TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLOGICO DE IZTAPALAPA

N° de control

INTEGRANTES: HERNÁNDEZ PERALES LUIS DAVID 181080378
ROCHA PEREA LUCERO ZOE 161080159
JOSE MANUEL LUIS LIRA 181080378
FRANCISCO JOEL JIMENES PICHARDO 161080157

PROFESOR: ABIEL TOMÁS PARRA HERNÁNDEZ

GRUPO: ISC-5AV

ACTIVIDADES PRÁCTICAS

IEEE - SWEBOK v3.0 2014 RESUMEN
CAPÍTULO 1
REQUISITOS DE SOFTWARE

El área de conocimiento de requisitos se ocupa de la obtención,análisis,especificación y validación de los requisitos durante todo el ciclo de vida del producto del software. Es ampliamente reconocido entre los investigadores, y profesionales de la industria que los proyectos de software son críticamente vulnerables cuando las actividades relacionadas con los requisitos se realizan de manera deficiente, los requisitos de software expresan las necesidades y limitaciones impuestas a un producto de software que contribuyen a la solución de algún problema del mundo real.
El término "ingeniería de requisitos" se utiliza ampliamente en el campo para denotar el manejo sistemático de requisitos. por razones de coherencia el término de ingeniería no se utilizara.
Para resolver algún problema en el mundo real, se puede automatizar parte de una tarea para que alguien apoye los procesos comerciales de una organización, para poder corregir las deficiencias del software existente o para controlar un dispositivo, por nombrar solo algunos de los muchos problemas para los que las soluciones de software son posibles.

Una propiedad esencial de todos los requisitos de software es que pueden verificarse como una característica individual como un requisito funcional a nivel de sistema como un requisito no funcional, puede resultar difícil o costoso verificar ciertos requisitos, por ejemplo, la verificación del requisito de rendimiento en un centro de llamadas puede requerir el desarrollo de simulación, las pruebas debes de garantizar que los recursos entren dentro de las imitaciones de recursos disponibles.

1.2 requisitos de producto y procesos
un requisito de producto es una necesidad o restricción del software que se va a desarrollar, entonces un requisito de proceso es esencialmente una restricción en el desarrollo del software, algunos requisitos de software generan requisitos de proceso implícitos.

2- Proceso de requisitos
2.1 modelos de proceso
el objetivo de este es proporcionar una comprensión del proceso de requisitos
-Identifica los requisitos de software como elementos de configuración y los gestiona utilizando las mismas prácticas de gestión de la configuración de software que otros productos de los procesos del ciclo de vida del software.
-necesita adaptarse a la organización y al contexto del proyecto.

2.2 actores de procesos

representan los roles de las personas que participan en el proceso de requisitos, este proceso es fundamentalmente interdisciplinario y el especialista en requisitos debe mediar entre el dominio del interesado y de la ingeniería de software, hay muchas personas involucradas además de especialistas en requisitos, cada una de las cláusulas tiene interes en el software.

CAPÍTULO 2 DISEÑO DEL SOFTWARE

El diseño se define como el proceso de definir la arquitectura, los componentes, las interfaces y otras características de un sistema o componente.
visto desde la perspectiva de la ingeniería de software en la que se analizan los requisitos del software a fin de producir una descripción de la estructura interna del software que servirá como base para su construcción.

el diseño juega un papel importante en el desarrollo de software durante el diseño , los ingenieros de software producen varios modelos que forman una especie de plano de la solución que se implementara,tambiens e pueden examinar y evaluar soluciones alternativas y compensaciones, finalmente se usan los modelos resultantes para planificar actividades de desarrollo posteriores, como la verificacion y validacion del sistema , además de utilizarlo como entrada y como punto de partida de la construcción y las pruebas.

el diseño arquitectónico de software a veces llamado diseño de alto nivel: desarrolla la estructura y organización de nivel superior del software e identifica los diversos componentes.

diseño detallado del software:
especifica cada componente con suficiente detalle para facilitar su construcción.

1- Fundamentos del diseño de software
Los conceptos , nociones y terminología introducidos aquí forman una base subyacente para comprender el papel y el alcance del diseño.

1.1 Conceptos generales de diseño
En un sentido general, el diseño puede verse como una forma de resolución de problemas, el diseño en su sentido general para comprender los límites del diseño.

1.2 Contexto del diseño de software
el diseño es una parte importante del proceso de desarrollo de software, en este punto debemos de ver cómo se encaja en el ciclo de vida del desarrollo de software, es importante comprender las características principales del análisis , las pruebas y mantenimiento.

