Las revisiones técnicas del código fuente pueden incluir una amplia variedad de preocupaciones tales como análisis de algoritmos, utilización de recursos informáticos críticos, adherencia a los estándares de codificación, estructura y organización del código para la con probabilidad y consideraciones críticas de seguridad. La decisión de salida de la inspección corre responde a una de las siguientes opciones:

1. Acepte sin modificaciones o, como máximo, con modificaciones menores

2. Aceptar con verificación de reelaboración

3.Inspeccionar.

Requisitos de calidad del software

Factores de influencia

Varios factores influyen en la planificación, gestión, y selección de actividades y técnicas de SQM, incluso

- El dominio del sistema en el que reside el software; las funciones del sistema pueden ser seguridad crítica, misión crítica, negocio crítico, seguridad crítica
- El entorno físico en el que reside el sistema de software
- Sistema y software funcional (lo que el sistema) y la calidad (qué tan bien el sistema realiza sus funciones) requisitos
- Los componentes comerciales (externos) o estándar (internos) que se utilizarán en el sistema
- Los estándares específicos de ingeniería de software aplicable
- Los métodos y herramientas de software que se utilizarán para desarrollo y mantenimiento y para la evaluación y mejora de la calidad
- El presupuesto, el personal, la organización del proyecto, los planes, y programación de todos los procesos
- Los usuarios previstos y el uso del sistema
- El nivel de integridad del sistema.

La información sobre estos factores influye en cómo los procesos de SQM están organizados y documentados, cómo se seleccionan las actividades específicas de SQM, qué recursos se necesitan y cuáles de esos los recursos imponen límites a los esfuerzos

Herramientas de calidad de software

Las herramientas de calidad del software incluyen estáticas y dinámicas herramientas de análisis. Fuente de entrada de herramientas de análisis estático codificar, realizar análisis sintáctico y semántico sin ejecutar el código y presentar los resultados a usuarios. Existe una gran variedad en la profundidad, minuciosidad y alcance de las herramientas de análisis estático que

Calidad del software 10-13

CAPÍTULO 11

Los ingenieros de software deben manejar problemas de ingeniería únicos, produciendo software con características y fiabilidad conocidas. Este requisito requiere ingenieros de software que poseen un conjunto adecuado de conocimientos, habilidades, formación y experiencia en la práctica profesional.

El concepto de práctica profesional puede ser visto como más aplicable dentro de las profesiones que tienen un cuerpo generalmente aceptado del conocimiento de códigos de ética y profesionales conducta con sanciones por violaciones en donde es aceptado procesos de acreditación, certificación y licencias y las sociedades profesionales para proporcionar y administrar todos estos.

Lo más interesante es que un ingeniero de software mantiene una práctica mediante la realización de todo el trabajo de acuerdo con prácticas, normas y en particular establecidas por la sociedad profesional aplicable. Cuando nosotros hablamos de "Licencia" en el área de software estamos hablando de la acción de dar a una persona la autorización para realizar ciertos tipos de actividades y asumir la responsabilidad de los productos de ingeniería resultantes. En donde el sustantivo "licencia" se refiere a ambos esa autorización y la grabación de documentos esa autorización.

