UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

INGENIERÍA DE SOFTWARE III

INGENIERÍA DE SISTEMAS

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

Vicerrectoría de Educación a Distancia y virtual

2016

AUTOR

Lina María Montoya Suarez
Ingeniera de Sistemas – Magister en Ingeniería de Software
linam.montoya@uniremington.edu.co

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

RESPONSABLES

Jorge Mauricio Sepúlveda Castaño
Decano de la Facultad de Ciencias Básicas e Ingeniería
jsepulveda@uniremington.edu.co

Eduardo Alfredo Castillo Builes
Vicerrector modalidad distancia y virtual
ecastillo@uniremington.edu.co

Francisco Javier Álvarez Gómez
Coordinador CUR-Virtual
falvarez@uniremington.edu.co

GRUPO DE APOYO

Personal de la Unidad CUR-Virtual
EDICIÓN Y MONTAJE

Primera versión. Febrero de 2011.
Segunda versión. Marzo de 2012
Tercera versión. noviembre de 2015
Cuarta versión 2016

TABLA DE CONTENIDO

Pág.

1 MAPA DE LA ASIGNATURA

INGENIERÍA DE SOFTWARE III

PROPÓSITO GENERAL DEL MÓDULO

El propósito del curso está orientado a que los estudiantes logren generar software con calidad, sin dejar atrás los principios y metodologías para el desarrollo, con el fin de dar continuidad al ciclo de vida de Software haciendo que mantenimiento de software perdure el tiempo y se actualice.

OBJETIVO GENERAL

Identificar las Prueba de software, Calidad de Software, gestión de proyecto entendiendo la importancia de la construcción productos con calidad, evitando los sobrecostos de mantenimiento o posibles fracasos.

OBJETIVOS ESPECÍFICOS

- Evaluar un producto de software de acuerdo a las normas internacionales de calidad, identificando los atributos relevantes del producto de software.
- Propiciar la mejora de calidad al interior de la empresa implementando un sistema de gestión PMBOK, que le permita a la organización la identificación de defectos en las etapas tempranas del proceso de desarrollo.
- Aplicar los principios, técnicas y metodologías de la ingeniería de software en el diseño, construcción, implantación y mantenimiento de soluciones informáticas.
- Aplicar las normas y estándares de calidad aceptadas por la comunidad académica y empresarial en el ámbito de la gerencia de tecnologías de la información y en proyectos de desarrollo de software.
- Desarrollar habilidades y destrezas que le permitan al estudiante desempeñarse como un tester de sistemas de información, implementando los criterios necesarios para la detección de la calidad del producto esperada por el usuario.

UNIDAD 1

Construcción de Software

UNIDAD 2

Calidad de Software

UNIDAD 3

Estándares

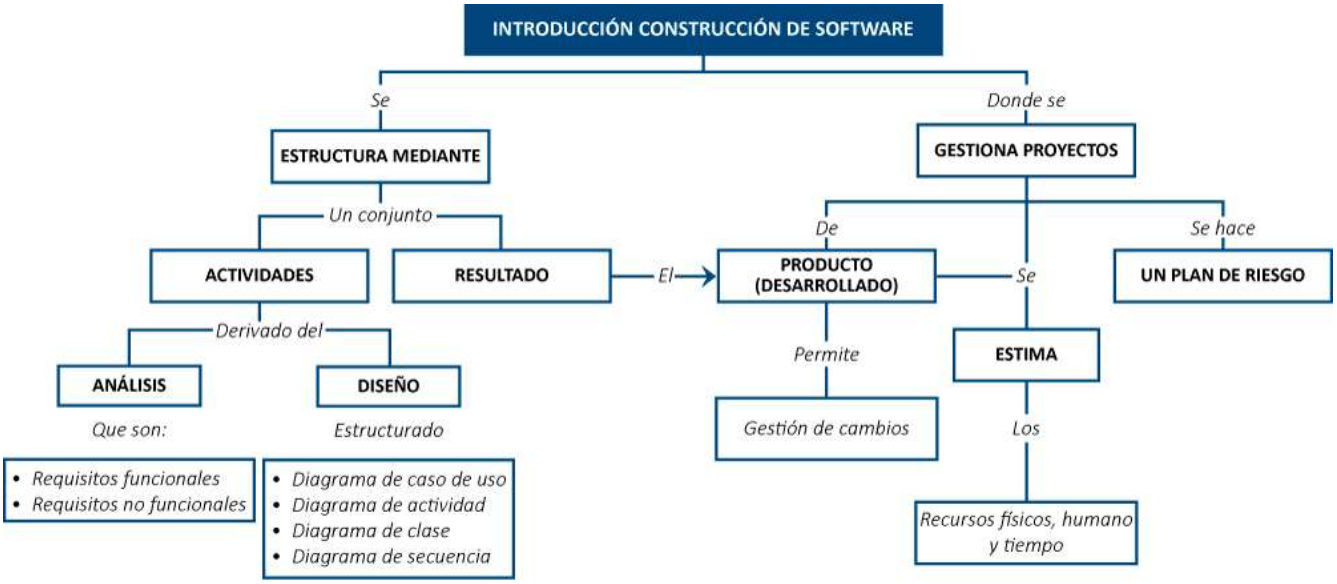
UNIDAD 4

Prueba y Métricas de Software

2 CONSTRUCCIÓN DE SOFTWARE

El propósito de Desarrollo de Software es la realización sistemática de un problema a resolver de la vida real haciendo énfasis en las actividades de análisis, diseño, codificación, pruebas y gestión de cambios con el fin de generar productos de software con calidad, dando cumpliendo a los requisitos especificados.

2.1.1 RELACIÓN DE CONCEPTOS



2.2 TEMA 1 INTRODUCCIÓN CONSTRUCCIÓN DE SOFTWARE

El proceso de software es un conjunto de actividades y resultados asociados que conducen a la creación de un producto de software" Sommerville 2005. Cuando el proceso implica la construcción de algún producto, solemos referirnos al proceso como Ciclo de Vida. El proceso de desarrollo de software suele denominarse ciclo de vida del software, porque describe la vida de un producto de software desde su concepción hasta su implementación, entrega, utilización y mantenimiento.

CPSLV1 - Señor developero. Opiniones sobre el oficio del desarrollo de software Enlace

Otra mirada del desarrollo de software, Ponencia CPSLV1 - Señor developero. Opiniones sobre el oficio del desarrollo de software.

2.2.1 CONCEPTOS

- **Ciclo de vida de software**

El término ciclo de vida del software describe el desarrollo de software, desde la fase inicial hasta la fase final, donde cada fase intermedia requiere validar el desarrollo de la aplicación, es decir, garantizar que el software cumpla los requisitos para la aplicación y verificación de los procedimientos de cada fase de desarrollo con respecto a una necesidad del cliente.

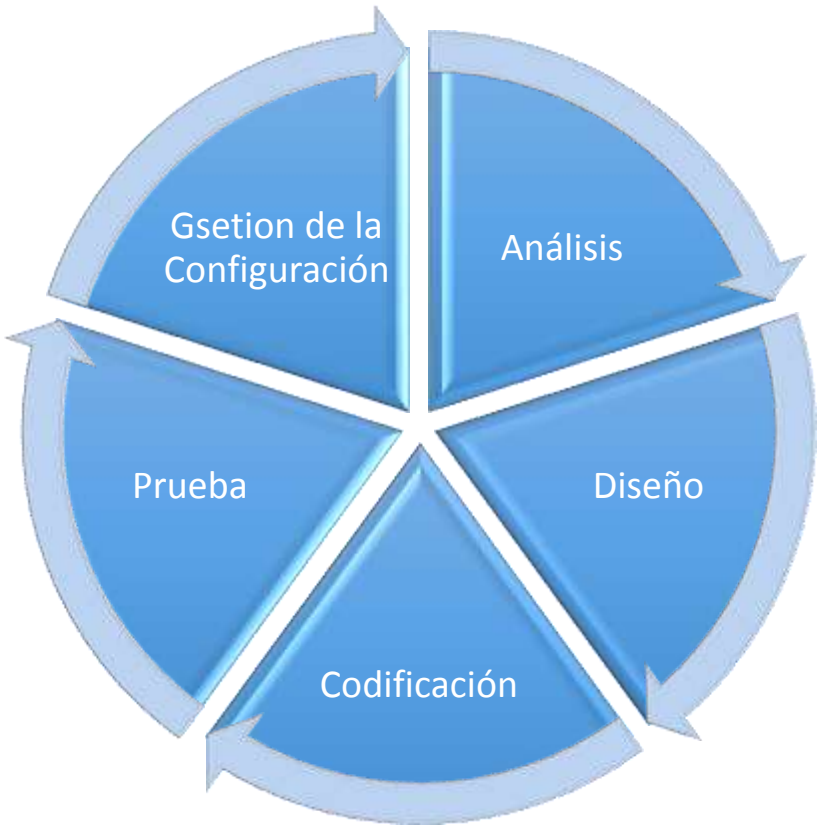
El ciclo de vida permite que los errores se detecten a tiempo, garantiza el funcionamiento de la aplicación, prueba y mantenimiento de esta.

El ciclo de vida consta de la siguiente fase:

DENOMINACIÓN	DESCRIPCIÓN
Análisis de los requisitos:	Recopilar, examinar y formular los requisitos del cliente y examinar cualquier restricción que se pueda aplicar.
Diseño	Documentación del software en UML, con base a los Requisitos extraídos.
Codificación (programación e implementación):	Es la implementación de un lenguaje de programación para crear las funciones definidas durante la etapa de análisis y diseño.

Prueba	Se valida y se constata los requisitos plasmado en la codificación, para garantizar que se implementaron de acuerdo con las especificaciones.
Gestión de la Configuración	Se hacen los procedimientos correctivos (mantenimiento correctivo) y las actualizaciones del software (mantenimiento continuo).
Nota: El orden y la presencia de cada uno de estos procedimientos en el ciclo de vida de una aplicación dependen del tipo de modelo de ciclo de vida acordado entre el cliente y el equipo de desarrolladores.	

Ciclo de vida de desarrollo de Software



Autoría propia

Etapa de desarrollo de Software

REPRESENTACIONES

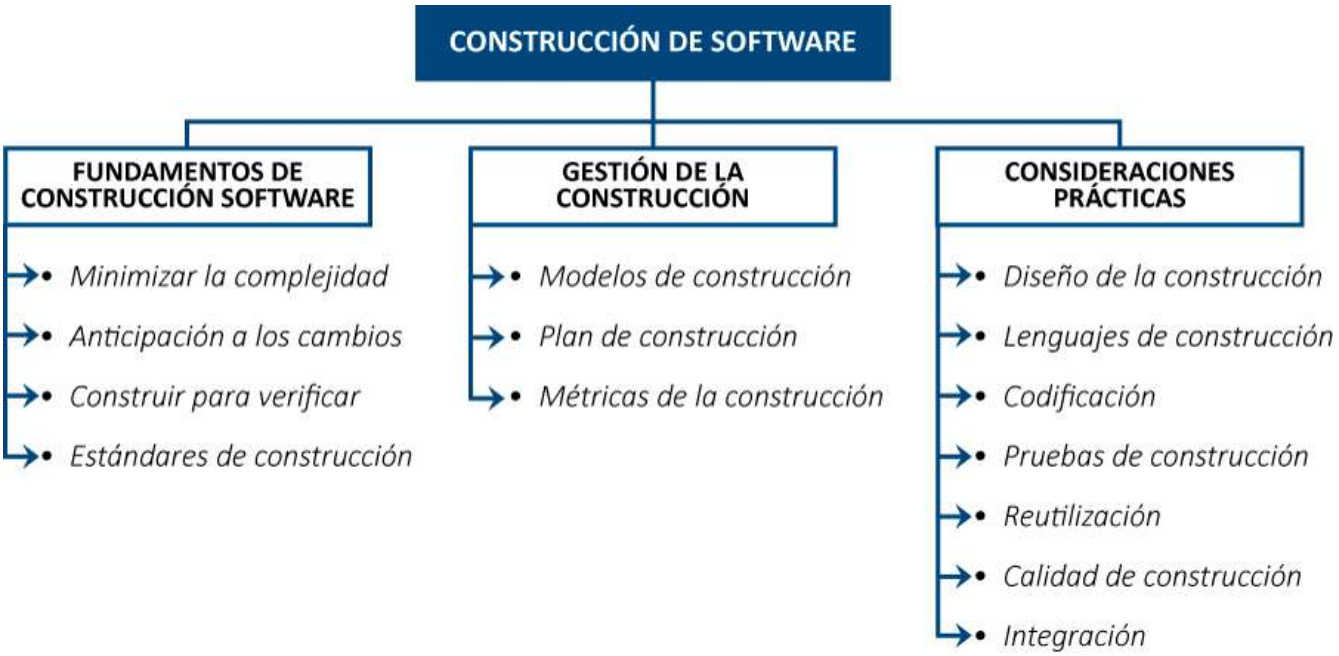
Tomado de: http://www.vision4its.com/soluciones-desarrollo.html

2.2.2 FUNDAMENTOS DE LA CONSTRUCCIÓN DE SOFTWARE

Los fundamentos de la construcción del software incluyen:

- Minimizar la complejidad
- Anticiparse a los cambios
- Construir para verificar
- Estándares en la construcción.

Los tres primeros conceptos se aplican tanto al diseño como a la construcción el ultimo define y describe cómo se aplican a la construcción.



En la construcción de un proyecto de software se presenta los siguientes problemas:

En realidad, son consejos obvios, pero que tendemos a olvidar cuando estamos metidos de lleno en ellos, por lo que es útil tenerlos a mano.

1. **Presión excesiva de calendario:**

   - Estimaciones objetivas.
   - Más recursos.
   - Mejores recursos.
   - Requerimientos priorizados.
   - Requerimientos sin alcances.
   - Entregas por fases.

2. **Necesidades cambiantes:**

   - Desarrollo iterativo.
   - Cambio de la gestión en el control/línea base.

3. **La falta de especificaciones técnicas:**

   - Creación de las especificaciones iniciales.
   - Actualización de las especificaciones en base a eventos.
   - Gestión de las especificaciones de la linea base.

- Un arquitecto de software asignado.

4. **Falta del documento de plan de proyecto:**

- Creación del documento de plan de proyecto inicial.
- Actualización del plan de proyecto de forma periódica y en base a eventos.
- Línea de base de gestión del plan del proyecto.
- Un jefe de proyecto asignado.

5.y 6. Innovaciones excesivas y secundarias:

- Control de la línea base.
- Evaluación de impacto.
- Gestión de riesgo de forma continuada.
- Un arquitecto de software asignado.

7. **Los requisitos cambian:**

- Línea base con los requisitos iniciales.
- Gestión de la línea base.
- Gestión del riesgo.
- Un arquitecto de software asignado.

8. **Falta de métodos científicos:**

- Prototipos.
- Desarrollo incremental.
- Medición del rendimiento técnico.

9. **Ignorar lo obvio:**

- Cálculos a grosso modo.
- Asimilación de lecciones aprendidas.

10. **Comportamiento poco ético:**

- Implantar una cultura de trabajo ética.
- Adhesión personal al código ético.

PROBLEMAS EN LA CONSTRUCCIÓN DE SOFTWARE

Table 1

Ten software project problems and some antidotes

Problem area	Antidotes
1. Excessive schedule pressure	Objective estimates More resources Better resources Prioritized requirements Descoped requirements Phased releases
2. Changing needs	Iterative development Change control/baseline management
3. Lack of technical specifications	Development of initial specifications Event-driven updating of specifications Baseline management of specifications A designated software architect
4. Lack of a documented project plan	Development of an initial plan Periodic and event-driven updating Baseline management of the project plan A designated project manager
5. & 6. Excessive and secondary innovations	Baseline control Impact analysis Continuous risk management A designated software architect
7. Requirements creep	Initial requirements baseline Baseline management Risk management A designated software architect
8. Lack of scientific methods	Prototyping Incremental development Technical performance measurement
9. Ignoring the obvious	Back-of-the-envelope calculations Assimilation of lessons learned
10. Unethical behavior	Ethical work environments and work cultures Personal adherence to a code of ethics

Tomado de: http://guti.bitacoras.com/10-problemas-en-proyectos-de-software/1717

2.2.3 EJERCICIO DE ENTRENAMIENTO

Apreciado estudiante, la siguiente actividad le servirá para afianzar los conocimientos en la Ingeniería de Software:

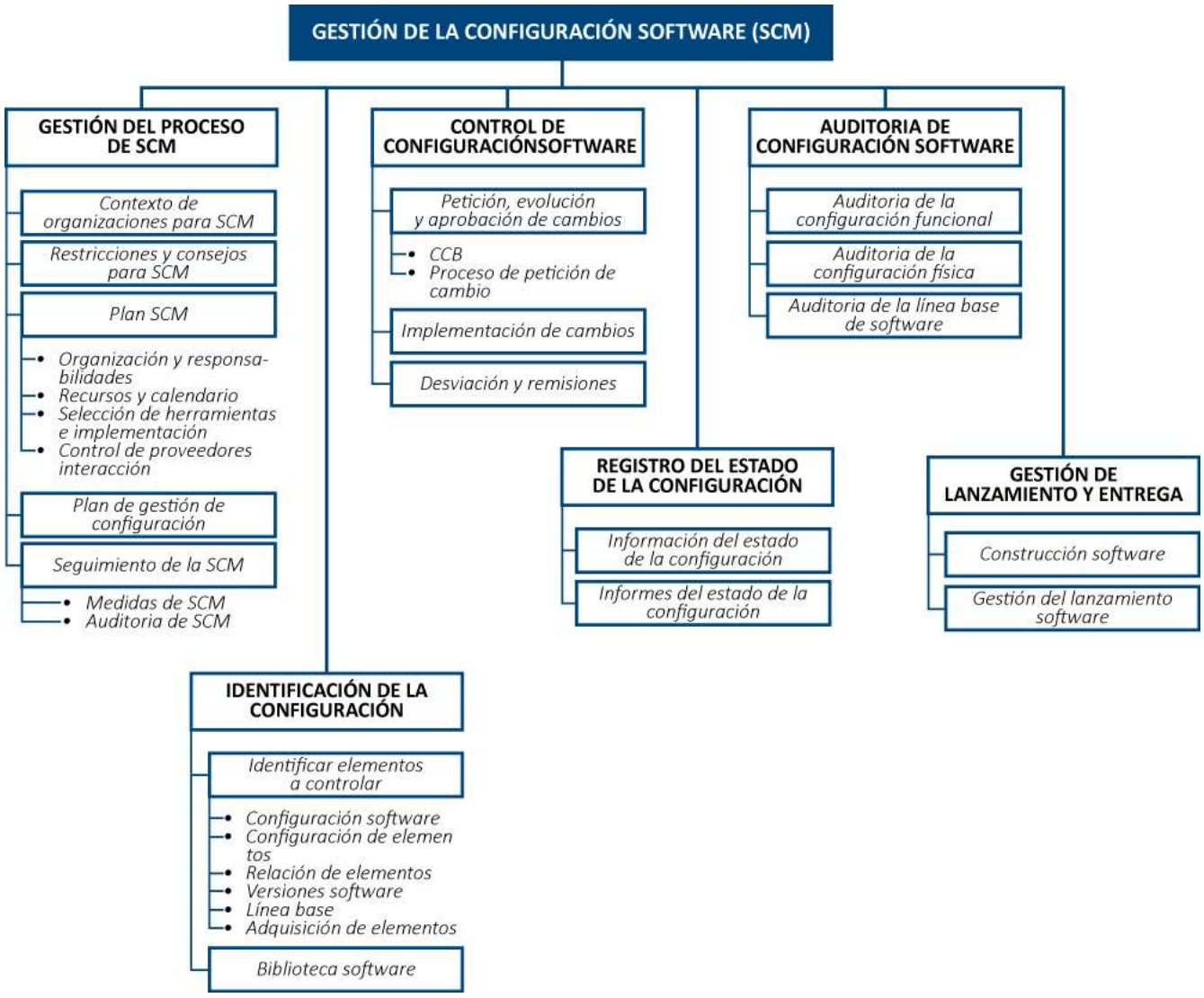
Elabore un ensayo donde describa la importancia de la ingeniería de software hoy y hacia dónde va. Máximo dos páginas.

2.3 TEMA 2 GESTIÓN DE LA CONFIGURACIÓN

La gestión de la configuración aplica procedimientos administrativos y técnicos durante todo el ciclo de vida del sistema para:

- Identificar, definir y establecer la línea base de los elementos de configuración del software del sistema.
- Controlar las modificaciones y las versiones de los elementos.
- Registrar el estado de los elementos y las peticiones de modificación.
- Asegurar la complexión, la consistencia y la corrección de los elementos.
- Controlar el almacenamiento, la manipulación y la entrega de los elementos.

El proyecto SWEBOK define la Gestión de la Configuración de Software (SCM) como la disciplina de identificación de la configuración en distintos puntos en el tiempo, con el propósito de que **sistemáticamente** se controlen los cambios de configuración y se mantenga la integridad y la trazabilidad de la configuración a lo largo del ciclo de vida (SWEBOK, 2004).



2.3.1 INTRODUCCIÓN

En el desarrollo de software, los cambios ocurren todo el tiempo, de modo que la administración del cambio es absolutamente esencial. Cuando un equipo de individuos desarrolla software, hay que cerciorarse de que los miembros del equipo no interfieran con el trabajo de los demás. Esto es, si dos personas trabajan sobre un componente, los cambios deben coordinarse. De otro modo, un programador podría realizar cambios y sobrescribir en el trabajo de otro. También se debe garantizar que todos tengan acceso a las versiones más actualizadas de componentes de software; de lo contrario, los desarrolladores pueden rehacer lo ya hecho. Cuando algo salga mal con una nueva versión de un sistema, se debe poder retroceder a una versión operativa del sistema o componente.

La gestión de la configuración es el nombre dado al proceso general de gestionar un sistema de software cambiante. La meta de la administración de la configuración es apoyar el proceso de integración del sistema, de modo que todos los desarrolladores tengan acceso en una forma controlada al código del proyecto y a los

documentos, descubrir qué cambios se realizaron, así como compilar y vincular componentes para crear un sistema.