1.3 Proceso de diseño de software
el diseño generalmente se consideran dos pasos
-diseño arquitectónico, describe cómo se organiza el software en componentes.
-El diseño detallado describe el comportamiento deseado de estos componentes.

CAPÍTULO 3 CONSTRUCCIÓN DE SOFTWARE

El término construcción de software se refiere a la creación detallada de software de trabajo a través de un combinación de codificación, verificación, pruebas unitarias, pruebas de integración y depuración.
El área de conocimiento de Construcción de software (KA) está vinculado a todos los demás KA, pero es más fuertemente vinculado al diseño de software y al software Pruebas porque el proceso de construcción del software implica un importante diseño y pruebas de software.
El proceso utiliza la salida del diseño y proporciona una entrada a la prueba ("diseño" y "prueba" en este caso refiriéndose a las actividades, no a los KAs). Límites entre diseño, construcción y pruebas (si cualquiera) variará según el ciclo de vida del software procesos que se utilizan en un proyecto.
Aunque se puede realizar algún diseño detallado antes de la construcción, mucho trabajo de diseño se realiza durante la actividad de construcción.
Por lo tanto, la construcción de software KA está estrechamente vinculado al Software Design KA.
Durante la construcción, los ingenieros de software tanto la prueba unitaria como la prueba de integración prueban su trabajo.

Por lo tanto, la construcción de software KA está estrechamente vinculado al Software Testing KA también.
La construcción de software normalmente produce mayor número de elementos de configuración que necesitan para ser administrado en un proyecto de software (archivos fuente, documentación, casos de prueba, etc.). Por lo tanto, la Software Construction KA también está estrechamente vinculado al Software Configuration Management KA.
Si bien la calidad del software es importante en todos los KA, el código es el producto final de un proyecto de software y, por lo tanto, la calidad del software KA es estrechamente vinculado al Software Construction KA.
Dado que la construcción de software requiere conocimientos de algoritmos y de prácticas de codificación, está estrechamente relacionados con Computing Foundations KA, que se ocupa de los fundamentos de la informática que sustentan el diseño y la construcción de

productos de software. También está relacionado con la gestión de proyectos, en la medida en que la gestión de la construcción puede presentar desafíos considerables.

CAPÍTULO 4 PRUEBAS DE SOFTWARE

1. Fundamentos de las pruebas de software
1.1. Terminología relacionada con las pruebas

1.1.2. Fallos contra fallos

Muchos términos se utilizan en la ingeniería de software.

literatura para describir un mal funcionamiento: en particular falla,

falla y error, entre otros. Esta terminología se define con precisión en [3, c2]. Es esencial

para distinguir claramente entre la causa de un mal funcionamiento (para lo cual se usará el término falla

aquí) y un efecto no deseado observado en el servicio prestado por el sistema (que se denominará

fracaso). De hecho, puede haber fallas en el

Software que nunca se manifiesta como fallas (consulte Limitaciones teóricas y prácticas

de Pruebas en la sección 1.2, Problemas clave). Por lo tanto, las pruebas pueden revelar fallas, pero son las fallas las que pueden

y debe eliminarse [3]. El término más genérico

defecto se puede utilizar para referirse a una falla o un

fracaso, cuando la distinción no es importante [3].

Sin embargo, debe reconocerse que la causa

de una falla no siempre se puede identificar de manera inequívoca. No existen criterios teóricos para definitivamente

determinar, en general, la avería que provocó una

fallo observado. Podría decirse que fue el

falla que tuvo que ser modificada para eliminar la falla,

pero otras modificaciones podrían haber funcionado simplemente

también. Para evitar la ambigüedad, uno podría referirse a

entradas que causan fallas en lugar de fallas, es decir,

2. Niveles de prueba

Las pruebas de software generalmente se realizan en diferentes niveles a lo largo de los procesos de desarrollo y mantenimiento. Se pueden distinguir niveles

basado en el objeto de prueba, que se llama

el objetivo, o en el propósito, que se llama el

objetivo (del nivel de prueba).

2.1. El objetivo de la prueba

El objetivo de la prueba puede variar: un solo módulo, un

grupo de dichos módulos (relacionados por propósito, uso,

comportamiento o estructura), o un sistema completo. Tres

Las etapas de prueba se pueden distinguir: unidad, integración y sistema. Estas tres etapas de prueba no

implican ningún modelo de proceso, ni ninguno de ellos

se supone que es más importante que los otros dos.