El software tiene efectos económicos en el individuo, nivel empresarial y social. Software "éxito" puede estar determinado por la idoneidad de un producto para un problema reconocido, así como por su eficacia cuando se aplica a ese problema a nivel individual, el empleo continuo de un ingeniero puede depender de su capacidad y disposición para interpretar y ejecutar tareas. en satisfacer las necesidades de los clientes o empleadores y Expectativas. A nivel social, los impactos directos del software, el éxito o el fracaso incluyen o excluyen accidentes, interrupciones y pérdida de servicio. Impactos indirectos incluyen el éxito o el fracaso de la organización que adquirió o produjo el software, aumentó o disminución de la productividad social, armoniosa u orden social disruptivo, e incluso el ahorro o pérdida de bienes y vidas. Se pueden proporcionar servicios de ingeniería de software bajo una variedad de relaciones cliente-ingeniero. El trabajo de ingeniería de software puede solicitarse como proveedor de empresa a cliente, consultoría de ingeniería a cliente, contratación directa o incluso trabajar como voluntario. En todas estas situaciones, el cliente y el proveedor acuerdan que se proporcionará un producto o servicio a cambio de algún tipo de consideración. La mayoría de los gobiernos del mundo otorgan exclusivos derechos de una obra original a su creador, generalmente por un tiempo limitado, promulgado como un derecho de autor. Los derechos de autor protegen la forma en que se presenta una idea, no la idea misma. Todos los profesionales del software deben conocer restricciones legales sobre la importación, exportación o reexportación de bienes, servicios y tecnología en las jurisdicciones en las que trabajan.

Dinámica de grupo y psicología

El trabajo de ingeniería se realiza muy a menudo en contexto de trabajo en equipo. Un ingeniero de software debe ser capaz de interactuar de manera cooperativa y constructiva con otros para determinar primero y luego satisfacer tanto las necesidades como las expectativas. Conocimiento de

La dinámica de grupo y la psicología es un activo cuando interactuar con clientes, compañeros de trabajo, proveedores, y subordinados para resolver la ingeniería de software problemas.

Dinámica de trabajo en equipos / grupos

Los ingenieros de software deben trabajar con otros. Por un lado, trabajan internamente en ingeniería equipos; Por otro lado, trabajan con clientes, miembros del público, reguladores y otras partes interesadas. Equipos de actuación: aquellos que demuestran una calidad constante de trabajo y progreso hacia las metas: son cohesivos y poseen un ambiente cooperativo, honesto y centrado.

Los objetivos individuales y de equipo están alineados para que los miembros se comprometen naturalmente y se sienten dueños de resultados compartidos.

Los miembros del equipo facilitan esta atmósfera al ser intelectualmente honesto, hacer uso del grupo pensar, admitir ignorancia y reconocer errores. Comparten responsabilidades, recompensas, y carga de trabajo de manera justa. Se encargan de comunicarse de forma clara, directa entre sí y en documentos, así como en código fuente, para que la información sea accesible para todos. Comentarios de pares sobre

Los productos de trabajo se enmarcan en un marco constructivo y forma no personal (consulte Revisiones y auditorías en el

Calidad del software KA). Esto permite que todos los miembros sigan un ciclo de mejora continua. Y crecimiento sin riesgo personal. En general, los miembros de equipos cohesionados demuestren respeto el uno para el otro y su líder.

CAPÍTULO 12

En este capítulo fue más entendible la parte de la economía en el software, puesto que hablaba de cómo un software consiste en tomar decisiones relacionadas con la ingeniería de software en un contexto empresarial. Y el éxito de un producto de software, servicio y solución depende de una buena gestión empresarial. Sin embargo, en muchas empresas y organizaciones, las relaciones comerciales de software para desarrollo de software e ingeniería siguen imprecisas.

En donde la economía de la ingeniería de software está preocupada con la alineación de las decisiones técnicas de software con los objetivos de negocio de la organización. En total las organizaciones, ya sea "con fines de lucro", "sin fines de lucro" o gubernamentales, esto se traduce en sostenible estancia en los negocios. En "para beneficio" organizaciones que, además, se relaciona con la consecución de un rendimiento tangible del capital invertido tanto los activos como el capital empleado. Así es como se observa el manejo de la economía en el software dependiendo si es con fines o sin fines de lucro. En algunas consideraciones prácticas finalizan el área de conocimiento.

Pero debemos tener en cuenta algo muy importante. La financiación es la rama de la economía en cuestión con la asignación, la gestión, adquisición e inversión de recursos las finanzas es un elemento de todas las organizaciones, incluyendo organizaciones de ingeniería de software. El ámbito financiero se ocupa de los conceptos de tiempo, dinero, riesgo y cómo están interrelacionados. También se ocupa de cómo se gasta y se presupuesta el dinero.