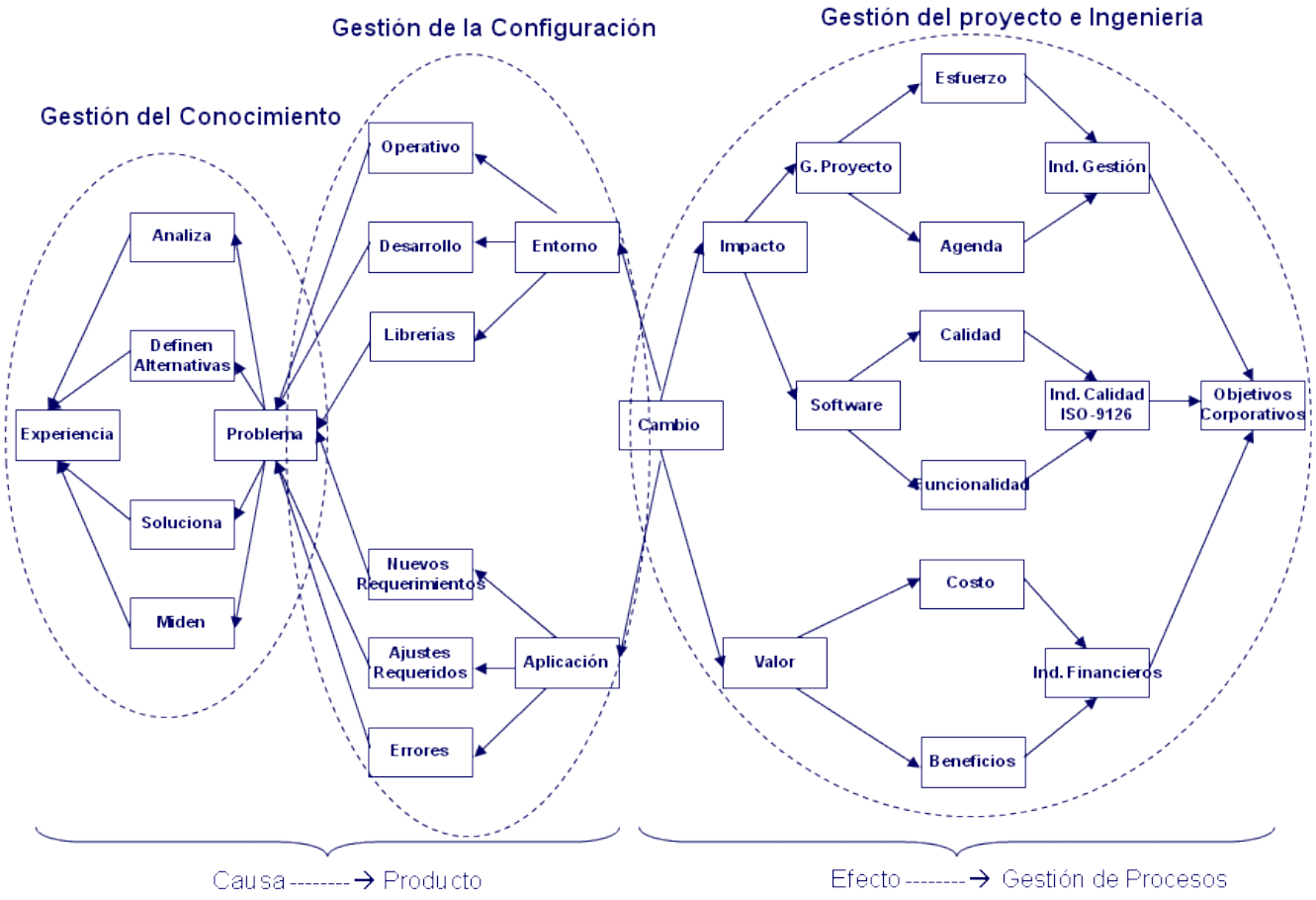
Comprende tres actividades fundamentales en la gestión de la configuración:

ACTIVIDAD	DESCRIPCIÓN
1. GESTIÓN DE VERSIONES	Donde se da soporte para hacer un seguimiento de las diferentes versiones de los componentes de software. Los sistemas de gestión de versiones incluyen facilidades para que el desarrollo esté coordinado por varios programadores. Esto evita que un desarrollador sobrescriba un código que haya sido enviado al sistema por alguien más.
2. INTEGRACIÓN DE SISTEMA	Donde se da soporte para ayudar a los desarrolladores a definir qué versiones de componentes se usan para crear cada versión de un sistema. Luego, esta descripción se utiliza para elaborar automáticamente un sistema al compilar y vincular los componentes requeridos.
3. RASTREO DE PROBLEMAS	Donde se da soporte para que los usuarios reporten bugs y otros problemas, y también para que todos los desarrolladores sepan quién trabaja en dichos problemas y cuándo se corrigen.

2.3.2 ¿QUÉ ES GESTIÓN DE LA CONFIGURACIÓN?

- Una práctica que nos permite implementar buenos procesos que garantice un adecuado control del cambio durante el ciclo de desarrollo de un sistema
- Una práctica que me permite controlar el mundo irracional de los cambios para mantener la integridad del producto, evaluar y controlar el cambio, hacer el producto visible a todo el equipo
- Es la práctica que se aplica con el objetivo de asegurar que un producto de software no se pierda, pueda ser rastreable y reconstruir sin inconvenientes
- Es el parte de la administración de un proyecto que se enfoca exclusivamente en controlar sistemáticamente los cambios en los artefactos de un producto de software que ocurren durante la ejecución de este.

Gestión del conocimiento, de la configuración y del proyecto e Ingeniería



Tomado de: Libro Swebok 2 edición

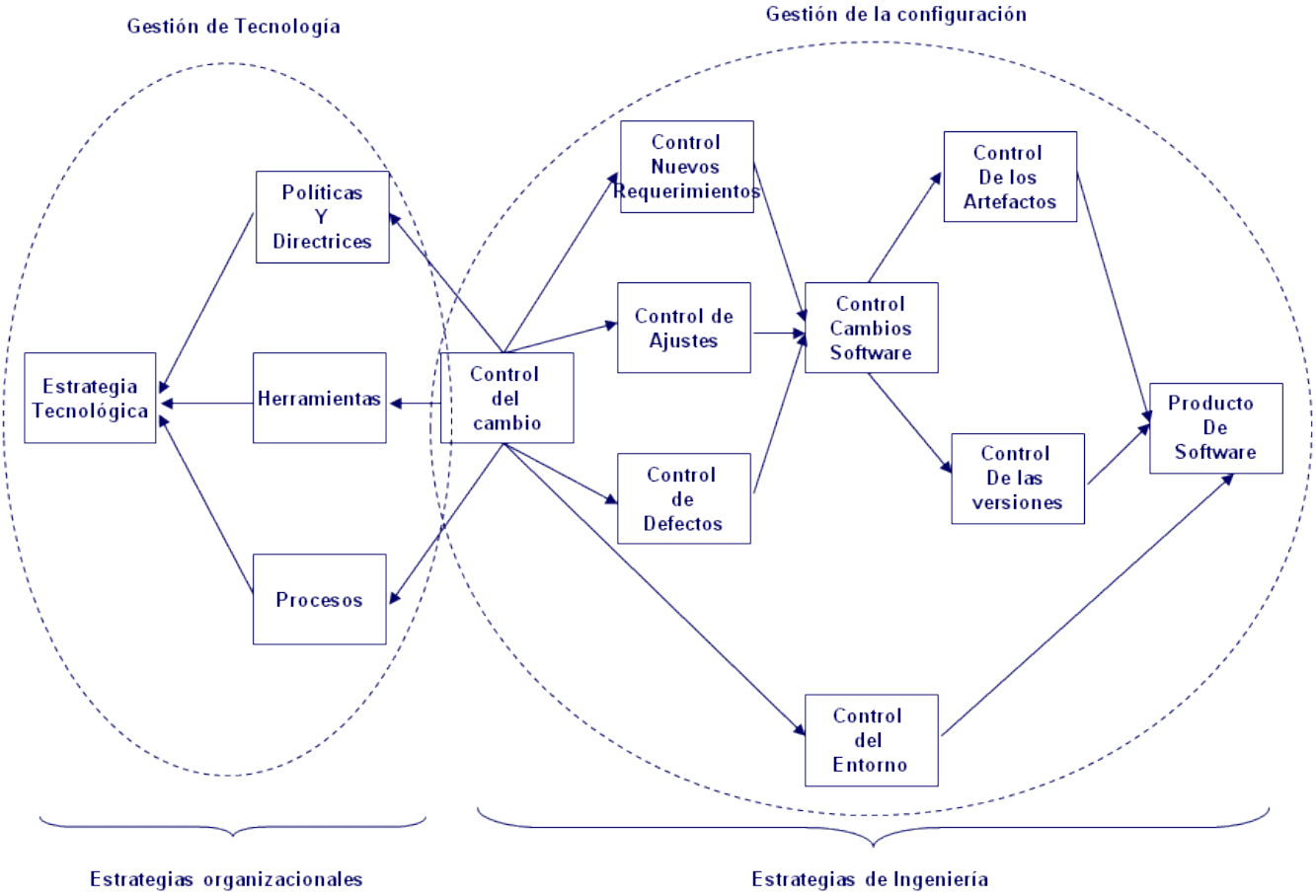
La relación sistémica en las variaciones que se pueden dar entre los diferentes elementos que intervienen en un cambio, requieren de igual forma de mecanismo que de forma sistémica integre elementos de control para garantizar:

¿Qué el producto de software se conserva íntegro?

→

¿Qué es trazable o rastreable durante el ciclo de vida del desarrollo o el mantenimiento?

Gestión de la tecnología y gestión de la configuración



Tomado de Libro Swebok 2 edición

La variación y relación sistemática que involucra todos estos elementos, solo se puede controlar, si se adoptan prácticas formales en las que se definen:

- Políticas, objetivos y metas
- Directrices de aplicación
- Procesos
- Indicadores de medición
- Herramientas que apoyen la gestión

Es fundamental tener presente lo siguiente: establecer y demostrar que el no implementar métodos formales para la Gestión de la Configuración y otras disciplinas, trae como consecuencia:

- Sobreesfuerzos producto del caos
- Probabilidad de desviación de la ruta crítica de los proyectos
- Falta de calidad en los productos
- Improductividad
- Productos no confiables
- Pérdida de competitividad

Además, hay relación directa entre diferentes variables, la variación de una tiene incidencia directa sobre la otra, para prácticas como SCM, estas variaciones son representativas y requieren una adecuada medición.

- **Efectividad vs Calidad**: en la medida que las prácticas sean efectivas, se puede lograr obtener mejores niveles de calidad. O también esta relación se puede leer como que si los niveles de calidad aumentan, se podría interpretar que las prácticas están siendo efectivas.
- **Calidad vs Productividad:** se ha comprobado que en la medida que los niveles de calidad aumentan, los niveles de productividad aumentan, dado que el esfuerzo que se requiere en darle solución a los errores, varía los indicadores de productividad de una organización.
- **Efectividad vs Sobreesfuerzo:** en la medida que las prácticas se han incorporado a la organización y son aplicadas de forma adecuada como herramientas, se puede constatar que los sobreesfuerzos producto del caos tienen una tendencia a la disminución.

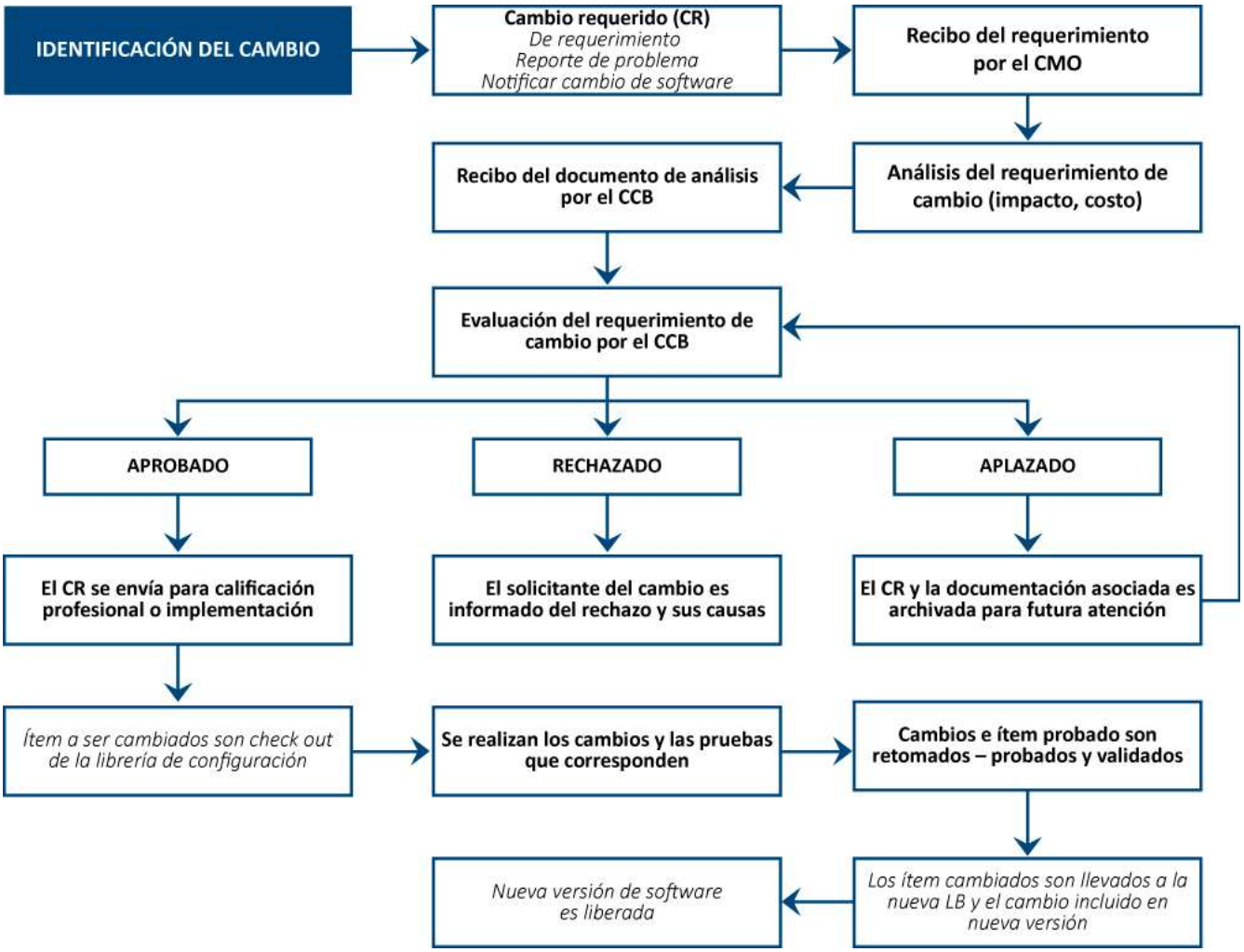
No tener implementadas buenas prácticas de Gestión de la Configuración, puede afectar la calidad interna y externa del producto de software (ISO/IEC 9126:2001):

- Funcionalidad
- Confiabilidad
- Facilidad de uso
- Eficiencia
- Mantenimiento
- Portabilidad

2.3.3 PUNTOS CLAVES:

- No importa en qué punto del ciclo de vida de un software se está para que se presenten cambios, el deseo del cambio es persistente.
- Todo lo que está alrededor de un sistema en construcción está expuesto al cambio.
- Todo cambio que no se controle, aumenta la probabilidad de la existencia de un producto sin calidad.
- El cambio no es el que incide sobre la calidad, es la ausencia de buenas prácticas que la deterioran.

Ciclo de un cambio



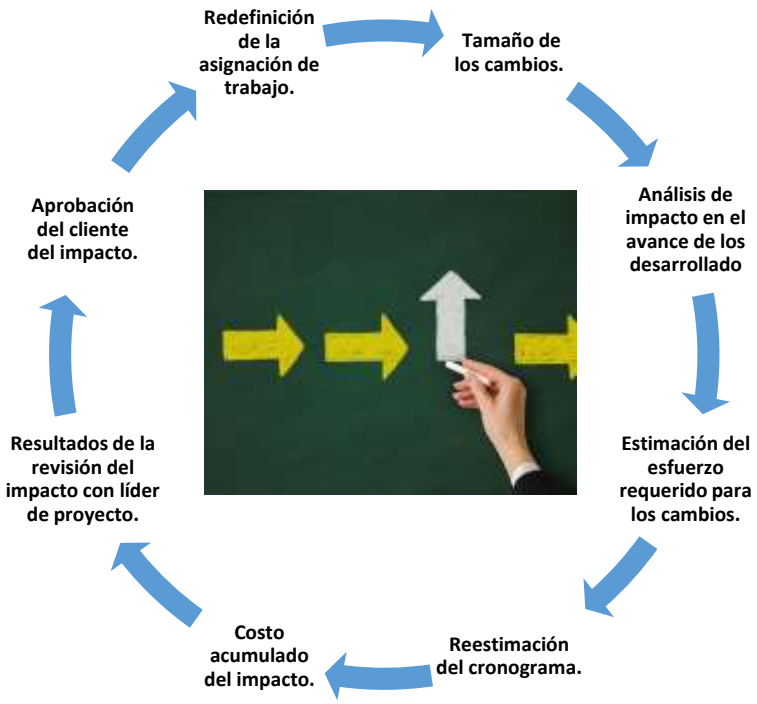
Autoría propia

Un cambio genera un impacto, para esto es importante analizar el impacto y establecer o estimar ¿cuál es el efecto en el software evaluado? y sus documentos asociados ante un cambio solicitado.

El análisis de impacto debe estar orientado a conocer:

- ¿Qué ítem relacionado afecta?
- ¿Qué esfuerzo se requiere para realizar el cambio?
- ¿Qué costos implican el cambio?

Esto nos debe dejar como resultado:



Autoría propia

2.4 TEMA 3 GESTIÓN DE PROYECTO

El proceso de gestión de un proyecto de software comienza con un conjunto de actividades que, globalmente, se denominan planificación del proyecto. “No podemos pedir exactitud a la fase de planificación, es solo una idea de cómo van a transcurrir las cosas. Hay que planificar el trabajo, los recursos humanos y la tecnología. Planificar es estimar. “La primera de estas actividades es la estimación.

Qué es un Proyecto Enlace

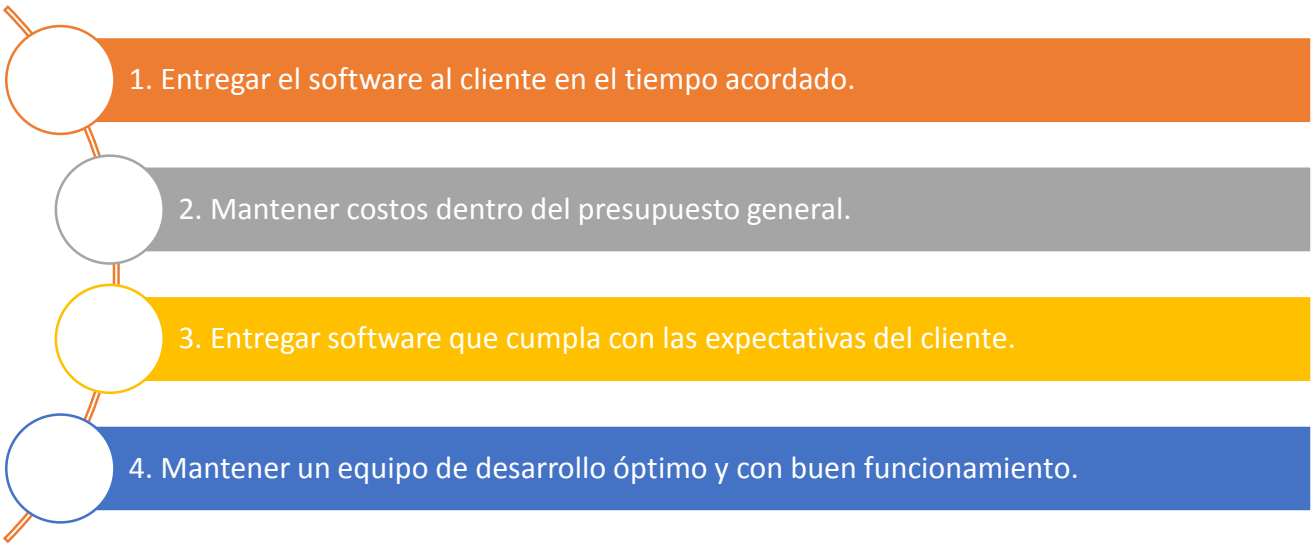
Gestión de Proyectos con PMBOK Enlace

PMBOK Guide Enlace

Introducción

La gestión de proyectos de software es una parte esencial de la ingeniería de software. Los proyectos necesitan administrarse porque la ingeniería de software profesional está sujeta siempre a restricciones organizacionales de presupuesto y fecha. El trabajo del administrador del proyecto es asegurarse de que el proyecto de software cumpla y supere tales restricciones, además de que entregue software de alta calidad. La buena gestión no puede garantizar el éxito del proyecto. Sin embargo, la mala gestión, por lo general, da como resultado una falla del proyecto: el software puede entregarse tarde, costar más de lo estimado originalmente o no cumplir las expectativas de los clientes (Sommerville 2011).

Los criterios de éxito para la gestión del proyecto varían de un proyecto a otro, pero, para la mayoría de los proyectos, las metas importantes son:



Autoría propia

Estas metas no son únicas para la ingeniería de software, pero sí lo son para todos los proyectos de ingeniería. Sin embargo, la ingeniería de software es diferente en algunas formas a otros tipos de ingeniería que hacen a la gestión del software particularmente desafiante.

Algunas de estas diferencias son:

- ***El producto es intangible:*** Un administrador de un astillero o un proyecto de ingeniería civil pueden ver el producto conforme se desarrolla. Si hay retraso en el calendario, es visible el efecto sobre el producto: es evidente que algunas partes de la estructura no están terminadas. El software es intangible. No se puede ver ni tocar. Los administradores de proyectos de software no pueden constatar el progreso con sólo observar el artefacto que se construye. Más bien, ellos se apoyan en otros para crear la prueba que pueden utilizar al revisar el progreso del trabajo.

- ***Los grandes proyectos de software con frecuencia son proyectos excepcionales:*** los grandes proyectos de software se consideran en general diferentes en ciertas formas de los proyectos anteriores. Por eso, incluso los administradores que cuentan con vasta experiencia pueden encontrar difícil anticiparse a los problemas. Aunado a esto, los vertiginosos cambios tecnológicos en computadoras y comunicaciones pueden volver obsoleta la experiencia de un administrador. Las lecciones aprendidas de proyectos anteriores pueden no ser aplicables a nuevos proyectos.
- ***Los procesos de software son variables y específicos de la organización:*** El proceso de ingeniería para algunos tipos de sistema, como puentes y edificios, es bastante comprendido. Sin embargo, los procesos de software varían considerablemente de una organización a otra. Aunque se ha producido un notorio avance en la estandarización y el mejoramiento de los procesos, no es posible predecir de manera confiable cuándo un proceso de software particular conducirá a problemas de desarrollo. Esto es especialmente cierto si el proyecto de software es parte de un proyecto de ingeniería de sistemas más amplio.