2.1.1. Examen de la unidad

La prueba unitaria verifica el funcionamiento de forma aislada

de elementos de software que se pueden probar por separado.

Dependiendo del contexto, estos podrían ser los

subprogramas individuales o un componente más grande

hecho de unidades altamente cohesivas. Normalmente, unidad

la prueba ocurre con el acceso al código que se está probando

y con el apoyo de herramientas de depuración. Los programadores que escribieron el código normalmente, pero no

siempre, realice pruebas unitarias.

2.1.2. Pruebas de integración

La prueba de integración es el proceso de verificar la

interacciones entre componentes de software. Las estrategias de prueba de integración clásicas, como de arriba hacia abajo y de abajo hacia arriba, se utilizan a menudo con software estructurado jerárquicamente.

Las estrategias de integración modernas y sistemáticas son

típicamente impulsado por la arquitectura, lo que implica

integrando de forma incremental los componentes de software o subsistemas basados en identificados

hilos funcionales. Las pruebas de integración suelen ser una

actividad continua en cada etapa del desarrollo

durante el cual los ingenieros de software se abstraen

perspectivas de nivel inferior y concentrarse en el

perspectivas del nivel en el que se están integrando. Para otro software que no sea pequeño y simple,

Las estrategias de prueba de integración incremental generalmente se prefieren a colocar todos los componentes

juntos a la vez, lo que a menudo se denomina "gran

bang ”prueba.

2.1.3. Prueba del sistema

La prueba del sistema se ocupa de probar el

comportamiento de todo un sistema. Unidad efectiva y

Las pruebas de integración habrán identificado muchos de

los defectos del software. Las pruebas del sistema suelen

considerados apropiados para evaluar los requisitos del sistema no funcionales, como seguridad, velocidad, precisión y confiabilidad (consulte Requisitos funcionales y no funcionales en el

Requisitos de software KA y requisitos de calidad de software en la calidad de software KA).

Interfaces externas a otras aplicaciones, utilidades,

dispositivos de hardware o los entornos operativos

también suelen evaluarse en este nivel.

aquellos conjuntos de entradas que provocan una falla en aparecer

CAPÍTULO 5 MANTENIMIENTO DEL SOFTWARE

Fundamentos de mantenimiento de software

Esta primera sección presenta los conceptos y

terminología que forma una base subyacente para

comprender el papel y el alcance del software

mantenimiento. Los temas proporcionan definiciones y

enfatice por qué es necesario el mantenimiento.

Las categorías de mantenimiento de software son fundamentales para

comprender su significado subyacente

Necesidad de mantenimiento

Se necesita mantenimiento para asegurar que el software

sigue satisfaciendo los requisitos del usuario. El mantenimiento es aplicable al software que se desarrolla

utilizando cualquier modelo de ciclo de vida del software (por ejemplo,

espiral o lineal). Los productos de software cambian debido

a acciones de software correctivas y no correctivas.

Se debe realizar el mantenimiento para

• corregir fallas;

• mejorar el diseño;

• implementar mejoras;

• interfaz con otro software;

• adaptar programas para que diferentes hardware,

se pueden utilizar software, funciones del sistema e instalaciones de telecomunicaciones;

• migrar software heredado; y

• retirar software.

Cinco características clave comprenden las actividades del mantenedor:

• mantener el control sobre las funciones diarias del software;

• mantener el control sobre el software

modificación;

• perfeccionar las funciones existentes;

• identificar amenazas de seguridad y corregir vulnerabilidades de seguridad; y

• evitar que el rendimiento del software

degradando a niveles inaceptables

CAPÍTULO 6 GESTIÓN DE CONFIGURACIÓN DE SOFTWARE

Gestión del proceso SCM

SCM controla la evolución e integridad de un

producto identificando sus elementos; gestión y

controlar el cambio; y verificar, registrar y

informar sobre la información de configuración. Desde el

perspectiva del ingeniero de software, SCM facilita

actividades de desarrollo e implementación de cambios. Una implementación exitosa de SCM requiere

planificación y gestión cuidadosas. Esto a su vez,

requiere una comprensión de la organización

contexto para, y las restricciones impuestas en, el

diseño e implementación del proceso SCM.