Así que mucho ojo con la economía en el software, ya que tenemos varios puntos a considerar gracias a esta guía.

La inflación describe las tendencias a largo plazo de los precios. La inflación significa que las mismas cosas cuestan más de lo que hicieron antes. Si el horizonte de planificación de una decisión comercial dura más de unos pocos años, o si la tasa de inflación es superior a un par de porcentajes puntos anualmente, puede causar cambios notables en el valor de una propuesta. La depreciación implica distribuir el costo de un activo tangible a lo largo de varios períodos de tiempo; se utiliza para determinar cómo las inversiones en activos capitalizados se cargan contra los ingresos varios años. La depreciación es una parte importante de determinar el flujo de efectivo después de impuestos, que es fundamental para abordar con precisión las ganancias y los impuestos. Los gobiernos cobran impuestos para financiar gastos que la sociedad necesita, pero en los que ninguna organización invertiría. Las empresas tienen que pagar impuestos sobre la renta, que pueden tomar una parte sustancial de la ganancia bruta de una corporación, Eficiencia económica de un proceso, actividad o tarea es la proporción de recursos realmente consumidos recursos que se espera consumir o que se desea

consumirse en la realización del proceso, actividad o tarea. Eficiencia significa "hacer las cosas bien". Una estimación es una evaluación bien fundada de recursos y tiempo que se necesitarán para lograr metas establecidas. Una estimación de software es utilizada para determinar si los objetivos del proyecto pueden alcanzarse dentro de las limitaciones previstas, presupuesto, características y atributos de calidad.

Las estimaciones se utilizan para analizar y pronosticar los recursos o tiempo necesarios para implementar los requisitos (consulte Esfuerzo, cronograma y estimación de costos en el Software Engineering Management KA y estimación de costos de mantenimiento en el software

Mantenimiento KA). Cinco familias de estimación existen técnicas:

• Juicio experto

• Analogía

• Estimación por partes

• Métodos paramétricos

• Métodos de estadística.

Ninguna técnica de estimación es perfecta, por lo que es útil utilizar una técnica de estimación múltiple.

Convergencia entre las estimaciones producidas por diferentes técnicas indica que las estimaciones probablemente sean precisos. La dispersión entre las estimaciones indica que ciertos factores podrían haber sido pasados por alto. Encontrar los factores que causaron la propagación y luego volver a estimar para producir resultados que converjan podrían conducir a una mejor estimación.

Abordar la incertidumbre

Debido a los muchos factores desconocidos durante el inicio y planificación del proyecto, las estimaciones son inherentemente incierto; que la incertidumbre debe ser abordada en las decisiones comerciales. Técnicas para abordar la incertidumbre incluyen:

• considerar rangos de estimaciones

• analizar la sensibilidad a los cambios de supuestos

• retrasar las decisiones finales.

La priorización implica clasificar alternativas basadas sobre criterios comunes para ofrecer el mejor el mejor posible valor. En proyectos de ingeniería de software, software a menudo se priorizan los requisitos para entregar el mayor valor al cliente dentro de las limitaciones de cronograma, presupuesto, recursos y tecnología, o para proporcionar incrementos de productos de construcción, donde los primeros incrementos proporcionan el valor más alto para el cliente (ver Requisitos

Clasificación y negociación de requisitos en los requisitos de software KA y software Modelos de ciclo de vida en la ingeniería de software Proceso KA).

Las decisiones bajo técnicas de riesgo se utilizan cuando el tomador de decisiones puede asignar probabilidades los diferentes resultados posibles (consulte Gestión de riesgos en la Gestión de ingeniería de software KA). Las técnicas específicas incluyen:

• toma de decisiones de valor esperado

• Variación de expectativas y toma de decisiones

• Análisis de Monte Carlo

• Árboles de decisión

• Valor esperado de la información perfecta.