Debido a estos conflictos, no es sorpresivo que algunos proyectos de software se retrasen y excedan el presupuesto. A menudo, los sistemas de software son nuevos y técnicamente innovadores. Los proyectos de ingeniería (como los nuevos sistemas de transporte) que son reformadores, normalmente también tienen problemas de calendario. Dadas las dificultades, quizá sea asombroso que tantos proyectos de software ¡se entreguen a tiempo y dentro del presupuesto! Es imposible efectuar una descripción laboral estándar para un administrador de proyecto de software. La labor varía enormemente en función de la organización y el producto de software a desarrollar.

No obstante, la mayoría de los administradores, en alguna etapa, toman la responsabilidad de varias o todas las siguientes actividades:

- **Planeación del proyecto**: Los administradores de proyecto son responsables de la planeación, estimación y calendarización del desarrollo del proyecto, así como de la asignación de tareas a las personas. Supervisan el trabajo para verificar que se realice de acuerdo con los estándares requeridos y monitorizan el avance para comprobar que el desarrollo esté a tiempo y dentro del presupuesto.

- **Informes Los administradores de proyectos**: por lo común son responsables de informar del avance de un proyecto a los clientes y administradores de la compañía que desarrolla el software. Deben ser capaces de comunicarse en varios niveles, desde codificar información técnica detallada hasta elaborar resúmenes administrativos. Deben redactar documentos concisos y coherentes que sinteticen información crítica de reportes detallados del proyecto. Es necesario que esta información se presente durante las revisiones de avance.

- **Gestión del riesgo:** Los administradores de proyecto tienen que valorar los riesgos que pueden afectar un proyecto, monitorizar dichos riesgos y emprender acciones cuando surjan problemas.

- **Gestión de personal**: Los administradores de proyecto son responsables de administrar un equipo de personas. Deben elegir a los integrantes de sus equipos y establecer formas de trabajar que conduzcan a desempeño efectivo del equipo.

- **Redactar propuestas**: La primera etapa en un proyecto de software puede implicar escribir una propuesta para obtener un contrato de trabajo. La propuesta describe los objetivos del proyecto y cómo se realizará. Por lo general, incluye estimaciones de costo y calendarización, además de justificar por qué el contrato del proyecto debería concederse a una organización o un equipo particular. La escritura de propuestas es una tarea esencial, pues la supervivencia de muchas compañías de software depende de contar con suficientes propuestas aceptadas y concesiones de contratos. Es posible que no haya lineamientos establecidos para esta tarea; la escritura de propuestas es una habilidad que se adquiere a través de práctica y experiencia.

2.4.1 PROYECTO DE SOFTWARE

Conjunto de actividades interdependientes orientadas a un fin específico

¿Qué significa terminar con éxito un proyecto?

- Cumplir con los objetivos dentro de las *especificaciones técnicas*, de *costo* y de *plazo de término*.

A un conjunto de proyectos orientados a un objetivo superior se denomina **programa**, y un conjunto de programas constituye un **plan,** para esto hay que planificar y planificar es estimar.

Estimar: Es la base de todas las demás actividades de planificación del proyecto y sirve como guía para una buena ingeniería del software. Echar un vistazo al futuro y aceptar cierto grado de incertidumbre.

"Es aplicar el sentido común y el conocimiento a un determinado problema"

No debe llevarse a cabo de forma descuidada. *"En informática, no podemos fiarnos de la intuición, una buena planificación acaba permitiendo abordar un proyecto y ayuda al proceso de refinamiento"*

Enfoque del que estima

MISIÓN	VISIÓN

La estimación de recursos, costos y planificación temporal de un proyecto requiere experiencia, una buena información histórica y confiar en medidas cuantitativas cuando todo lo que conocemos son datos cualitativos.

La estimación conlleva un riesgo que lleva a la incertidumbre.

- **La complejidad del proyecto** tiene un gran efecto en la incertidumbre, presente en oda planificación y , es una medida relativa que se ve afectada por la familiaridad con esfuerzos anteriores (experiencia adquirida en proyectos anteriores).

- **El tamaño del proyecto** es otro factor importante que puede afectar a la precisión de las estimaciones. Conforme aumenta el tamaño, crece rápidamente la interdependencia entre varios elementos del software.

El grado de incertidumbre en la especificación de requisitos tiene efecto en el riesgo de la estimación. La disponibilidad de información histórica también determina el riesgo de la estimación.

Actividades que se derivan de la estimación

Establecer el ámbito del software y estimación de los recursos requeridos.

- Establecer el ámbito del software: Describe la función, el rendimiento, las restricciones, las interfaces y la fiabilidad.

Se evalúan las funciones descritas en el enunciado del ámbito o se refinan las consideraciones de rendimiento abarcan los requisitos de tiempos de respuesta y de procesamiento. Las restricciones marcan los límites del software debidos al hardware externo, la memoria disponible y otros sistemas existentes.

Para la obtención de la información necesaria para el ámbito, o sea para acercar al cliente y al desarrollador (ingeniero del software, analista), y para hacer que comience el proceso de comunicación es establecer una reunión o entrevista preliminar.

Se sugiere que el analista comience haciendo preguntas de que lleven a un entendimiento básico del problema.

- El primer conjunto de cuestiones se centra en el cliente, en los objetivos globales y en los beneficios.

- El siguiente conjunto de cuestiones permiten que el analista comprenda mejor el problema y que el cliente exprese sus percepciones sobre una solución.

- Estimación de los recursos requeridos. Es la segunda tarea en la planificación, para acometer el esfuerzo de desarrollo software se estima lo siguiente:

  - Recursos humanos.

  - Recursos Tecnológicos (software y de hardware).

  - Recursos físicos.

2.4.2 ACTIVIDADES QUE SE DERIVAN DE LA PLANIFICACIÓN.

- Fijar los objetivos y metas
- Desarrollar estrategias
- Desarrollar políticas
- Anticipar futuras situaciones
- Conducir un establecimiento de riesgos
- Determinar posibles cursos de acción
- Tomar decisiones de planificación
- Fijar procedimientos y reglas
- Desarrollar los planes del proyecto
- Preparar presupuestos
- Documentar los planes del proyecto.

2.4.3 LOS PRINCIPALES PROBLEMAS EN LA PLANIFICACIÓN DE UN PROYECTO DE LA INGENIERÍA DE SOFTWARE

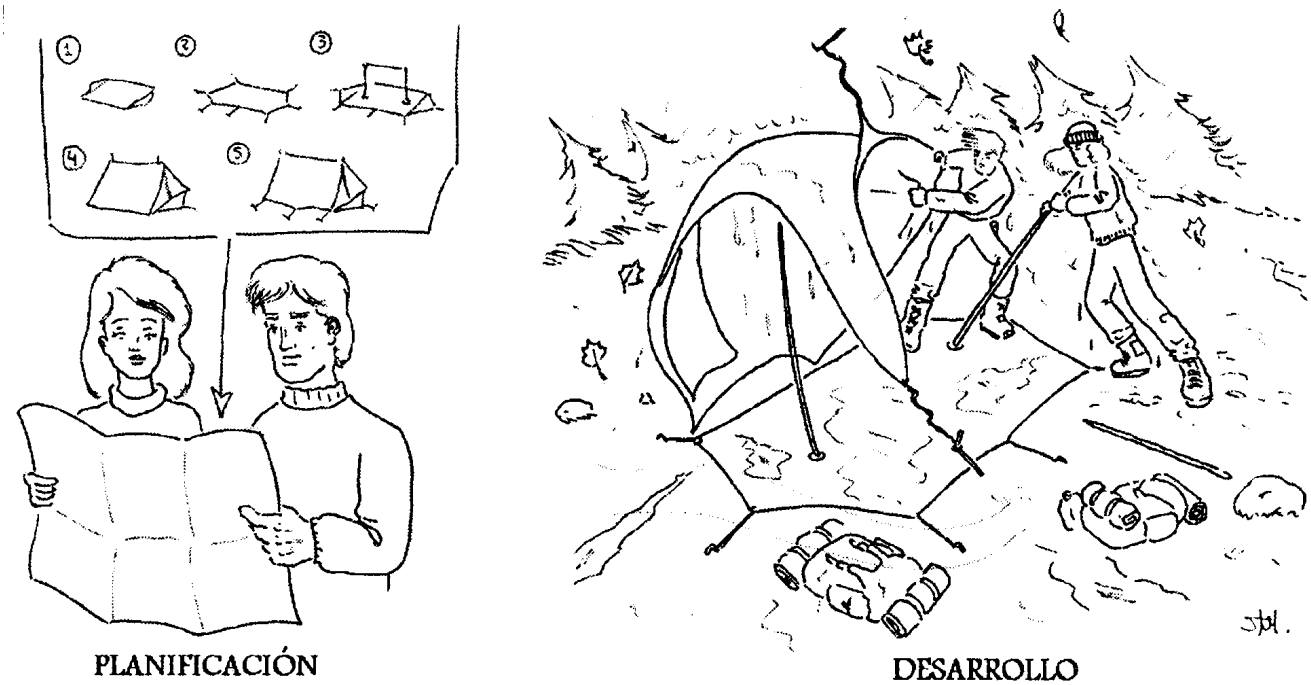
- Requerimientos incorrectos e incompletos.
- Muchas especificaciones de requerimientos son inestables y sujetas a cambios mayores.
- La planificación no se lleva a cabo por la creencia errónea de que es una pérdida de tiempo y los planes cambiarán de todos modos.
- La planificación de costos y plazos no es actualizada y se basa en necesidades de mercadeo y no de los requerimientos del sistema.
- Estimar el tamaño y complejidad del proyecto de software de modo de realizar una estimación de costos y plazos realista.
- Los costos y plazos no son re estimados cuando los requerimientos del sistema o el ambiente de desarrollo cambia.
- No se manejan factores de riesgo.
- La mayoría de las organizaciones de desarrollo de software no recolectan datos de proyectos pasados.

- Las compañías no establecen políticas o procesos de desarrollo de software.

¿Cuál es el centro del Proyecto de Software?

Tomado de Libro Swebok 2 edición

El proyecto: su crudeza y realidad

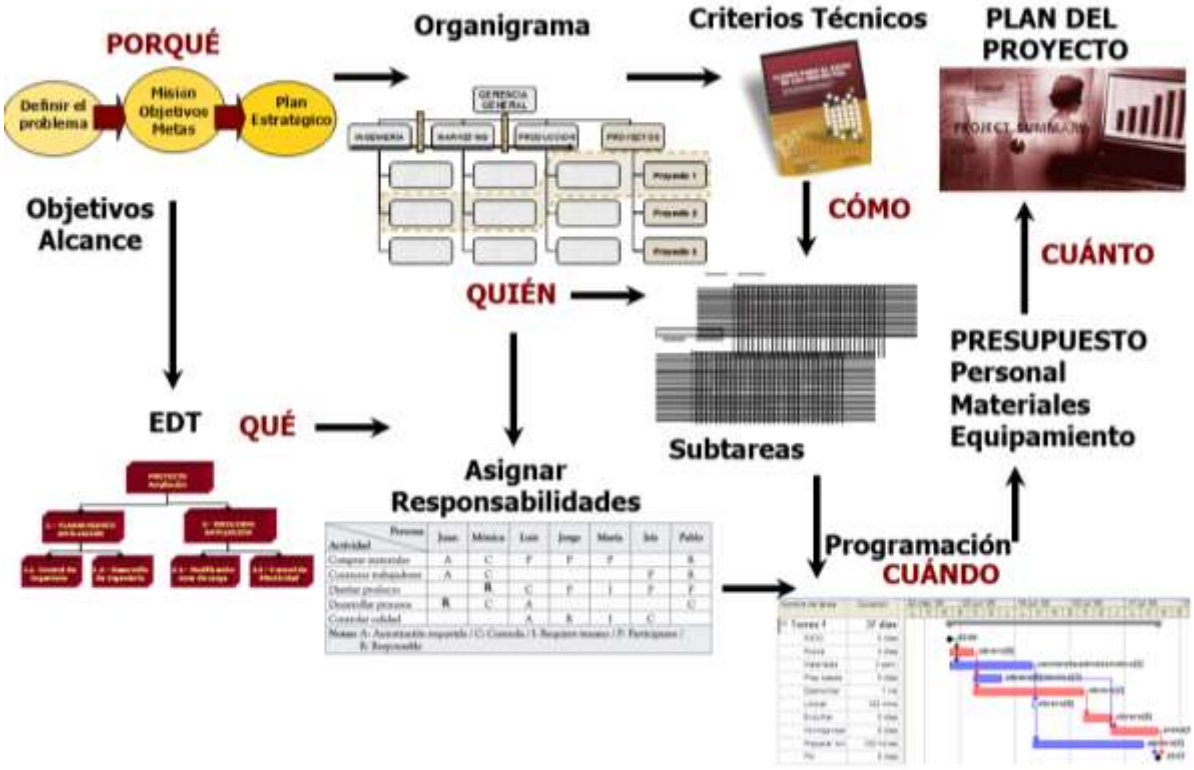


Tomado de http://es.slideshare.net/alfredoramosaq/gestion-de-proyectos1enjoy-29719748

2.4.4 BUENAS PRÁCTICAS PARA UN PROYECTO DE SOFTWARE

- **Segmentación** - el proyecto debe ser separado en un número manejable de actividades y tareas.
- **Interdependencia** – la planificación debe reflejar la segmentación de tareas, y las relaciones entre ellas. Hay algunas tareas ocurren en secuencia, otras en paralelo, algunas son requisito de otras, etc.
- **Asignación de tiempo** - a cada tarea se le debe asignar un cierto número de unidades de trabajo (personas-días, etc) y fechas de inicio y término.
- **Validación del esfuerzo** – se debe verificar que la gente asignada a una tarea esté disponible y además que sea suficiente.
- **Definición de responsabilidades** - cada tarea debe tener un responsable.
- **Salida definida** - cada tarea debe tener un "producto" bien definido. Por ejemplo: diseño de un módulo, plan de pruebas, etc.
- **Definición de metas** - cada grupo de tareas se debe estar asociado con una meta.

Planificación Integral



Autoría propia

2.4.5 GESTIÓN DEL RIESGO

La gestión del riesgo es una de las tareas más sustanciales para un administrador de proyecto. Proyecto o la calidad del software a entregar, y posteriormente tomar acciones para evitar dichos riesgos (Hall, 1998; Ould, 1999).

Se puede considerar un riesgo como algo que es preferible que no ocurra. Los riesgos pueden amenazar el proyecto, el software que se desarrolla o a la organización. Por lo tanto, existen tres categorías relacionadas de riesgo:

RIESGOS	DESCRIPCIÓN
RIESGOS DEL PROYECTO	Los riesgos que **alteran** **el calendario** o **los recursos** del proyecto. Un ejemplo de riesgo de proyecto es la renuncia de un diseñador experimentado. Encontrar un diseñador de reemplazo con habilidades y experiencia adecuadas puede demorar mucho tiempo y, en consecuencia, el diseño del software tardará más tiempo en completarse.
RIESGOS DEL PRODUCTO	Los riesgos que **afectan** **la calidad** o **el rendimiento** del software a desarrollar. Un ejemplo de riesgo de producto es la falla que presenta un componente que se adquirió al no desempeñarse como se esperaba. Esto puede afectar el rendimiento global del sistema, de modo que es más lento de lo previsto.
RIESGOS EMPRESARIALES	Riesgos que **afectan** a **la organización** que desarrolla o adquiere el software. Por ejemplo, un competidor que introduce un nuevo producto es un riesgo empresarial. La introducción de un producto competitivo puede significar que las suposiciones hechas sobre las ventas de los productos de software existentes sean excesivamente optimistas.

Ejemplo

Riesgos comunes para el proyecto, el producto y la empresa.

Riesgo	Repercute en	Descripción
Rotación de personal	Proyecto	Personal experimentado abandonará el proyecto antes de que éste se termine.
Cambio administrativo	Proyecto	Habrá un cambio de gestión en la organización con diferentes prioridades.
Indisponibilidad de hardware	Proyecto	Hardware, que es esencial para el proyecto, no se entregará a tiempo.
Cambio de requerimientos	Proyecto y producto	Habrá mayor cantidad de cambios a los requerimientos que los anticipados.
Demoras en la especificación	Proyecto y producto	Especificaciones de interfaces esenciales no están disponibles a tiempo.
Subestimación del tamaño	Proyecto y producto	Se subestimó el tamaño del sistema.
Bajo rendimiento de las herramientas CASE	Producto	Las herramientas CASE, que apoyan el proyecto, no se desempeñan como se anticipaba.
Cambio tecnológico	Empresa	La tecnología subyacente sobre la cual se construye el sistema se sustituye con nueva tecnología.
Competencia de productos	Empresa	Un producto competitivo se comercializa antes de que el sistema esté completo.

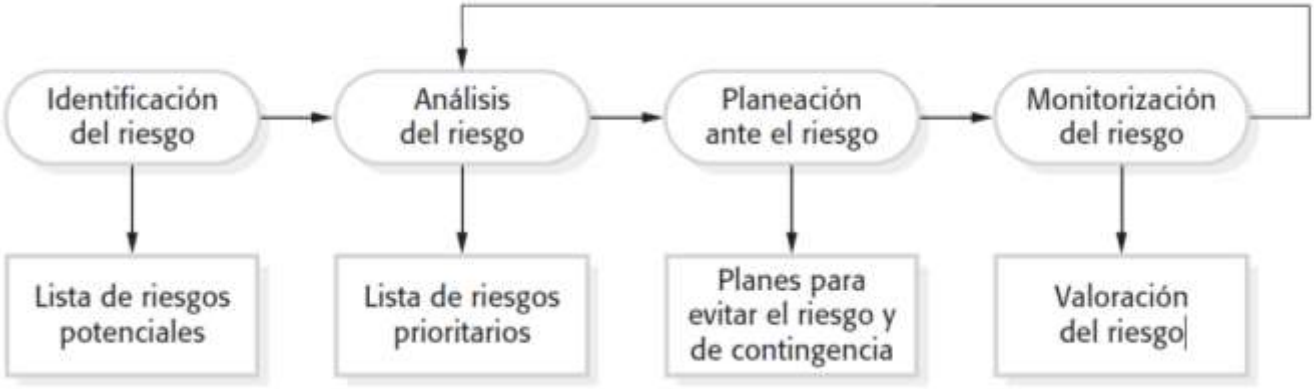
Tomado de Libro Ingeniería de Software 9 edición (Sommerville, 2011).

En la tabla anterior se ilustra una idea general del proceso de gestión del riesgo.
Comprende varias etapas:

2.4.6 EJERCICIO DE APRENDIZAJE

ETAPAS	DESCRIPCIÓN
IDENTIFICACIÓN DEL RIESGO	Hay que identificar posibles riesgos para el proyecto, el producto y la empresa.
ANÁLISIS DE RIESGOS	Se debe valorar la probabilidad y las consecuencias de dichos riesgos.
PLANEACIÓN DEL RIESGO:	Es indispensable elaborar planes para enfrentar el riesgo, evitarlo o minimizar sus efectos en el proyecto.
MONITORIZACIÓN DEL RIESGO:	Hay que valorar regularmente el riesgo y los planes para atenuarlo, y revisarlos cuando se aprenda más sobre el riesgo.
Nota: Para lograr e identificar los riesgos es fundamental tener presente el proceso como se observa en la figura ***"El proceso de gestión del riesgo"***	

El proceso de gestión del riesgo



Tomado de Libro Ingeniería de Software 9 edición (Somerville, 2011).

2.4.7 PUNTOS CLAVE

- La buena gestión de proyectos de software es esencial si los proyectos de ingeniería de software deben desarrollarse dentro del plazo y el presupuesto establecidos.
- La gestión del software es distinta de otras administraciones de ingeniería. El software es intangible. Los proyectos pueden ser novedosos o innovadores, así que no hay un conjunto de experiencias para orientar su gestión. Los procesos de software no son tan maduros como los procesos de ingeniería tradicionales.
- La gestión del riesgo se reconoce ahora como una de las tareas más importantes de la gestión de un proyecto.
- La gestión del riesgo implica la identificación y valoración de los grandes riesgos del proyecto para establecer la probabilidad de que ocurran; también supone identificar y valorar las consecuencias para el proyecto si dicho riesgo surge. Debe hacer planes para evitar, gestionar enfrentar los posibles riesgos.
- Las personas se sienten motivadas por la interacción con otros individuos, el reconocimiento de la gestión y sus pares, y al recibir oportunidades de desarrollo personal.
- Los grupos de desarrollo de software deber ser bastante pequeños y cohesivos. Los factores clave que influyen en la efectividad de un grupo son sus integrantes, la forma en que está organizado y la comunicación entre los miembros.
- Las comunicaciones dentro de un grupo están influidas por factores como el estatus de los miembros del grupo, el tamaño del grupo, la composición por género del grupo, las personalidades y los canales de comunicación disponibles.

2.4.8 EJERCICIO DE ENTRENAMIENTO

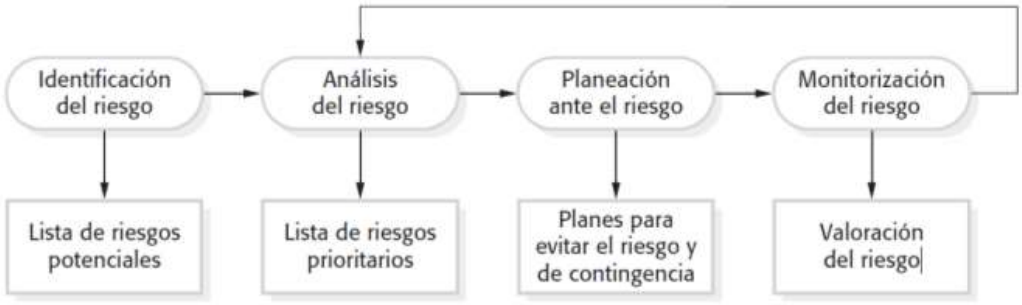
Teniendo como modelo el ejercicio de aprendizaje realice la siguiente actividad:

- Elabore a su proyecto de desarrollo de software:

  - Elabore un prototipo para su proyecto donde evidencie el proceso y el producto a desarrollar.
  - Se puede utilizar Word o adjunto este link donde hay diferentes herramientas http://www.campusmvp.es/recursos/post/Herramientas-de-prototipado-de-aplicaciones-Web.aspx

- Elabore un cronograma de actividades donde especifique las actividades a desarrollar, fecha de Inicio, fecha de fin.

- Realiza un plan de cambio, que especifique y evidencie los cambios realizados dentro del proyecto. Tenga presente el ciclo de cambio.



Tomado de Libro Ingeniería de Software 9 edición (Somerville, 2011).

