

MANUAL DE APRENDIZAJE



CÓDIGO: 89001708 Profesional Técnico



|  |
| --- |
| AUTORIZACIÓN Y DIFUSIÓN |

MATERIAL DIDÁCTICO ESCRITO

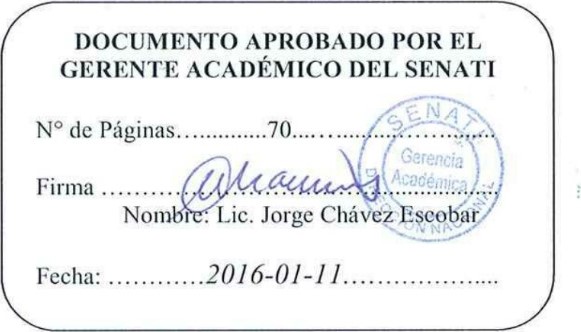
FAM. OCUPACIONAL  COMPUTACIÓN E INFORMÁTICA

CARRERA  DESARROLLO DE SOFTWARE

NIVEL  PROFESIONAL TÉCNICO

Con la finalidad de facilitar el aprendizaje en el desarrollo de la formación y capacitación en la Carrera de DESARROLLO DE SOFTWARE a nivel nacional y dejando la posibilidad de un mejoramiento y actualización permanente, se autoriza la APLICACIÓN Y DIFUSIÓN de material didáctico escrito referido a CALIDAD DE SOFTWARE.

Los Directores Zonales y Jefes de Centro de Formación Profesional son los responsables de su difusión y aplicación oportuna.



Registro de derecho de autor:



íNDlCE

|  |  |  |
| --- | --- | --- |
| TAREA | CONTENIDO | N O PÁG. |
|  | PROCESOS DE SOFTWARE | 6 |
|  | 1.1. Actividad Práctica 1. | 6 |
|  | 1.2. Actividad Práctica 2. | 8 |
|  | 1.3. Evolución y Desarrollo de Software. | 10 |
|  | 1.4. Procesos de Software | 12 |
|  | 1.5. IMCM | 14 |
|  | 1.6. Patrones del Proceso de Software. | 15 |
|  | 1.7. Modelo de Proceso de Software Personal-PSP. | 15 |
|  | 1.8. Modelo de Proceso en Equipo-TSP. | 17 |
| ll. | CALIDAD DE SOFTWARE | 21 |
|  | 2.1. Actividad Práctica 3. | 21 |
|  | 2.2. Actividad Práctica 4. | 22 |
|  | 2.3. Calidad de Software. | 26 |
|  | 2.4. Modelos para Revisiones Técnicas. | 30 |
| III. | MODELOS DE SOFTWARE. | 33 |
|  | 3.1. Actividad Práctica 5. | 33 |
|  | 3.2. Actividad Práctica 6. | 34 |
|  | 3.3. Actividad Práctica 7. | 37 |
|  | 3.4. Modelos de Software. | 38 |
| IV. | MÉTRICAS DE SOFTWARE. | 45 |
|  | 4.1. Actividad Práctica 8. | 45 |
|  | 4.2. Actividad Práctica 9. | 47 |
|  | 4.3. Métricas de Software. | 48 |
|  | INTEGRACIÓN DE MODELOS DE MADUREZ DE CAPACIDADES-CMMI. | 57 |
|  | 5.1. Actividad Práctica 10. | 57 |
|  | 5.2. Actividad Práctica 11. | 59 |
|  | 5.3. CMMl-lntegración del Modelo de Madurez de Capacidades. | 61 |

PROCESOS DE SOFTWARE.

OBJETIVO GENERAL.

Utilizar los conceptos relacionados a los procesos de software para efectuar pruebas que permitan comprobar la integridad de diversas aplicaciones.

OBJETIVOS ESPECíFlCOS.

* Utilizar el marco de trabajo para un proceso de software.
* Utilizar IMCM integrándolo a sus marcos de trabajo en sus procesos de software.
* Utilizar patrones durante un proceso de software.
* Evaluar el modelo de proceso personal.
* Evaluar el modelo de proceso en equipo.

1.1. ACTIVIDAD PRÁCTICA 1.

En esta primera actividad se instalará una aplicación de software libre que permite un entorno de pruebas para aplicaciones basadas en la web. El software a instalar es el Selenium, debido a que permite su ejecución en los distintos navegadores de Internet y su instalación en diferentes sistemas operativos avanzados como LINUX y Windows.

.

1.2. ACTIVIDAD PRÁCTICA 2.

Funcionamiento y exploración del software Selenium.

Paso 1. Verificar que la barra de Menú del Mozilla Firefox se encuentre activa, de lo contrario activarla, para activarla efectuar:

Click derecho sobre la barra de título del explorador Mozilla y activar la casilla de verificación "Barra de menú"

Paso 2. Para poder visualizar Selenium IDE hay que activarlo desde la opción:

* Ver Barra lateral Selenium IDE
* Crtl+Mayus+S

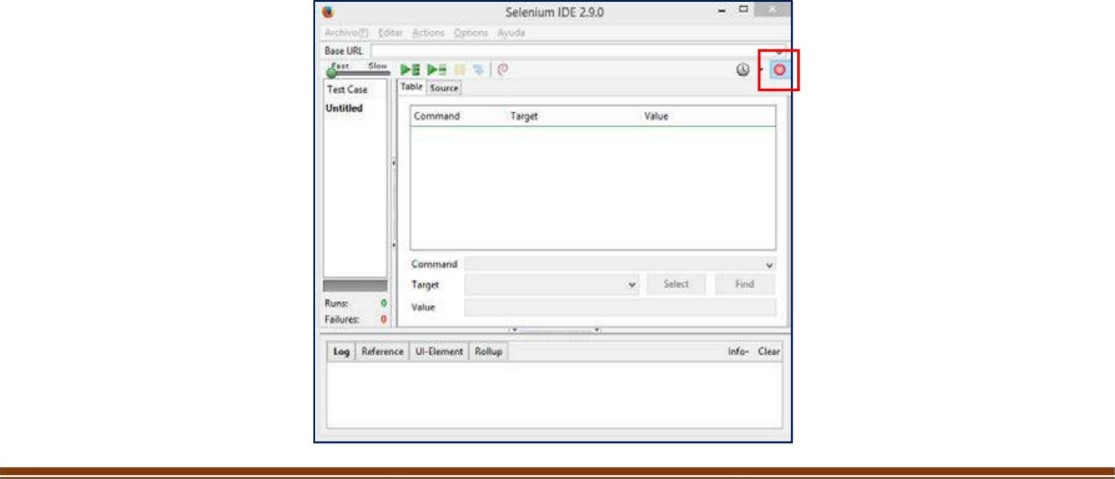
También desde:

* Herramientas -> Selenium IDE



Paso 3. En la barra de marcadores apareceré el botón que identifica que tenemos el Selenium IDE activo. Se muestra dentro del recuadro color rojo en la siguiente figura:

Paso 4. Verificar que se muestre la ventana del entorno de trabajo del software Selenium IDE, con el botón en rojo, como se muestra en el recuadro color azul de la figura:



Paso 5. Manejo de opciones del Selenium. Se muestra el detalle de las opciones generales del plugin Selenium (Barra principal del plugin):



Paso 6. Explorar diversas opciones del software a partir de las siguientes descripciones:

* Archivo (E).

|  |  |  |
| --- | --- | --- |
| Detalle |  | Descripción |
| Archivo(E) Editar Actions Opti |  | New Test Case: Generación de un caso de prueba.  Open...: Abrir caso de prueba.  Save Test Case: Guardar caso de prueba. Save Test Case As...: Guardar caso de prueba como se indique.  Export Test Case As...: Exportar caso de prueba en formato de lenguaje de programación Selenium RC.  Recent Test Cases: Casos de prueba usados recientemente.  New Test Suite: Generación de un juego de pruebas.  Open Test Suite...: Abrir juego de pruebas.  Save Test Suite: Guardar juego de pruebas. Save Test Suite As...: Guardar juego de pruebas como se indique  Export Test Suite: Exportar caso de prueba.  Recent Test Suite: Juegos de prueba usados recientemente. |
| New Test Case Ctrl+N  Qpen.„ Ctrl+O  Save Test Case CtrI+S  Save Test Case As...  Export Test Case As...  Recent Test Cases Add Test Case... Ctrl+D    New Test Suite Open Test Suite.„  Save Test Suite  Save Test Suite As.„ aport Test Suite As„.  Recent Test Suites    Cerrar(X) Ctrl+W |
|  |

* Edición. En esta parte el software (Plugin Selenium) tiene acciones típicas de edición.
* Options. Provee las siguientes prestancias:
* Codificación del script (Por ejemplo: UTF-8).
* Valor por defecto del timeout en la grabación.
* Extensiones Selenium Core.
* Extensiones Selenium IDE.
* Recordar URL base.
* Grabar assertTitle automáticamente.
* Grabar URL absoluta.
* Opciones de formato.
* Format: es el formato en el cual se realizara la grabación del lenguaje script del software Selenium.
* Clipboard Format: es el formato de grabación en el portapapeles del lenguaje script del software Selenium.

|  |  |
| --- | --- |
|  | Reproducir Prueba |
|  | Reproducir Prueba |
|  | Parar prueba |
|  | ejecución paso a paso |
|  | Reproducir con Selenium  TestRunner |
|  |  |
|  | Grabar Prueba |
| slo•w | Velocidad de ejecucción de la prueba |

Paso 6. Identificación de los detalles de los distintos botones e iconos representativos del plugin Selenium. Las representaciones son las que comúnmente se encuentran en este tipo de aplicaciones, con opciones de reproducción, pausa, entre otras. Los detalles se muestran en la figura de la derecha.

Paso 7. Reconocimiento del panel de comandos y panel de códigos, que contiene las siguientes opciones:

* Base URL: Aquí se coloca la dirección de la página web a la que se efectuara el test.
* Panel de comandos y panel de códigos.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test Case | |  |  |  | | --- | --- | --- | | Command | Target | Value | |  |  |  |   Command  Select Find  Value |
| Failures: |

* Panel de información:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Log Reference Ul-EIement Rollup | Info• Clear | |  |  | |

1.3. EVOLUCIÓN Y DESARROLLO DEL SOFTWARE.

Los primeros pasos en software se inician en el año de 1950, casi inmediatamente después del término de la Segunda Guerra Mundial, teniendo como escenario el ámbito militar de los Estados Unidos de Norteamérica.

Se requería en ese entonces contar con programas que permitan el control de la navegación área y sistemas de control de tiro ante posibles ataques aéreos de la URSS (bloque soviético) hacia USA, época de inicios de la "Guerra Fría". Por esos años el desarrollo de software, los programas y aplicaciones eran orientados y específicos, es decir, estaban definidos para un determinado

hardware, en otras palabras, el software formaba parte de los sistemas electrónicos diseñados.

Una vez terminados estos sistemas pasaban por una serie de pruebas muy rigurosas que permitían afianzar sus beneficios y asegurar su performances (calidad). Las empresas que más contribuyeron en esa época son DEC (Digital Equipment Corporation) e IBM (International Business Machines). Cabe resaltar que, no existen datos oficiales acerca de las fallas detectadas y lo errores cometidos o daños ocasionados por software en esa época.

Una de las características diferenciadas del software implantado en los sistemas militares de la época era la estabilidad del producto y el cumplimiento de requerimientos a partir de las exigencias del cliente, lo cual diferencia de manera importante de los desarrollados actualmente.

Con el vertiginoso avance tecnológico de la electrónica (hardware) y la aparición de novedosos lenguajes de programación en alto nivel, se originó una nueva directriz en el desarrollo de software, se comenzaron a desarrollar sistemas no militares y totalmente independientes del hardware. Un ejemplo de estos nuevos sistemas fueron los que hicieron su aparición en la década del 50 y 60, podemos citar a los primeros sistemas de reservación de pasajes aéreos para aplicaciones netamente comerciales.

Para la década siguiente, (60 — 70), los progresos fueron muy significativos y estuvieron llevados por grandes inversiones e investigaciones, tanto en universidades como en la industria. Todos encaminados a producir sistemas de propósito general. Estos avances, tanto en las plataformas físicas (hardware) y lógicas (lenguajes de programación) hicieron cada vez más sofisticado este desarrollo.



Para finales de la década del '70, las empresas de distintas partes del mundo desarrollaban productos que buscaban compatibilidad entre ellas, pero por

sobre todo con IBM, quien en ese entonces llevaba la delantera en avances tecnológicos. Se trabaja en estos desarrollos desde Inglaterra hasta Japón, con empresas importantes como RCA (Radio Corporation of América), GE (General Electric), Fujitsu, Hitachi entre otras.

En 1935, el británico Alan Turing propone la primera teoría sobre el software en su ensayo relacionado a números computables, con un programa cuyo fin era la toma de decisiones. Pero el término "software" recién fue utilizado en 1958 por el norteamericano John W. Tukey, quien diseño uno de los primeros algoritmos, "Algoritmo Cooley-Tukey".

En la actualidad debido al alto nivel de integración del hardware y a su bajo costo, las empresas requieren mayor cantidad de programadores que diseñadores de hardware, debido a que las herramientas y aplicaciones de software han logrado automatizar muchas funciones de hardware.

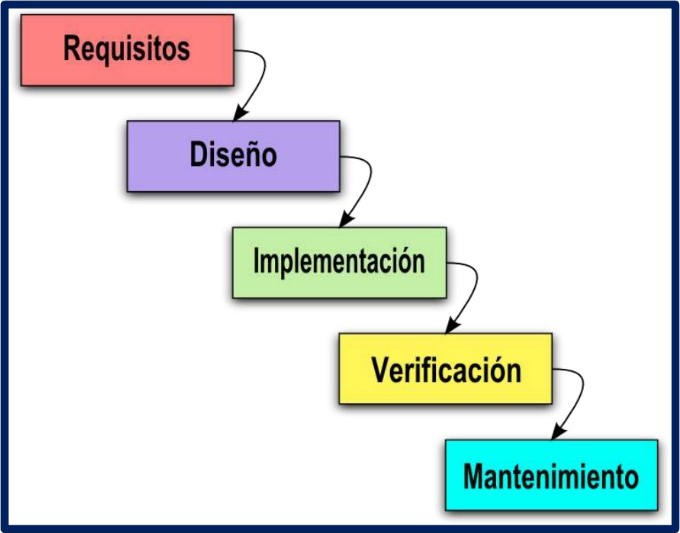
1.4. PROCESOS DE SOFTWARE.

El Proceso para el desarrollo de software, o también llamado "Ciclo de Vida del Desarrollo de Software" son los pasos a seguir para la ejecución de un proyecto de software. Existen diversos modelos que guían el desarrollo de software, cada uno con descripciones y enfoques particulares sobre las actividades a ejecutar en este proceso.

En la actualidad la mayoría de empresas utilizan diversas metodologías para el desarrollo de sus plataformas softwares, que no cumplen con los requisitos para una certificación, pero tambien existen compañías que se dedican al desarrollo de software aplicando metodologías normalizadas. La ISO (Organización Internacional de Estándares), promueve la Norma ISO 12201 para regular el método de selección, implementación y monitoreo del ciclo de vida de un producto o servicio basado en software.

Es una preocupación constante encontrar procesos reproducibles y predecibles que permitan obtener una mayor productividad y un nivel más alto de calidad. Todos con la intención de organizar el trabajo al desarrollar un software. Cabe resaltar que la amenaza en cuanto al desarrollo de software es constante y radica en problemas como: demora en la entrega de proyectos y exceso en el presupuesto asignado para su desarrollo.