CAPÍTULO 13

Debido a que ningún software puede existir en un vacío o se ejecutan sin un ordenador, el núcleo de un entorno de este tipo es la computadora y sus diversos componentes. Conocimientos sobre él y sus principios subyacentes de hardware y software sirven como marco qué ingeniería de software está anclada. Por lo tanto, todos los ingenieros de software deben tener una buena comprensión. Generalmente se acepta que la ingeniería de software se basa en la ciencia de la computación. Tanto la informática como la ingeniería de software se ocupan de computadoras, computación, y software. La ciencia de la computación, como un cuerpo de conocimiento, es el núcleo de ambos. Por lo tanto, en el núcleo de la ingeniería de software es una comprensión de la informática.

Mientras que pocas personas negarán el papel de la computadora la ciencia juega en el desarrollo de software ingeniería tanto como disciplina y como cuerpo de la importancia de la informática a la ingeniería de software no se puede enfatizar en exceso.

Bueno y como segundo lugar, en algunos temas tratados en esta directriz no existen como cursos independientes en programas de ciencias de la computación de pregrado o posgrado. En consecuencia, de todo esto es que tales temas pueden no ser adecuadamente cubiertos en un desglose puramente basado en cursos. Ya que me parece que el (KA) no cuenta con estos criterios. Ahora para poder solucionar problemas podemos tener en cuenta como la resolución de problemas se refiere al pensamiento y las actividades realizadas para responder o derivar una solución un problema. Hay muchas maneras de resolver un problema, y cada camino empleado sea considerar diferentes herramientas y utilizar diferentes procesos.

Los programas funcionan con datos. Pero los datos deben ser expresados y organizados dentro de las computadoras antes de ser procesados por programas. Esta organización y la expresión de datos para el uso de programas es el sujeto de la estructura y representación de los datos. En pocas palabras, una estructura de datos intenta almacenar y organizar datos en una computadora de tal manera que los datos puedan ser utilizados de manera eficiente. Las estructuras de datos son representaciones informáticas de datos. Las estructuras de datos se utilizan en casi todos los programas. En cierto sentido, ningún programa significativo puede ser construido sin el uso de algún tipo de datos estructura. Como se mencionó anteriormente, diferentes perspectivas pueden utilizarse para clasificar estructuras de datos. sin embargo, la perspectiva predominante utilizada en la clasificación se centra en el ordenamiento físico y lógico entre elementos de datos. Esta clasificación divide las estructuras de datos en estructuras lineales y no lineales. Operaciones básicas compatibles con todas las estructuras de datos incluyen crear, leer, actualizar y eliminar (CRUD).

• Crear: inserte una nueva entrada de datos en la estructura.

• Leer: recupera una entrada de datos de la estructura.

• Actualizar: modifica una entrada de datos existente.

• Eliminar: elimine una entrada de datos de la estructura.

Algunas estructuras de datos también admiten operaciones:

• Encuentra un elemento particular en la estructura.

• Clasifique todos los elementos según algún orden.

• Recorre todos los elementos en un orden específico.

• Reorganizar o equilibrar la estructura.

Pero la diferencia es que una base de datos suele ser externa a programas individuales y permanente en existencia en comparación con las estructuras de

datos. Las bases de datos se utilizan cuando el volumen de datos es grande o lógico las relaciones entre elementos de datos son importantes. Los factores considerados en el diseño de la base de datos incluyen rendimiento, simultaneidad, integridad y recuperación de fallas de hardware.

Desde la perspectiva de una computadora, una amplia

Existe una brecha semántica entre su comportamiento previsto y el funcionamiento de la electrónica subyacente. Dispositivos que realmente hacen el trabajo dentro de la computadora. Esta brecha se cierra a través de la organización informática, que combina varios dispositivos eléctricos, electrónicos y mecánicos en un solo dispositivo. Que forma una computadora. Los objetos que la computadora la organización se ocupa de los dispositivos, conexiones y controles. La abstracción incorporada en la organización informática es la computadora.