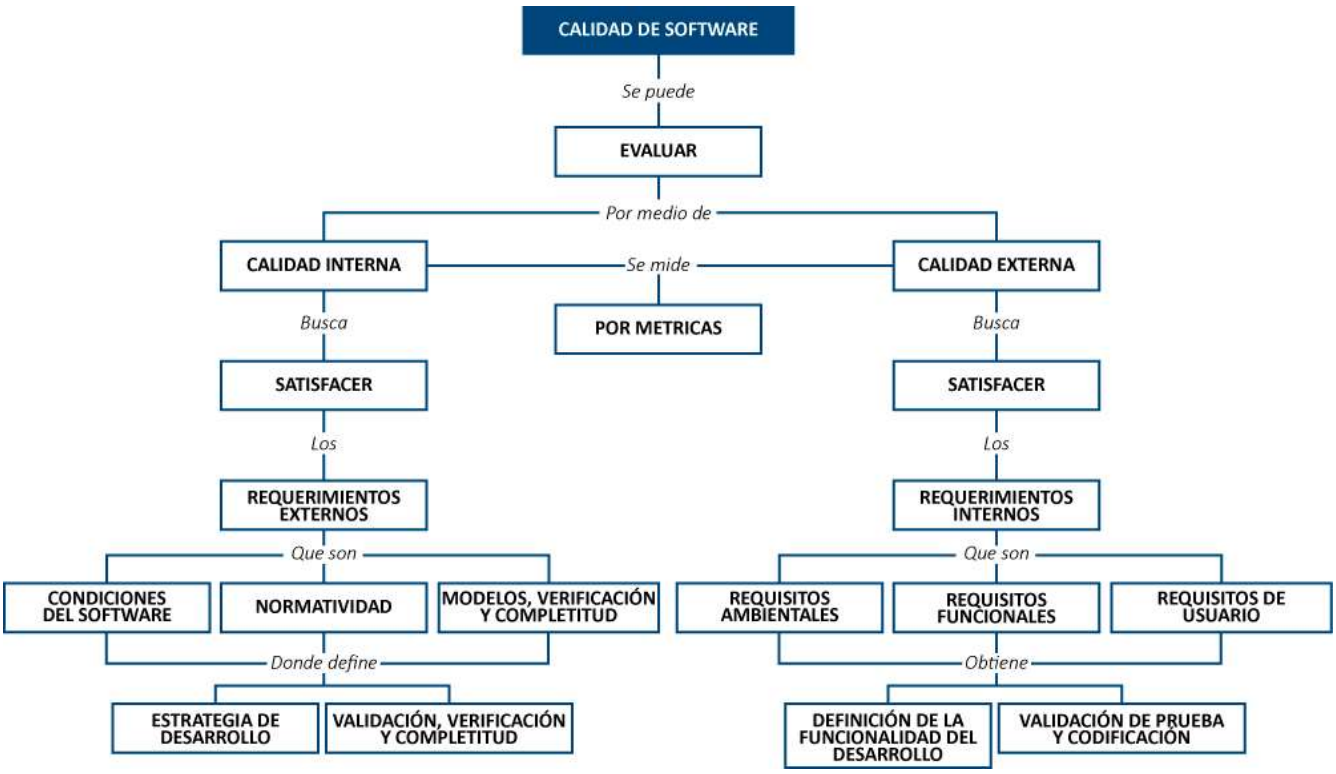
- Diseñe un plan donde estime recurso humano, tecnológico, físico con base a las especificaciones planteadas, con base a la siguiente tabla:

RECURSOS	ESPECIFICACIONES	COSTO	CANTIDAD
Humano			
Físicos			
Tecnológico			
Total			

3 UNIDAD 2 CALIDAD DE SOFTWARE

Se identifica un conjunto de características, definidas por la industria, de tal manera que se garantiza su característica (Funcionabilidad: que el usuario pueda utilizar el software, Confiabilidad: que los datos sean íntegros, Usabilidad: fácil de usar, fácil de aprender a usar, Portabilidad: compatible con otras plataformas, Compatibilidad: visible y ejecutable en la plataforma que corra, Corrección: capaz de darle mantenimiento, Eficiente: hace lo que debe bien, lo hace a tiempo y no derrocha recursos, Robustez: que se mantenga en un rito que debe Oportunidad: fácil de acceder, en cualquier momento), con base a los requerimientos solicitado por el clientes, donde se debe buscar que el grado de satisfacción sea llenada con sus expectativas.

3.1.1 RELACIÓN DE CONCEPTOS



3.2 TEMA 1 INTRODUCCIÓN A LA CALIDAD DE SOFTWARE

El objetivo general de la ingeniería de software es **la producción de software de calidad**. La calidad del software puede ser considerada desde dos perspectivas diferentes; la óptica del desarrollador y la del cliente o usuario final. Los factores que afectan al desarrollador se denominan Internos y los del cliente Externos.

El Compromiso con la calidad de software hoy en día es lograr gestionar y controlar innumerables sistemas que afectan a prácticamente todos los aspectos de nuestra actividad diaria, muchas de ellas críticas. El incorrecto funcionamiento o el no funcionamiento de alguna parte de él puede producir incomodidades y/o ciertos costes, pero también pueden conllevar graves riesgos económicos o incluso poner en peligro para la salud o la vida de

las personas además Lograr ofrecer productos de calidad es una responsabilidad que comienza por un compromiso personal: “Poner todos los medios para conseguir que proceso de desarrollo produzca bienes que cumplan las especificaciones requeridas.”

La Importancia de la Calidad en la Producción de Sofware Enlace

Calidad del software Enlace

3.2.1 DEFINICIÓN DE CONCEPTOS

- **Calidad**

  - Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor. Esta tela es de buena calidad. (RAE, 2010)
  - Condición o requisito que se pone en un contrato. (RAE, 2010)
  - Característica o atributo de algo (American Heritage Dictionary, 2003)
  - Conjunto de propiedades y de características de un producto o servicio, que le confieren aptitud para satisfacer una necesidad explícita o implícita (ISO 8402).
  - El grado en que un sistema, un componente o un proceso satisface las necesidades o expectativas de un cliente o usuario. (IEEE Std 610.12-1990)

En general la calidad, se refiere a características mensurables, longitud, color, propiedades eléctricas, maleabilidad, etc.

La ISO 8402 define:

- **Calidad:** "Conjunto de propiedades y características de un producto o servicio que le confieren su aptitud para satisfacer unas necesidades explícitas o implícitas"
- **Control de calidad:** "Conjunto de técnicas y actividades de carácter operativo, utilizadas para verificar los requerimientos relativos a la calidad del producto o servicio".
- **Garantía de calidad:** "Conjunto de acciones planificadas y sistemáticas necesarias para proporcionar la confianza adecuada de que un producto o servicio satisfará los requerimientos dados sobre calidad".
- **Gestión de la calidad:** "Aspecto de la función de gestión que determina y aplica la política de la calidad, los objetivos y las responsabilidades y que lo realiza con medios tales como la planificación de la calidad, el control de la calidad, la garantía de calidad y la mejora de la calidad".

  Nota: Es responsabilidad de todos los niveles ejecutivos, pero debe estar guiada por la alta dirección. Su realización involucra a todos los miembros de la organización y se tienen en cuenta también criterios de rentabilidad.

Cuenta con 4 etapas:

ETAPA	DESCRIPCIÓN
PLANEAR	Se establecen **los objetivos** del proceso, **requisitos** a cumplir, **la interacción** (entradas y salidas) con los otros procesos.
HACER	Se establecen las actividades que se realizan con el fin de dar cumplimiento a lo planeado, definiendo **responsables de la ejecución** de la actividad y **salidas** que a su vez serán insumo para otros procesos.
VERIFICAR	Se establecen **los mecanismos de verificación** necesarios para apoyar **seguimiento** y si así se requiere **medición de los procesos.**
ACTUAR	Se establecen las **acciones correctivas** y/o **preventivas** a implementar con el fin de alcanzar **los resultados planificados** y l**a mejora continua de** estos procesos.

Calidad de Software

> "La calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario". (IEEE, Std. 610-1990).

Proceso de la Calidad en el desarrollo de software

Para que el software pueda satisfacer al cliente debe ser usable y proporcionar una solución a una necesidad (explícita o implícita) del cliente, a un coste razonable.

- **Calidad es capacidad para ser usado.** (Juran)
- **Calidad es valor para el cliente**. (Weinberg)
- **Calidad es cumplir con los requisitos** (Crosby, Quality is free, 1979).
- **Calidad es cumplir con los requisitos de una persona determinada** (Weinberg, Software quality management).

3.2.2 EJERCICIO DE APRENDIZAJE

Un editor de textos que sea incapaz de manejar ficheros de más de 200 o 300 páginas será un software de calidad para el que sólo escribe documentos de pocas páginas. Sin embargo, el mismo editor de textos será un software de mala calidad para el que escriba libros o documentos extensos.

- ¿cero defectos => software de calidad?
- ¿muchas posibilidades => software de calidad?
- ¿código elegante => software de calidad?
- ¿rapidez => software de calidad?
- ¿precio => software de calidad?
- ¿facilidad de uso => software de calidad?

- **Diferencia entre software de calidad y calidad del software:**

- La **primera:** es **la percepción** que tiene un cliente.

- La **segunda**: es algo **objetivo** y **cuantificado**.

- **Política de Calidad**

Intenciones globales y orientación de una organización relativas a la calidad tal como se expresan formalmente por la Alta Dirección.

Proporciona el marco de referencia para el establecimiento de los objetivos de la calidad. (ISO 9000:2000)

Ejemplo:

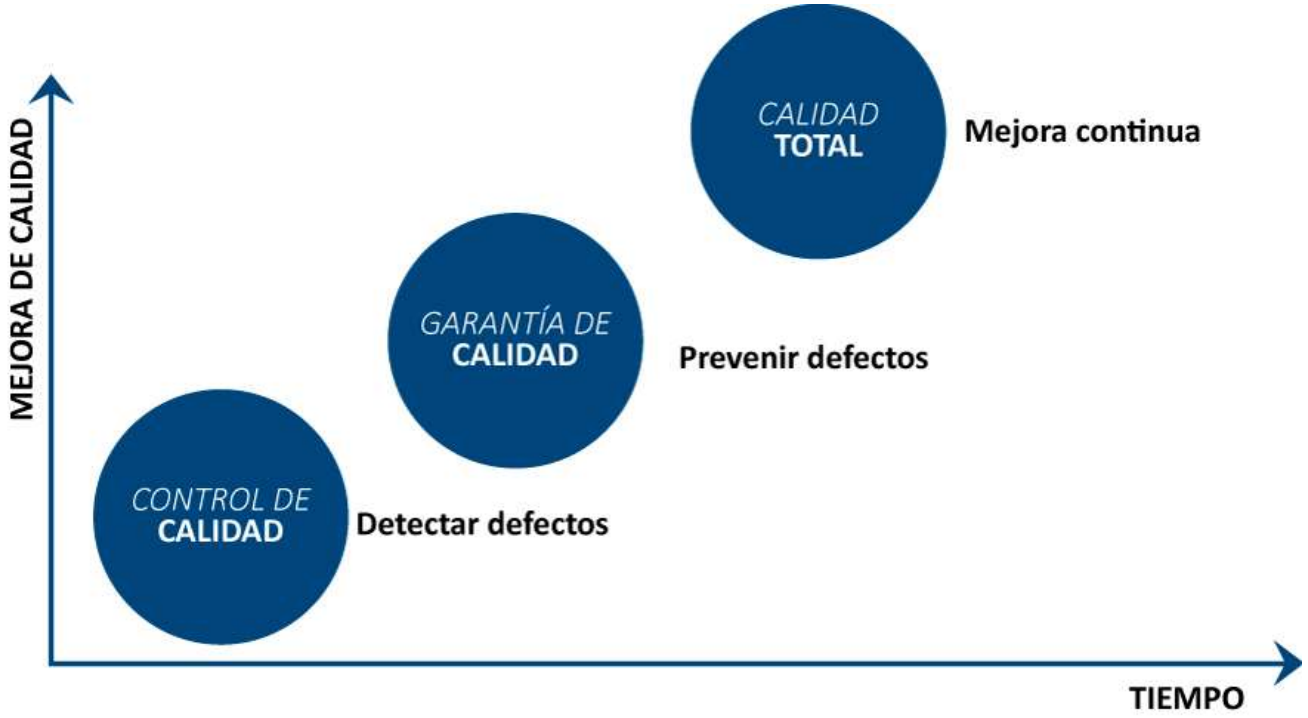
- Minimizar defectos
- Satisfacción del cliente.
- Prevención de defectos.
- Rapidez de entrega.
- Mínimo coste.
- Mantenibilidad.

3.2.3 FACTORES QUE DETERMINAN LA CALIDAD DEL SOFTWARE

- Se pueden clasificar en dos grandes grupos (Pressman):
  - Factores que pueden ser **medidos directamente.**
  - Factores que solo pueden ser **medidos indirectamente**.
- Se centran en tres aspectos importantes de un producto software (McCall):
  - **Características operativas**
    - ✓ **Corrección**. ¿Hace lo que quiero?
    - ✓ **Fiabilidad.** ¿Lo hace de forma fiable todo el tiempo?
    - ✓ **Eficiencia**. ¿Se ejecutará en mi hardware lo mejor que pueda?
    - ✓ **Seguridad** (Integridad). ¿Es seguro?
    - ✓ **Facilidad de uso**. ¿Está diseñado para ser usado?
  - **Capacidad** de soportar los cambios.
    - ✓ **Facilidad** de mantenimiento. ¿Puedo corregirlo?
    - ✓ Flexibilidad. ¿Puedo cambiarlo?
    - ✓ Facilidad de prueba. ¿Puedo probarlo?

- **Adaptabilidad** a nuevos entornos.
  - ✓ **Portabilidad**. ¿Podré usarlo en otra máquina?
  - ✓ **Reusabilidad.** ¿Podré reutilizar alguna parte del software?
  - ✓ **Interoperabilidad**. ¿Podré hacerlo interactuar con otro sistema?

- **Mejora de la calidad Vs Tiempo**



- **Situación actual en el desarrollo de software**

  - La industria del software no ha acabado de salir de la fase artesanal.
  - Padecemos de **"prisa patológica",** que es consecuencia directa de:
    - Desorganización.
    - Falta de planificación.
  - Alta dependencia de los "héroes".
  - Dedicamos nuestros esfuerzos de hoy a arreglar lo que se hizo mal ayer.
  - El producto (software) es algo intangible y no constreñido por las leyes físicas.
  - La disciplina, ingeniería del software, es relativamente reciente y muchos de sus conceptos importantes están aún inmaduros.

- Carencia de un corpus de conocimiento aceptado mayoritariamente que sirva como fundamentos.
- Escasa presión del mercado.
- Procesos software normalmente improvisados.
- Si se han especificado los requisitos, no se siguen rigurosamente.
- Organización reactiva (resolver crisis inmediatas).
- Planes y presupuestos excedidos sistemáticamente, al no estar basados en estimaciones realistas.
- No existen bases objetivas para juzgar la calidad del producto.
- Cuando los proyectos están fuera de plan, las revisiones o pruebas se recortan o eliminan.

- **Gestión de calidad en los procesos**

	Procesos	Objetivos	Resultados	
Gestión de la calidad	Planeamiento de la calidad	Precisar los clientes internos y externos	Calidad Planificada	Calidad asegurada
		Determinar las necesidades y elaborar productos y servicios que las satisfagan		
	Control de la calidad	Monitorear, medir, comparar y ajustar productos y servicios de acuerdo a lo planificado	Imperfecciones de la calidad detectadas	
	Mejoramiento de la calidad	Mejorar los procesos criticos y eliminar las actividades que no agregan valor	Imperfecciones de la calidad corregidas	

Tomado de Libro Ingeniería de Software 9 edición (Somerville, 2011).

3.3 TEMA 2 LAS 4P EN LA CALIDAD, PERSONAS, PRODUCTO, PROYECTO Y PROCESO

Para garantizar el proceso de calidad en el desarrollo de software se debe tener claro lo siguiente:

3.3.1 PERSONAS

Una persona de calidad en el proceso de software es quien **construye conocimiento**, no lo prueba, quien se rige por los requerimientos de los usuarios, quien está presto para mejorar el software.

3.3.2 PRODUCTO

La calidad del producto está estrechamente ligada al **proceso** que se lleva a cabo para producirlo, El sistema de producción es **calibrado** hasta conseguir **la producción de productos de calidad** teniendo presente lo siguiente:

CALIDAD LANIFICADA:	CALIDAD OBTENIDA:	CALIDAD REQUERIDA:
Técnicas, actividades técnicas y operativas que llevan a verificar los requisitos de calidad	Relacionada con las funcionalidades, puedo hacer lo que necesito	Relativa al cliente/producto que será mejor o peor.

Autoría propia

La relación calidad del proceso y producto es compleja:

- Dificultad de medición de atributos de calidad.
- Dificultad de entendimiento de las características de calidad en el proceso de software.
- Dificultad de predecir la influencia de los cambios en el proceso de calidad del producto.

La gestión de la calidad del producto implica:

1. **Definir estándares de proceso** revisión con fechas de calendario para llevarlas a cabo.
2. **Supervisar el proceso de desarrollo** para garantizar que se sigan esos estándares.
3. **Hacer informes periódicos** tanto para el equipo que desarrolla el proyecto como para el propietario o comprador.

- **Definición del Proceso para el desarrollo del Producto**

3.3.3 PROYECTO

El proceso de gestión de un proyecto de software comienza con un conjunto de actividades que globalmente, se denominan planificación del proyecto.

"No podemos pedir exactitud a la fase de planificación, es solo una idea de cómo van a transcurrir las cosas. Hay que planificar el trabajo, los recursos humanos y la tecnología. Planificar es estimar."

La primera de estas actividades es **la estimación**. La estimación conlleva un riesgo que lleva a **la incertidumbre.**

La Estimación es la base de todas las demás actividades de planificación del proyecto y sirve como guía para una buena ingeniería del software. Echar un vistazo al futuro y aceptar cierto grado de incertidumbre.

"Es aplicar el sentido común y el conocimiento a un determinado problema."

Nota: No debe llevarse a cabo de forma descuidada.

"En informática, no podemos fiarnos de la intuición, una buena planificación acaba permitiendo abordar un proyecto y ayuda al proceso de refinamiento."

La estimación de recursos, costes y planificación temporal de un proyecto requiere experiencia, una buena información histórica y confiar en medidas cuantitativas cuando todo lo que conocemos son datos cualitativos.

- **La complejidad del proyecto** tiene un gran efecto en la incertidumbre, presente en oda planificación y es una medida relativa que se ve afectada por la familiaridad con esfuerzos anteriores (experiencia adquirida en proyectos anteriores).

- **El tamaño del proyecto** es otro factor importante que puede afectar a la precisión de las estimaciones. Conforme aumenta el tamaño, crece rápidamente la interdependencia entre varios elementos del software.

  - ✓ El grado de incertidumbre en la especificación de requisitos tiene efecto en el riesgo de la estimación.

  - ✓ La disponibilidad de información histórica también determina el riesgo de la estimación.

Tomado de http://biblioguias.biblioteca.deusto.es/c.php?g=149240&p=2089803

Los principales problemas en la planificación de un proyecto de ingeniería de software incluyen los siguientes:

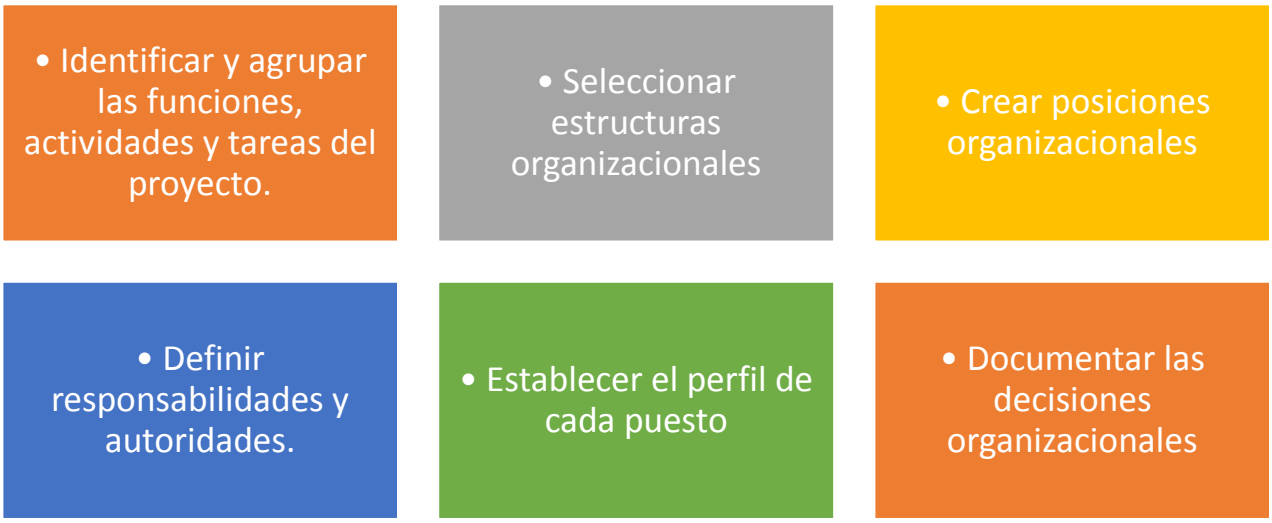
- Requerimientos incorrectos e incompletos.
- Muchas especificaciones de requerimientos son inestables y sujetas a cambios mayores.
- La planificación no se lleva a cabo por la creencia errónea de que es una pérdida de tiempo y los planes cambiarán de todos modos.
- La planificación de costos y plazos no es actualizada y se basa en necesidades de mercadeo y no de los requerimientos del sistema.
- Es difícil estimar el tamaño y complejidad del proyecto de software de modo de realizar una estimación de costos y plazos realista.
- Los costos y plazos no son re estimados cuando los requerimientos del sistema o el ambiente de desarrollo cambia.
- No se manejan factores de riesgo.
- La mayoría de las organizaciones de desarrollo de software no recolectan datos de proyectos pasados.
- Las compañías no establecen políticas o procesos de desarrollo de software.

Actividades que se derivan de la planificación de un proyecto de software:

- Fijar los objetivos y metas
- Desarrollar estrategias

- Desarrollar políticas
- Anticipar futuras situaciones
- Conducir un establecimiento de riesgos
- Determinar posibles cursos de acción
- Tomar decisiones de planificación
- Fijar procedimientos y reglas
- Desarrollar los planes del proyecto
- Preparar presupuestos
- Documentar los planes del proyecto.