El proceso para el desarrollo de software pasa por las siguientes actividades:



* Requisitos. Los requisitos son las funcionalidades y bondades que el usuario final espera del sistema. Estos requerimientos son recogidos del usuario final a través de entrevistas, para ser analizados y validar su incorporación el sistema a desarrollar. Se deberá documentar los requerimientos para ser utilizados como guía en la siguiente etapa.
* Diseño. Los detalles de los requerimientos de la fase anterior son estudiados en esta fase y el diseño del sistema es preparado. Esta fase permite especificar los requisitos del software y hardware, además ayuda a definir la arquitectura del sistema. Esta fase alimenta a la fase de implementación.
* Implementación. En esta etapa el diseño es dividido en módulos o unidades y para dar inicio al proceso de codificación. Primero se desarrollan partes de programas llamadas "unidades", las cuales se integraran en la fase de verificación.
* Verificación. Las unidades desarrolladas en la fase de implementación son probadas aquí, para posteriormente ser integradas en un sistema más complejo y se procede a verificar si todos los módulos o unidades se coordinan entre si y si su desempeño es de acuerdo a lo esperado a partir de los requerimientos del cliente, si es así queda listo para su entrega.
* Mantenimiento. Es la fase final del proceso y se encarga de los problemas posteriores al desarrollo del software, se evidencian durante su uso en producción por el cliente. La solución de problemas presentados en esta etapa forma parte de la fase de mantenimiento.

1.5. IMCM.

El IMCM es uno de los procesos de mejora cuyo objetivo es brindar apoyo las diversas organizaciones a potenciar su desempeño a través de procesos de software. IMCM es utilizada como guía en:

* La mejora de procesos durante la ejecución y desarrollo de proyectos.
* Gestión y administración de divisiones y departamentos de empresas en su totalidad.

El IMCM forma parte de la ingeniería de software y desarrollo organizacional de las empresas, permite a las organizaciones contar con elementos esenciales para lograr procesos más eficaces. IMCM es una marca comercial propiedad del Instituto de Ingeniería de Software de Carnegie Mellon University. El SEI (Instituto de Ingeniería de Software), en el 2008, a través de IMCM ayuda a:

* Integrar la organización de funciones separadas tradicionalmente.
* Establecer objetivos de mejora de procesos y prioridades.
* Proporcionar una guía para los procesos de calidad.
* Proporcionar un punto de referencia para evaluar los procesos actuales.

En la actualidad el IMCM concentra tres áreas de interés:

* Desarrollo de productos y servicios - IMCM para el Desarrollo (IMCM-DEV),
* Establecimiento de servicios, administración y entrega - IMCM para servicios

(IMCM-SVC), Y

* Adquisición de productos y servicios - IMCM para la adquisición (IMCMACQ).

El modelo IMCM busca que las empresas cumplan sus objetivos institucionales. Otra utilidad de los modelos IMCM es la evaluación de la madurez de los procesos en una organización. IMCM proviene de la ingeniería de software, pero en la actualidad, ha sido muy generalizada y abarca otras áreas de interés de las Tl, como:

* Desarrollo tecnológico de productos de hardware.
* Entrega de todo tipo de servicios en Tl,
* Adquisición de productos y servicios de Tl.

La principal ventaja de IMCM es que justifica su aporte como metodología eficaz que permite mejoras de gran impacto en procesos de desarrollo de productos software.

1.6. PATRONES DEL PROCESO DE SOFTWARE.

Un patrón de proceso de software ofrece una plantilla, es decir, un método consistente que se utiliza para describir una característica importante del proceso de software. Se pueden definir en cualquier nivel de abstracción, como:

* Un proceso completo.
* Una actividad del marco de trabajo importante.
* Una tarea dentro de una actividad del marco de trabajo.

Patrones:

* Patrones de Tarea. Son aquellos que definen una acción de software o una tarea relevante.
* Patrones de Escenario. Son aquellos que definen una acción perteneciente al marco de trabajo para el proceso.
* Patrones de fase. Son aquellos que definen la sucesión de actividades dentro del marco de trabajo que ocurre paralelo a un proceso.

Plan para describir un patrón de proceso. Se recomienda la siguiente secuencia:

1. Nombre del patrón. Describe su función dentro del software.
2. Propósito. Objetivo del patrón.
3. Tipo. De tarea, de escenario o de fase.
4. Contexto inicial. Se describen las condiciones en las cuales se aplica el patrón.
5. Solución. Se describe la implementación del patrón.
6. Contexto resultante. Se describen las condiciones que habrá una vez que el patrón haya sido implementado con éxito. Definen una acción de la ingeniería de software o una tarea de trabajo importante.
7. Patrones relacionados. Lista de todos los patrones de proceso directamente relacionados con éste.
8. Usos conocidos/Ejemplos. Se indican los ejemplos específicos en los cuales el patrón es aplicable.

1.7. MODELO DE PROCESO DE SOFTWARE PERSONAL - PSP.

El PSP fue propuesto en el año 2005 por el físico y pensador norteamericano Watts S. Humphrey, conocido como el padre de la calidad de Software. El PSP ha sido diseñado con la intención de ayudar a los desarrolladores de software a realizar bien su trabajo. Trata de cómo aplicar en las actividades cotidianas,

métodos avanzados de ingeniería, proporcionando en forma detallada la planificación y estimación del control de su rendimiento frente a objetivos y procesos definidos que guían la acción laboral frente a una tarea.

Ventajas:

* Aumento de la productividad de las personas.  Mejora en la actividad cotidiana de programación.
* Detección oportuna de defectos y riesgos.
* Aumento de la calidad.
* Reducción del ciclo de vida del producto.
* Establecimiento de una mejor comunicación.

Desventajas:

* Requiere compromiso y disciplina.  Requiere documentación y registros.
* Demanda planificación.
* Capacitación del equipo.
* Entrenamiento a nuevos miembros.

El PSP permite mejorar la calidad de trabajo en el desarrollo de software. Para ello propone las siguientes actividades:

* Medición de tiempos.
* Planteamiento de actividades.
* Análisis de procesos.
* Cambios en las actividades diarias.

El PSP propone un proceso de mejora para cambiar la forma de ejecutar un trabajo. Propone los siguientes pasos:

* Definición de objetivos de calidad que se pretende alcanzar.
* Medición de la calidad de propio producto.  Entender el funcionamiento del proceso.
* Ajustar el proceso.
* Aplicar el proceso ajustado.
* Medir los resultados.
* Comparar los resultados con los objetivos esperados.
* Realimentar y establecer un ciclo de mejora continua.

Bajo la aplicación del PSP se propone el registro de las siguientes actividades:

* Cuaderno de ingeniero (programador).  Cuaderno para registrar tiempos.
* Resumen de actividades semanales.
* Cuaderno de trabajos realizados y pendientes.
* Cuaderno de compromisos (acuerdos).  Programar actividades.
* Cuadernos de historial de defectos.
* Listas de verificación para la revisión de códigos.

1.8. MODELO DE PROCESO EN EQUIPO - TSP.

El modelo TSP, es una metodología para dirigir un equipo en el trabajo de mejora y desarrollo de software de manera efectiva normal y natural. En concordancia con el Personal Software Process (PSP), el modelo llamado Team Software Process (TSP) o Modelo de Proceso en Equipo, determina un marco de trabajo de procesos determinados cuyo fin es ayudar a equipos de gerentes e ingenieros en la organización y producción de proyectos de software de gran envergadura, que comprendan un número por encima de varios miles de líneas de código. La finalidad del TSP es mejorar el grado de calidad y productividad de un equipo durante el desarrollo de un proyecto de software, cumpliendo con las exigencias de entrega en tiempo y requerimientos sin afectar el costo del proyecto.

La versión inicial del TSP fue desarrollada en 1996 por Watts Humphrey y en el año 2000 fue publicado su primer reporte, a cargo del DOD (Departamento de Defensa de los Estados Unidos de Norteamérica). El TSP tiene como intención el detalle y se enfoca en el proceso de la construcción de un equipo desarrollador de software, delimitando objetivos, estableciendo roles y tareas de trabajo en equipo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | psp |   Planes personales  Método planeaciúl  Valor agregado  Métricas calidad  Procesos definidos | |  | | --- | | TSP creación de equipo |   Planes agresivos  Calidad propia  Objetivos proyecto Plan propio  Plan detallado Recursos de equipo | |  | | --- | | TSP trabajo en equipo |   pnorjdad en calidad  Costo de calidad  Seguir el proceso  Revisión de status y calidad  Comunicación |

Estructura del TSP

Objetivos del TSP.

Propone los siguientes objetivos:

* Generación de un marco basado en PSP.
* Desarrollo de productos para varios ciclos.
* Establecimiento de normas para medir la calidad y el comportamiento.
* Aplicar métricas para equipos.
* Evaluación de roles personales y equipos.
* Guías para resolución de problemas en equipos.

Ventajas:

* Incremento de la productividad de las personas.
* Perfeccionamiento de los hábitos de programación.  Detección temprana de defectos y riesgos. - Incremento de la calidad

Desventajas:

* Compromiso y disciplina para seguir el plan por miembros del equipo.
* Cumplir con la documentación requerida.
* Contar con un buen conjunto de métricas y escalas para establecer niveles de calidad.
* Todos los miembros del equipo deben conocer el PSP.

Fases del Ciclo de Vida TSP

El ciclo de vida del TSP comprende:

1. Lanzamiento (Launch). Esta fase comprende:  Estudio de objetivos.
   * Asignación de equipos y roles.
   * Descripción de necesidades del cliente.

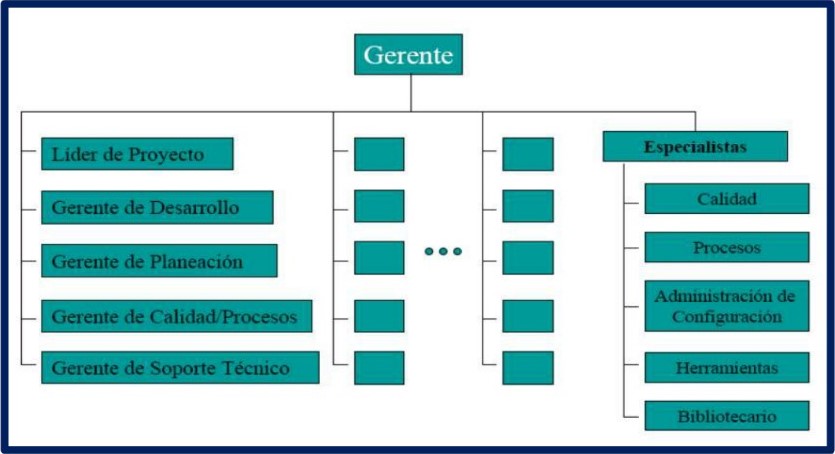
 Establecimiento de metas individuales y del equipo.

1. Estrategia (Strategy). Esta fase comprende:
   * + Creación del diseño conceptual del producto.
     + Establecimiento de estrategias de desarrollo: toma de decisiones sobre productos a entregar.
     + Estimaciones iniciales de esfuerzos y tamaño.
     + Plan de administración de la configuración.
     + Reciclaje de plan previo.
     + Delimitación de riesgos administrativos.
2. Requerimientos (Requirements). Esta fase comprende:
   * Análisis de necesidades del cliente y se entrevistan.
   * Caracterización de requerimientos.
   * Revisión de requerimientos.
   * Diseño de plan de pruebas del sistema.
3. Diseño (Design). Esta fase comprende:
   * + Diseño de alto nivel.
     + Especificaciones de diseño.
     + Revisión del diseño.
     + Ejecución de plan de pruebas de integración.
4. Implementación (Implementation). Esta fase comprende:  Aplicar PSP en la implementación de módulos y unidades.
   * Creación del diseño detallado de los módulos y unidades.
   * Inspección del diseño.
   * Conversión del diseño al código.
   * Revisión del código.
   * Compilación y pruebas de módulos y unidades.  Análisis de calidad de los módulos y unidades.
5. Pruebas (Test). Esta fase comprende:  Construcción e integración del sistema.

 Ejecución de pruebas del sistema. - Documentación de usuario final.

1. Después de la Muerte (Postmortem). Esta fase comprende:
   * Análisis de resultados.
   * Presentación de reporte del ciclo.  Evaluación de pares y equipo.

Organización del TSP.



Lanzamiento TSP, checklist para planeación.

Las cuales son:

* + Establecimiento de productos y objetivos institucionales.
  + Establecimiento de roles y objetivos de equipo.
  + Definición de estrategia para el desarrollo.
  + Definición de un plan general.
  + Definición de un plan de calidad.
  + Balance de cargas de trabajo.
  + Proyección de riesgos.
  + Diseño de informes administrativos.
  + Inspección del plan administrativo.
  + Análisis después del ciclo de vida, revisión del proceso por nuevo equipo.

ll. CALIDAD DE SOFTWARE.

OBJETIVO GENERAL.

Utilizar los conceptos relacionados a la calidad de software para efectuar pruebas que permitan comprobar la integridad de diversas aplicaciones.

OBJETIVOS ESPECíFlCOS.

* + Relacionar los conceptos de calidad.
  + Reconocer la importancia de la calidad de software.
  + Reconocer la importancia de la revisión de software.
  + Reconocer la importancia de la fiabilidad de software.

2.1. ACTIVIDAD PRÁCTICA 3.

En esta tercera actividad se utiliza el software Selenium para efectuar el primer testeo de una página web a través de una cuenta registrada en Gmail, utilizando el navegador Mozilla Firefox, esto permitirá contar con un script que se utilizara en las pruebas.

Testeando una página WEB con Selenium IDE.

De acuerdo a las recomendaciones de buenas prácticas un producto de software debe probarse durante su proceso de desarrollo, a continuación con el software Selenium se efectúa la prueba de software a una página web (producto final).

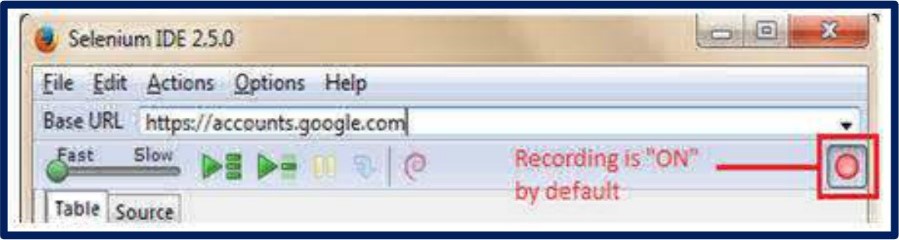
Paso 1. Inicializar la página web de Google (http://www.google.es) utilizando el navegador Mozilla Firefox. Se recomienda contar siempre con la última versión del navegador.

Paso 2. A continuación se requiere activar el plugin Selenium IDE. Para ello siga la ruta que se describe a continuación: Ver -> Panel lateral -> Selenium

IDE

Paso 3. A continuación deberá ubicarse en el cuadro de texto del buscador Google y escribir Gmail para acceder a una cuenta autorizada. Efectúe la búsqueda de manera normal. Seleccione la página indicada de los resultados obtenidos.

Paso 4. Proceder a activar el botón de grabación. A partir de este momento el software Selenium se encuentra grabando cada una de las acciones que se ejecutan en la página Web seleccionada, cabe señalar que de manera automática la grabación se inicia al inicializar el plugin, si se requiere detener la grabación e iniciarla posteriormente, solo deberá deshabilitar el botón de grabar y activarlo cuando lo necesite.



Paso 7. Una vez realizadas las anteriores acciones, deberá guardar el script cuyo lenguaje de grabación por defecto es el HTML, esta acción generará el código respectivo.

Paso 8. En este punto usted se encuentra listo para entender el funcionamiento del software Selenium IDE como aplicación de software libre que permite la ejecución de pruebas básicas de aplicaciones Web.