Descripción general de la organización informática

Una computadora generalmente consta de una CPU, memoria, dispositivos de entrada y dispositivos de salida. Abstractamente hablando, la organización de una computadora puede ser dividida en cuatro niveles (Figura 13.4). La macro

El nivel de arquitectura es la especificación formal de todas las funciones que una máquina en particular puede realizar y se conoce como arquitectura de conjunto de instrucciones (ES UN). El nivel de microarquitectura es la implementación de ISA en una CPU específica, en otras palabras, la forma en que las especificaciones de la ISA se llevan a cabo realmente. El nivel de los circuitos lógicos es el nivel donde cada componente funcional del micro arquitectura se compone de circuitos que toman decisiones basadas en reglas simples. El nivel de dispositivos es el nivel donde, finalmente, cada lógica El circuito está construido con dispositivos electrónicos como como semiconductores complementarios de óxido de metal (CMOS), semiconductores de óxido metálico de canal n (NMOS) o transistores de arseniuro de galio (GaAs).

Cada nivel proporciona una abstracción al nivel superior y depende del nivel inferior. A un programador, la abstracción más importante es la ISA, que especifica cosas como el nativo tipos de datos, instrucciones, registros, direccionamiento modos, la arquitectura de memoria, interrupción y manejo de excepciones y las E / S. En general, el ISA especifica la capacidad de una computadora y qué se puede hacer en la computadora con programación.

En el nivel más bajo, se realizan cálculos por los dispositivos eléctricos y electrónicos dentro de una computadora. La computadora usa circuitos y memoria para contener cargas que representan la presencia o ausencia de voltaje. La presencia de voltaje es igual a 1 mientras que la ausencia de voltaje es cero. En el disco, la polaridad del voltaje está representada por 0 y 1 que a su vez representa

los datos almacenados. Todo, incluida la instrucción y datos: se expresan o codifican mediante ceros digitales y unos. En este sentido, una computadora se convierte en un sistema digital. Por ejemplo, el valor decimal 6 puede codificarse como 110, la instrucción de suma puede codificarse como 0001, y así sucesivamente. El componente de la computadora como la unidad de control, ALU, memoria y E / S utilizan la información para calcular las instrucciones.

CAPÍTULO 14

Los profesionales del software viven con programas. En un lenguaje muy simple, se puede programar sólo para algo que sigue una lógica bien entendida, no ambigua. Las Fundaciones Matemáticas área de conocimiento (KA) ayuda a los ingenieros de software comprender esta lógica, que a su vez se traduce en el código del lenguaje de programación. KA es una parte fundamental para poder ayudar a los ingenieros del software a cumplir con los requisitos necesarios para cumplir con un proyecto o con un producto. El lenguaje y los métodos de la lógica que se discuten aquí nos permiten describir las pruebas matemáticas para inferir de manera concluyente la absoluta verdad de ciertos conceptos más allá de los números.

Aquí también podremos observar como un conjunto es una colección de objetos, denominados elementos del conjunto. Un conjunto se puede representar enumerando sus elementos entre llaves, y así sucesivamente, realmente en este capítulo se aprende mucho en como ver los conjuntos y sus funciones un poco más allá.