Actividades que se derivan del proceso en el desarrollo de software en la Organización



Autoría Propia

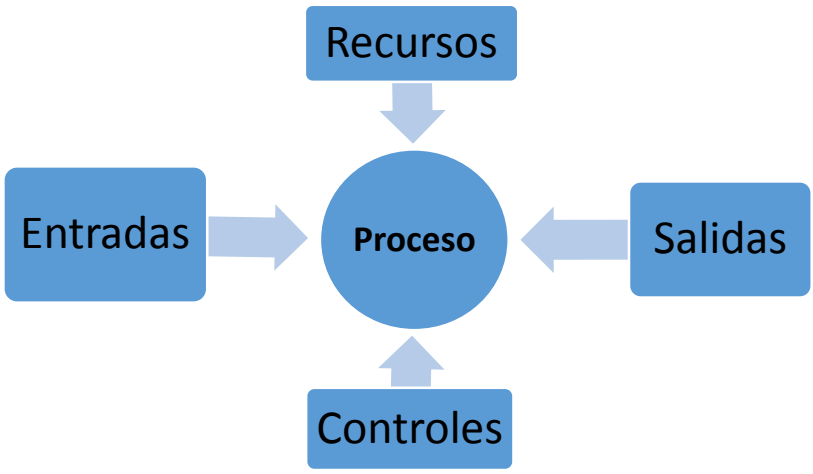
3.3.4 PROCESO

Un proceso se define como un conjunto de tareas, actividades o acciones interrelacionadas entre sí que, a partir de una o varias entradas de información, materiales o de salidas de otros procesos, dan lugar a una o varias salidas también de materiales (productos) o información con un valor añadido Además…

- Es repetible en el tiempo.
- Tiene un dueño.

- Se relaciona con otros procesos.

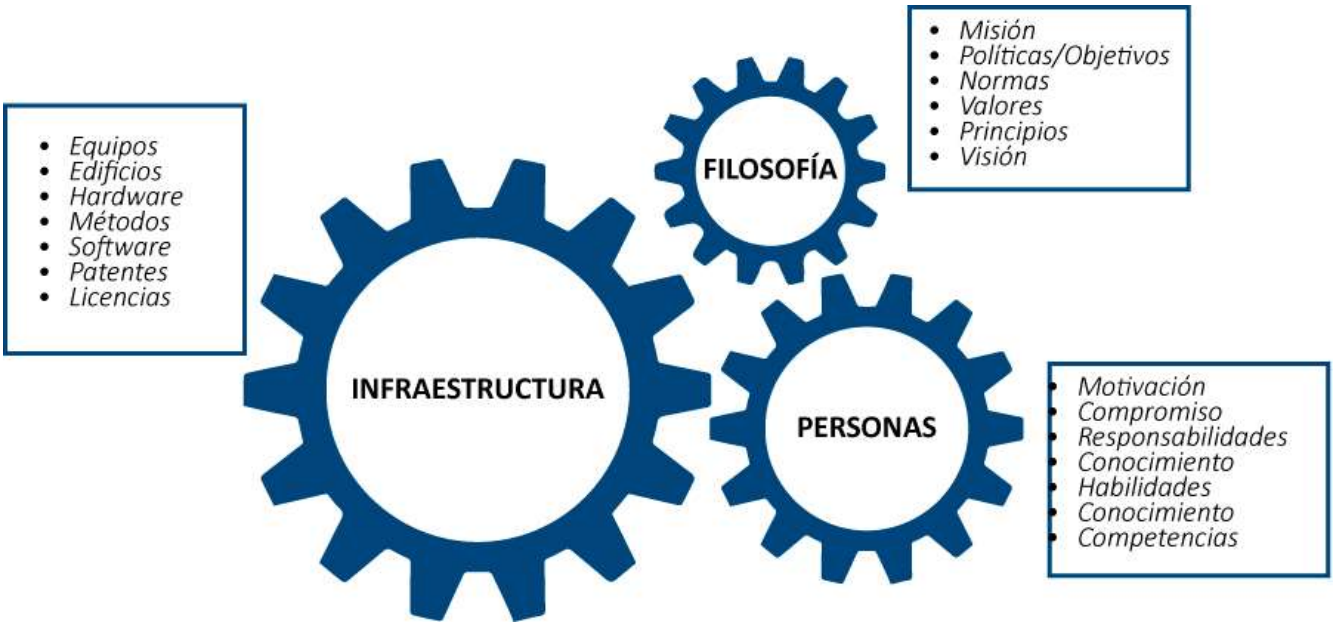
Control de proceso



Autoría propia

La importancia de los procesos:

- Los procesos son recolectores de conocimiento.
- Los procesos se pueden utilizar como referentes para valorar el desempeño.
- Los procesos hacen repetibles las tareas en aras de lograr una efectividad y desempeño previstos y uniformes.
- Los procesos pueden servir para apoyar los objetivos de negocio y estrategia empresarial si se conciben adecuadamente.
- Los procesos dotan a la organización de una identidad empresarial.



Autoría propia

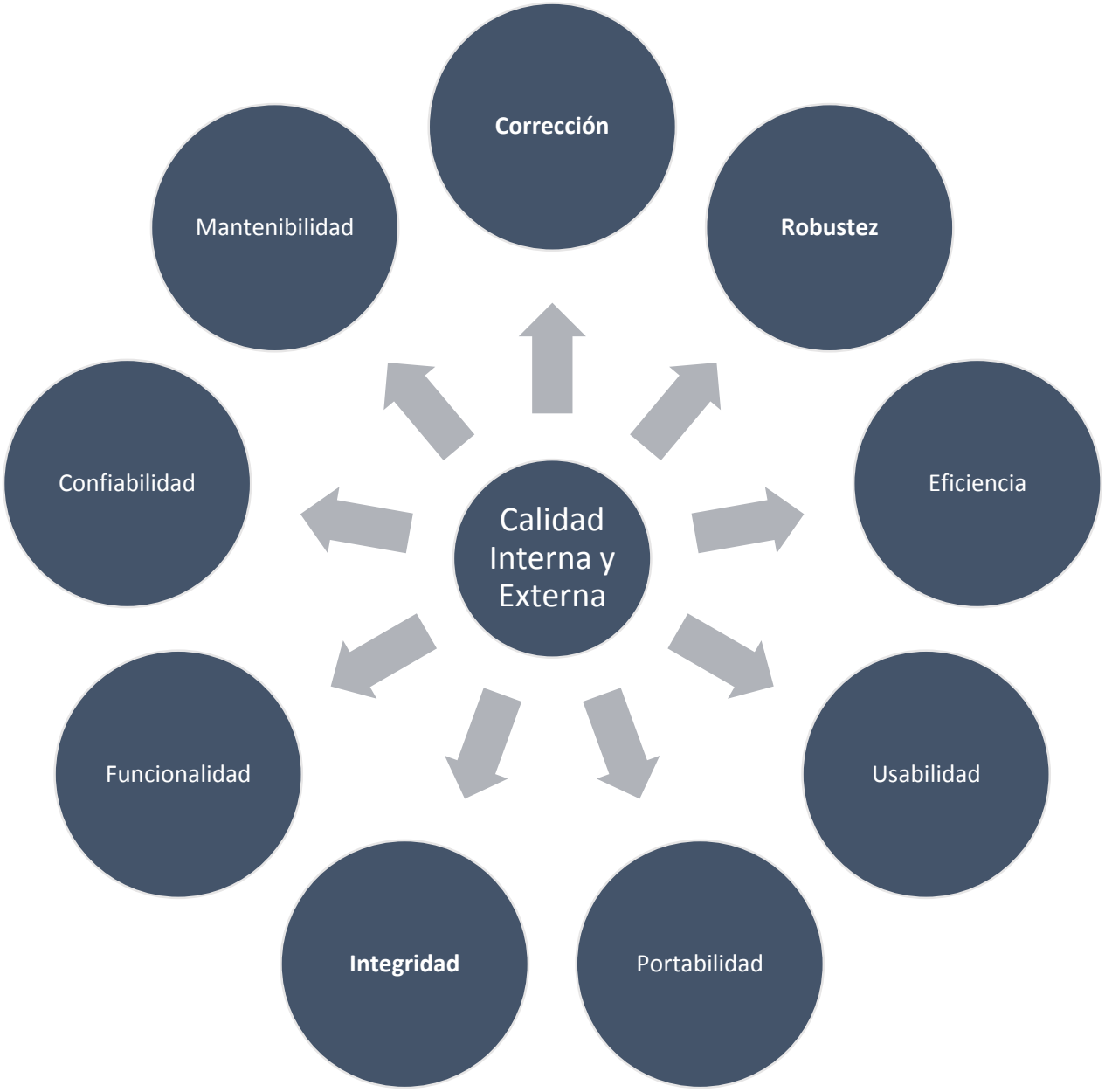
3.4 TEMA 3 CALIDAD INTERNA, CALIDAD EXTERNA

La calidad del software no es algo en lo que se empieza a pensar una vez que se ha generado el código, es una "actividad de protección" que se aplica a lo largo de todo el proceso de ingeniería software:

1. Un **enfoque de gestión** de **la calidad**.
2. Tecnología de ingeniería del software o del conocimiento **efectivo** (métodos y herramientas).
3. **Revisiones técnicas formales** que se aplican durante cada paso de la ingeniería del software o del conocimiento. Apoyado en una estrategia de pruebas por niveles.
4. **Procesos de gestión de la configuración**. El control de la documentación del software y de los cambios realizados.
5. **Un procedimiento** que asegure un ajuste a los estándares de desarrollo del software (cuando sea posible).
6. Mecanismos de **medición** y de **información.**
7. Calidad **interna** y **externa**.

Nota: Las características para calidad interna y externa se subdividen y manifiestan externamente cuando el software es usado como parte de un sistema Informático, donde los resultados son atributos internos de software.

- **Calidad Interna y Externa**



Autoría Propia

CORRECCIÓN

Es la capacidad de los productos software para **realizar con exactitud las tareas expresadas** en su especificación.

Uno de los problemas de la corrección es que se presupone la confianza en los distintos componentes involucrados en la producción del sistema; compilador, bibliotecas, módulos, Sistema operativo, etc.

ROBUSTEZ

Es la capacidad de los productos software de **reaccionar apropiadamente ante condiciones excepcionales**.

La robustez viene a ser el complemento de la corrección. En implementación se cuenta con el mecanismo de ***excepciones*** el cual garantiza el correcto flujo de ejecución del código. (Programación por contrato)

EFICIENCIA

Es la capacidad del software para hacer buen **uso de los recursos que manipula**.

Una práctica muy común en los desarrolladores es la optimización excesiva, lo importante es mantener un balance adecuado entre eficiencia y corrección.

PORTABILIDAD

Es la **facilidad** con que un sistema software puede **ser migrado entre diferentes plataformas hardware o software.**

La **facilidad de uso** es un factor determinante en términos de mercadeo y venta, ya que es el principal elemento que afecta al usuario final. La facilidad de uso incluye prestancia en ***instalación, operación*** *y* ***supervisión***

INTEGRIDAD

Es la característica de un sistema de ser **capaz de proteger sus diferentes componentes contra los procesos o elementos** que no tengan derecho de acceso a los mismos.

La integridad es un factor muy importante en sistemas contables, administrativos y gerenciales ya que de ellos depende el capital de la empresa.

USABILIDAD

Es la **facilidad** con la que un **usuario puede interactuar con un sistema software**.

La facilidad de uso es un factor determinante en términos de mercadeo y venta, ya que es el principal elemento que afecta al usuario final. La facilidad de uso incluye prestancia en ***instalación, operación*** *y* ***supervisión***.

- **VERIFICABILIDAD**

Es la **facilidad de verificación de corrección de un software**. Que tan sencillo es la realización de pruebas que garanticen la funcionalidad del sistema.

La prueba exhaustiva de un software es un concepto imposible de ejecutar debido al carácter infinito de flujos de ejecución del sistema. Sin embargo, la Ingeniería de software plantea estrategias generales para garantizar un alto grado de corrección.

- **COMPATIBILIDAD**

Es la **facilidad combinar diferentes elementos software** con el fin de ejecutar una labor en conjunto.

La mayoría de los sistemas son abiertos (interactúan con otros sistemas), y el dinamismo inherente a la realidad hace muy probable que los sistemas software tengan que intercambiar información entre sí. Esto hace que la compatibilidad sea un factor muy serio al momento de modelar el sistema.

- **REUTILIZACIÓN**

Es la **capacidad de los productos software** para funcionar como módulos básicos de la construcción de diferentes aplicaciones.

El objetivo general de la reutilización es adaptar la industria del software al modelo de otras industrias (como la electrónica), donde la producción se basa en un conjunto de elementos discretos preexistentes con una función genérica (circuitos integrados, resistencias, transformadores), que se ensamblan para dar origen a un nuevo producto.

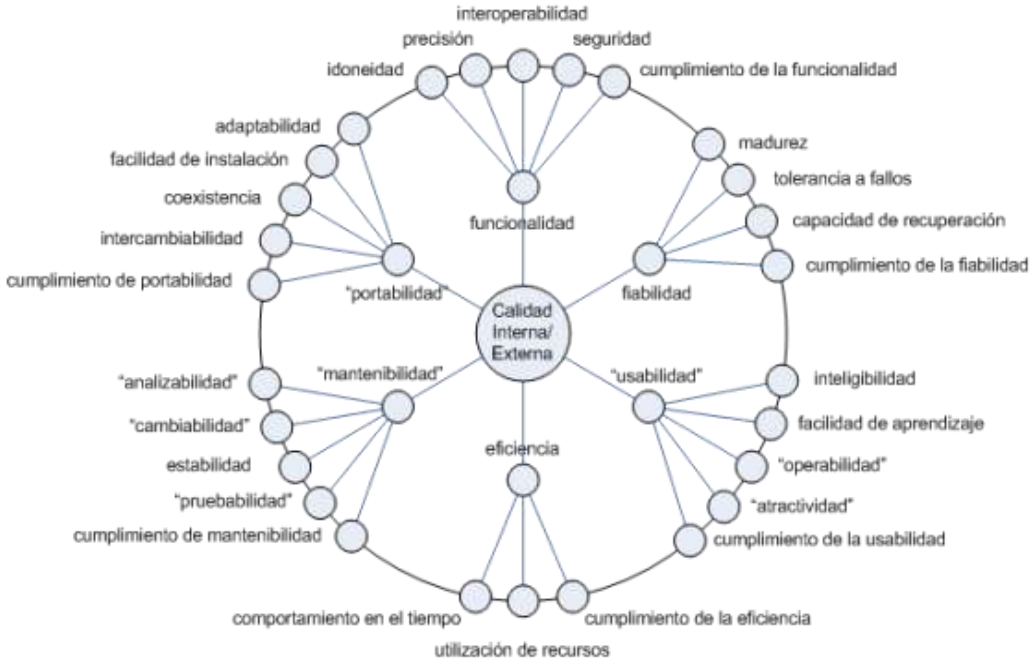
La reutilización es uno de los aspectos más importantes de la producción de software.

Características y subcaracterísticas

CARACTERISTICA	PREGUNTA	SUBCARATERISTICA	PREGUNTA
FUNCIONALIDAD	¿Las funciones y Propiedades satisfacen las necesidades Explícitas e implícitas; esto es, el qué?	ADECUACIÓN	¿Tiene el conjunto de funciones apropiadas para las tareas especificadas?
		EXACTITUD	¿Hace lo que fue acordado en forma esperada y correcta?
		INTEROPERABILIDAD	¿Interactúa con otros sistemas especificados?
		CONFORMIDAD	¿Está de acuerdo con las leyes o normas y estándares, u otras prescripciones?
CONFIABILIDAD	¿Puede mantener el nivel de rendimiento, bajo ciertas condiciones y por cierto tiempo?	MADUREZ	¿Con qué frecuencia presenta fallas por defectos o errores?
		TOLERANCIA A ERRORES	¿Si suceden fallas, como se comporta en cuanto a la performancia especificada?
		RECUPERABILIDAD	¿Es capaz de recuperar datos en caso de fallas?
USABILIDAD	¿El software, es fácil de usar y de aprender?	ENTENDIMIENTO	¿Es fácil de entender y reconocer la estructura y la lógica y su aplicabilidad?
		APRENDIZAJE	¿Es fácil de aprender a usar?
		OPERABILIDAD	¿Es fácil de operar y controlar?
		ATRACCIÓN	¿Es atractivo el diseño del software?
EFICIENCIA	¿Es rápido y minimalista en cuanto a uso de recursos, bajo ciertas condiciones?	COMPORTAMINETO DE TIEMPOS	¿Cuál es el tiempo de respuesta y performancia en la ejecución de la función?
		UTILIZACION DE RECURSOS	¿Cuántos recursos usa y durante cuánto tiempo?
CAPACIDAD DE MANTENIMINETO	¿Es fácil de modificar y testear?	CAPACIDAD DE SER ANALAIZADO	¿Es fácil diagnosticar una falla o identificar partes a modificar?
		CAMBIALIDAD	¿Es fácil de modificar y adaptar?
		ESTABILIDAD	¿Hay riesgos o efectos inesperados cuando se realizan cambios?
		FACILIDAD DE PRUEBA	¿Son fáciles de validar las modificaciones?
PORTABILIDAD	¿Es fácil de transferir de un ambiente a otro?	ADAPTABILIDAD	¿Es fácil de adaptar a otros entornos con lo provisto?
		FACILIDAD DE INSTALACION	¿Es fácil de instalar en el ambiente especificado?
		REMPLAZABILIDAD	¿Es fácil de usarlo en lugar de otro software para ese ambiente?
		COEXISTENCIA	¿Comparte sin dificulta recursos con otro software o dispositivo?

Tomado de: guia_tecnica_para_evaluacion_de_software.pdf

- **Calidad Interna y externa**



Tomado de http://olgacarreras.blogspot.com.es/2012/03/estandares-formales-de-usabilidad-y-su.html

3.4.1 EJERCICIO DE ENTRENAMIENTO

Apreciado estudiante, el siguiente ejercicio le servirá para evidenciar los conceptos aprendidos de la Unidad, trabájela con gran responsabilidad, si es del caso revise conceptos anteriores, ejemplos y ejercicios que te ayuden en la solución del mismo, recuerde que son conceptos básicos y de gran aplicabilidad para la calidad de software, por lo tanto es importante que los tenga siempre presentes.

- Describa un caso real e identifique problemas relacionados con calidad que usted considere Importantes.
- Clasifique para su proyecto la Calidad interna y externa con base a las características.
- Responda la siguiente pregunta: **¿CÓMO CONSEGUIR LA CALADID?**

Solución

- Describa un caso real e identifique problemas relacionados con calidad que usted considere Importantes.
  - No hacer un buen levantamiento de los requisitos de software.
  - No hacer una buena del software que va a ser controlado
  - No Seleccionar una medida que pueda ser aplicada al objeto de control.
  - No determinar los métodos de valoración de los indicadores.

- No Definir las regulaciones organizativas para realizar el control (quiénes participan en el control de la calidad, cuándo se realiza, qué documentos deben ser revisados y elaborados, etc.)
- Ignorar las opiniones de los stakeholder.
- La falta de cumplimiento de los requerimientos funcionales y no funcionales explícitamente definidos.

Clasifique para su proyecto la Calidad interna y externa con base a las características.

Según el proyecto que se esté laborando se puede clasificar de la siguiente manera:

- **Funcionalidad:**
  - Idoneidad en el acceso al sistema.
  - Cumplimiento de la funcionalidad en el acceso al módulo.
- **Usabilidad:**
  - Integralidad de los datos.
- **Eficiencia:**
  - Utilización de los recursos.

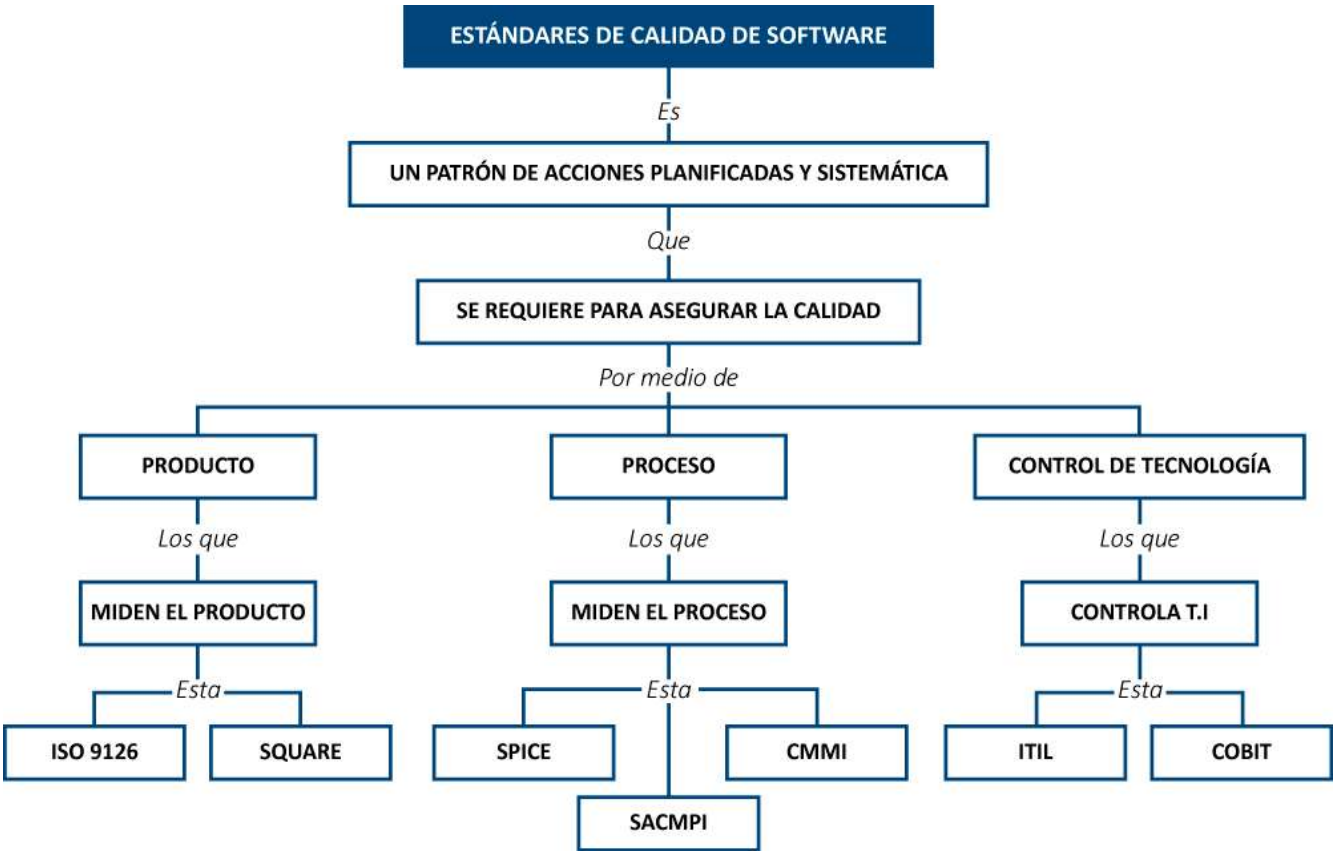
La calidad se consigue por medio de:

- Mediante la utilización de metodologías o procedimientos estándares para el análisis, diseño, programación y prueba del software que permitan constatar lo que el cliente solicito.
- La adopción de una buena política de Calidad.
- Un buen control o evaluación de los requisitos.
- Definir el software que va a ser controlado
- Seleccionar una medida que pueda ser aplicada como control en cuanto al producto y proceso.
- Crear o determinar los métodos de valoración de los indicadores (Métricas).
- Generar métodos y herramientas de análisis, diseño, codificación y prueba.
- Revisiones y técnicas formales que se aplican en cada fase de la ingeniería de software.
- Generar estrategias de procedimiento que asegure un ajuste a los estándares de desarrollo.
- Mecanismos a medida y de información.