Restricciones y orientación para el SCM

Proceso

Restricciones que afectan al SCM y orientación para él

proceso provienen de varias fuentes. Políticas y procedimientos establecidos a nivel corporativo u otro

los niveles organizacionales pueden influir o prescribir

el diseño e implementación del proceso SCM para un proyecto dado. Además, el contrato

entre el adquirente y el proveedor puede contener disposiciones que afecten al proceso de SCM. por

ejemplo, ciertas auditorías de configuración pueden ser

requerido, o podría especificarse que ciertos elementos

colocarse debajo de CM. Cuando los productos de software

desarrollarse tienen el potencial de afectar al público

seguridad, los organismos reguladores externos pueden imponer

limitaciones. Finalmente, la vida particular del software

proceso de ciclo elegido para un proyecto de software y

el nivel de formalismo seleccionado para implementar el

El software afecta el diseño y la implementación de

el proceso SCM.

Orientación para diseñar e implementar un

El proceso SCM también se puede obtener de "best

práctica ", como se refleja en los estándares de software

Selección e implementación de herramientas

Como en cualquier área de la ingeniería de software, la

selección e implementación de herramientas SCM

debe planificarse cuidadosamente. Deben considerarse las siguientes preguntas:

• Organización: ¿qué motiva la adquisición de herramientas desde una perspectiva organizacional?

• Herramientas: ¿podemos utilizar herramientas comerciales o

desarrollarlos nosotros mismos?

• Medio ambiente: cuáles son las limitaciones

impuesta por la organización y su contexto técnico?

• Legado: ¿cómo utilizarán (o no) los proyectos el

nuevas herramientas?

• Financiamiento: quién pagará las herramientas

adquisición, mantenimiento, formación y

personalización?

• Alcance: cómo se implementarán las nuevas herramientas—

por ejemplo, a través de toda la organización

o solo en proyectos específicos?

• Propiedad: ¿quién es responsable de la introducción de nuevas herramientas

CAPÍTULO 7 GESTIÓN DE INGENIERÍA DE SOFTWARE

Otros aspectos de la gestión organizacional ejercen un impacto en la ingeniería de software (por ejemplo, políticas y procedimientos organizacionales que proporcionan el marco en el que se llevan a cabo los proyectos de ingeniería de software). Además, es posible que sea necesario implementar o establecer una serie de políticas específicas para la ingeniería de software para una gestión eficaz de la ingeniería de software a nivel organizacional.

Por ejemplo, las políticas suelen ser necesarias para establecer procesos o procedimientos específicos en toda la organización para tareas de ingeniería de software, como diseño de software, construcción de software, estimación, seguimiento y generación de informes. Dichas políticas son importantes para la gestión eficaz a largo plazo de los proyectos de ingeniería de software en una organización.

El personal de ingeniería de software puede presentar capacitación o personal único desafíos de gestión (por ejemplo, mantener moneda en un contexto donde la tecnología subyacente sufre cambios rápidos y continuos). La reutilización eficaz requiere visión estratégica que refleje las ventajas y desventajas de la reutilización.

Además de comprender los aspectos de gestión que están influenciados exclusivamente por proyectos de software, los ingenieros de software deben tener algún conocimiento de los aspectos más generales de manejo que se discuten en este KA (incluso en los primeros años después de la graduación).

También se proporciona información adicional en el otras referencias y lecturas adicionales para este KA. Esta gestión de ingeniería de software KA Consiste en los procesos de gestión de proyectos de software en los primeros cinco temas (Inicio y definición del alcance, Planificación del proyecto de software, Implementación, revisión y Evaluación, Cierre), más Ingeniería de Software Medición en el sexto tema y software.

Herramientas de gestión de ingeniería en el séptimo tema. Si bien la gestión de proyectos y la gestión de mediciones a menudo se consideran separados, y de hecho cada uno posee muchos atributos únicos, la estrecha relación ha llevado a tratamiento combinado en este KA.

un proceso de software también puede incluir sus criterios de entrada y salida y descomposición de las actividades laborales en tareas, que son las unidades de trabajo más pequeñas sujetas a gestión responsabilidad. Una entrada de proceso puede ser un evento desencadenante o la salida de otro proceso. Entrada Los criterios deben satisfacerse antes de que un proceso pueda comenzar.

Un proceso de software puede incluir subprocesos.

Por ejemplo, la validación de requisitos de software es un proceso utilizado para determinar si los requisitos proporcionarán una base adecuada para el software desarrollo; es un subproceso del software proceso de requisitos. Las entradas para la validación de requisitos suelen ser una especificación de requisitos de software y los recursos necesarios para realizar la validación (personal, herramientas de validación, tiempo suficiente).