La recursividad es el término general para la práctica de definir un objeto en términos de sí mismo. Existen algoritmos recursivos, funciones definidas recursivamente, relaciones, conjuntos, etc. Un algoritmo es recursivo si resuelve un problema reduciéndolo a una instancia del mismo problema con una entrada más pequeña. Se dice que un fenómeno es aleatorio si los resultados individuales son inciertos, pero el patrón a largo plazo de muchos resultados individuales es predecible. Aquí, comencemos con los conceptos de distribución de probabilidad y probabilidad discreta. Un modelo de probabilidad es una descripción matemática de un fenómeno aleatorio que consta de dos partes: un espacio muestral S y una forma de asignar probabilidades a eventos. Un sistema informático puede abstraerse como un mapeo de estado a estado impulsado por entradas. En otra palabra, un sistema puede considerarse como una transición función T: $S \times I \rightarrow S \times O$, donde S es el conjunto de estados e I, O son las funciones de entrada y salida.

La gramática de un lenguaje natural nos dice si una combinación de palabras hace una válida frase. A diferencia de los lenguajes naturales, un lenguaje formal se especifica mediante un conjunto bien definido de reglas para sintaxis. Las oraciones válidas de un lenguaje formal pueden ser descritas por una gramática con la ayuda de estas reglas, conocidas como reglas de producción.

Un lenguaje formal es un conjunto de longitud finita palabras o cadenas sobre un alfabeto finito, y una gramática específica las reglas para la formación de estas palabras o cadenas. El conjunto completo de palabras que son válidas para una gramática constituye el lenguaje de la gramática. Por tanto, la gramática G es cualquier definición matemática compacta y precisa de un lenguaje L en lugar de solo una lista sin procesar de todos de las oraciones legales del idioma o ejemplos de esas oraciones.

Una gramática implica un algoritmo que genera todas las oraciones legales del idioma.

Existen diferentes tipos de gramática. Una estructura de frase o gramática de tipo 0 G = (V, T, S, P) es una tupla de 4 en la que El término libre de contexto deriva del hecho de que

A siempre se puede reemplazar por a, independientemente del contexto en el que ocurre.

Un lenguaje formal está libre de contexto si lo genera una gramática libre de contexto. Los lenguajes libres de contexto son la base teórica de la sintaxis de la mayoría de los lenguajes de programación. Gramática regular. Todos los fragmentos en el RHS son terminales simples o un par construido por un terminal y no terminal; es decir, si $A \rightarrow a$, entonces ya sea $a \in T$, o $a = cD$, o $a = Dc$ para $c \in T$, $D \in N$.

Si $a = cD$, entonces la gramática se llama derecha gramática lineal. Por otro lado, si $a = Dc$, entonces la gramática se llama gramática lineal izquierda. Ambas las gramáticas lineal derecha e izquierda son gramática regular o tipo 3.

El lenguaje L (G) generado por un la gramática G se llama lenguaje regular.

Una expresión regular A es una cadena (o patrón) formado a partir de las siguientes seis piezas de información: $a \in S$, el conjunto de alfabetos, e, 0 y el operaciones, O (+), PRODUCTO (.), CONCATENACIÓN (*). El idioma de G, L (G) es igual a todas esas cadenas que coinciden con G, $L (G) = \{x \in S^* \mid x \text{ coincide con } G\}$.

CAPÍTULO 15

Algo muy interesante es que la IEEE define la ingeniería como "la aplicación de un enfoque sistemático, disciplinado y cuantificable a estructuras, máquinas, productos, sistemas o procesos", Aplicación de este conocimiento, según proceda, permitirá ingenieros de software para desarrollar y mantener más eficiente y eficaz. Completar su trabajo de ingeniería de manera eficiente. Posiblemente un método de ingeniería para la resolución de problemas implica proponer soluciones o modelos de soluciones y luego realizar experimentos o pruebas para estudiar las soluciones o modelos propuestos.

Algo muy interesante de un estudio observacional o de caso es una investigación que hace observaciones de los procesos o fenómenos dentro de un contexto de la vida real. Mientras un experimento ignora deliberadamente el contexto, un observacional o caso de estudio incluye el contexto como parte de la observación. En donde un estudio retrospectivo implica el análisis de datos históricos. Para llevar a cabo sus responsabilidades, los ingenieros deben entender cómo los diferentes y las características del proceso varían. Ingenieros a menudo se encuentran con situaciones en las que la relación entre diferentes variables debe ser estudiada.