4 UNIDAD 3 ESTÁNDARES

El software se rige por licencias de utilización, es decir, en ningún momento un usuario compra un programa o se convierte en propietario de él, tan sólo adquiere el derecho de uso, incluso así haya pagado por él. Las condiciones bajo las cuales se permite el uso del software, o sea las licencias, son contratos suscritos entre los productores de software y los usuarios, en esta unidad se reconocerán las Normatividad nacional e Internacional en cuanto se rige la norma.

4.1.1 RELACIÓN DE CONCEPTOS



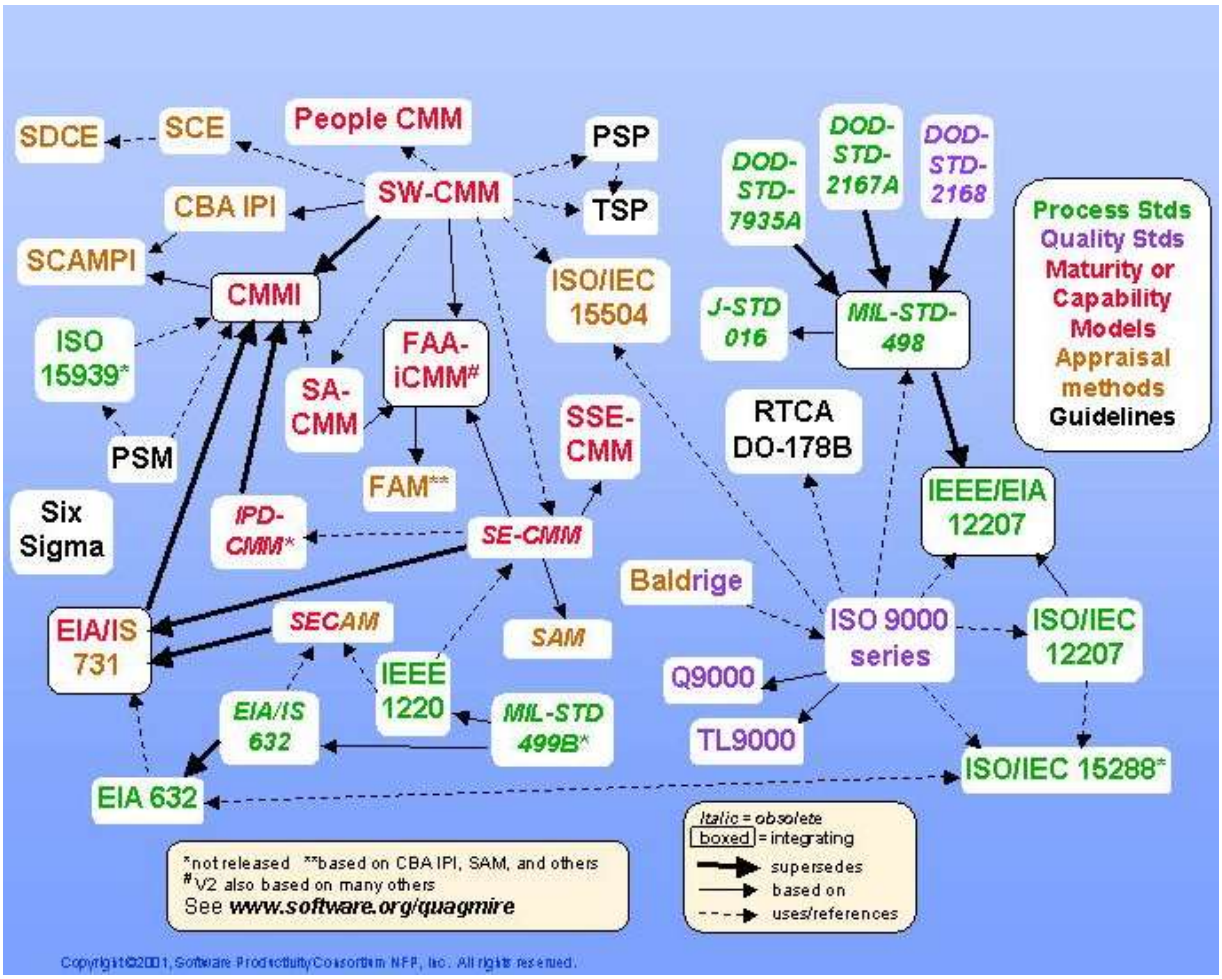
Calidad del software - MODELOS Y ESTANDARES DE UN PRODUCTO Enlace

4.2 TEMA 1 INTRODUCCIÓN A LA NORMATIVIDAD NACIONAL E INTERNACIONAL SOBRE SOFTWARE

Los Estándares de calidad aplicados al desarrollo de software , permite mejorar la calidad un tema fundamental importante para poder satisfacer las necesidades puntuales de los clientes, La calidad es un fundamento en los desarrolladores por el cual buscan opciones del como poder desarrollar software de calidad y en ello se han creado desde hace mucho tiempo atrás los estándares que hoy en día rigen en torno a este mundo para el desarrollo correcto de aplicaciones de calidad cumpliendo con sus normas y parámetros en aras de conseguir la ansiada calidad.

Para lograr el éxito en el desarrollo de software es necesario hacerlo con eficiencia y demostrar su buena usabilidad. Esto sólo es posible con los modelos de Calidad ISO, IEEE, CMMI, SPICE, SQUARE, ITIL.

A continuación, se refleja en el grafico la normatividad y modelos para la adopción de la Calidad.



Tomado de www.software.org/quagmire

Para entender mejor el proceso se tiene lo siguientes estándares de calidad basado en el producto, proceso y los que controlan el producto y proceso

LOS QUE MIDEN EL PRODUCTO	LOS QUE MIDEN EL PROCESO	LOS QUE CONTROLA Y SE ENFOCAN EN TI
ISO 9126 SQUARE	SPICE CMMI SCAMPI	ITIL COBIT

Autoría Propia

4.3 TEMA 2 PMBOK (PROJECT MANAGEMENT BODY OF KNOWLEDGE)

Contiene prácticas que han sido compiladas y mejoradas durante los últimos veinte años gracias al esfuerzo de profesionales y académicos de diversos ámbitos de ingeniería.

Es uno de los más importantes documentos publicados en la actualidad por el Project Management Institute que produce documentos y prácticas generalmente aceptadas de **dirección y de gestión** de proyectos.

Objetivo

- Desarrollar una metodología integral para la concepción, **planificación** y ejecución de proyectos.
- Identificar los **procesos** de administración de proyectos siguiendo al **PMBOK**.

Proyecto: Conjunto de actividades interdependientes orientadas a un fin específico.

¿Qué significa terminar con éxito un proyecto?

Cumplir con los objetivos ***dentro de las especificaciones técnicas***, ***de costo*** y ***de plazo de término***.

Los proyectos orientados a un objetivo superior se denominan **PROGRAMA** y un conjunto de Programas constituye un **PLAN.** Responder al ¿Qué? ¿Cómo? ¿Por qué?

- **Dimisiones PMBOK**
  - **Técnica:**
    - **Aplicar los conocimientos específicos** de cada área de trabajo, cumpliendo con una forma de trabajar y unos requisitos ("know how") que cada profesión impone.
    - **Disponer de los conocimientos adecuados** para resolver el problema en cuestión o realizar la obra encomendada.
    - **La importancia de esta faceta técnica** no debe eclipsar el resto de aspectos que intervienen en la consecución de un proyecto, y que otorgan a esta actividad de una trascendencia y complejidad mayor.
  - **Humana:**
    - Un proyecto es un complejo entramado de relaciones personales, donde se dan cita un gran número de intereses a veces contrapuestos.
    - Regularmente existen diferencias que surgen por ejemplo entre el jefe de proyecto y cliente o proveedores.

- Disputas entre departamentos al momento de repartir los recursos de que se dispone, pues son varios los proyectos que se pueden estar llevando a cabo paralelamente en la organización.

- **Gestión:**

- Dimensión menospreciada porque no es tan espectacular o visible como otros elementos

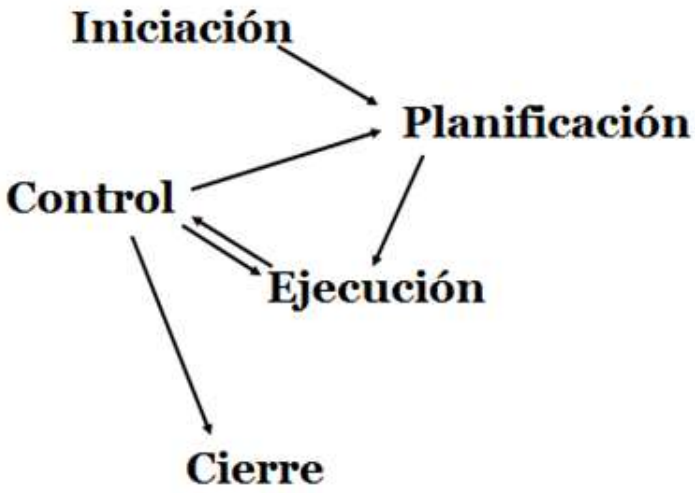
- "Lamentablemente" es el catalizador que permite que el resto de los elementos se comporten adecuadamente.

- El éxito del proyecto depende en gran medida de la gestión.

¿Porque PMBOK?

Aporta todas las bases para una gestión adecuada independiente de la ingeniería o construcción.

Etapas del Proceso



Autoría propia

- **Buenas prácticas para la gestión de Proyecto**

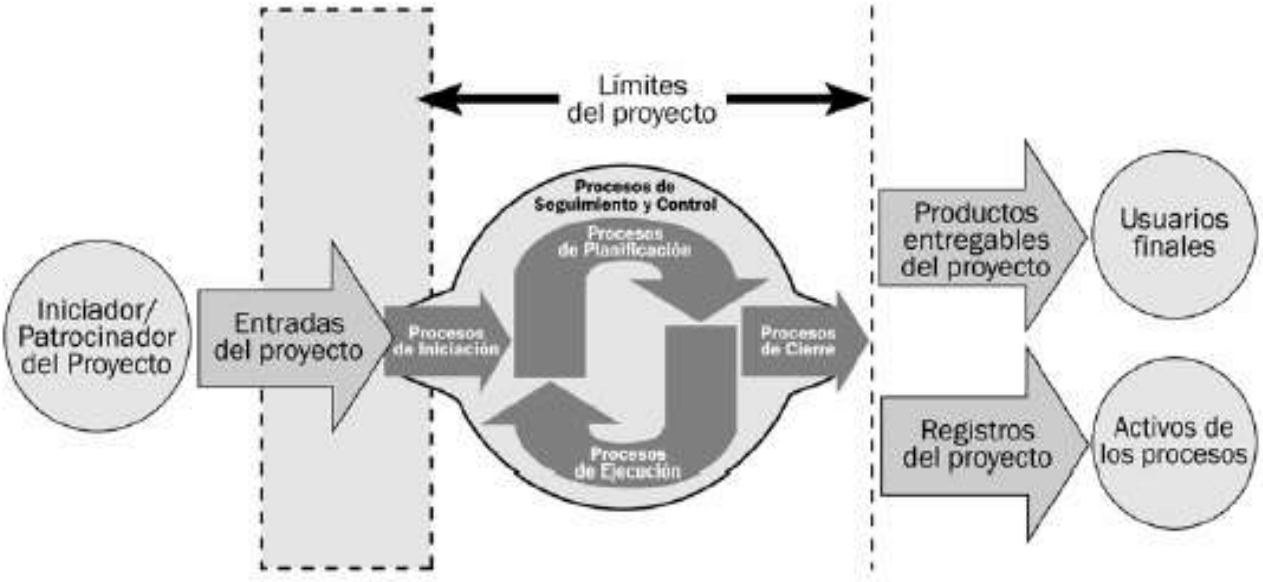
- **Segmentación:** el proyecto debe ser separado en un número manejable de actividades y tareas.

- **Interdependencia:** la planificación debe reflejar la segmentación de tareas, y la relaciones entre ellas. Hay algunas tareas ocurren en secuencia, otras en paralelo, algunas son requisito de otras, etc.

- **Asignación de tiempo:** a cada tarea se le debe asignar un cierto número de unidades de trabajo (personas-días, etc.) y fechas de inicio y término.

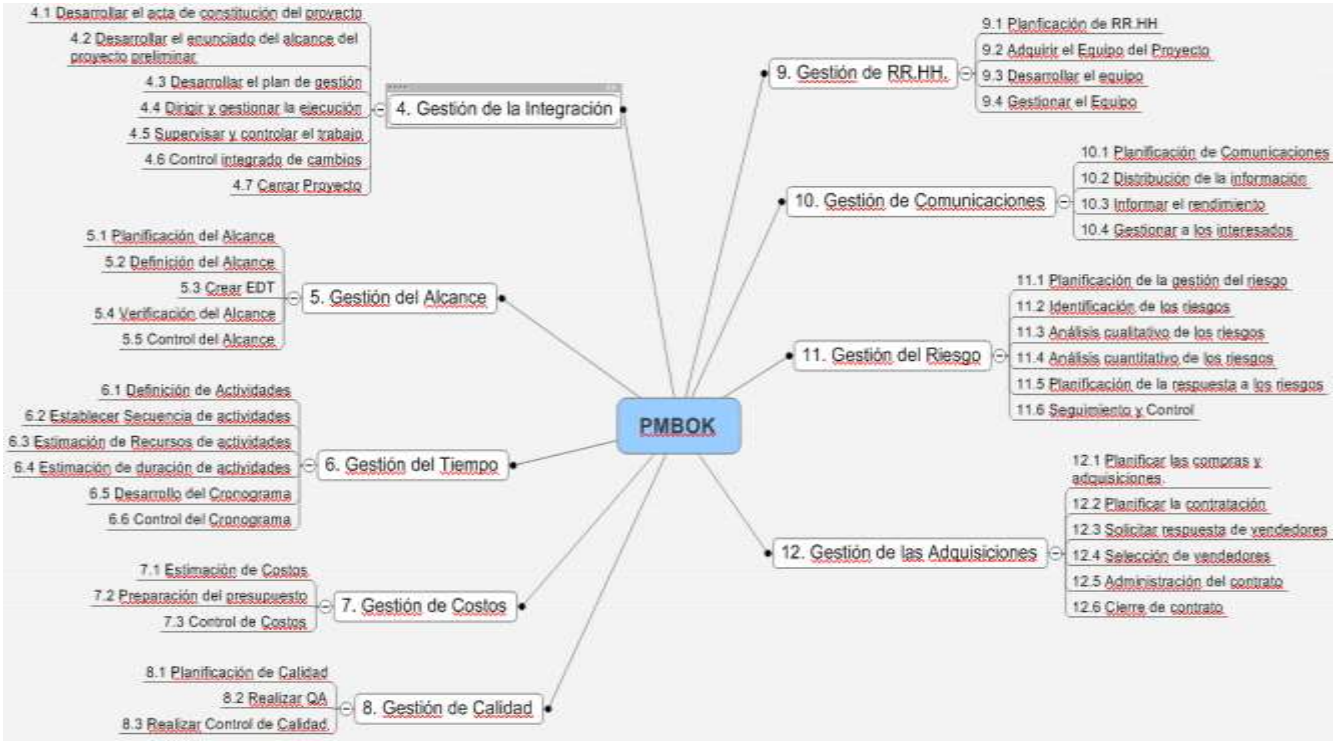
- **Validación del esfuerzo:** se debe verificar que la gente asignada a una tarea esté disponible y además que sea suficiente.
- **Definición de responsabilidades:** cada tarea debe tener un responsable.
- **Salida definida:** cada tarea debe tener un "producto" bien definido. Por ejemplo: diseño de un módulo, plan de pruebas, etc.
- **Definición de metas:** cada grupo de tareas se debe estar asociado con una meta.

Límites del proyecto



Tomado de http://slideplayer.es/slide/3929857

Resumen: Cuerpo PMBOK



Tomado de Libro Vista ampliada para Gerencia de Proyectos usando mejores prácticas del PMBok®

4.4 TEMA 3 MODELO DE CALIDAD ISO, IEEE, CMMI, SQUARE, SPICE, ITIL

4.4.1 ISO 9126

Estándar ISO/IEC 9126/1991

→ ISO/IEC 9126: Calidad del producto software.

→ ISO/IEC 14598: Evaluación del producto software.

Estándar ISO/IEC 25000

→ Es una nueva versión de la Norma ISO/IEC 9126

Proporciona una guía para el uso de las nuevas series de estándares internacionales llamados Requerimientos y evaluación de calidad de productos de software (SQUARE).

Es un estándar internacional para la evaluación del Software.

Está dividido en cuatro partes:

- **Modelo de calidad.**
- **Métricas externas.**
- **Métricas internas.**
- **Calidad en las métricas de uso.**

El modelo de calidad presenta un conjunto estructurado de características y subcaracterísticas de calidad.

4.4.2 SQUARE (SOFTWARE QUALITY REQUERIMENTS AND EVALUATION)

Es una revisión y nueva serie de normas que se basa en ISO/IEC 9126 y en ISO 14598 (Evaluación de software) y conserva las mismas caracteristicas de calidad del software, se dedica unicamente a la calidad del software. Nace con el objetivo de responder a las necesidades de los usuarios a traves de un conjunto de documentos unificados cubriendo tres procesos de calidad complementarios: Especificación de Requisitos, Medidas y Evaluación.

Es una nueva serie de normas que se basa en ISO9126 y en ISO14598 (Evaluación de Software). Es una revisión de estas normas. Surge una nueva versión de la norma ISO9126, la norma ISO25000.

Es un estándar internacional que guía el desarrollo de los productos de software con la especificación y evaluación de requisitos de calidad. Se centra en el lado del producto. Incluye un estándar de requerimientos de calidad, que se basa en la calidad del producto software.

Proceso

Autoría propia

> **Nota:** El proceso SQUARE se puede reutilizar para determinar cómo el cambio podría afectar a las condiciones de seguridad del sistema.

4.4.3 SPICE

Evaluación y **mejora** de procesos de **desarrollo** y **mantenimiento** de sistemas de información y productos de software.

Objetivo

- **Proporcionar** un marco en el cual se puedan armonizar los distintos enfoques prácticos existentes.
- **Aplicar** el estándar desarrollado en la industria del software.
- **Promover** la transferencia de conocimiento y de tecnología sobre los procesos de software a nivel mundial.

4.4.4 CMMI

Integración de Modelos de Madurez de Capacidades o **Capability Maturity Model Integration** (CMMI) es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software.

Las mejores prácticas CMMI se publican en los documentos llamados modelos. En la actualidad hay tres áreas de interés cubiertas por los modelos de CMMI:

- **Desarrollo.**
- **Adquisición.**
- **Servicio.**

Proceso CMMI

Tomado de http://blogs.epakistan.com/is-cmmi-extremely-hard-to-implement/

- **Finalidad**

  - Permite definir y gestionar los procesos.
  - Responde la necesidad de mitigar los problemas que se presentan continuamente al momento de contratar empresas desarrolladoras de software, **por la progresiva elevación de costos y desfase de las fechas de entrega**.
  - Establece un conjunto de prácticas o procesos clave agrupados en Áreas Clave de Proceso (KPA - Key Process Area).
  - Para cada área de proceso define un conjunto de buenas prácticas que habrán de ser:
    - Definidas en un procedimiento documentado.
    - Provistas (la organización) de los medios y formación necesarios.
    - Ejecutadas de un modo sistemático, universal y uniforme (institucionalizadas).

- Medidas.
- Verificadas.

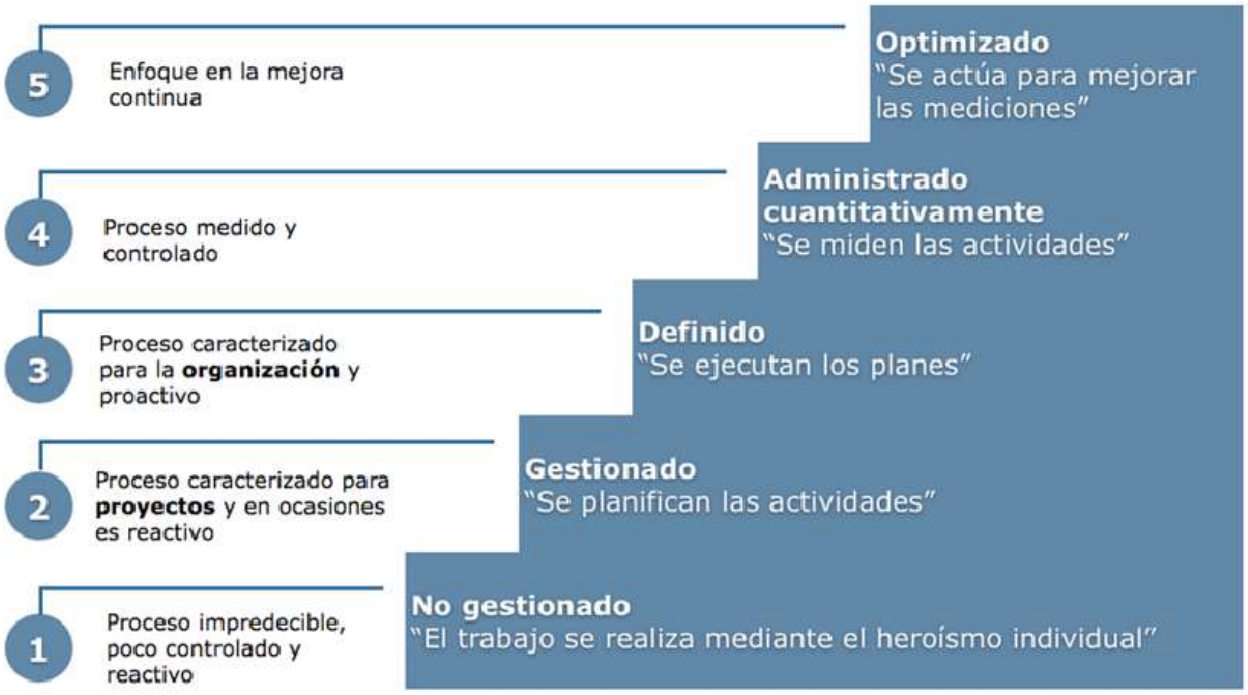
- **Niveles de madurez**

El modelo CMM establece una medida del progreso conforme avanza, en niveles de madurez. Cada nivel a su vez cuenta con un número de áreas de proceso que deben lograrse. El alcanzar estas áreas se detecta mediante la satisfacción o insatisfacción de varias metas claras y cuantificables.