Estas tareas implican trabajo asignaciones para individuos y equipos. La salida de la validación de requisitos es típicamente una validación especificación de requisitos de software que proporciona Entradas al proceso de diseño y prueba de software.

2. Ciclos de vida del software

un ciclo de vida de desarrollo de software (SDLC) incluye los procesos de software utilizados para especificar y transformar los requisitos de software en un producto de software entregable. La vida de un producto de software ciclo (SPLC) incluye un desarrollo de software ciclo de vida más procesos de software adicionales que proporcionar implementación, mantenimiento, soporte, evolución, retiro y todos los demás procesos de inicio a retiro de un producto de software, incluida la gestión de la configuración del software y procesos de aseguramiento de la calidad del software que son aplicado a lo largo del ciclo de vida de un producto de software.

La salida de un El proceso de software a menudo proporciona la entrada para otros (por ejemplo, los requisitos de software proporcionan entrada para un proceso de diseño arquitectónico de software y el construcción de software y procesos de prueba de software). Ejecución concurrente de varios programas

Procesos de software además de los enumerados anteriores incluyen lo siguiente.

Los procesos de gestión de proyectos incluyen procesos de planificación y estimación, recursos gestión, medición y control, liderazgo, gestionar el riesgo, gestionar las partes interesadas y coordinar el principal, de apoyo, organizativo, y procesos cruzados de proyectos de desarrollo y mantenimiento de software.

Por ejemplo, las preocupaciones de seguridad durante el desarrollo de software pueden

Necesita uno o más procesos de software para proteger la seguridad del entorno de desarrollo y reducir el riesgo de actos maliciosos. Los procesos de software también se pueden desarrollar para proporcionar motivos adecuados para establecer la confianza en la integridad del software

CAPÍTULO 9 MODELOS DE INGENIERÍA DE SOFTWARE Y MÉTODOS

Un modelo es una abstracción o simplificación de un componente de software. Una consecuencia de usar. La abstracción es que ninguna abstracción describe completamente un componente de software

Esta simplificación conduce a un conjunto de supuestos sobre el contexto en el que se desarrolla el modelo colocado que también debe ser capturado en el modelo. Los modelos se construyen para representar el mundo real objetos y sus comportamientos para responder preguntas sobre cómo se espera el software para operar. Interrogando a los modelos, ya sea mediante exploración, simulación o revisión, puede exponer áreas de incertidumbre dentro del modelo y el software al que se refiere el modelo.

Una entidad puede representar artefactos concretos (por ejemplo, procesadores, sensores o robots) o artefactos abstractos (por ejemplo, módulos de software o protocolos de comunicación). Las entidades modelo están conectadas a otras entidades mediante relaciones (en otras palabras, líneas u operadores textuales en entidades objetivo).

El significado de una entidad puede estar representado por su forma, atributos textuales o ambos. Generalmente, la información textual se adhiere a estructura sintáctica específica del lenguaje.

Para los lenguajes textuales, la sintaxis está definida utilizando una gramática de notación que define construcciones de lenguaje válidas (por ejemplo, Backus-Naur Formulario (BNF)). Para lenguajes gráficos, la sintaxis es definidos mediante modelos gráficos denominados meta modelos. Al igual que con BNF, los meta modelos definen las construcciones sintácticas de un modelo gráfico idioma; el meta modelo define cómo se pueden componer estos constructos para producir modelos válido

El análisis de interacción se centra en las comunicaciones o las relaciones de flujo de control entre entidades. utilizado para realizar una tarea o función específica dentro del modelo de software. También puede ser importante que algunas aplicaciones de software examinen las interacciones entre la aplicación de software y la interfaz de usuario. software.

Capitulo 10: software quality 10-1

La creciente preocupación por la calidad en la industria del software tiene como objetivo principal el desarrollo sistemático de productos y servicios de mejor calidad y el cumplimiento de las necesidades y expectativas de los clientes. En el presente artículo se hace una introducción a la calidad y al modelo de calidad adoptado por Colciencias, CMMI. Pretendemos unir esfuerzos con esta iniciativa y motivar a la comunidad académica a trabajar en calidad con las empresas desarrolladoras de software para mejorar la competitividad y la calidad global de esta industria.

PALABRAS CLAVES: Calidad, calidad de software, industria de software, CMMI, niveles CMMI.

ABSTRACT The growing concern for quality in the software industry have as its main objective the systematic development of products and services of better quality and fulfilling the needs and expectations from customers. This article provides an introduction to quality and the quality model adopted by Colciencias, CMMI. We seek to join forces with this initiative and motivate the academic community to work out quality in software development companies to improve competitiveness and overall quality of this industry.