2.2. ACTIVIDAD PRÁCTICA 4.

En esta cuarta actividad se utiliza el software Selenium para crear el primer script siguiendo los procesos requeridos por la aplicación.

Creando el primer script con Selenium IDE.

Paso 1. Identificar los procesos para la creación de un script, se requiere de tres procesos bajo el software Selenium, los cuales son:

* Recording, es el primer proceso y es el que ayuda a grabar las interacciones entre el usuario y el navegador. A este proceso también se le conoce como Selenium IDE script.
* Playing back, este segundo proceso se encarga de ejecutar el script grabado para verificar y monitorear su estabilidad y eventos presentados.
* Saving, este último y tercer proceso se encarga del grabado de una secuencia de scripts estable, quedando almacenado para su en futuras pruebas.

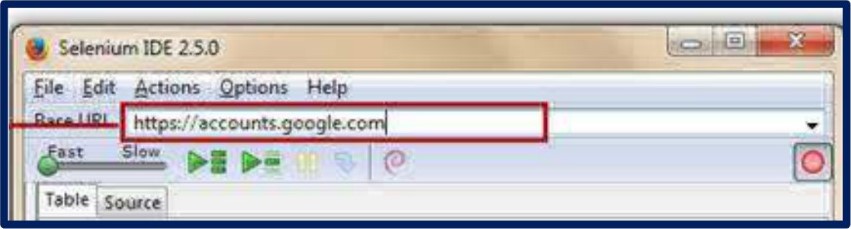
Paso 2. Efectuar la ejecución de los procesos a partir de la activación de los procesos.

Paso 3. Grabando un script de prueba, se requiere que ejecute y verifique siguiente escenario:

* Abra "https://accounts.google.com'  Declare el nombre de la aplicación.
* Ingrese a una cuenta con un nombre de usuario y contraseña válidos.
* Verificar que el usuario se redirigido a la página.

Paso 4. Ejecute Firefox y abra Selenium IDE desde la barra de menú.

Paso 5. Ingrese dentro de la caja de texto "Base URL" la dirección de la aplicación web a la cual se ejecutara el test. ("https://accounts.google.com")



Paso 6. Por defecto, el botón de grabación se encuentra en estado "ON' (activo y listo). De no encontrarse activo ajuste en "OM" si se encontrara en estado "OFF" a fin de habilitar el modo de grabación.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | a Selenium1DE 2.50 | | | Eile Edit Actacns Qpoons Heip     |  | | --- | | httpsd/acccunts.googie.c |   Base URL | | |  | O | | by defau}t  Table Source' | | |

|  |  |
| --- | --- |
| Sign in with your Google Account   |  | | --- | | Email |   Password      Create an account |

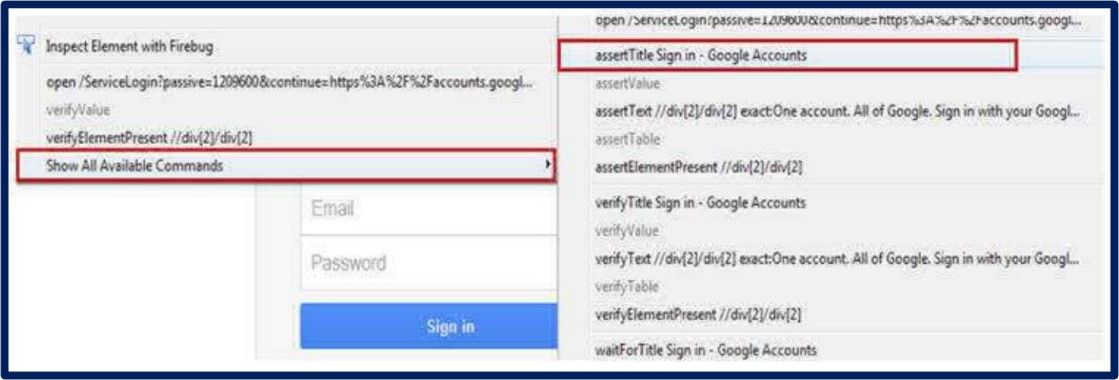
Paso 7: A continuación abra la aplicación a testear en explorador web Mozilla Firefox (https://accounts.google.com). Se mostrara la ventana clásica de acceso y registro a las cuentas de Google, como se muestra en la figura de la derecha. (Recuerde que debe de contar con una cuenta de Google, de lo contrario crear una de para las pruebas).

Paso 8. Verifique que el nombre de la aplicación sea el correcto. A continuación, haga clic derecho en cualquier parte del escritorio de la aplicación excepto en los hipervínculos o imágenes, de inmediato se abrirá un menú contextual del Selenio IDE, mostrando una lista de comandos a ejecutar.

Paso 9. Haga click en la opción "Show Available Commands" para obtener una lista completa de todos los comandos disponibles.

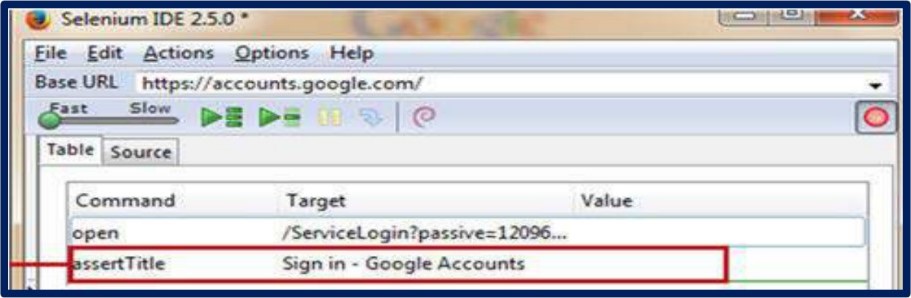
Paso 10. A continuación se abrirá otro menú que contiene un nuevo grupo de comandos disponibles y aplicables para el ejercicio.

Seleccione "assertTitle Sign in — Google Accounts ", esta opción sirve para verificar el título de la página. Utilice de guía la siguiente figura:



Paso 11. Tan pronto como se haga click en la opción "assertTitle Sign in — Google Accounts", se dará inicio al proceso de prueba bajo el editor de pruebas del software Selenium IDE.

Paso 12. Verifique que el entorno de trabajo del plugin Selenium tenga el aspecto que se muestra en la figura, el contenido del recuadro en color rojo debe presentarse como: "assert in — Google Accounts"



Paso 13. A continuación, ingrese un nombre de usuario valido en la caja de texto de Gmail "Email".

Paso 14. Luego, ingrese una clave de acceso valida en la caja de texto 'Password"

Paso 15. Verifique la simulación de estas acciones de los usuarios en el editor de pruebas de la aplicación Selenium IDE.

Paso 16. De no contar con una cuenta Google, usted deberá crear previamente una cuenta de prueba en Gmail. También puede hacer uso de sus propias cuentas.

Paso 17: Haga click sobre el botón "Sign in" para completar el ingreso a la cuenta de Gmail. Como puede observar se ha creado previamente una cuenta de prueba llamada "TestSelenium1607@gmail.com", muchos estudiantes optan por esta opción, crear cuentas de prueba en "Gmail". El usuario deberá poder ingresar a la página principal proporcionando las credenciales correctas.

Paso 18. Compare la información de su escritorio con la información que aparece dentro de los recuadros en rojo de la figura anterior. La información a revisar es:



Paso 19. Para finalizar la grabación deberá dar click en el botón rojo (Recording stopped).

Paso 20. Verifique que la información de su escritorio muestre la secuencia de comandos grabada en el siguiente orden:



• assertTitIe--—-----Sign in —Google Accounts



Paso 20. Observe en el computador, que el elemento web correspondiente se resalta en amarillo con un borde de color verde fluorescente alrededor. Si no se resalta, el usuario deberá corregir el problema y tendría que otorgar algún otro valor al localizador.

Paso 21. Selenium fue desarrollada en el 2004 por Jason Huggins, e inmediatamente se unieron al esfuerzo otras personas especialistas en pruebas y programación. Selenium es un software de código abierto bajo la licencia apache2.0 que puede ser utilizada de manera libre y responsable.

Nota: El nombre viene a partir de una anécdota de Huggins al mofarse de un rival llamado Mercury (mercurio). Huggins decía que el envenenamiento por mercurio puede ser curado tomando complementos de Selenio. Los miembros de su equipo decidieron tomar el nombre y siguieron hasta estos días.

Además del Selenium, existe una gran cantidad de proyectos que se desarrollan en paralelo con el Selenium (En primer lugar, Selenium Grid, para probar diversas pruebas tanto a clientes remotos o locales, y en segundo lugar, Flash Selenium para la prueba de programas escritos en Adobe Flex o Selenium Silverlight).

2.3. CALIDAD DE SOFTWARE.

Medición de Calidad de Software.

Al desarrollar software, la calidad de diseño forma parte a la calidad de los requerimientos, del usuario sobre el sistema. La calidad del diseño tiene que estar en fase con la implementación, los objetivos, requisitos y rendimiento del sistema a implementar.

Cuando se trata de medir la calidad de un producto de software, el usuario como el desarrollador quieren lograrlo, sin embargo, hay controversias respecto a la definición de calidad, con el fin de satisfacer ambas partes, quedando más afectado el cliente.

Definición de la Calidad de Software.

John W. Tukey en 1957, utilizo por primera vez el término software, según la IEEE (IEEE 729,1983), "un producto de software es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de computación".

Del concepto anterior de desprender, que el software es todo lo intangible, es decir todo aquello que es "no físico" y se relaciona con las ordenes e instrucciones de generadas por la computadora.

Cualquier producto de software, emana de requerimientos y necesidades. La IEEE 830 (IEEE 830,1998) para la "Especificación de Requerimientos de Software" (SRS), señala dos tipos de requerimientos para el producto de software: Funcionales y No funcionales.

De acuerdo a los avances en el desarrollo de software, las particularidades no funcionales de un producto software, han elevado su importancia, ya que los clientes al usarlas y no conseguirlas, las van solicitando como necesidades, tal es el caso de la calidad.

Cada producto de software busca lograr la más alta calidad posible, es decir que cumpla con las expectativas de los usuarios, pero se debe considerar que los productos de software no son perfectos, y en ellos hay cierto grado de calidad, encontrándose productos malos, buenos y excelentes.

Bajo esta situación se hace difícil definir la calidad del software, pues tiene una relación estrechamente subjetiva, dependiendo de la persona y proyecto. Es totalmente relativo dependiendo del usuario. Por otro lado, determinados proyectos requerían satisfacer diferentes necesidades entre sí.

En conclusión la calidad contiene muchas aristas a considerar como punto de observación, y no se encuentra determinada de manera discreta (1 0 0), es decir, tiene calidad o no tiene.

Aplicación de Estándares de calidad de software.

Ante la necesidad de homogenizar las características que definen la calidad del software, se creó un estándar para permitir esta necesidad, actualmente hay dos generaciones de estándares, una relacionada al ISO/IEC 9126 Information Technology — Software Product Evaluation- Quality Characteristics, divulgada en 1991 cuya intención era fomentar un ámbito para la evaluación de la calidad del software.

Posteriormente en 1994 se hacen modificaciones y se adaptan a éste estándar, incluyéndose definiciones de calidad interna (dependiente de la construcción y no de la ejecución) y calidad externa (dirigida al software en ejecución o percepción del cliente).

La segunda generación está dada por la norma ISO/IEC 25000 — SQuaRE (Software Product Quality Requirements and Evaluation), que es la actualización de la ISO 9126; presentada en el 2007, se define a la calidad del software en la actualización del 2008, como conjunto estructurado de características y subcaracterísticas.

Evaluación de la Calidad de Software.

A partir de las características y subcaracterísticas definidas por el estándar, la calidad del producto de software (Interna) y el uso del producto de software (Externa), son altamente subjetivas, la mejor forma de evaluar el software es por medio de la utilización y experiencia de los usuarios o clientes finales, generalmente en el desarrollo de software, por otro lado, tal situación impide ejecutar comparaciones objetivas y resalta la falta de madurez en esta disciplina.

Importancia de la Calidad de Software.

El concepto de calidad del software parece ser transparente a un gran número de personas en la vida diaria, concepción realmente errada. Si en la computadora aparece el famoso pantallazo azul, nos encontramos frente a un problema de calidad del software; cuando falla el sistema bancario y provoca retrasos, también nos encontramos frente a un problema de calidad del software; cuando por algún motivo no pasa la tarjeta de crédito en una tienda, también hay un problema de calidad del software.

Alguna de las situaciones anteriores afecta a las personas ocasionando retrasos, pérdidas de tiempo, dinero, entre otras. Ahora si esta situación se presenta en sistemas más elaborados el impacto definitivamente es mayor, por ejemplo si se presentara en la Bolsa de Valores de Lima, ocasionaría grandes pérdidas económicas. La situación más catastrófica por fallo de software se presentó en el año 1996, cuando en el primer minuto de haber lanzado el cohete Arianna 5 de la Agencia Espacial Europea hizo explosión. Se perdió 10 años de investigación y un monto de 7 mil millones de Euros.

Es un peligro latente que cualquier sector productivo, pueda verse afectado por una falla de software en cualquier parte del mundo.

Sin embargo en la actualidad se desarrolla software fiable y correcto a costos accesibles. Los desarrolladores profesionales y grandes empresas utilizan técnicas de control y prevención, para disminuir estas situaciones.

Pruebas de software para la calidad.

Por encima de los certificados y estándares generales, se encuentran las pruebas de software y forman parte de las técnicas a considerar en todos los proyectos, estas tratan de la ejecución de ensayos sobre el funcionamiento de las aplicaciones en ambientes bajo control, con la finalidad de detectar defectos antes de la entrega del producto de software.

Si son ineficientes las pruebas, permitirán que el producto llegue con defectos al usuario final. Es de gran importancia esta etapa durante el desarrollo de un proyecto de software.

Factores que afectan negativamente a las pruebas.

Ante esta situación, se plantea un análisis con el fin de identificar las causas que afectan a las pruebas de software. Los expertos indican que uno de los factores es el tiempo para las pruebas, que normalmente se dejan para la última etapa del proyecto, queriendo optarse por la facilidad de disminuir la calidad o recursos cuando se presentan ajustes en el tiempo, otro factor es la falta de formación de programadores y gerentes de Tl.

Se recomienda concientizar y formar a los integrantes de equipos de desarrollo de proyectos de software para lograr un mejor nivel en la calidad de los productos de software entregados. La naturaleza del software obedece a un gran esfuerzo netamente intelectual y es determinante contar con personas capacitadas para el desempeño como programador.

Importancia de la Revisión de software.

La revisión de software actúa como filtro durante el proceso de desarrollo de un proyecto de software. Además permite descubrir errores y defectos con la posibilidad de eliminarlos antes de la entrega del producto.

La revisión tiene como objetivo principal identificar errores antes de que pasen a un siguiente nivel de desarrollo del software o que sean entregados al cliente de manera formal.

Analizando las métricas de revisión.

Las revisiones pueden ser formales o informales, en las formales tenemos las siguientes métricas de revisión y su empleo:

* Esfuerzo de preparación (Ep): Revisión del producto antes de la revisión real.
* Esfuerzo de evaluación (Ee): Se refiere a la revisión real.
* Esfuerzo de la repetición (Er): Corrección de errores encontrados durante la revisión.
* Tamaño del producto del trabajo (TPT): Medición del tamaño del producto del trabajo revisado.
* Errores menores detectados (Errmenores): Cantidad de errores detectados que se clasifican con el menor nivel.  Errores mayores detectados (Errmayores): Cantidad de errores detectados que se clasifican con el mayor nivel.

|  |  |
| --- | --- |
| |  | | --- | | Erevision | |

La densidad del error demuestra a los errores encontrados por cada unidad de producto de trabajo ejecutado:



Eficiencia de las revisiones.