Los niveles son:

Inicial o Nivel 1	•Este es el nivel en donde están todas las empresas que no tienen procesos. Los presupuestos se disparan, no es posible entregar el proyecto en fechas, te tienes que quedar durante noches y fines de semana para terminar un proyecto. No hay control sobre el estado del proyecto, **el desarrollo del proyecto es completamente opaco**, no sabes lo que pasa en él.
Repetible - Gestionado o Nivel 2	•Quiere decir que el éxito de los resultados obtenidos se pueden repetir. La principal diferencia entre este nivel y el anterior es que **el proyecto es gestionado y controlado durante el desarrollo** del mismo. El desarrollo no es opaco y se puede saber el estado del proyecto en todo momento.
Definido o Nivel 3	•Alcanzar este nivel significa que la **forma de desarrollar proyectos (gestión e ingeniería) esta definida**, pero definida quiere decir que esta establecida, documentada y que existen métricas (obtención de datos objetivos) para la consecución de objetivos concretos.
Cuantitativamente Gestionado o Nivel 4	•Los proyectos usan objetivos medibles para alcanzar las necesidades de los clientes y la organización. Se usan métricas para gestionar la organización.
Optimizado o Nivel 5	•Los procesos de los proyectos y de la organización están orientados a la mejora de las actividades. Mejoras incrementales e innovadoras de los procesos que mediante métricas son identificadas, evaluadas y puestas en práctica.

Autoría propia



Tomado de https://einsupiicsa.wordpress.com/2011/12/06/cmmi-sus-niveles-y-su-importancia/

4.4.5 SCAMPI (STANDARD CMMI APPRAISAL METHOD FOR PROCESS IMPROVEMENT)

Es un método desarrollado por el SEI con el fin de proporcionar clasificaciones de referencia de calidad en los procesos de software de una organización, también se define como conjunto de criterios de alto nivel para desarrollar, definir y usar métodos de evaluación basados en CMMI.

- **Finalidad**

Los resultados de un SCAMPI permiten a la organización:

- ✓ **Conocer la situación actual de sus procesos**,
- ✓ **Establecer prioridades**,
- ✓ **Enfocar** las actividades de mejora,
- ✓ **Reforzar áreas de oportunidad**, así como
- ✓ **Tener las bases** sobre las cuales **establecer el siguiente ciclo de mejora**.

Estas evaluaciones son **hechas por un Asesor Líder acreditado por el SEI.**

- **Un SCAMPI C :Menor duración y alcance**, se ve el uso de los procesos en la organización y las iniciativas de mejora con relación al modelo CMMI. Permiten identificar una tendencia en el uso del proceso.

- Un **SCAMPI B** :**Mayor duración que un C** ,permite identificar la implementación del proceso en la organización con una muestra más amplia de información.

  A-B:No da un nivel de madurez.

- Un **SCAMPI A** :**Mayor duración**, permite ver la institucionalización de los procesos en la organización. Es mas riguroso en cuanto a la muestra de proyectos a observar y da un nivel de madurez a la organización.

  - **¿Quién realiza la evaluación?**

La evaluación la realizan **calificadores autorizados por el SEI**, también existen algunas **evaluaciones informales de análisis de brecha** basados en este método que hacen consultores o un equipo de la misma organización, en donde también pueden participar **personas externas** si así lo desean, y **son orientados por el Asesor Líder** quién **define las guías**, que se cumplan **lineamentos del método**, y hace que **el equipo trabaje**.

Proceso de Evaluación:

EVALUACIÓN	DESCRIPCIÓN
Analizar:	Las evaluaciones estabilizan el proceso y priorizan el cambio.
Motivar:	Producen cambio involucrando y motivando a las organizaciones en esfuerzos de auto análisis.
Transformar:	Suavizando una cultura de censura, permitiendo al personal la libertad de pensar acerca de que se hace de forma equivocada y cómo corregirlo.
Educar:	Las evaluaciones educan proporcionando a las personas en las organizacionales un amplio conocimiento de su propia compañía y estimulando a las organizaciones a contemplar las mejores prácticas de la industria y compararlas con la organización.

Proceso

- **Elegir** una metodología de evaluación y un modelo de referencia.
- **Seleccionar** un evaluador líder.
- **Establecer** los objetivos de negocio y alcance de la evaluación.
- Asegurar que la organización comparte un entendimiento acerca de lo que se desea lograr con la evaluación.

- **Principio**
- **Comenzar** con un modelo de referencia de evaluación.
- **Usar** un proceso de evaluación formalizado.
- **Involucrar** a la alta gerencia como patrocinador de la evaluación.
- **Abordar** la evaluación de forma colaborativa.

- **Evidencia**

Información basada en **observación**, **medición** o **prueba** y que **es verificable**. Esta evidencia Incluye:

- **Instrumentos**: Información escrita relacionada con la implementación en la unidad organizacional.
- **Presentaciones**: Información que incluye normalmente resúmenes de visión general y demostraciones de herramientas o capacidades.
- **Documentos**: Artefactos que reflejen la implementación de una o más prácticas del modelo: políticas de organización, procedimientos y artefactos a nivel de implementación.
- **Entrevistas**: Interacción cara a cara con aquellos que implementan o usan los procesos dentro de la unidad organizacional.

- **Puntuación**
  - Juicio.
  - Evidencias de los datos.
  - Grado de definición e implantación.
  - Única puntuación.

- **Escala de Puntuación**
  - **Fully Implemented (FI)**: Enfoque completo y consistente en cuanto a implantación de la practica en toda la organización sin existir ninguna debilidad significativa
  - **Largely Implemented (LI):** Enfoque adecuado en la definición de la práctica a nivel de proceso y su implantación en los proyectos es ordenada, aunque existen variaciones. Hay evidencia de algunas debilidades no significativas.
  - **Partially Implemented (PI):** Enfoque adecuado en la definición de la práctica a nivel de proceso pero su implantación no es evidenciada en los proyectos, puede no ser consistente ni predecible.

- **Not Implemented (NI):** Poca o ninguna evidencia de que la práctica está implementada.

- **Metas**

- **Satisfecho**: Las prácticas organizativas asociadas al componente CMMI están implantadas e institucionalizadas de acuerdo al estándar o mediante una alternativa adecuada.

- **Parcialmente Satisfecho:** Hay inconsistencias o cobertura parcial en la implantación e institucionalización de las prácticas asociadas al componente CMMI.

- **No Satisfecho:** Hay debilidades significativas en la implantación e institucionalización de las prácticas asociadas al CMMI y no existe una alternativa adecuada.

- **No aplicable:** Las prácticas asociadas al componente CMMI no son aplicables en el contexto de la organización.

- **No puntuado:** Los hallazgos de la evaluación no cumplen los criterios de cobertura o el elemento CMMI está fuera del alcance de la evaluación.

4.4.6 ITIL

Están**dar mundial de la Gestión de Servicios Informáticos**, es una guía que proporciona una forma eficiente de gestión de servicios TI (Tecnología de la Información).

- **Sirve para:**

  - Ayudar a establecer el control, operación y gestión de los recursos (ya sean propios o de los clientes).

  - Permite hacer una revisión y reestructuración de los procesos existentes en caso de que ser necesario.

  - Establecer una filosofía de mejora continua

  - Genera una cultura hacia la calidad y permite la generación de documentación pertinente, permitiendo que toda la organización esté al tanto de los cambios, se organiza, sistematiza y se generan importantes registros para la toma de decisiones.

  - Permite un adecuado seguimiento del entorno que se maneja en la organización a nivel de TI (Tecnología de la Información).

- **Objetivo**

  - Mejorar el profesionalismo en una organización.

  - Reducir costos.

- Mejorar la disponibilidad de los servicios IT.
- Incrementar los resultados.
- Optimizar los recursos.
- Mejorar la escalabilidad.
- Alinear los servicios de IT con los requerimientos del negocio.
- Proveer un servicio óptimo a un costo razonable.

- **Implementación**

S	**Services (IT´s)** Principio fundamental
P	**Processes & procedures** Independencia de la tecnologia
O	**Organization** Cambio de enfoque
T	**Tools & technology** Soporte a los procesos

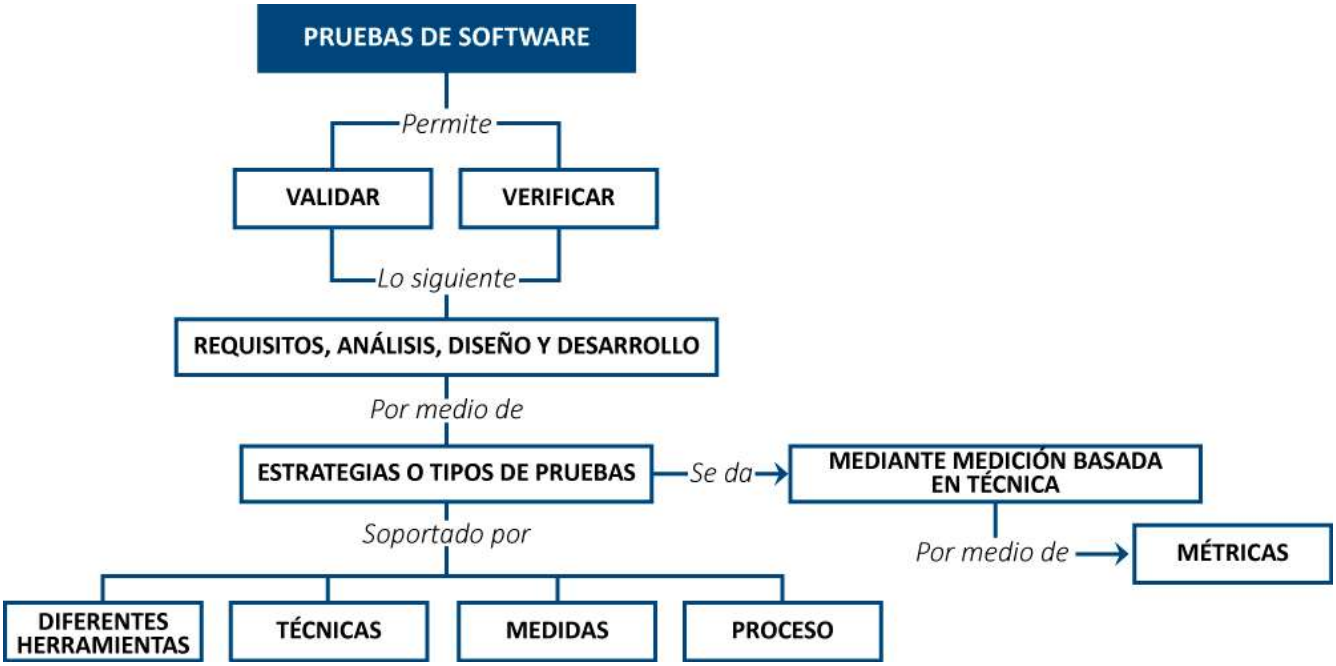
4.4.7 EJERCICIO DE ENTRENAMIENTO

Apreciado estudiante, el siguiente ejercicio le servirá para evidenciar los conceptos aprendidos de la Unidad, trabájela con gran responsabilidad, si es del caso revise conceptos anteriores, ejemplos y ejercicios que te ayuden en la solución del mismo, recuerde que son conceptos básicos y de gran aplicabilidad en los estándares de calidad softwares, por lo tanto es importante que los tenga siempre presentes.

- Elabore para el proyecto que este implementado, un plan de estándar de calidad teniendo presente los modelos y PMBOK.

5 UNIDAD 4 PRUEBAS Y MÉTRICAS DE SOFTWARE

5.1.1 RELACIÓN DE CONCEPTOS



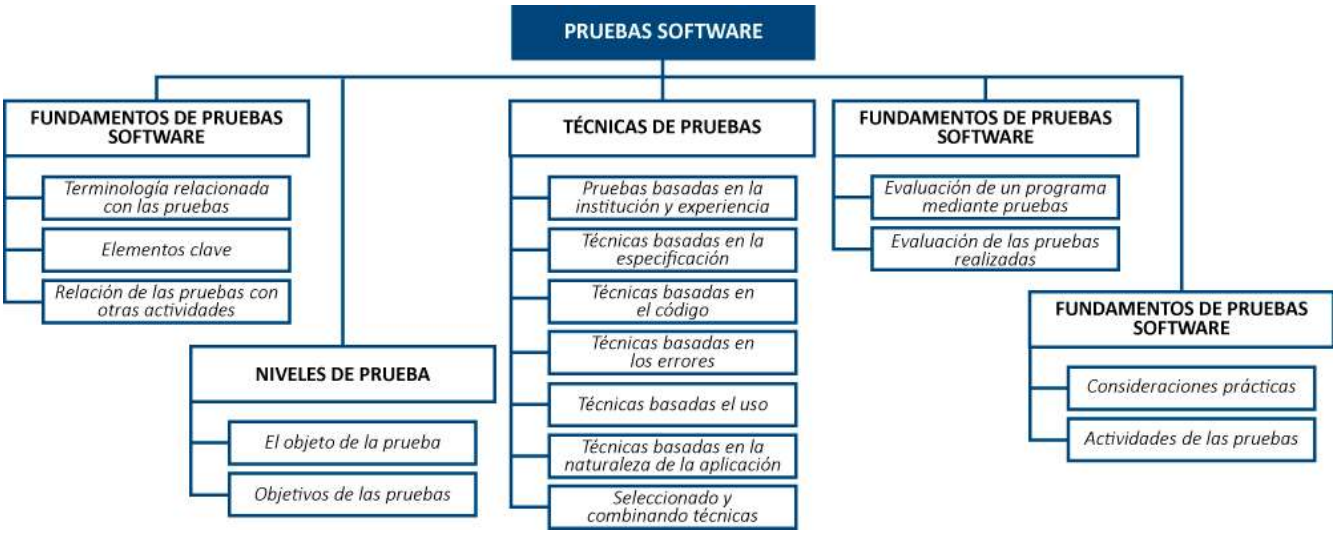
5.2 TEMA 1 INTRODUCCIÓN A LAS PRUEBAS DE SOFTWARE

Las pruebas son fundamentales para el proceso de desarrollo y son técnicas de comprobación dinámica donde implican la ejecución del programa (codificación), estas Permiten: evaluar la calidad de un producto y mejorarlo identificando error, defectos y fallos.

Pruebas de Software (Testing de Software) - Entrevista a Javier Marchese Enlace

[1/4] Proceso de Pruebas: Pieza clave en la Calidad del Software Enlace

Factores y Metricas de Calidad de Software Enlace



Tomado de: Libro Swebok 2 edición

Niveles de pruebas				
Test	**Objetivo**	**Participantes**	**Ambiente**	**Método**
Unitario	Detectar errores en los datos, lógica, algoritmos	Programadores	Desarrollo	Caja Blanca
Integración	Detectar errores de interfaces y relaciones entre componentes	Programadores	Desarrollo	Caja Blanca, Top Down, Bottom Up
Funcional	Detectar errores en la implementación de requerimientos	Testers, Analistas	Desarrollo	Funcional
Sistema	Detectar fallas en el cubrimiento de los requerimientos	Testers, Analistas	Desarrollo	Funcional
Aceptación	Detectar fallas en la implementación del sistema	Testers, Analistas, Cliente	Productivo	Funcional

Tomado de http://www.it-mentor.com.ar/pdf/propuestaCursoPruebasUsandoIC.pdf

5.3 TEMA 2 ERROR, FALLO Y DEFECTO

El objetivo de las pruebas es descubrir errores en el software; Un error es una equivocación cometida por un desarrollador, el fallo es la incapacidad de un sistema o de alguno de sus componentes para realizar las funciones requeridas por el usuario y un defecto es el resultado de un fallo o deficiencia durante el proceso de codificación.

- **Pruebas** *(**test**):*

Es una actividad en la cual un sistema o uno de sus componentes se ejecutan para verificar el funcionamiento de un proceso, los resultados se observan y registran para realizar una evolución de dicho proceso.

Referente a la programación una prueba **de software**, en inglés *testing* son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación.

- **Caso de prueba** *(**test case**):*

Un conjunto de entradas, condiciones de ejecución y resultados esperados desarrollados para un objetivo particular, un caso de prueba es utilizado por el analista para determinar si el requisito de una aplicación es parcial o completamente satisfactorio.

- **Defecto** *(**defect, fault, bug**):*

Un defecto de software, es el resultado de un fallo o deficiencia durante el proceso de creación de programas de ordenador o computadora u otro dispositivo. Por ejemplo, un proceso, una definición de datos o un paso de procesamiento incorrectos en un programa.

- **Error (error)**:

Es una equivocación cometida por un desarrollador. Algunos ejemplos de errores son: un error de tipeo, una malinterpretación de un requerimiento o de la funcionalidad de un método, una acción humana que conduce a un resultado incorrecto. Por ejemplo: Divisiones entre cero. Es una tipo de manifestación del defecto en el sistema que se ejecuta.

- **Fallo *(failure):***

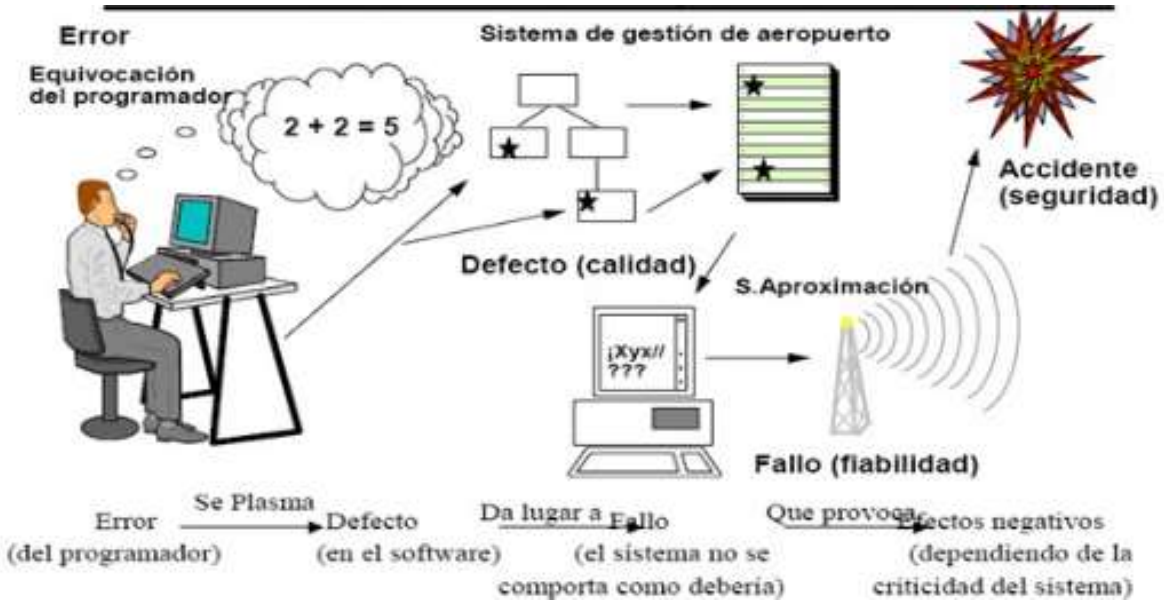
Puede presentarse en cualquiera de las etapas del ciclo de vida del software aunque los más evidentes se dan en la etapa de desarrollo y programación. Es la incapacidad de un sistema o de alguno de sus componentes para realizar las funciones requeridas dentro de los requisitos de rendimiento especificados.

- **Verificación:**

La verificación del software es el proceso a través del cual se analiza y revisa que el software satisfaga los objetivos propuestos al inicio del desarrollo.

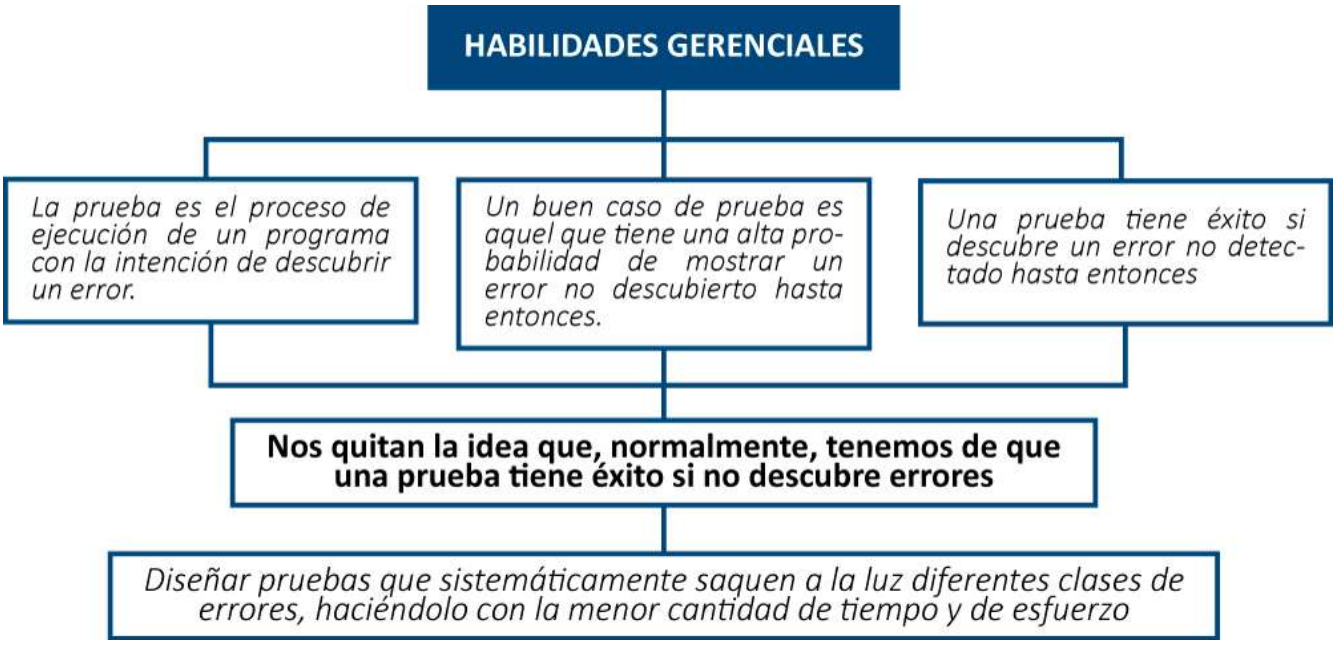
- **Validación:**

El proceso de evaluación de un sistema o de uno de sus componentes durante o al final del proceso de desarrollo para determinar si satisface los requisitos marcados por el usuario.



Tomado de http://plaza-entretenimiento.blogspot.es/img/dos.jpg

Un error puede conducir a uno o más defectos. Un defecto se encuentra en un artefacto y puede definirse como una diferencia entre la versión correcta del artefacto y una versión incorrecta. Un defecto es haber utilizado el operador "<" en vez de "<=". En este caso una falla es la discrepancia visible que se produce al ejecutar un programa con un defecto, respecto a la ejecución del programa correcto. Es decir, una falla es el síntoma de un defecto. Por ejemplo: una consulta que no arroje ningún resultado.