KEYWORDS: Quality, software quality, software industry, CMMI, CMMI levels

3 CALIDAD DE SOFTWARE En la industria del software se pueden evidenciar necesidades de satisfacción del cliente de productos o servicios de software, de reducción de recursos invertidos en proyectos de software y de la efectiva asignación de recursos humanos. Si hablamos de la calidad del software, una de las primeras definiciones aseguraba que “la calidad de un programa o sistema se evaluaba de acuerdo al número de defectos por cada mil líneas de código. (KLOC: Kilo Lines Of Code)”. 5 La definición de la calidad del software según la IEEE, Std. 610-1990, es “el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”6 .

capitulo 11: software profesional de la ingenieria practica

La versión corta del código resume las aspiraciones a un alto nivel de abstracción; las cláusulas que se incluyen en la versión completa proporcionan ejemplos y detalles acerca de cómo estas aspiraciones modifican nuestra manera de actuar como profesionales de la ingeniería de software. Sin las aspiraciones los detalles pueden convertirse en tediosos y legalistas; sin los detalles las aspiraciones pueden convertirse en altisonantes pero vacías; juntas, las aspiraciones y los detalles forman un código cohesivo. Los ingenieros de software deberán comprometerse a convertir el análisis, especificación, diseño, implementación, pruebas y mantenimiento de software en una profesión respetada y benéfica. De acuerdo a su compromiso con la salud, seguridad y bienestar social, los ingenieros de software deberán sujetarse a los ocho principios siguientes:

1. Sociedad. Los ingenieros de software actuarán en forma congruente con el interés social.
2. Cliente. y empresario. Los ingenieros de software actuarán de manera que se concilien los mejores intereses de sus clientes y empresarios, congruentemente con el interés social.
3. Producto. Los ingenieros de software asegurarán que sus productos y modificaciones correspondientes cumplen los estándares profesionales más altos posibles.
4. Juicio. Los ingenieros de software mantendrán integridad e independencia en su juicio profesional.
5. Administración. Los ingenieros de software gerentes y líderes promoverán y se suscribirán a un enfoque ético en la administración del desarrollo y mantenimiento de software.
6. Profesión. Los ingenieros de software incrementarán la integridad y reputación de la profesión congruentemente con el interés social.
7. Colegas. Los ingenieros de software apoyarán y serán justos con sus colegas.
8. Personal. Los ingenieros de software participarán toda su vida en el aprendizaje relacionado con la práctica de su profesión y promoverán un enfoque ético en la práctica de la profesión.

Las computadoras tienen un papel central cada vez mayor en el comercio, industria, gobierno, medicina, educación, entretenimiento, y sociedad. Los ingenieros de software son aquellos que contribuyen, mediante la participación directa o enseñanza, al análisis, especificación, diseño, desarrollo, certificación, mantenimiento y pruebas de sistemas de software. Debido a sus funciones en el desarrollo de sistemas de software, los ingenieros de software tienen suficientes oportunidades para causar beneficio o generar daño y para habilitar o influenciar a otros a causar daño o beneficio. Para asegurar, en la medida de lo posible, que sus esfuerzos se utilizarán para hacer el bien, los ingenieros de software deben comprometerse a hacer de la ingeniería del software una profesión benéfica y respetada.

capitulo 12: Software ingenieria Economicos

El software, como bien de capital
El software es un bien de capital, y algo más Todos los bienes de capital operan dentro de una vasta y evolutiva estructura de interrelaciones, la estructura de producción Los bienes de capital son conocimiento empaquetado acerca de cómo realizar algún tipo de producción El conocimiento que hay que empaquetar es disperso, incompleto, cambiante, en gran parte tácito y crecientemente complejo El proceso de desarrollo de nuevos bienes de capital (Sw) es un proceso de aprendizaje social
Complejidad del software
Efectos de la escala en la ingeniería del software Aspectos y áreas involucrados Modelo CMM Código ACM/IEEE-CS de ética y práctica profesional
Gestión del conocimiento
Qué se entiende por conocimiento en la empresa Qué es la gestión del conocimiento El conocimiento como activo. Capital intelectual El conocimiento como ventaja competitiva El software, componente del capital intelectual Un ejemplo concreto: Los Factores Claves del Éxito (de BMBS)
Peopleware: El Individuo
Indicadores de personalidad: MBTI La inteligencia, según la teoría factorialista Nuevos conceptos de inteligencia Inteligencia emocional Apuntes breves sobre el "hardware" de la inteligencia Anexo: Amigdala Training
Desde un punto de vista económico, el software es un medio de producción englobado dentro del capital intelectual de la empresa. Como cualquier otro medio de producción, en su fase de uso puede verse como conocimiento empaquetado. Su fase de desarrollo, consistente en reunir y empaquetar en la forma adecuada los conocimientos necesarios para realizar algún tipo de producción, se ha transformado, debido a las características específicas del software, a la complejidad creciente de los objetivos de producción y a la naturaleza muy evolutiva de los conocimientos necesarios, en un sofisticado proceso de aprendizaje social, sometido por ende a las reglas de productividad y competencia económicas. De forma general, la producción y el uso (precedido de la contratación) de software forman parte del amplio campo de lo que ahora se llama "gestión del conocimiento"