El cálculo del costo-beneficio de la eficiencia de las revisiones solo se puede dar después de que éstas han finalizado, de que las unidades de medida de la revisión se han recabado, de que los datos promedio han sido calculados y de que la calidad posterior del software ha sido medida (mediante pruebas).

Esfuerzo ahorrado por error = Epruebas -Erevisiones

Las revisiones se aplican dentro de un nivel de formalidad adecuado para el producto a desarrollar, considerando la programación y tiempo del proyecto y las personas del equipo de desarrolladores.

|  |
| --- |
| Tiempo  Planeación |

2.4. MODELOS PARA REVISIONES TÉCNICAS.

Cada una de las particularidades del modelo de referencia nos permite definir el nivel de formalidad de la revisión. A continuación se muestra el incremento del nivel de formalidad de una revisión: Revisiones técnicas.

La revisión técnica tiene como objetivo detectar errores y revelar aspectos que tendrían un impacto negativo en el producto de software. Si se adelanta este descubrimiento es menos probable que se difundan a otros productos del software y se propague en todo el sistema, lo que demandaría un mayor esfuerzo para su corrección.

Revisión informal.

Una revisión informal es una reunión casual entre colegas. Este tipo de reunión al o contar con planeación, agenda o puntos de reunión, carece del nivel de eficacia de las revisiones fórmales.

Revisión formal.

Una revisión técnica formal (RTF) es una actividad del control de calidad del software que ha sido previamente programada y ejecutada por el equipo de desarrolladores de software (y otras personas). Este tipo de revisión contiene los siguientes puntos:

* Descubrir los errores en funcionamiento, lógica o implementación.
* Verificar que el software que se revisa cumple sus requerimientos.
* Garantizar que el software está representado de acuerdo con estándares predefinidos.
* Obtener software desarrollado de manera uniforme.  Ejecutar proyectos más manejables.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Reunión de revisión    preparación previa, pero no más de dos horas de trabajo   |  | | --- | | Debo ser de al monos dos horas | | Centra en una parte específica (y pequeña) del software general. | | Reporte y registro de la revisión   |  | | --- | | Se produce la lista de pendientes de la revisión |     Elabora un reporte técnico formal de la revisión   |  | | --- | | El reporte de la revisión es una sola pigme (quizá eon anexos) | | Identificar las áreas de problemas en el producto | |

Las revisiones técnicas formales obedecen a los siguientes lineamientos:

 Revise el producto, no al productor.

* Establezca una agenda y sígala.
* Limite el debate y las contestaciones.
* Enuncie áreas de problemas, pero no intente resolver cada uno.
* Tome notas por escrito.
* Limite el número de participantes e insista en la preparación previa.
* Desarrolle una lista de verificación para cada producto que sea probable que se revise.
* Asigne recursos y programe tiempo para las RTF.

 Dé una capacitación significativa a todos los revisores.  Revise las primeras revisiones.

Fiabilidad del Software.

Las fallas en el funcionamiento de un software pueden tener su origen en, una especificación inadecuada, errores de diseño del software, averías en el hardware, Interferencias transitorias o permanentes en la comunicación de datos.

La fiabilidad o reliability de un software, es una medida de su aceptación con una especificación autorizada de su comportamiento, mientras que una avería o failure, es una desviación del comportamiento del software respecto a su especificación.

Las averías se presentan en el desenvolvimiento externo del software, pero son el resultado de errores internos. Las causas mecánicas o algorítmicas de los errores se llaman fallos o faults.

Tipos de fallos.

Los fallos deben prevenirse de todas formas e impedir que causen averías. Los fallos pueden ser de 3 tipos:

 Fallos transitorios. Son aquellos que desaparecen solos al cabo de un tiempo. Ejemplo: la interferencias en comunicaciones.

* Fallos permanentes. Son aquellos que permanecen hasta que se reparan. Ejemplo: problemas de hardware, errores de diseño de software.
* Fallos intermitentes. Son aquellos fallos transitorios que ocurren de vez en cuando. Ejemplo: el calentamiento de un componente de hardware.

Aumento de la fiabilidad del software.

Existen dos maneras de aumentar la fiabilidad de un software:

* Prevención de fallos. Tiene como fin evitar que se inserten fallos al software antes de que entre en funcionamiento. Se realiza en dos etapas:

1. Evitación de fallos. Su función es impedir la introducción de fallos durante la construcción del software.
2. Eliminación de fallos. Su función es encontrar y eliminar los fallos que se producen en el software una vez construido

 Tolerancia de fallos. Tiene como fin conseguir que el software continúe funcionando aunque se produzcan fallos. Y maneja los siguientes grados de tolerancia ante la presencia de fallos:

1. Tolerancia completa (fail operational). El software sigue funcionando, por un determinado tiempo, sin perder el funcionamiento ni servicios básicos.
2. Degradación aceptable (fail soft, graceful degradation). El software sigue funcionando con una pérdida parcial de sus funciones hasta la reparación del fallo.
3. Parada segura (fail safe). El software se detiene en un estado que asegura la integridad del entorno hasta que se repare el fallo.

En ambos casos el objetivo es desarrollar software con tipos de averías bien definidos previamente para que su mantenimiento correctivo a incidencias se ya conocido, tomando en consideración las recomendaciones de buenas prácticas de los diferentes modelo de madurez para el desarrollo de software y mejora de procesos en las organizaciones.

III. MODELOS DE SOFTWARE.

OBJETIVO GENERAL.

Aplicar los modelos de calidad de software para el desarrollo de aplicaciones que logre satisfacer necesidades implícitas e explícitas del cliente con el fin de mejorar el producto de software.

OBJETIVOS ESPECíFlCOS.

 Describir los tipos de modelo de calidad de software.

* + Reconocer el modelo de Calidad de producto.
  + Reconocer el modelo de Calidad de proceso.
  + Reconocer el modelo de Calidad de uso.
  + Reconocer los estándares de calidad de software.

3.1. ACTIVIDAD PRÁCTICA 5.

En esta quinta actividad se utiliza el software Selenium para probar la estabilidad del script creado en la actividad 4. Esta prueba forma parte del proceso número 2 que corresponde al Playing back.

Playing back / executing a test script.

|  |  |  |  |
| --- | --- | --- | --- |
| Selenium IDE 2.5.0 • | | | |
| Elle Edit Qpt•ons Help  Base URL https://accounts. le.com/ | | | |
|  | |  |  |
|  | Table Source | |

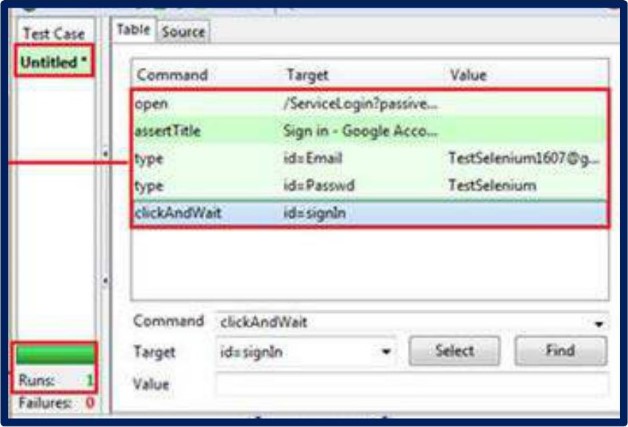
Paso 1. Ahora que se ha creado el primer script en Selenium IDE, se probara su estabilidad. Haga clic en el botón de reproducción para ejecutar el script. La figura de la derecha encierra en un recuadro color rojo el botón de reproducción.

Dar click en el icono representativo para dar inicio a la reproducción del script.

Paso 2. Luego de realizar la ejecución del script, se presentara en color verde las líneas de código que tengan funcionamiento correcto.

Paso 3. Verifique que en el área de comandos al efectuar todo el test los colores del código aparezcan en verde, esto indica una ejecución del script satisfactoria.

Paso 4. Verifique el área de estado de la prueba (Test Case). Deberá aparecer en color verde la cantidad de veces que esta haya corrido, el verde indicara el correcto funcionamiento y el rojo indicara la presencia de problemas en su ejecución.

Paso 5. Verifique todos los pasos anteriormente señalados comparando la información de su computador con el contenido de los recuadros en rojo de la siguiente figura, ubicada al lado derecho de este párrafo:

Guardando el script de prueba.

Paso 1. Una vez reproducido el script de prueba, este deberá de guardarse para posteriores usos. Para guardar el script, haga click en el menú "File" y seleccione la opción "Save Test Case".

Paso 2. El software solicitara indicar el lugar donde guardar y proporcionar un nombre al script. Proporcione el nombre "Gmail\_Login" y haga clic en el botón "Save". Por defecto se guardara en formato HTML.

Paso 3. Verificar que el script de prueba se encuentre en la ubicación proporcionada en la etapa anterior. Observe que el script de prueba por defecto se guarda y utiliza el formato HTML.

3.2. ACTIVIDAD PRÁCTICA 6.

En esta sexta actividad se utiliza el software Selenium para comprobar las características y opciones comunes que pueden realizar sobre la ejecución de las pruebas de software. Como por ejemplo el control de la ejecución de un script durante el proceso de prueba de software.

Utilizando las características comunes del Selenium.

Paso 1. Al probar la velocidad de ejecución a partir de las pruebas de software en aplicaciones web, es muy probable se presenten diversas situaciones que pueden provocar una carga de la página. Razón por la cual se tiene que considerar esta posibilidad en el proceso.

Paso 2. Para llevar un mejor control sobre la observación de fallas durante la reproducción de un determinado escenario de prueba, configure la velocidad de ejecución al mínimo.

3. En su escritorio (software Selenium) ubique el área de control de velocidad, para efectuar dicho control, el área a cargo de esta funcionalidad se muestra en la figura de la derecha encerrada en un recuadro de color rojo.

Utilizando comando "Execute" (Ejecutar).

Paso 1. Selenium permite la ejecución de pruebas de manera sencilla y deja al usuario ejecutar una determinada parte del test sin necesidad de ejecutar todo el script. La opción para logar esta acción es a través del comando "Execute this command".

|  |  |
| --- | --- |
| mail\_Login.html - Selenium IDE 2.5.0 | |
| Edit  URL st | Acttons Qptions Help |
| Record  elay entire test suite  Play current test case  Pause Resume  Step  Fastest Q)  Faster  Slower  Slowest Q)  Toggle Breakpoint  Set / Clear Start Point |
| ble |
|  |
|  |
| clic |
|  | Execute this command |

Paso 2. La opción "Execute this command" se utiliza cuando requiera depurar o cuando necesite ver el comportamiento de una determinada etapa de la prueba de manera particular.

Paso 3. La opción "Execute this command" se puede utilizar de 4 formas:

* Forma 1. Utilizando la opción "Actions" ubicada en la barra de Menú.
* Forma 2. Usando la tecla de atajo "X".
* Forma 3. Haciendo click derecho en el área de prueba y seleccionando la opción "Execute this command".
* Forma 4. Haciendo doble click en cualquier área del Selenium durante el proceso de ejecución de la prueba.

Paso 5. Considere para cualquier forma de ejecución del paso 4, que el usuario deba haber seleccionado la etapa de ejecución de la prueba.

Utilizando "Steps to be followed" (Detener la prueba).

Paso 1: Abra el navegador web y en Base URL colocar el link "https://accounts.google.com"

Paso 2. Seleccione el área del código que desea testear.

Paso 3. Verifique encontrarse en la página web correcta para reducir las posibilidades de error.

4. Presione "X" para ejecutar el test seleccionado. Puede utilizar cualquier otra forma para ejecutar este paso.

Paso 5. Observe que se ejecuten los pasos de la prueba seleccionada. En el código se indicara en color verde para prueba satisfactoria o rojo cuando hubo errores en su ejecución.

Paso 6. Verifique que todas las acciones que realice en el explorador web sean testeadas por el Selenium.

Paso 7. Tenga en cuenta que el usuario es responsable de la ejecución del script en Firefox. Hay una probabilidad de fallo si el usuario no ha abierto una página web legítima.

Utilizando "Start point" (Punto de inicio).

Paso 1. Selenium permite al usuario especificar el punto de inicio o "Start point" dentro de un script de prueba. El punto de inicio es el lugar donde queremos iniciar la ejecución del script. Se utiliza cuando no se requiere ejecutar desde un inicio todo el script.

Paso 2. El Start point (punto de inicio) se puede configurar de la siguiente manera:

* Forma 1. Utilizando la opción "Actions" de la barra de "Menú"  Forma 2. Utilizando la tecla de atajo "S".
* Forma 3. Utilizando la selección desde el botón derecho del mouse con la opción: "Set/Clear Start Point".

Paso 3. Aparecerá un menú similar a los detalles descritos en los pases anteriores. Para cualquier caso se espera que el usuario con estas alternativas pueda seleccionar el inicio de la prueba desde cualquier punto. Tan pronto como el usuario marque el punto de inicio de la prueba aparecerá el icono respectivo, como se muestra en la figura:

|  |  |  |
| --- | --- | --- |
| assertTitle | Sign in • Google Accounts id: Email TestSeleniumH7@gmail.com | |
| type | id: Passwd | TestSelenium |
| IclKkAndWatt | id: signln |  |

Paso 4. Ahora cada vez que ejecute el script de prueba, se iniciara desde el punto de inicio señalado, es decir, para el caso del ejemplo es la cuarta línea, cuyo contenido es:

# • Type-—m id =

5. Considere que solo podrá generar un punto de inicio en toda la prueba. Este punto de inicio puede eliminarse si lo requiere. Recuerde que Selenium es un software libre y se puede descargar sin costo alguno de la página web del fabricante. Además para la realización de los ejercicios debe contar con una cuenta en Gmail.

3.3. ACTIVIDAD PRÁCTICA 7.

En esta séptima actividad se utiliza el software Selenium para probar la ejecución de pausas (Break point) durante la ejecución de un script de prueba. Estas pausas están orientadas a generar puntos de "stop" líneas de código determinadas por el ejecutor de las pruebas. Utilizando "Break point" (Pausa)

Paso 1. Selenium IDE permite al usuario especificar pausas (Break point) dentro de un script de prueba. Las pausas (Break point) permiten detener el script en un punto determinado de la prueba. Los "Break points"o pausas se utilizan en los momentos donde es necesario detener la ejecución de un script y observar su comportamiento. Un "Break point" o pausa puede ejecutarse de 3 maneras:

 Utilizando "Actions"de la barra de Menú.

* Utilizando el click derecho del mouse y seleccionando "Toggle Breakpoint".  Utilizando la tecla de atajo "B'

Paso 2. Tan pronto como el usuario active la pausa (break point) durante la prueba, un icono en color amarillo aparecerá al lado izquierdo, como se indica en la figura:

|  |  |  |
| --- | --- | --- |
| assertTitle | Sign in - Google Accounts id=EmaiI | TestSelenium1607@gmail.com |
| [AJtype | id: Passwd | TestSelenium |
| chckAndWait | id:signln |  |

Paso 3. Ahora, cada vez que ejecuta el script de prueba, se detendrá en el punto de pausa (break point), es decir, en la línea que tiene el contenido:

* Type id = PasswdTestSelenium

Paso 4. Compruebe el contenido de las líneas durante la ejecución de la prueba. Como se muestra a continuación:

* assertTitle—-Sign in — Google Accounts-——-





Paso 5. En la siguiente actividad se demostrara como ejecutar la aplicación de múltiples puntos de pausa durante la ejecución del script.

3.4. MODELOS DE SOFTWARE.

Modelos de Calidad de Software.