También se conocen estudios retrospectivos como estudios históricos. Este tipo de estudio utiliza datos que han sido archivados a lo largo del tiempo. Entonces por estas razones realmente se debe apreciar la ingeniería de software y por muchas razones más.

Los verdaderos valores de los parámetros de una distribución. generalmente son desconocidos y deben estimarse a partir de las observaciones de la muestra. Las estimaciones son funciones de los valores muestrales y se denominan estadísticas. De manera similar, la tasa de aparición de defectos estimada a partir de la muestra es una estadística y sirve como la estimación de la tasa poblacional de tasa de defectos por línea de código. Propiedades de los estimadores. Varias estadísticas Las propiedades de los estimadores se utilizan para decidir sobre la idoneidad de un estimador en una determinada situación. Un objetivo principal de muchas investigaciones estadísticas es establecer relaciones que permitan predecir una o más variables en términos de otros. Aunque es deseable predecir una cantidad exactamente en términos de otra cantidad, rara vez es posible y, en muchos casos, tenemos que ser satisfecho con estimar el promedio o esperados valores. Con la escala de intervalo, llegar a una forma cuantitativa en el sentido corriente de la palabra. Casi todas las medidas estadísticas habituales son aplicables aquí, a menos que requieran conocimiento de un verdadero punto cero. se encuentran con bastante frecuencia en las ciencias físicas. Estas escalas de medidas se

caracterizan por el hecho de que las operaciones existen para determinar las 4 relaciones: igualdad, rango orden, igualdad de intervalos e igualdad de razones.

Las medidas pueden ser directas o derivadas (a veces llamadas medidas indirectas). Un ejemplo de una medida directa sería un recuento de cuántas veces ocurrió un evento, como el número de defectos encontrados en un producto de software. Una medida es aquella que combina medidas directas en de alguna manera que sea consistente con la medición método.

Una pregunta básica que debe hacerse para cualquier método de medición es si el método de medición propuesto está midiendo realmente el concepto con buena calidad. La fiabilidad y la validez son los dos criterios más importantes para abordar esta cuestión. La fiabilidad de un método de medición es la medida en que la aplicación del método de medición produce resultados de medición consistentes. Esencialmente, la fiabilidad se refiere a la coherencia de los valores obtenidos cuando el mismo elemento se mide varias veces. Cuando los resultados están de acuerdo entre sí, se dice que el método de medición es confiable. La fiabilidad suele depender de la definición operativa. Se puede cuantificar utilizando el índice de variación, que se calcula como la relación entre la desviación estándar y la media. Cuanto menor sea el índice, más fiables será el resultado de la medición. La validez se refiere a si el método de medición realmente mide lo que pretendemos medir. La validez de un método de medición puede examinarse a partir de tres…

La resolución de problemas de ingeniería comienza cuando se reconoce una necesidad y ninguna solución existente satisfará esa necesidad. Como parte de esta resolución de problemas, se deben identificar los objetivos de diseño que debe alcanzar la solución. Además, se debe definir y utilizar un conjunto de criterios de aceptación para determinar qué tan bien una solución propuesta satisfará la necesidad. Una vez que se ha identificado la necesidad de una solución a un problema, el proceso de diseño de ingeniería tiene los siguientes pasos genéricos: a)definir el problema b)recopilar información pertinente c)generar varias soluciones d)analizar y seleccionar una solución e)implementar la solución Todos los pasos de diseño de ingeniería son iterativos, y los conocimientos adquiridos en cualquier paso del proceso se pueden utilizar para informar tareas anteriores y desencadenar una iteración en el proceso. Estos pasos se expanden en las secciones siguientes. Un. Defina el problema. En esta etapa, se recopilan los requisitos del cliente. La información específica sobre las funciones y características del producto también se examinan de cerca..