CAPÍTULO 13 FUNDAMENTOS DE COMPUTADORA

La resolución de problemas se refiere al pensamiento y las actividades realizadas para responder o derivar una solución a un problema.

Por ejemplo, la ingeniería de software se enfoca en resolver problemas usando computadoras y software. Si bien diferentes problemas justifican diferentes soluciones y pueden requerir diferentes herramientas y procesos, la metodología y las técnicas utilizadas en la resolución de problemas, siga algunas pautas y a menudo se puede generalizar como resolución de problemas Técnicas

• Formule el problema real.

• Analice el problema.

• Diseñar una estrategia de búsqueda de soluciones

Cuatro tipos de análisis incluyen análisis de situación, en el que los aspectos más urgentes o críticos de un la situación se identifica primero; análisis de problemas, en cuál debe determinarse la causa del problema; análisis de decisiones, en el que la (s) acción (es) necesario para corregir el problema o eliminar su debe determinarse la causa; y problema potencial análisis, en el que las acciones necesarias para prevenir debe determinarse cualquier reaparición del problema o el desarrollo de nuevos problemas.

La programación implica diseño, escritura, pruebas, depuración y mantenimiento. El diseño es la concepción o invención de un esquema para convertir un requisito del cliente de software de computadora en software operativo.

Pruebas es la actividad para verificar que el código que uno escribe realmente hace lo que se supone que debe hacer. La depuración es la actividad para encontrar y corregir errores (fallas) en el código fuente (o diseño). El mantenimiento es el actividad para actualizar, corregir y mejorar los programas.

CAPÍTULO 14 FUNDAMENTOS MATEMÁTICOS

INTRODUCCIÓN

Los profesionales del software conviven con los programas. en un lenguaje muy simple, se puede programar solo para algo que sigue una lógica bien entendida y no ambigua. Los fundamentos matemáticos el 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. Las matemáticas que son el enfoque principal en este KA son bastante diferente de la aritmética típica, donde los números son tratados y discutidos. La lógica y el razonamiento son la esencia de las matemáticas que un software ingeniero debe abordar.

Las matemáticas, en cierto sentido, son el estudio de sistemas. La palabra "formal" se asocia con precisión, por lo que no puede haber ninguna ambigüedad o interpretación errónea del hecho. Las matemáticas son, por tanto, el estudio de todos y cada uno de los verdades sobre cualquier concepto. Este concepto puede ser sobre números, así como sobre símbolos, imágenes, sonidos, video, casi cualquier cosa. En resumen, no solo los números y las ecuaciones numéricas están sujetos a precisión. Al contrario, un software ingeniero necesita tener una abstracción precisa en un dominio de aplicación diverso.

1. Conjunto, relaciones, funciones

[1 *, c2]

Conjunto. Un conjunto es una colección de objetos, llamados elementos. del set. Un conjunto se puede representar enumerando sus elementos entre llaves, por ejemplo, S = {1, 2, 3}.

El símbolo ∈ se usa para expresar que un elemento pertenece a un conjunto o, en otras palabras, es un miembro del conjunto. Su negación está representada por ∉, por ejemplo, 1 ∈ S, pero 4 ∉ S.

En una representación más compacta del conjunto usando establecer la notación del constructor, {x | P (x)} es el conjunto de todo x tal que P (x) para cualquier proposición P (x) sobre cualquier universo del discurso. Los ejemplos de algunos conjuntos importantes incluyen los siguientes:

N = {0, 1, 2, 3,…} = el conjunto de no negativos enteros.