El presente capítulo define a la calidad del Software como la conjunción de características que pueden ser medidas y especificadas en relación a un sistema, a partir del software a producirse, para determinar su beneficio y presencia. La etapa de desarrollo del producto es uno de los principales problemas del software. Tanto para los vendedores de software y desarrolladores, razón por la cual se realiza un análisis de los modelos de calidad existentes, tales como: La Norma ISO/IEC, integración del modelo de maduración de la capacidad (CMMI) y el (IT MARK) modelo de calidad para PYMES, ambos orientados a determinar los pro y contras que presentan ante el desarrollo de software con calidad.

Al aplicar estos estándares de calidad se generan nuevos beneficios a las empresas que deciden adoptarlas. Reduciendo costos de manera considerable y aumentando sus ingresos económicos debido a la mejora de satisfacción y perfección de los clientes internos y externos. A finales de la década de los '80 se funda la European Foundation for Quality Managment (EFQM), organización que propone modelos de gestión de calidad total tales como, GTC o TQM, con tácticas enfocadas a perfeccionar los recursos, disminuir costos y optimizar los resultados, con el fin de dar una mejora continua al proceso productivo.

Los Modelos de Calidad permiten a las empresas e instituciones la optimización de sus procesos y la Mejora productos y servicios de un nivel de excelencia considerable. Los modelos de calidad de manera general son:

* Modelo de Dirección por Calidad 2006-2008 (SMCT).
* Modelo de Calidad para Gobiernos Municipales 2005.
* Modelo Iberoamericano de Excelencia en la Gestión (FUNDIBEQ).  Modelo de Excelencia de European Foundation Quality Management (EFQM).
* Modelo Shingo-Excellence in Manufacturing (E.U.A).
* Modelo Malcom Baldrige (E.U.A).

 Modelo Deming (Japón)

Existen factores que determinan la calidad de software ante la aplicación de estos modelos, los cuales son:

* Capacidad de corrección. ¿Hace lo que quiero?
* Tener fiabilidad. ¿Lo hace de forma fiable todo el tiempo?
* Contar con eficiencia. ¿Se ejecutará en mi hardware lo mejor que pueda?
* Tener seguridad (Integridad). ¿Es seguro?
* Amigable de uso. ¿Está diseñado para ser usado?

En una empresa que se inicia en la construcción y desarrollo de software se presentan prácticas erradas que deben corregirse, estas son las siguientes características:

* Constante improvisación en los procesos de software.
* No se siguen los procesos de software de acuerdo a las especificaciones.

 Costumbre por la atención inmediata ante crisis. (sin plan de acción)  Planes y presupuestos excedidos sistemáticamente, al no estar basados en estimaciones realistas.

Es primordial contar son datos objetivos que ayuden a mejorar la calidad del software. La mayoría de los modelos contienen métricas para evaluar diferentes atributos de calidad del software con frecuencia desde el diseño o código. Los actuales modelos están dirigidos a mejorar la calidad de estos procesos.

La relevancia al medir un producto de software se encuentra relacionado con la definición, recolección y administración conjunta de dos modelos fundamentales, como lo modelos empíricos (basado en el mundo real) y los modelos numéricos (basado en un contexto empírico). El que más se utiliza de los dos es el modelo empírico.

MODELO CMMI.

Es un factor estándar de calidad de software para industrias utilizados en Europa, sobretodo en España, el cual tiene como modelo la capacidad de los procesos que se utilizan en una organización, cuyo fin es entregar un producto de calidad esperada y consistente, el modelo nace a partir de la unión de la Ingeniería de sistemas e Ingeniería de Software, lo cual elimina redundancias y lo hace consistente.

Fundamentalmente el CMMI es un conjunto de normas para calidad orienta al entorno del Software. Aplicable en cada uno de los diferentes procesos a seguir para poder obtener software con calidad, razón por la cual es resaltable que igual que las normas ISO 90003, el modelo indica lo que se debe hacer, pero no la forma de hacerlo. El modelo CMMI ayuda en:

 Describir cada uno de los componentes del modelo y la relación entre ellos.

 Comprender el funcionamiento de las áreas de proceso.

 Ubicar información importante en el modelo.

Aplicar los conocimientos a su entorno de trabajo y en un equipo de evaluación de componentes y sus relaciones de un modelo.

La madurez del modelo de acuerdo a CMMI se divide en seis niveles (incompleto, ejecutado, gestionado, definido, cuantitativamente gestionado y optimizado). Estos niveles proporcionan a la medición la madurez de los procesos pues incorpora una nueva área denominada "Medición y Análisis", su alcance es mayor y más explícito comparándolo con la medición que se proporciona en el modelo CMM.

Esta nueva área entrega una gestión con el enfoque y la visibilidad que las organizaciones necesitan como guía en la medición y el mejoramiento continuo. El fin principal del área es desarrollar e instituir una capacidad de medición para dar soporte al resto de áreas de proceso entregando un marco de trabajo a las organizaciones a la hora de alinear los objetivos y necesidades de medición, proporcionando resultados objetivos y útiles para la toma de decisiones y acciones correctivas.

Esta orientación se encuentra alineada al Goal, Question, Metric (GQM) y el estándar ISO 15939, que tambien son procesos de medición.

Para establecer un proceso de medición efectivo se requiere de dos objetivos fundamentales:

Alinear las actividades de análisis de la medición.

Para lograr esto en CMMI se requiere:

* Establecer los objetivos de la medición.
* Especificar medidas.
* Especificar procedimientos de recogida y almacenamiento.  Especificar procedimientos de análisis.
* A partir de estas prácticas se establece un plan para la medición y el análisis.
* Con el que se pretende resolver cuestiones tales como: ¿Por qué se mide? , ¿Qué se va a medir? , ¿Cómo se va a medir?, etc.

Proporcionar los resultados de la medición.

Las prácticas asociadas a este objetivo son:

 Recoger los datos de la medición.

* Analizar los datos de la medición.
* Almacenar los datos y resultados. - Comunicar.

Estas dos etapas tienen como objetivo incorporar un correcto proceso de recojo de información y difusión de los resultados, ya que éstos deben proporcionarse a la persona adecuada para compensar sus necesidades de información.

Ventajas:

 Facilita la organización interna y permite homogenizar procesos de actuación.

* Detecta tempranamente errores en el desarrollo del producto o en la prestación de nuestro servicio.
* Reduce en una cantidad importante el número de incidencias.
* Control continuo de los proyectos, en cuanto a: Riegos, Recursos, Esfuerzo, Costos, Plazos.
* Obtención de un importante caudal de datos históricos susceptibles de ser considerados bajo análisis.
* Fomenta la cultura del compartir conocimientos.
* Mejora e servicio de entrega a los clientes.

 Promueve nuevas oportunidades de negocio.

Desventajas:

* Delimitado plazo para la implantación del proyecto.
* Normalmente aplicado a empresas grandes.
* Es de gran tamaño y complejidad, muy superior a los modelos vigentes.  Requiere de una evaluación con procesos más costosos en tiempo y esfuerzo.  La evaluación compleja alterar la definición de objetivos concretos de madurez.

MODELO ISO/ IEC 15504.

Este estándar SPICE, fue diseñada por la alta competitividad del mercado de desarrollo de software, utiliza procesos de ingeniería del tipo cliente-servidor, gestión de la organización y del soporte, permite identificar los riesgos, cumplir con el calendario, controlar los costos y mejorar la eficiencia y calidad, también permite una evaluación de los procesos de software relacionando con el conocimiento y la experiencia aportado por otros modelos como: CMM, Trillium, Bootstrap y ISO 9001 entre otros. El modelo CMMI, a lo igual que ISO,

define seis niveles de madurez. Cinco categorías de procesos y nueve atributos de procesos para medir el grado de calidad de los procesos.

El modelo ISO/ IEC 15504, trae consigo una guía para evaluar proyectos, que encierra, medir el proceso, utilizando métricas de calidad, administrar datos (considerando datos históricos), y el manejo de métricas en la organización, cuyo fin principal es entregar métricas tanto de procesos como de productos, de tal forma se pueda dar soporte a una programación eficiente y así generar valor en los productos. El modelo engloba los procesos y sus potencialidades tomando en cuenta la experiencia de compañías grandes, medianas y pequeñas. El modelo ISO/ IEC 15504 contiene 9 unidades entre normativas e informativas.

Ventajas:

* Es específico cuando se requiere desarrollo y mantenimiento del software.
* Se puede definir agrupación de procesos.
* Su orientación busca mejorar los procesos en beneficio de los objetivos del negocio.

Desventajas:

* Cuenta solamente con líneas cuyo fin es brindar un mecanismo que permite la evaluación.

 Recientemente es una norma internacional.

* No es un modelo específico para la industria de software.

MODELO IT MARK.

IT Mark provee un servicio certificado a nivel internacional aplicado a los procesos técnicos y de negocio, su diseño está dirigido a PYMES del giro Tl, con la finalidad de medir Excelencia en Tecnologías de la Información y su reconocimiento en la industria de Tl. También podemos decir que es un servicio clave diseñado para las PYMES de Tl, con el fin de posicionarlas a la vanguardia bajo una constante mejora continua y que esta sea sostenible. El IT Mark, también es aplicable en grandes organizaciones y considera los siguientes puntos:

* Gestión y desarrollo de software.
* CMMI-DEV vl .2, niveles de madurez 2 y 3.
* Gestión del negocio.
* Gestión de la seguridad de información. - ISO/IEC 27002:2005.
* Código de práctica para administración de seguridad de la información.
* ISO/IEC 27001 :2005 Técnicas de Seguridad.
* I.T. Mark.
* I.T. Mark Premium.
* Seguridad y Desarrollo de Software. - I.T. Mark Elite.

Ventajas:

* Desde el punto de vista técnico y administrativo aumenta el servicio de las organizaciones.

 El reconocimiento ha mejorado en el mercado respecto a capacidades de Tl.  Forma parte de los indicadores del mercado por medio de un sello de excelencia o Visibilidad al interior de sus propias capacidades de desarrollo de Tl.

* Identifica fortalezas y debilidades, así como oportunidades de mejora.  Provee una ruta de mejora en cada uno de los niveles de madurez y su compatibilidad y complemento con otros modelos existentes.
* Es Económico, rápido y rentable.
* Se incluyen valoraciones en Gestión del Negocio y en Seguridad Informática.

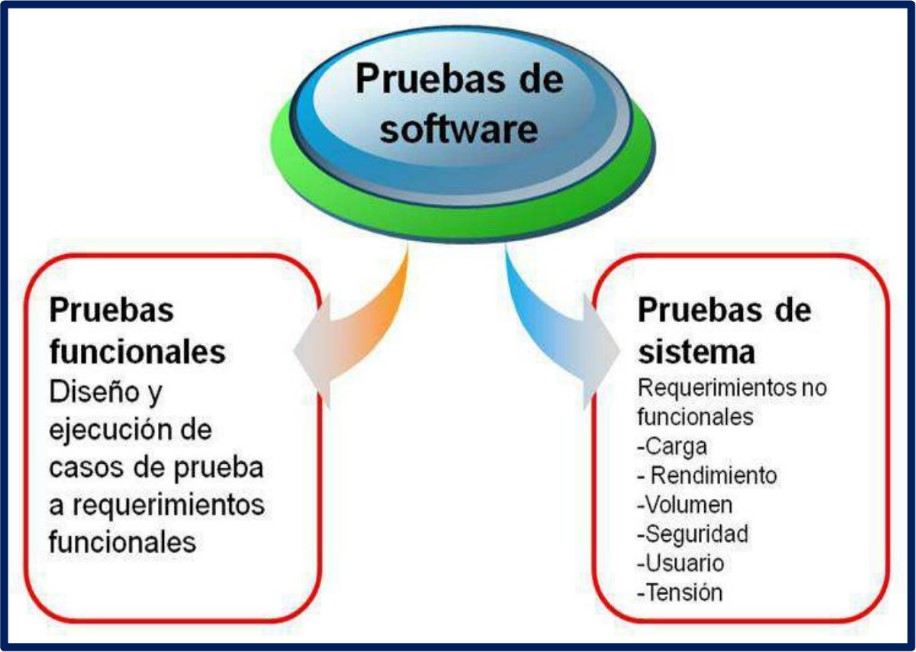
Elección de un modelo.

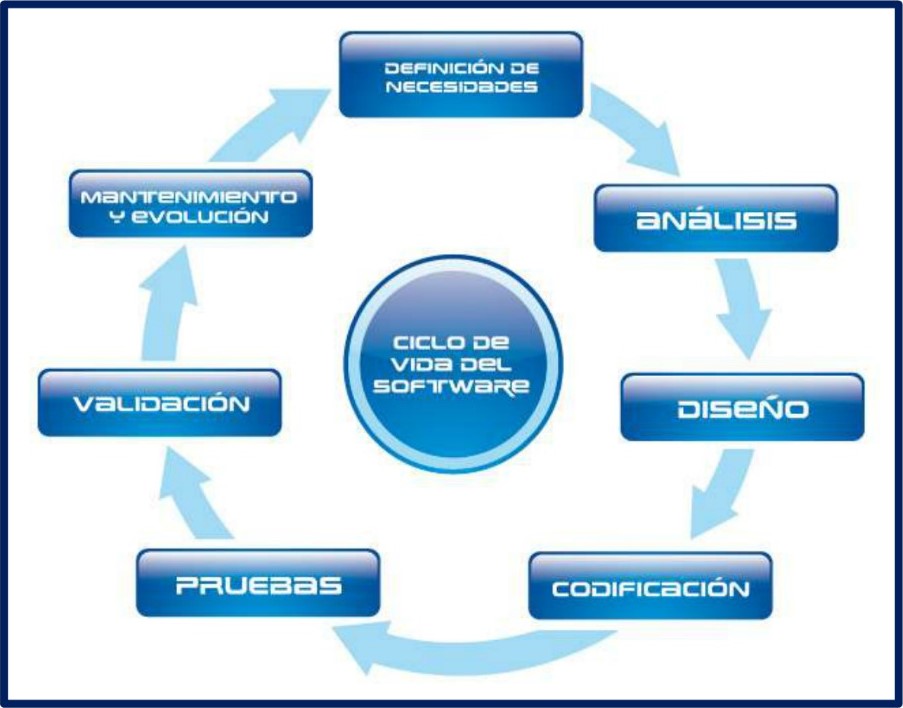
El modelo CMMI, permite mejorar la forma de trabajar en los diferentes negocios y delimitar objetivos, no debe considerarse como una certificación. Este modelo no detalla el desarrollo de los procesos. Solo define "el qué" mas no demuestra "el cómo", razón por la cual es poco adecuado requiere ser apoyado para su uso eficiente. Su principal fortaleza es haber sido probado en la industria y verificar que mejora la madurez y el rendimiento de las organizaciones, contrariamente no minimiza una mala gestión o decisiones estratégicas equivocadas.

El modelo ISO/ IEC 15504, es un estándar SPICE y provee procesos de ingeniería, cliente-servidor, gestión de la organización y del soporte, nace debido a la alta competencia presente en el mercado de desarrollo de software, y da frente a la complicada actividad de identificar los riesgos, cumplir con el calendario, controlar los costos y mejorar la eficiencia y calidad, conjuntamente ejecuta una valoración de los procesos de software concerniendo con el conocimiento y la experiencia dejada por otros modelos.

IT Mark es reconocido como el primer modelo de calidad internacional planteado concretamente para las pequeñas y medianas empresas del rubro de las Tecnologías de la Información y las Comunicaciones.

Actualmente, las empresas tienen e incorporan a la organización, diferentes tipos de herramientas que están disponibles en el mercado, con la finalidad de cumplir con las diferentes actividades. Este grupo de herramientas, por lo general permanecen desvinculadas unas con otras, propiciando la dificultad para gestionarlas, mantenerlas y controlar, la tarea está en integrarlas y complementarlas.





IV. MÉTRICAS DE SOFTWARE.

OBJETIVO GENERAL.