Z = {…, −3, −2, −1, 0, 1, 2, 3,…} = el conjunto de

enteros.
Conjunto finito e infinito. Un conjunto con un número finito de elementos se denomina conjunto finito. Por el contrario,
cualquier conjunto que no tenga un número finito de elementos es un conjunto infinito. El conjunto de todo natural
números, por ejemplo, es un conjunto infinito.
Lógica proposicional
Una proposición es una declaración que es verdadera
o falso, pero no ambos. Consideremos declarativo
oraciones para las que es significativo asignar
cualquiera de los dos valores de estado: verdadero o falso. Algunos
A continuación se dan ejemplos de proposiciones.
1.El sol es una estrella
2.Los elefantes son mamíferos.
3. 2 + 3 = 5.
Sin embargo, a + 3 = b no es una proposición, ya que es
ni verdadero ni falso. Depende de los valores de
las variables ay b.
La ley del medio excluido: para cada proposición p, p es verdadera op es falsa.
La ley de la contradicción: para cada proposición p, no se da el caso de que p sea tanto verdadera como falsa.
La lógica proposicional es el área de la lógica que
se ocupa de las proposiciones. Se muestra una tabla de verdad
la relación.
Un sistema informático puede abstraerse como un mapeo de estado a estado impulsado por entradas. En otra
palabras, un sistema puede considerarse como una transición
función T: S × I → S × O, donde S es el conjunto de
estados e I, O son las funciones de entrada y salida.
Si el conjunto de estados S es finito (no infinito), el sistema se denomina máquina de estados finitos (FSM).
Alternativamente, una máquina de estados finitos (FSM) es un
abstracción matemática compuesta de un finito
número de estados y transiciones entre esos
estados. Si el dominio S × I es razonablemente pequeño,
entonces uno puede especificar T explícitamente usando diagramas
similar a un diagrama de flujo para ilustrar la forma en que la lógica
flujos para diferentes entradas. Sin embargo, esto es práctico solo para máquinas que tienen un tamaño muy pequeño.
capacidad de información.
Un FSM tiene una memoria interna finita, una entrada
característica que lee símbolos en una secuencia y una
a la vez y una función de salida.

El funcionamiento de un FSM comienza desde un principio
estado, pasa por transiciones dependiendo de la entrada
a diferentes estados y puede terminar en cualquier estado válido.
Sin embargo, solo unos pocos de todos los estados marcan un flujo de operación exitoso.
Estos se llaman aceptar
estados

CAPITULA 15 FUNDAMENTOS DE INGENIERÍA

Un método de ingeniería para la resolución de problemas implica proponer soluciones o modelos de solo ciones y luego realizar experimentos de pruebas para estudiar las soluciones propuestas como modelo. Esto, los ingenieros deben entender borw para crear un conocimiento y analizar los resultados del experimento es orden. Evaluar la solución popneod Las millas empíricas y las técnicas experimentales ayudan a los pobres a describir y comprender la van bilidad de sus reservas, a identificar A medida que la teoría y la práctica de la ingeniería de software cambian, es cada vez más evidente que la ingeniería de asombro es un disco de ingeniería para todas las disciplinas de la ingeniería. El más ligero es un modelo de perros. prototipos y sllatina ndarde, und vot co atysis. Arelictios de todos aparecen A fin de llevar a cabo sus respeesibilidades, engi. nors mint comprende cómo varían las características de los diferentes productos y procesos. Los ingenieros a menudo se encuentran con situaciones en las que la relación entre diferentes variables debe establecerse. Por lo tanto, los ingenieros deben desarrollar una comprensión adaptada de las técnicas estáticas para recopilar datos confiables en términos de muestreo y análisis para llegar a resultados que puedan generalizarse. Estas técnicas se analizan a continuación.

Población. El conjunto de todos los encuestados o elementos ta (posibles unidades de muestreo) que se van a estudiar forma la población en. Como ejemplo, considérese el caso de un estudio de la usabilidad percibida de un producto de software. En este caso, el conjunto de todos los posibles usuarios de la población Al definir la población, se debe tener cuidado para comprender el estudio y la población objetivo. Hay casos en que la población de estudios y la población.

Las mediciones pueden ser físicas, ambientales, económicas, operativas o de algún otro tipo de medición que sea significativa para el proyecto en particular. Esta sección explora la teoría de la medición y cómo es fundamental para la ingeniería.

La medición comienza como una conceptualización y luego pasa de conceptos abstractos a definiciones del método de medición y a la aplicación real de ese método para obtener un resultado de medición. Cada uno de estos pasos debe entenderse, comunicarse y emplearse correctamente para generar datos utilizables.