Aplicar las métricas de calidad de software para permitir la organización y eficacia de un proceso de software existente, sobre los programas, documentación, desarrollo y mantenimiento del mismo.

OBJETIVOS ESPECÍFICOS.

* + Reconocer los objetivos de las métricas de calidad de software.
  + Integrar el uso de métricas de calidad de software. Reconoce el modelo de Calidad de proceso.

4.1. ACTIVIDAD PRÁCTICA 8.

En esta octava actividad se utiliza el software Selenium para probar la ejecución de múltiples puntos de pausa (Break point) durante la ejecución de un script de prueba. Estas pausas están orientadas a generar puntos de "stop' sobre las líneas de código, determinadas por el ejecutor de la prueba. Aplicar múltiples puntos de pausa.

Paso 1. El software Selenium permite al usuario aplicar múltiples puntos de interrupción en un solo script de prueba. Una vez que se inicia la ejecución de la primera parte del código de prueba, el script se detendrá cuando alcanza el punto de pausa.

Paso 2. Para ejecutar estos puntos de prueba, se requiere que el usuario ejecute cada uno de ellos de forma separada. En el dibujo se muestra un caso con más de un punto de prueba

|  |  |  |
| --- | --- | --- |
| Command | Target Value  /ServiceLo in? | |
| assertTrtle | Sign in - Google Accounts | |
|  | id: Email | TestSeIentumi607@ mail.com |
| '22 type | id: Passwd | TestSelenium |
| clickAndWait | id: signln |  |

Paso 3. Verifique que en el script de prueba anterior, la ejecución se detiene en la línea:

* "assertTitle / Sign in — Google Accounts"

Paso 4. Después de ejecutar totalmente cada etapa de prueba, el control pasa a la siguiente etapa de prueba en la secuencia:

* "type I id = Email / TestSelenium1607@gmail.com"

Se vuelve a detener en:

* "type I id = Passwd / TestSelenium"

Paso 5. Revise esta función y permita al usuario realizar la ejecución de cada paso del testeador de manera más efectiva.

Paso 6. Considere que puede contar con tantos puntos de pausa como requiera el script.

Paso 7. Verifique que los puntos de ruptura puedan borrarse de la misma forma en la que se generaron.

Usando "Find Button" (Botón de búsqueda).

Paso 1. Uno de los puntos importantes de las pruebas de scripts bajo Selenium es encontrar y localizar elementos web en una página web. A veces, hay elementos web que tienen propiedades análogas asociados con ellos, por lo que es difícil para un usuario identificar un elemento web en particular.

Paso 2. Para solucionar este problema, Selenium proporciona el botón Buscar. El botón Buscar (Find) se utiliza para determinar el valor localizado de manera correcta e identifica el elemento web designada en la interfaz gráfica de usuario.

Paso 3. Considere el script de prueba creado. Seleccione cualquier comando y observe el cuadro de texto de destino (Target). Haga clic en el botón Buscar (Find) al lado del cuadro de texto Destino (Target).

Paso 4. Observe en el computador, que el elemento web correspondiente se resalta en amarillo con un borde de color verde fluorescente alrededor. Si no se resalta, el usuario deberá corregir el problema y tendría que otorgar algún otro valor al localizador.

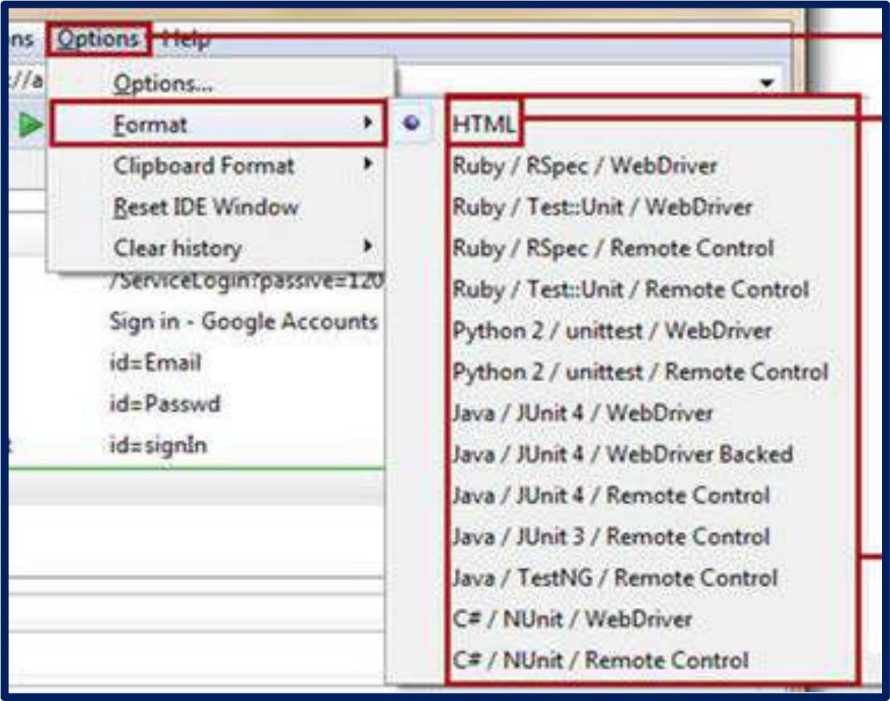
4.2. ACTIVIDAD PRÁCTICA 9.

Conversión del scripts de Selenium IDE a otros lenguajes.

En esta séptima actividad se utiliza el software Selenium para la conversión de sus archivos a otros lenguajes de programación y poder utilizarlos en otros formatos (como HTML).

Paso 1. Haga clic en la pestaña de opciones (Options) en la barra de Menú y seleccione el formato de lenguaje de programación de la opción de formato (Format) con el fin de convertir el script de prueba al formato deseado.

Paso 2: Tan pronto como selecciona el formato del lenguaje de programación deseado ("java / junit4 / webdriver" para este caso), aparece un cuadro de mensaje que dice changing format is now marked experimental, si continúa, la grabación y reproducción presentaran errores y no podrá trabajar.



Paso 3: A continuación para evitar que los cambios efectuados se pierdan, por precaución debe de hacer una copia del código script en un editor de texto.

Paso 4: Finalmente aparecerá el mensaje "do you still want to proceed?" (¿Todavía quiere continuar?), haga click en "ok " si está seguro de continuar y convertir el código por defecto en HTML en algún otro. Para comprobar, verifique el código siguiente:



Paso 5: Verifique que la información del escritorio de su computador presente información del nuevo código convertido al lenguaje JAVA. Puede comparar con la figura de la página anterior.

4.3. MÉTRICAS DE SOFTWARE.

El software y su medición.

El objetivo primordial de la ingeniería del software es promover un método, programa o producto con un alto nivel calidad. Para cumplir este fin, los desarrolladores de software utilizan métodos efectivos al lado de herramientas actuales dentro un ámbito perteneciente a un proceso maduro frente al desarrollo del software. En paralelo a este contexto, un efectivo desarrollador de software y administrador de técnicas de software ha de medir si hay un alto nivel de calidad. En este capítulo se presentan un grupo de métricas de software que pueden emplearse para la valoración del nivel cuantitativo de la calidad alcanzado por el software.

El producto de software, puede ser de buena calidad y de mala calidad, esta calidad está compuesta de distintos aspectos, que van desde el estético, (la apariencia y agrado que tiene frente al usuario), funcionalidad (facilidad de uso), eficiencia (fácil de ejecutar, con rapidez y precisión los procesos), etc.

Lo que distingue al software de otros productos industriales es que no es de naturaleza material, no se puede tocar. Por tanto no resulta viable hacer una valoración del mismo en base a una impresión rápida o análisis del aspecto ni en base al coste de materiales componentes.

Métrica.

Es un concepto extraído de la lengua y literatura. En software, la palabra métrica se refiere a medir el software a partir de medidas preestablecidas, por ejemplo el número de líneas que contiene un código. En algunos casos se utiliza el término "Indicadores" del software. Un buen número de

desarrolladores lo usan como sinónimos mientras que otros no y consideran significados distintos. A continuación se presentan algunas métricas o indicadores:

* índice de productividad.  Tasa de defectos.

También se define un conjunto de factores de calidad, que debe ser el primer paso hacia el desarrollo de métricas de la calidad del software. Estos permiten evaluar el software desde tres ángulos diferentes:

* Operación del producto (utilizándolo).
* Revisión del producto (cambiándolo).

 Transición del producto (modificándolo)

La ISO 9001 es una norma para asegurar la calidad en las empresas de cumplimiento optativo, donde se definen los procesos que se ejecutan en una empresa que requiere estandariza y controlar sus procesos. El cumplimiento de la norma no garantiza obtener un producto final de calidad. Simplemente garantiza que la empresa ha adoptado una organización definida y controlada.

CMMI sirve para evaluar si una organización conserva ciertos niveles de calidad en relación al software. CMMI es un estándar que un gran número de empresas lo ha asumido como modelo, pero hay un buen número que aplica otras normas y modelos, como CMM, ISO 9001, SPICE, psp, TSP, ISO 20.000, ITIL, etc.

Si se desea la acreditación bajo CMMI se debe pasar por una evaluación, la cual tiene diferentes niveles dentro del modelo CMMI, y está de acuerdo al cumplimiento de requisitos a cumplir más o menos exigentes, a partir de lo que se espera como objetivo en la empresa. El CMMI puede implantarse tanto en empresas grandes como en PYMES.

CMMI no tiene costo y se puede descargar desde Internet (en pdf o doc.). También se encuentra en la web una guía de CMMI para pequeñas organizaciones.

Las métricas sirven para hacer mediciones del software. Cuando se implanta un modelo de calidad como CMMI, se utilizan las métricas con el fin de comprobar que se ejecuten los cambios de manera real en el software. Se espera que al implantar este tipo de sistema haya mejoras reales que puedan medirse.

Finalmente una buena cantidad de empresas hacen la evaluación sobre si una nueva tecnología les aporta mejora y calidad utilizando las métricas. Por ejemplo, la disminución considerable del número de líneas de código en un programa o aplicación.

Medición de la calidad del software.

Actualmente existen muchas medidas de la calidad de software, la corrección, facilidad de mantenimiento, integridad y facilidad de uso suministran indicadores útiles para el equipo del proyecto. Muchos especialistas en el tema sugieren definiciones y medidas aplicables, tales como:

* Corrección.
* Facilidad de mantenimiento.
* Integridad.
* Fácil de usar.

A una aplicación le corresponde trabajar correctamente de lo contrario no entregara valor a sus usuarios. Se debe entender que la corrección es el grado en el que el software realiza una tarea requerida. La medida más común de corrección son los defectos por KLDC, en donde un defecto se define como una falla verificada de conformidad con los requisitos.

En cuanto al mantenimiento del software, es el que requiere más esfuerzo que cualquier otra actividad del desarrollo de software. El proponer un fácil mantenimiento permitirá corregir un programa al encontrar un error, se puede adaptar si su entorno cambia u optimizar si el cliente desea un cambio de requisitos.

Sobre la integridad, en la actualidad sobre los intrusos informáticos y los virus, la integridad del software es un papel de gran importancia. Este atributo evalúa la habilidad de un sistema para dar frente a los ataques, ya sean, accidentales como intencionados, en contra la seguridad. Los ataques se presentan ante cualquiera de los tres componentes del software, ya sea en los programas, datos o documentos.

Por último, la facilidad de uso hace referencia a que tan "amigable es con el usuario final". Si el programa no es "amigable con el usuario", está condenado a fracasar, así realice tareas valiosas.

Métricas en práctica.

Las pruebas de las métricas de software se deben acoplar a partir de que el software adquiere forma. Estas pruebas y puesta en marcha de las métricas son fundamentales. En esta etapa de pruebas se utilizan los denominados casos de prueba: una secuencia de pasos que debe llevarnos a un resultado esperado.

El tiempo utilizado en las pruebas de software repercute en costos económicos para la organización, pues incluye gastos de personal a cargo de las pruebas como retrasos en la entrega del producto al cliente. Con el fin de mejorar estos procesos se aplican metodologías de pruebas y se registran los resultados generándose con el fin de contar con un historial.

Es ideal que, nunca se entregue un proyecto que no haya sido pasado por pruebas rigurosas. En la praxis, el personal a cargo del desarrollo considera más importante el tiempo de entrega, dejando de lado la ejecución de pruebas y métricas de software.

Ante esta situación, lo ideal es contar con un equipo de prueba distinto al equipo de desarrollo y que componga una incidencia por cada anomalía encontrada en el software. Lo frecuente es que los mismos desarrolladores realicen las pruebas.

A los especialistas en estas pruebas son conocidos como "tester" y los errores que encuentra "bugs", términos de origen inglés.

Métricas aplicables a los modelos.

En la presente etapa se consideran los requisitos y se instaurará las bases del diseño. Por lo cual es necesario contar con una visión interna a la calidad del modelo de análisis. Por al contrario se cuenta con pocas métricas de análisis y especificación, pero es de conocimiento la posibilidad de adaptar métricas obtenidas para la aplicación de un proyecto, con el fin de que examinan el modelo de análisis para predecir el tamaño del sistema producido, se puede predecir que el tamaño y la complejidad del diseño estén directamente relacionadas.

A continuación se presentan diversas métricas:

* Métricas basadas en la función.
* Métricas Bang.
* Métricas de la Calidad de Especificación.
* Métrica del modelo del diseño.
* Métricas de diseño de alto nivel.
* Métricas de diseño en los componentes.
* Métricas de cohesión.

 Métricas de acoplamiento.

* Métricas de complejidad.
* Métricas de diseño de interfaz.
* Métricas de código fuente.
* Métricas para pruebas.
* Métricas de mantenimiento.

Métricas basadas en la función.

Esta métrica de punto de función (PF) se utiliza como medio para pronosticar el tamaño de un sistema que se va a producir de un modelo de análisis. Para instruir el empleo de la métrica basada en punto de fusión, se supondrá una sencilla representación del modelo de análisis, en donde se simboliza un diagrama de flujo de datos, de una función para una aplicación cuyo nombre es "Hogar Seguro".

Esta función administra la relación con el usuario, utilizando una contraseña de habilitación del sistema, lo cual permite consultas sobre el estado de las zonas de seguridad y la inclusión de varios sensores de seguridad.

Por último, esta función consigna una serie de mensajes de petición y envía señales adecuadas para el control a diferentes elementos del sistema de seguridad.

Métrica Bang.

La métrica Bang se emplea para el desarrollo de una indicación de la extensión del software a producir a partir del modelo de análisis. Esta métrica fue desarrollada por Tom De Marco.

La métrica Bang es una premisa, independiente de la implementación, y de la extensión del sistema. Para su cálculo, el desarrollador de software debe valorar en primer lugar un grupo de primitivas y elementos del modelo de análisis que no pasen por otra subdivisión.

Métricas de la Calidad de Especificación.

Esta métrica es correspondiente con la especificación de las siguientes características:

* Especificidad.

 Corrección.

* Compleción.
* Comprensión.
* Capacidad de verificación.
* Consistencia externa e interna.
* Capacidad de logro.
* Concisión.
* Trazabilidad.
* Capacidad de modificación.
* Exactitud.

 Capacidad de reutilización.

Todas las características anteriores pueden ser de naturaleza cuantitativa, y se pueden representar usando una o más métricas.

Métrica del modelo del diseño.

Bajo la métrica del modelo de diseño para software, se considera que estas no son perfectas; un buen número de especialistas explican que se requiere mayor investigación y experimento hasta que se utilicen bien las métricas de diseño. Por otro lado el diseño sin medición es totalmente una opción inaceptable. Este tipo de métrica permite al diseñador un mejor panorama interno lo cual beneficia en un mejor nivel de calidad del software producido.

Métricas de diseño de alto nivel.

Las métricas de diseño de alto nivel, se orientan y dedican a las características de la estructura del programa dando relevancia a la estructura arquitectónica y eficiencia de los módulos. En otras palabras este tipo de métricas son de caja negra, ya que no se requiere ningún conocimiento del trabajo interno de ningún modo en particular del sistema. A partir de ello se proponen tres medidas de complejidad del software:

* Complejidad estructural.
* Complejidad de datos.

 Complejidad del sistema.

Métricas de diseño en los componentes.

Las métricas de diseño a nivel de componentes se basan en las características internas de los diversos elementos (componentes) del software e incluyen las medidas de:

* Cohesión.
* Acoplamiento.
* Complejidad del módulo.

Estas tres medidas proveen de ayuda al desarrollador de software para el emitir un juicio sobre la calidad de un diseño a nivel de componentes (elementos del software).

Estas métricas son de "caja blanca" ya que necesitan el saber cuál es el trabajo interno del módulo en cuestión. Estas métricas son aplicables a partir de que se ha desarrollado un diseño procedimental. Cabe la posibilidad que se pueden retrasar hasta tener disponible el código fuente.

Métricas de cohesión.

Las métricas de cohesión definen una recopilación de distintas métricas que suministran una indicación de la cohesión de un módulo. Estas métricas presentan cinco conceptos y medidas, los cuales son:

* Porción de datos. - Símbolos léxicos.
* Señales de unión.
* Señales de súper unión.  Cohesión.

En primer lugar, con respecto a la porción de datos, hace referencia a una marcha atrás por un módulo que busca encontrar valores de datos que afectan a la localización del módulo en el que empezó la marcha atrás. Sobresale que permite definir tanto partes de programas (centrados en enunciados y condiciones) como partes de datos.

En segundo lugar, para los símbolos léxicos de datos. Las variables definidas para un módulo pueden definirse como señales de datos para el módulo. En tercer lugar, en cuanto a las señales de unión. El conjunto de señales de datos que se encuentran en uno o más porciones de datos. En cuarto lugar, para las señales de súper unión son comunes a todas las porciones de datos de un módulo. Por último, la cohesión relativa de una señal de unión es directamente proporcional al número de porciones de datos que liga.

Métricas de acoplamiento.

Las métricas de acoplamiento de módulo proporcionan una indicación de la "conectividad" de un módulo con otros módulos, datos globales y entorno exterior. Se propone una métrica para el acoplamiento del módulo con el fin de combinar el:

* Acoplamiento de flujo de datos y de control.
* Acoplamiento global.
* Acoplamiento de entorno.

Los medios requeridos para calcular el acoplamiento de módulo vienen dados a partir de cada uno de los tres tipos de acoplamiento apuntados anteriormente.

Métricas de complejidad.

Para determinar la complejidad del flujo de control del programa se pueden calcular una variedad de métricas del software. La mayoría se basan en una representación denominada grafo de flujo, un grafo es una representación compuesta de nodos y enlaces. Cuando se dirigen los enlaces (es decir las aristas), el grafo de flujo es un grafo dirigido.

Se identifica un alto número de usos para las métricas de complejidad, que se emplean para predecir información sobre la fiabilidad y mantenimiento de sistemas software, también suministran la información durante el proyecto de software con el fin de proveer control a la actividad de diseño, en las pruebas y mantenimiento, proporcionan información sobre los módulos software para ayudar a resaltar las áreas de inestabilidad.

Métricas de diseño de interfaz.

Sobre las métricas de diseño de interfaces hombre-máquina, existe muy poca información que proporcione una visión interna de la calidad y facilidad de empleo de la interfaz. Los especialistas sugieren la coexistencia de la representación (CR) como una valiosa métrica de diseño para interfaces hombre-máquina. Una GUI (Interfaz Gráfica de Usuario) típica usa entidades de representación, iconos gráficos, texto, menús, ventanas y otras para ayudar al usuario a completar tareas.

Métricas de código fuente.

Este tipo de métricas utiliza las basadas en la teoría de Halstead. En la ciencia del software es la más conocida y estudiada para obtener medidas compuestas de la complejidad. La disciplina del software señala leyes cuantitativas para el desarrollo del software de computadora. Esta teoría parte de que el hombre sigue un conjunto de reglas bastante rígido (tal cual algoritmo). Entonces el software utiliza un grupo de medidas primitivas que se recogen una vez que se ha generado o estimado el código después de completar el diseño.

Métricas para pruebas.

La mayoría de las métricas anteriores se concentran en el proceso de pruebas, no en las características técnicas de las pruebas mismas. La persona a cargo de las pruebas debe asegurarse del análisis, diseño y código para que les guíen en el diseño y ejecución los casos de prueba. Las métricas fundamentadas en la función se emplean para predecir el esfuerzo global de las pruebas. El equipo posteriormente podrá predecir el producto final a partir de las características del proyecto actual.

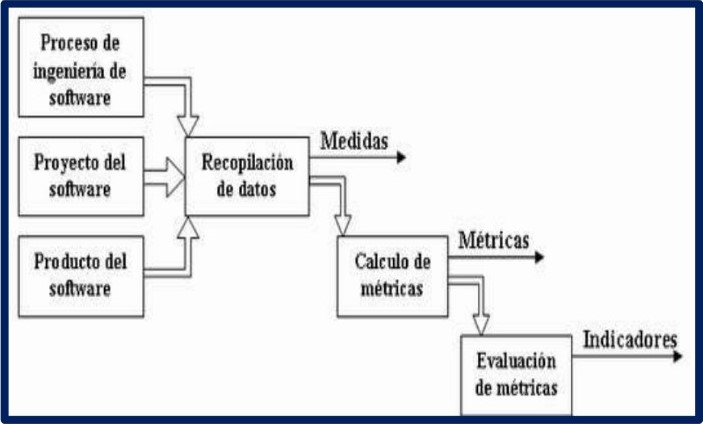
Métricas de mantenimiento.

Las métricas de mantenimiento están diseñadas explícitamente para actividades de mantenimiento. La norma y estándar 982.1-1988 de la IEEE sugiere un índice de madurez del software (IMS) que proporciona una predicción de la estabilidad de un producto de software (reconociendo las mejoras que ocurren con cada nuevo upgrade del producto).

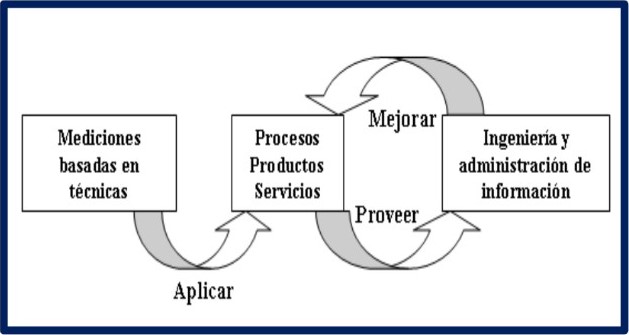
La siguiente tabla presenta un ejemplo de métricas de calidad externa. Características y confiabilidad.

|  |  |  |
| --- | --- | --- |
|  | Confiabilidad | |
| Elemento | Métrica | Propone medir |
| Madurez | Grado de solución ante  fallos totales | ¿cuántas condiciones de fallo total están resueltas? |
| Erradicación de fallos | ¿Cuántos fallos han sido corregidos? |
| Densidad de fallos totales contra casos de prueba | ¿cuántos fallos totales fueron detectados durante un período de pruebas definido? |
| Madurez de las pruebas | ¿Está bien probado el producto? |

El siguiente diagrama presenta una propuesta de métrica para el perfeccionamiento de gestión de un proyecto de software



Propuesta de métrica de mejora en una organización.



v. INTEGRACIÓN DE MODELOS DE MADUREZ DE CAPACIDADES 

CMMI.

OBJETIVO GENERAL.

Aplicar los conceptos del CMMI y sus componentes para la mejora y evaluación de procesos en el desarrollo de software.

OBJETIVOS ESPECíFlCOS.

* + Reconocer los conceptos del CMMI.
  + Reconocer los componentes del CMMI.

5.1. ACTIVIDAD PRÁCTICA 10.

En las actividades anteriores, se explicó cómo crear y grabar scripts automatizados utilizando las principales funciones y características del software Selenium. En esta actividad se demostrara el manejo de características avanzadas del Selenium.

Cómo usar Firebug para Crear scripts en Selenium.

Firebug es un add-on de Mozilla Firefox. Esta herramienta ayuda en la identificación o para ser más concreto en la inspección del HTML, CSS y los elementos de JavaScript en una página web. La identificación de los elementos se da forma única en una página web. Los elementos se pueden encontrar únicamente en función de sus tipos de localización que estaríamos discutiendo más adelante en este tutorial.

Instalación de Firebug.

Paso 1. Inicie el navegador Mozilla Firefox y diríjase a la página web del enlace: https://addons.mozilla.org/en-US/firefox/addon/firebug/ aquí se encuentran los complementes necesarios para instalar el Firebug. La URL anterior nos lleva a Firefox sección de complementos.

Paso 2. Haga clic en "Add to Firefox" botón presente en la página web. Consulte la siguiente figura para la misma.

|  |  |  |
| --- | --- | --- |
| @ADD-ONSEXTENSIONS I THEMES I COLLECTIONS I MORE.„  » Extensions » Firebug  Firebug 1.12.6 NO RESTART by Joe Hevv-itt, Jan Odvarko, robcee, FirebugWorkingGroup  Firebug integrates v.ith Firefox to put a wealth of de', while you browse. You ran edit, debug, and monitor any web page...   |  |  | | --- | --- | | |  | | --- | | + Add to Firefox | | |

Paso 3. Una vez que hacemos clic en el botón "Add to Firefox", podría aparecer un cuadro de alerta de seguridad, haga clic en el botón "Permitir" para continuar.

Paso 4. Ahora Firefox descarga el complemento muestra una barra de progreso de este proceso.

Paso 5. En cuanto finalice la descarga, aparecerá la ventana de instalación del software. Ahora haga clic en el botón "Install NoW'.

|  |  |
| --- | --- |
| Firebug  Joe He•xitt,  while you any web pase. |  |
| + Addt0V |
|  |

Paso 6. Una vez finalizada la instalación, un pop-up aparecerá indicando que el Firebug se ha instalado correctamente. Ahora debe cerrar esta ventana.

Nota: A diferencia de Selenium IDE, no estamos obligados a reiniciar el Firefox para completar la instalación de Firebug.

Paso 7. Para iniciar Firebug, puede optar por cualquiera de las siguientes formas:

* Pulse FIZ
* Haga clic en el icono de Firebug presente en la esquina extrema superior derecha de la ventana de Firefox.



* Haga clic en la barra de menú de Firefox -> Web Developer -> Firebug -> Open Firebug

Paso 8. Ahora el Firebug se puede ver en la parte inferior de la ventana de Firefox.

Ahora que se ha descargado e instalado Firebug, sigue la creación de los diferentes tipos de localizadores usando Firebug.

5.2. ACTIVIDAD PRÁCTICA 11.

A diferencia del Selenium, el Firebug, crea scripts automatizados manualmente, mediante la unión de múltiples pasos con el fin de formar una secuencia de comandos lógica y coherente.

Creación de scripts para Selenium usando Firebug.

Se explicará el proceso paso a paso. Para lo cual se requiere contar con el siguiente escenario:

* Abrir "https://accounts.google.com".
* Contar con un nombre de usuario y contraseña válida para iniciar sesión.

Paso 1. Iniciar el Firefox y ejecutar Selenium desde la barra de menú.

Paso 2. Escribir la dirección de la aplicación a examinar en el cuadro de texto Base URL. (Recuadro en color rojo)

|  |  |  |
| --- | --- | --- |
|  | | |
| Test Case |  | O |
|

Paso 3. Por defecto, el botón de grabación se encuentra en estado ON. Recuerde que el botón en OFF desactiva el modo de grabación. Coloque en ON, para dar inicio a la grabación de las interacciones con el navegador web. Paso 4. Abrir la dirección que se someterá a prueba con el Mozilla Firefox, en este caso:

## https://accounts.google.com

Paso 5. Ejecute Firebug desde el navegador web. Dar click en pestaña HTML como se muestra en la figura inferior:



Paso 6. Seleccione la primera línea de prueba del área Editor. En este momento se encuentra sin contenido. (Recuadro en color rojo).

|  |  |  |  |
| --- | --- | --- | --- |
| Table I Source | |  |  |
|  | |  |  |
|  | Command | Target | Value |
|  | | |

Paso 7. Haga click y de la caja de texto Command seleccione la opción Open del panel Editor. El comando "Open" abre la URL especificada en el navegador web.

Paso 8. Ubique esta imagen en el explorador y copie el contenido que se muestra en la figura:

|  |  |  |
| --- | --- | --- |
|  | | |
|  |  | 'cten |
| é.x•vidtb: |

Paso 9. Seleccione la segunda línea de prueba del Editor.

Paso 10. Haga click y de la caja de texto seleccione "assertTitle" en el panel Editor. El comando "assertTitle" devuelve el título de la página actual y lo compara con el título especificado.

Paso 11. Pegue el título copiado en el paso 8 en el campo Target de la segunda línea.

Paso 12. Ahora seleccione la tercera línea de prueba vacía del panel Editor

Paso 13. Seleccione "type" en el cuadro de texto del comando. El comando "type" introduce un valor en el elemento web en la interfaz gráfica de usuario.

5.3. CMMI INTEGRACIÓN DEL MODELO DE MADUREZ DE CAPACIDADES.

Entendiendo los modelos de madurez.

Hoy en día las empresas anhelan otorgar superiores servicios y productos en menor tiempo y al menor costo. Por otro lado, y en paralelo al mundo de la alta tecnología actual, la mayoría de empresas han ido entregando productos y servicios con mayor valor y complejidad. Actualmente es difícil encontrar que las empresas generen por si solas todos los elementos necesarios para entregar un producto o servicio. Casi siempre, una parte se fabrica en la empresa y otros se adquieren de terceros; para luego integrar todos los elementos y dar por entregado el producto o servicio final. Razón por la cual estas empresas deber contar con la capacidad de gestionar y tener el mando y supervisión de este elaborado proceso de dar vida a un producto servicio.

La problemática que afrontan estas empresas requiere soluciones que abarca a toda la empresa y que demandan una interacción cada vez más integrada. La gestión eficaz de los activos de la organización es crítica para el éxito de su actividad. En principio, las empresas que generan productos y servicios requieren la forma para gestionar una interacción más integra en sus acciones de mejora, como fracción para alcanzar sus objetivos estratégicos.

Actualmente el mercado, cuenta con guías, metodologías, estándares, y modelos que apoyan a las empresas para mejorar sus actividades de desarrollo de productos y servicios. Por al contrario, la mayoría de las aproximaciones de mejora disponibles se centran en una parte específica de su actividad y no adoptan una aproximación sistémica a los problemas a los que se enfrentan la mayoría de las organizaciones. Se centran en la mejora del área de negocio, equivocadamente estos modelos han perdurado las vías y las brechas existentes en el corazón de las organizaciones.

El modelo de madurez de integración de capacidades otorga la posibilidad de evitar o eliminar estas vías y brechas, sustentándose en los modelos integrados que trascienden disciplinas. El modelo de madurez de integración de capacidades para desarrollo examina las buenas prácticas referentes a las acciones de desarrollo y mantenimiento aplicadas a productos y servicios.

Contempla las tareas asociadas el ciclo de vida del producto desde el inicio hasta la entrega y el mantenimiento de un proyecto. Dando prioridad al trabajo ineludible para construir y mantener el producto perfecto.

Aplicación sobre los Modelos de madurez y de capacidad.

El Instituto de Ingeniería de Software, en sus estudios relacionados a la ayuda de empresas para que desarrollen y mantengan sus productos y servicios de calidad, ha detallado diversas dimensiones, sobre las que una empresa debe considerar para obtener mejoras en su desarrollo.

Son tres las dimensiones claves sobre las cuales generalmente se concentran las empresas, estas son:

* Personas.

 Métodos y procedimientos.

* Herramientas y equipamiento.

|  |
| --- |
| Métodos y procedimientos que definen las relaciorvs entre tareas  Personas con habilidades, formaciónHerramientas y moüvacióny equipamiento |

Figura de las tres dimensiones críticas.

¿Qué le da sostenibilidad a todo este conjunto? Los procesos manejados en la empresa. Dichos procesos facilitan el alineamiento y la forma de operar de la empresa, su desarrollo y la manera de incorporar los conocimientos de cómo obtener mejores resultados. Los procesos tambien facilitan la explotación mejorada de los recursos y entender las directrices de su actividad.

Esto no quita importancia a otros componentes como las personas y la tecnología.

En el mundo actual la tecnología es cambiante y cada vez lo hace de una manera más frecuente. En paralelo, las personas laboran normalmente para distintas empresas durante el desarrollo de su vida profesional.

Hoy en día se tiene un mundo totalmente dinámico. Con un centro en el proceso que proporciona la infraestructura necesaria para dar cara al actual mundo cambiante, potenciar al máximo la productividad de las personas y aplicar de manera adecuada la tecnología para mantenerse a la vanguardia.

El sector de la industria en Tl, reconoce desde hace un buen tiempo la jerarquía de la eficiencia y de la eficacia del proceso. En la actualidad, muchas empresas industriales y empresas del sector servicios reconocen la preponderancia de contar con procesos de calidad.

El proceso permite que los miembros de una organización puedan alcanzar los objetivos estratégicos facilitando su trabajo de una manera más inteligentemente. Los procesos eficaces también suministran una vía para la introducción y uso de nuevas tecnologías de forma que accedan a responder de una mejor manera a los objetivos estratégicos de la empresa.

Walter Shewhart, en 1931, comenzó a proponer la mejora de procesos iniciando los principios del control estadístico de la calidad. Dichos principios los mejoro W. Edwards Deming en 1986, Phillip Crosby en 1979 y Joseph Juran en 1988. Watts Humphrey, Ron Radice y otros tambien aportaron y lo aplicaron al software durante su permanencia en IBM y en el Instituto de Ingeniería de Software. Humphrey en su libro Administración de Procesos de Software (Managing the Software Process), explica y define los principios que sirvieron de base para muchos de los modelos de madurez y de capacidad (CMMs).

El Instituto de Ingeniería de Software (SEI) tomo en cuenta de la gestión de proceso el siguiente enunciado, "la calidad de un sistema o de un producto está muy influenciada por la calidad del proceso empleado para desarrollarlo y para mantenerlo". El apego a este principio fundamental forma la base de los movimientos de calidad de todo el mundo, como lo anuncia la Organización Internacional de Estándares y la Comisión Internacional de Electrónica, (ISO/IEC) en sus diferentes normas y estándares.

Perspectivas del CMMI.

Para poder entender lo que se viene en software, se debe considerar como se inició todo, para llevarnos a la situación actual. A continuación se explicara los inicios del CMMI, las primeras ideas para su diseño, sus detractores actuales, y lo que se espera del modelo de madurez.

Existen cinco ideas principales extraídas de una gama amplia de dominios que inspiraron el modelo original y su proceso de evaluación CMMI. Las ideas que iniciaron todo fueron:

* Planificación, seguimiento y gestión del proyecto.
* Definir requerimientos y controlar las configuraciones.

 Evaluar el desarrollo del proceso.  Medir la calidad y la mejora continua.  Mejora evolutiva

Hoy en día se sabe que estos conceptos son aplicables al desarrollar software y sistemas, pero en la década de los 80 esto no era así, o posteriormente cuando se da inicio al desarrollo del CMM, o cuando por primera vez se aplica en IBM en el área de desarrollo de software en 1966.

Planificación, seguimiento y gestión del proyecto.

Según Judson Neff la única forma de gestionar operaciones altamente elaboradas era mediante el control que contenga una planificación detallada y precisa, condición que se aplica cuando se requiere recuperar proyectos que se encuentran con un alto grado de desorganización. Razón por la cual se recomienda como primer punto la planificación, seguimiento y gestión del proyecto, lo cual garantiza el dominio y control total de la gestión del proyecto.

Definir requerimientos y controlar las configuraciones y los cambios.

Aquí se deben considerar dos puntos muy importantes. En primer lugar, se deben permitir los cambios necesarios para cumplir con los requerimientos, debido a que, si estos no están bien definidos se corre el riesgo de desarrollar un producto equivocado y perder todo el esfuerzo en su construcción. En segundo lugar, si hay cambios, es necesario tener un riguroso control sobre ellos, de lo contrario nunca acabará el desarrollo de un producto.

Evaluar el desarrollo del proceso.

En IBM en el área encargada del desarrollo y la producción, se solicitó la identificación de las dificultades el proceso de fabricación de semiconductores de la marca. En ese entonces se utilizó un método para dicha evaluación a partir de uno ya ensayado y aplicado en los empleados, este consistía en dar solución a sus propios problemas.

En la fábrica de semiconductores de IBM en USA se explicó que cabía la posibilidad de comprar componentes importados de Asia (Japón) a precios por debajo del valor de producción generado en USA. Pero ni el mismo IBM aceptaría tal situación.

Si el área a cargo del proceso de fabricación de semiconductores no se hacía productiva, pasaría a mejor vida. El equipo a cargo evaluó cada una de sus

actividades, y dio solución al problema de costos, convirtiéndose más tarde en el productor de semiconductores de menor coste de fabricación en el mundo.

Medir la calidad y la mejora continua.

Los ingenieros a cargo del área de fabricación de semiconductores lograron controlar sus costes a través de la gestión de la producción. Esto se logró porque se aplicó el medir, dar seguimiento, y analizar cada etapa del trabajo de cada uno de los componentes del equipo.

Mejora evolutiva.

La gestión de la calidad bajo el modelo de Phil Crosby, empleaba un modelo de madurez de cinco niveles. Lo cual se aplicó para disminuir tiempos en la mejora del desarrollo de software de IBM.

La primera evaluación de un laboratorio era generalmente exitosa, pero la segunda y tercera no lo eran. La limitación del modelo de niveles de madurez de Crosby es que se basaba en juicios de actitudes subjetivas y no se tomaban en cuenta las actividades específicas de software.

Integración del Modelo de Madurez en DOD.

Había que mejorar la calidad de proveedores de software para la Fuerza Aérea de los EE.UU, con el fin de potenciar las capacidades de la organización. En ese entonces se percibía que las empresas que utilizaban las mejores prácticas técnicas y de gestión en sus proyectos de desarrollo estaban resueltas a efectuar el mejor trabajo, así que se propuso un cuestionario de 85 preguntas que cubría los siguientes temas:

* Planificación del proyecto.
* Seguimiento del proyecto.

 Gestión del proyecto.

* Gestión de los requerimientos.
* Control de configuración.
* Medida de la calidad.
* Mejora continua de los procesos.

Estas preguntas se agruparon en un marco de trabajo de madurez y terminaron convirtiéndose en la primera versión de lo que actualidad es el CMMI o Modelo de madurez de Integración de Capacidades.

Retos del modelo de madurez.

Las opiniones que dieron vida a CMMI aparecieron de diversos puntos y se enriqueció de los saberes previos de personas de diferentes disciplinas. Por al contrario actualmente existen tres puntos en los cuales se debe seguir innovando.

En primer lugar, con la globalización y la competencia del mercado, las empresas se interesan generalmente en los niveles de madurez más que en la capacidad de proceso. Está claro que los niveles de madurez carecen de capacidad para medir de manera clara las aptitudes organizativas, por al contrario pueden indicar áreas de proceso arriesgadas o redireccionar la mejora de los procesos, puntualizando un conjunto mínimo de actividades necesarias. Actualmente se observan casos en las cuales la calificación de madurez elevada no está acorde con las prácticas efectivas de elevada madurez. Lo cual necesariamente no significa que el proceso de evaluación sea defectuoso o que las organizaciones cuenten la verdad, por al contrario simplemente es un indicativo de que, el marco de madurez no es lo suficientemente fino para hacer un examen pleno de todas las prácticas de la organización.

En segundo lugar, la siguiente cuestión hace referencia al ajuste del marco de CMMI y los métodos utilizados al evaluar este problema. Si no se ejecutan cambios, es probable esperar una mayor cantidad de casos donde las mediciones de alta madurez no están en correspondencia con un funcionamiento mejor. Ante esta situación se sugiere lo siguiente:

* Si se requiere un verdadero control de una actividad que contiene un elevado nivel de complejidad, se deben planificar planes detallados y precisos.
* Se debe contar con la capacidad de medir y gestionar la calidad.

Esto debe considerarse un punto de partida o una guía para los equipos encargados del desarrollo de software para la aplicación de los principios en su trabajo.

El Instituto de Ingeniería de Software (SEI) desarrolló el proceso PSP (Personal Software Process) destinado a las personas y el proceso TSP (Team Software Process), que involucra a los equipos. Al aplicar tanto el PSP como el TSP, los evaluadores del modelo han descubierto estas prácticas por medio de las evaluaciones con CMMI. En conclusión, al parecer el PSP y el TSP apoyan en la adopción de prácticas maduras aplicables al desarrollo de software. Definitivamente, la adopción del PSP y TSP por parte del Instituto de Ingeniería de Software (SEI) beneficia al desarrollo de sistemas.

En tercer lugar, se da la cuestión, que hace referencia a la flexibilidad. El CMMI no enseña cómo hacer el desarrollo del software, solo indica lo que se debe hacer. A diferencia de, PSP y TSP que son específicos e indican cómo se deben realizar el plan del proyecto, su seguimiento y la gestión de la calidad.

Lo interesante de todo esto, es como incorporar estas prácticas y principios en el modelo y método de CMMI sin perder el punto al cual se quiere llegar y la vía

por donde conducir ello. El requerimiento implica enlazar los principios y prácticas probadas, sin restringir las posibilidades de desarrollo de las empresas a medida que avanza la tecnología. El Instituto de Ingeniería del Software (SEI) busca resolver estas cuestiones a la par de mejorar la eficacia en cuanto a métodos y modelos.

Proceso evolutivo de los modelos de madurez.

Está bastante claro que el CMM se encarga de la mejora de los procesos de una organización. Estos comprenden componentes primordiales de eficacia de los procesos diferentes ámbitos labores e indican la ruta de mejora evolutiva permitiendo el pase de procesos inmaduros a procesos disciplinados y maduros con una mejor calidad y eficacia.

El Instituto de Ingeniería del Software (SEI) da vida a la primera versión del CMM orientado al apoyo de empresas relacionadas al desarrollo de software en su publicación de 1995, "The Capability Maturity Model: Guidelines for Improvement the Software Process".

Esta primera aplicación puso en práctica los principios introducidos hace casi 100 años respecto de la mejora de procesos y ha trascendido a través del tiempo.

Actualmente las empresas han aumentado su productividad y hacen constante uso de técnicas para obtener una mejora de la calidad en sus productos, además de obtener mejoras en la duración del ciclo productivo, planificaciones y presupuestos más precisos y fiables. Al incorporar una nueva versión de los modelos CMMI, es recomendable ver la que se hasta este punto y dar alguna explicación sobre hacia dónde se dirigirá estos modelos.

Los modelos con niveles de mejora retoman una vez más el interés en la calidad de la producción. Al poco tiempo de crearse el SEI, la aviación norteamericana solicito identificar los mecanismos claves que tenía que cumplir un proveedor para entregar software confiable. Luego en 1991, se mejora este seguimiento de prácticas y de medición por medio de la incorporación de nuevas etapas para la mejora. Todo a partir del CMM original.

La confianza de este modelo para diversas áreas de la ingeniería origino nuevos esfuerzos de similar importancia con otros componentes dentro de la comunidad de desarrollo de productos. La preferencia por estos modelos para la mejora de los procesos de la ingeniería de sistemas dio como resultado el nacimiento de dos nuevos modelos producidos en el año de 1994. El CMM para la Ingeniería de Sistemas, lanzado por la Enterprise Process Improvement Collaboration (EPIC), con la aportes del SEI.

Otro modelo es el SECAM o System Engineering Capability and Assessment Method lanzado por el International Council on Systems Engineering (INCOSE). Después de cuatro años, ambos modelos fueron combinados con éxito en el Electronic Industries Alliance (EIA) Interim Standard 731 como producto del trabajo cooperativo de colaboración entre EIA, EPIC e INCOSE. Mientras que en 1996, se creó una similar al SW-CMM para cubrir las prácticas clave en la adquisición de software —el Software Acquisition Capability Maturity Model, o SA-CMM. El interés por preservar y realzar las capacidades del personal de ingeniería de desarrollo condujo al SEI a crear el Modelo de Madurez y de Capacidad de Personal (P-CMM) en 1995.

En el año 1996, el SEI tenía la responsabilidad de producir una nueva versión actualizada del SW-CMM, con el fin de entregar un modelo que contendría prácticas asistentes de ingeniería en un CMM para desarrollo de productos integrados. El gestor de esta investigación fue el Departamento de Defensa de los EE.UU, se consideró que estos esfuerzos deberían contar con un modelo integrado que actualmente se conoce como CMMI (Capability Maturity Model Integration).

La posibilidad de constituir un conjunto heterogéneo de modelos de madurez había sido justificada a inicios del mismo año por la Aviación Civil Norteamericana, que lanzo el modelo FAA-iCMM vl .0. (FAA Integrated Capability Maturity Model).

Respecto a los diferentes esfuerzos en el proceso y producto integrados (IPPD) también desarrollado por los militares norteamericanos y la industria, se definieron las primeras actividades relacionadas al CMMI considerándose principalmente la integración de la ingeniería de sistemas, la ingeniería del software y el desarrollo integrado de productos y procesos (IPPD, Integrated Process and Product Development).

Se presentaron inicialmente dos versiones del CMMI con antelación a los que sería la versión oficial lanzada en el año 2000, considerándose áreas de la ingeniería de sistemas, ingeniería del software, y de desarrollo integrado de productos y procesos. Con cambios de poca relevancia con respecto a futuras entregas (vl .02 en diciembre de 2000 y vl .02d)

Después de un año se recolectaron las impresiones de la primera versión antes de instituir la actualización, que comprendía un producto más estable por un largo período de tiempo. Estas primeras versiones de modelo se lanzaron en diciembre del 2001 (vl .1). Con la venia de los directivos del CMMI (CMMI Steering Group), al año siguiente en el mes de abril, se agrega un cambio que consistía en nuevas prácticas relativas a la adquisición. Convirtiéndose esta versión en los cimientos para la siguiente edición.

En la actualidad la aplicación del CMMI supero lo previsto. El modelo se hizo un estándar importante en el desarrollo de sistemas y software demostrando su competencia para medir los procesos durante las auditorías internas y externas de una empresa.

Finalmente, el desarrollo y mantenimiento del software necesita contar con normas y estándares de calidad que apoyen el proceso de mejora continua en las diferentes áreas de una empresa, para poder entregar un producto de software con valor agregado y calidad.

Se debe de priorizar la medición de la calidad del software durante el desarrollo de todo el proceso y no solo centrarse en el producto final. Para poder lograrlo se requiere el uso de las mejoras prácticas en gestión e ingeniería de software, tales como ITIL, COBIT, CMMI, ISO, Etc.

El grafico representa los diferentes niveles de madurez que señala el CMMI respecto a los procesos en las organizaciones.