Autoría propia

5.4 TEMA 3 TIPO DE PRUEBA

Los tipos de pruebas principalmente garantizan el cumplimiento de las especificaciones de los requisitos planteadas desde un inicio por el analista o el cliente y ayuda encontrar posibles errores de codificación.

Objetivo: Descubrir errores en el software.

- Es necesario crear buenos casos de prueba (aquél que tiene una alta probabilidad de mostrar errores aún no descubiertos)
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Pruebas relacionadas con el rendimiento del sistema:

- **Rendimiento** (tiempos de respuesta adecuados)
- **Volumen** (funcionamiento con grandes volúmenes de datos)
- **Sobrecarga** (funcionamiento en el umbral límite de los recursos)
- **Disponibilidad de datos** (cuando se produce una recuperación ante fallos)
- **Facilidad de uso** (usabilidad)
- **Operación e instalación** (operaciones de rearranque, actualización de software)

- **Entorno** (interacciones con otros sistemas) y comunicaciones
- **Seguridad** (control de acceso e intrusiones, ej. Inyección código SQL)

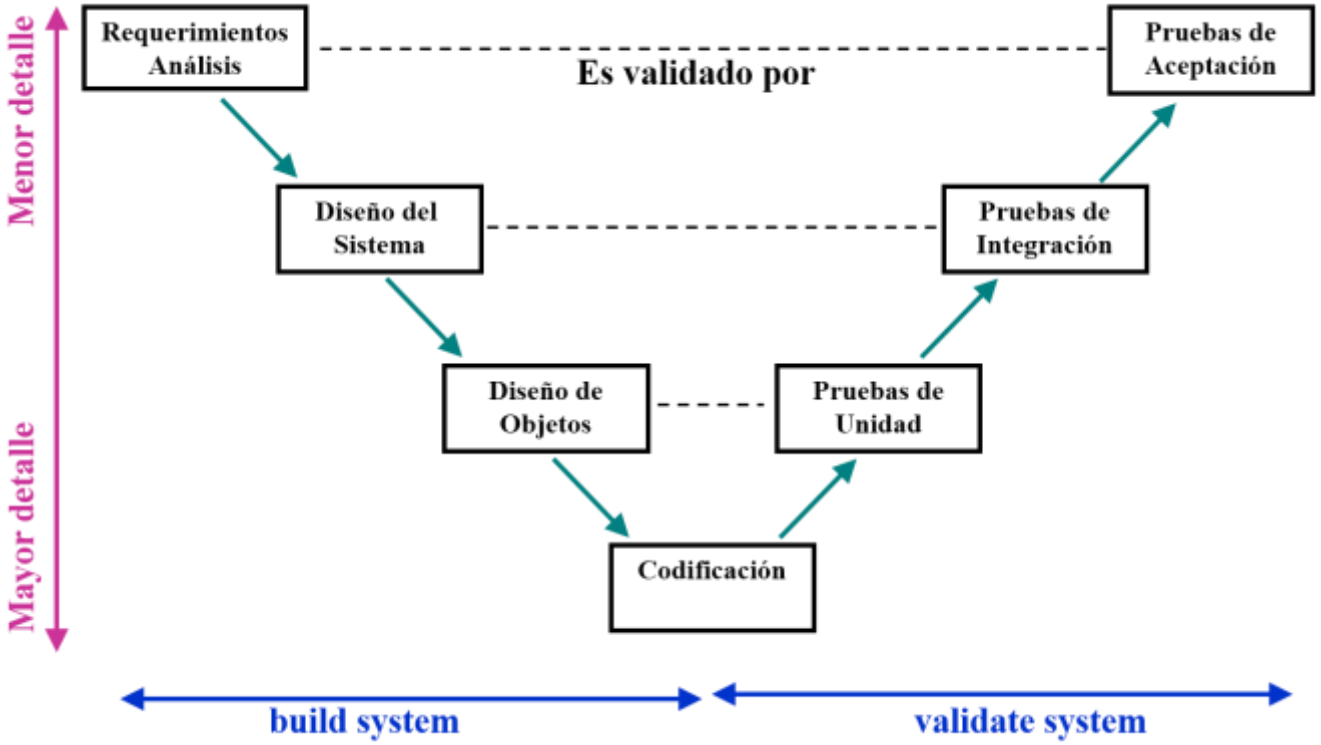
Principio de Prueba

PRINCIPIOS DE LAS PRUEBAS	A todas las pruebas se les debería poder hacer un seguimiento hasta cumplir con los requisitos del cliente.
	Las pruebas deberían planificarse mucho antes de que empiecen.
	El principio de Pareto es aplicable a la preba del software.
	Las pruebas deberían empesar por "lo pequeño" y progresar hacia "lo grande"
	No son posibles las pruebas exhaustivas.
	Para ser más eficaces las pruebas deberían ser realizadas por un equipo independiente.

Autoría propia

Principio de Prueba

- A todas las pruebas se les debería hacer un seguimiento hasta los requisitos del cliente.
- Las pruebas deberían planificarse mucho antes de que empiecen.
- El principio de Pareto es aplicable a la prueba de software. (El 80% de los errores descubiertos con las pruebas surgen de hacerle seguimiento sólo al 20% de todos los módulos del programa). El problema es aislar esos módulos sospechosos y probarlos concienzudamente.
- Las pruebas empiezan por lo pequeño y progresan hasta lo grande.
- No son posibles las pruebas exhaustivas.
- Para ser más efectivas, las pruebas deberían ser conducidas por un equipo independiente.



Tomado de http://u5-modelodeimplementacion.blogspot.com.co/

5.4.1 EJERCICIO DE ENTRENAMIENTO

Apreciado estudiante, la siguiente actividad le servirá para afianzar los conocimientos sobre prueba de Software:

Con base a los tipos de Prueba describa lo siguiente para cada una la definición, un ejemplo, ventaja, desventaja y que software apoya este proceso.

- Prueba Unitaria.
- Pruebas de desempeño.
- Prueba de Integración.
- Pruebas de Carga.
- Prueba de Regresión.
- Prueba de Caja negra.
- Prueba de Caja blanca.

- Prueba de Humo.
- Prueba del Sistema.
- Prueba de Stress.
- Prueba de Volumen.
- Prueba de Recuperación.
- Prueba de Seguridad.
- Prueba de Aceptación.
- Prueba de Instalación.
- Prueba de Alfa.
- Prueba de Beta.

5.5 TEMA 4 MÉTRICAS DE SOFTWARE

Las métricas permiten generar medición y se refieren a un amplio elenco de medidas para la codificación de Software donde despliega una serie de información relevante para el proceso, con base a esto las métricas implican medir: medir involucra números; el uso de números para hacer cosas mejores en la codificación además pretende mejorar las técnicas, para estimar, predecir costos, medir la productividad y la calidad del producto.

Conceptos de Métricas

Está asociada con las palabras **medición** y **medida**, aunque estas tres son distintas.

"La medición es el proceso por el cual los números o símbolos son asignados a **atributos** o **entidades** en el mundo real tal como son descritos de acuerdo a reglas claramente definidas" (Fenton).

La IEEE define **métrica** como:

"una **medida cuantitativa** del grado en que un sistema, componente o proceso posee un atributo dado"

En las métricas hay que tener en cuenta que la mayor parte de la documentación se basa en métricas aplicadas a desarrollos realizados por codificación manual, afortunadamente hoy en día se está trabajando métricas para desarrollo de aplicaciones orientadas a objetos.

Es importante aclarar que esta guía plantea métricas solo para la evaluación de productos finales y realizados desde la visión del cliente. Esto lleva a plantear unas métricas sencillas pero significativas a la hora de aplicarlas en el proceso de evaluación.

Proceso para la métrica con el fin de evaluar un software (guia_tecnica_para_evaluacion_de_software.pdf)

El proceso de evaluación de software se inicia con una visión cualitativa y deriva en una evaluación cuantitativa, siendo todo el proceso documentado y cumpliendo los siguientes pasos:

- **Estado del Software**

Conocimiento del estado del software, estableciendo si se trata de un desarrollo sin terminar o un producto terminado para la entrega al cliente.

- **Identificar el tipo de software**

Especificar el tipo de software a evaluar, si es un sistema operativo, software de seguridad, software de ofimática, lenguaje de programación, base de datos, aplicativo a la medida, entre otros.

- **Perfiles de Evaluadores**

Teniendo como marco conceptual al estándar ISO [ISO/IEC9126], se consideran tres perfiles de usuario, a un alto nivel de abstracción para desarrollo de software, usuarios finales, desarrolladores, y gerentes.
El estándar afirma que la relativa importancia de las características de calidad (como usabilidad, funcionalidad, confiabilidad, eficiencia, portabilidad, mantenibilidad y calidad en uso) varía dependiendo del punto de vista considerado y de la crítica de los componentes del software a evaluar.

La visión del usuario final, concierne al interés de los mismos en usar el software, como así también su performance, su eficiencia, su facilidad de uso, entre otros aspectos. Los usuarios finales no están interesados en características internas o de desarrollo del software (sin embargo, atributos internos contribuyen a la calidad de uso).

La visión de calidad del desarrollador debe considerar no sólo los requerimientos del software para la visión del usuario sino también la calidad para los desarrollos intermedios resultantes de las actividades de la fase de desarrollo. Se debe tener en cuenta que los desarrolladores están preocupados en características de calidad del software como mantenibilidad y portabilidad.

La visión de calidad del gerente es una visión integradora, que incorporar requerimientos de negocio a las características individuales.

Ejemplo, un gerente está interesado en el equilibrio entre la mejora del software y los costos y tiempos establecidos

- **Especificar los Objetivos**

Conocer los objetivos tanto generales como específicos del software

- **Aplicar el modelo de calidad**

Elaborar un instrumento o formato donde aplique el modelo de calidad externo e interno y calidad de uso.
Si existe un comité o conjunto de personas encargadas de la evaluación, el instrumento debe ser aprobado por los participantes.

- **Criterios de la evaluación**

Los criterios para evaluar el software se dividen en dos grandes bloques: uno dedicado a criterios que son aplicables a cualquier tipo de software (criterios generales), y otro conjunto compuesto por criterios adaptables al grupo de software evaluados (criterios específicos). En este caso se definen los criterios de la evaluación según el tipo de software, para el cual debe conformar un equipo evaluador, este ejercicio ayuda a definir que opciones se deben evaluar con más detalle y valor.

Seleccionar métricas

La selección de métricas se obtiene a partir de los indicadores especificados en el modelo.

Niveles o escalas

• A cada métrica seleccionada le asigna un puntaje máximo de referencia.
• La suma de los puntajes máximos de todas las métricas debe ser igual o aproximado a 100 puntos.
• El personal que participa en la evaluación debe establecer niveles de calificación cualitativa con base a los puntajes, por ejemplo:

- ✓ De 0 a 1 Inaceptable.
- ✓ De 2 a 3 mínimo aceptable
- ✓ Más de 3 Aceptable o satisfactorio

Otro ejemplo de calificación cualitativa puede ser:

- ✓ Deficiente
- ✓ Insuficiente
- ✓ Aceptable
- ✓ Sobresaliente
- ✓ Excelente

• Se permite usar números enteros o hasta con un decimal de aproximación.

• Definir por cada métrica, un puntaje mínimo de aprobación, y al final de la evaluación, dependiendo del puntaje si es mayor o menor a lo propuesto, considerar si el software cumple o no cumple con los objetivos propuestos.

Establecer criterios

Las persona que participa en el proceso de evaluación debe tener criterios con aspecto al indicador que se está analizando, Es importante tener en cuenta que el criterio debe ajustar al tipo de software que se va a evaluar.

Tomar medidas

Para la medición, las métricas seleccionadas se aplican al software. Los resultados son valores expresados en las escalas de las métricas, definidos previamente.

Resultados

El proceso de evaluación genera un cuadro de resultados por cada uno de los principales indicadores y el total final de resultado.

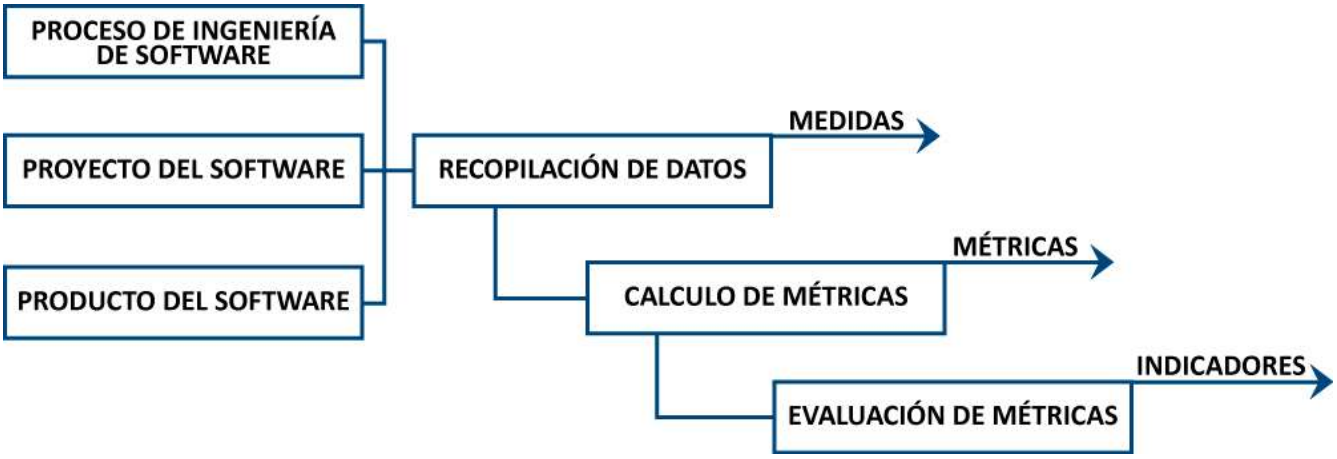
Documentación

El proceso de evaluación se documenta, indicando la fecha, empresa, los cargos, nombres y apellidos, dependencia de las personas que participan en el proceso de evaluación, especificando las etapas en las que participaron.

Seguimiento

Si el resultado de la evaluación tiene observaciones o indicadores de calidad bajos, y el personal que lo evalúa permite realizar la corrección, se programa otra evaluación donde se verifique que el proceso mejora, el tiempo que se estime debe influir en los criterios de la aproxima evaluación

Proceso de recopilación de métricas



Tomado de http://slideplayer.es/slide/3929857

Las métricas pueden ser:

Directas	Indirectas	Indicadores
• LCF: líneas de código fuente escritas. • HPD: horas-programador diarias. • CHP: coste por hora-programador, en unidades monetarias.	• HPT: horas-programador totales. • LCFH: líneas de código fuente por hora de programador. • CTP: coste total actual del proyecto, en unidades monetarias. • CLCF: coste por línea de código fuente.	• PROD: productividad de los programadores.

Tomado de http://slideplayer.es/slide/3929857

Se obtiene:

Directas	Indirectas	Indicadores
• LCF = Contar las líneas de código. • HPD = Contar cada día las horas dedicadas por los programadores al proyecto. • CHP = Consultar el plan de proyecto.	• HPT = Σ HPD • LCFH = LCF/HPT • CTP = CHP*HPT • CLCF = LCF/CTP	• PROD: Establecer criterios o rangos de valores.

Tomado de http://slideplayer.es/slide/3929857

5.5.1 EJERCICIO DE ENTRENAMIENTO

Apreciado estudiante, la siguiente actividad le servirá para afianzar los conocimientos en las métricas de Software: es importante que responda lo siguiente:

1. ¿Cuál es la finalidad de las pruebas de software?
2. ¿Cuál es la importancia de las pruebas en un desarrollo?
3. ¿Qué ventajas y desventajas general la elaboración de pruebas?
4. ¿Cuál es la finalidad de implementar pruebas en un proyecto de software?
5. Elaborar un mapa mental sobre las pruebas.

5.5.2 EJERCICIO DE ENTRENAMIENTO

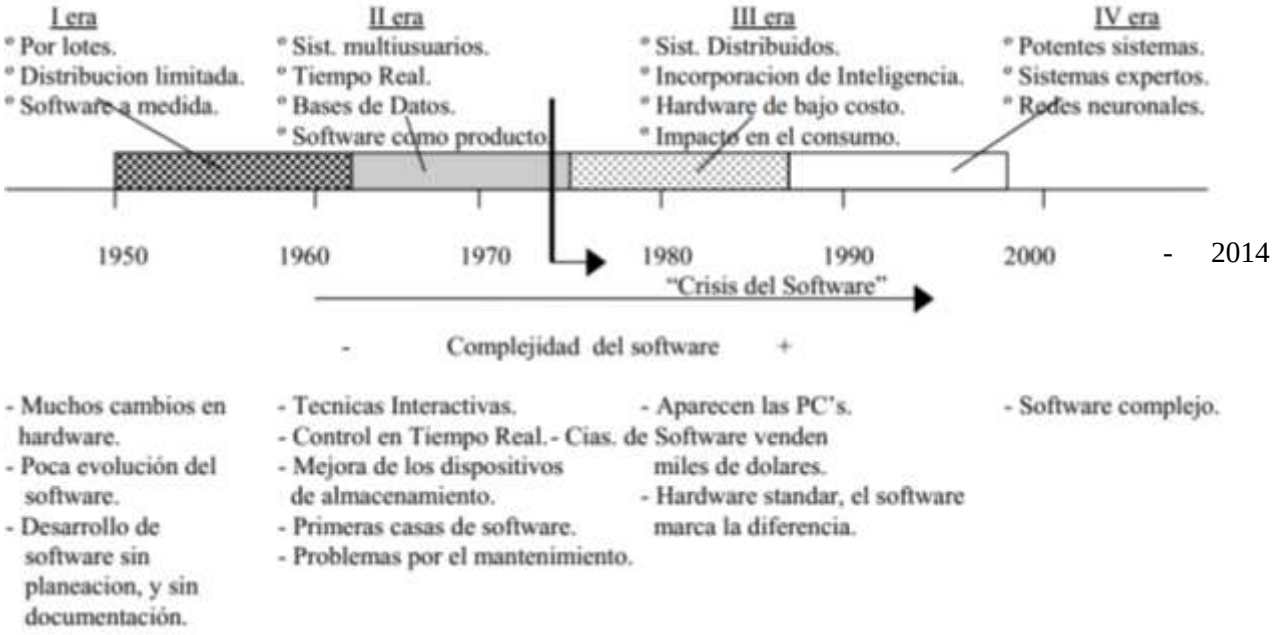
Teniendo como modelo el ejercicio de aprendizaje realice la siguiente actividad:

- Elabore a su proyecto de software:
  - ✓ Un plan de modelo de calidad (CMMI, SPICE, SQUARE etc) que se vea reflejado.
  - ✓ Un plan de pruebas y métricas con respecto a lo aprendido.

5.5.3 EJERCICIO DE ENTRENAMIENTO

Apreciado estudiante, la siguiente prueba bien realizada le como repaso y afianzamiento de toda la unidad vista, trabájela con gran responsabilidad, si es del caso revise conceptos anteriores, ejemplos y ejercicios que te ayuden en la solución del mismo, recuerde que son conceptos básicos y de gran aplicabilidad para la Ingeniería de Software, por lo tanto es importante que los tenga siempre presentes.

1. **La evolución del Software se observa a continuación.**



En las pasadas décadas los ejecutivos y desarrolladores se hacían las siguientes preguntas, a hoy que está pasando con las empresas:

¿Cuál es tu respuesta a estas preguntas? Responda con base a lo que ha leído y ha aprendido:

a) ¿Por qué lleva tanto tiempo terminar los desarrollos?
b) ¿Por qué es tan elevado el costo?
c) ¿Por qué no podemos detectar los errores antes de entregar el software a los clientes?
d) ¿Por qué resulta tan difícil comprobar el progreso del desarrollo del software?
e) ¿Cuál es el mayor impacto que ha generado en esta época desarrollar software?
f) ¿Cómo influye la Calidad de Software en el tiempo desde su inicio a hoy?

2. **Escriba lo siguiente, se entiende por:**

- Error.
- Fallo.
- Defecto.

Además de dos ejemplos de cada uno donde se puede evidenciar el error, el fallo y el defecto en un proyecto de Software.

3. Lea con atención.

2.12 PRODUCTO Y PROCESO

Si el proceso es débil, el producto final va a sufrir indudablemente. Aunque una dependencia obsesiva en el proceso también es peligrosa. En un breve ensayo, Margaret Davis [DAV95] comenta la dualidad producto y proceso:

Cada diez años o cinco aproximadamente, la comunidad del software vuelve a definir «el problema» cambiando el foco de los aspectos de producto a los aspectos de proceso. Por consiguiente, se han abarcado lenguajes de programación estructurados (producto) seguidos por métodos de análisis estructurados (proceso) seguidos a su vez por encapsulación de datos (producto) y después por el énfasis actual en el Modelo Madurez de Capacidad de Desarrollo del software del Instituto de ingeniería de software (proceso).

Mientras que la tendencia natural de un péndulo es parar en medio de dos extremos, el enfoque de la comunidad del software cambia constantemente porque se aplica una fuerza nueva cuando falla el último movimiento. Estos movimientos son perjudiciales por sí mismos y en sí mismos porque confunden al desarrollador del software medio cambiando radicalmente lo que significa realizar el trabajo, que por sí mismo lo realiza bien. Los cambios tampoco resuelven «el problema», porque están condenados al fracaso siempre que el producto y el proceso se traten como si formasen una dicotomía en lugar de una dualidad.

Cita:
[Si se desarrolla sin pensar y se aplica descuidadamente, el proceso puede convertirse] en la muerte del sentido común.
Philip K. Howard

En la comunidad científica hay un precedente que se adelanta a las nociones de dualidad cuando contradicciones en observaciones no se pueden explicar completamente con una teoría competitiva u otra. La naturaleza dual de la luz, que parece ser una partícula y una onda al mismo tiempo, se ha aceptado desde los años veinte cuando Louis de Broglie lo propuso. Creo que las observaciones que se hacen sobre los mecanismos del software y su desarrollo demuestran una dualidad fundamental entre producto y proceso. Nunca se puede comprender el mecanismo completo, su contexto, uso, significado y valor si se observa sólo como un proceso o sólo como un producto…

Toda actividad humana puede ser un proceso, pero cada uno de nosotros obtiene el sentido de autoestima ante esas actividades que producen una representación o ejemplo que más de una persona puede utilizar o apreciar, una u otra vez, o en algún otro contexto no tenido en cuenta. Es decir, los sentimientos de satisfacción se obtienen por volver a utilizar nuestros productos por nosotros mismos o por otros.

Así pues, mientras que la asimilación rápida de los objetivos de reutilización en el desarrollo del software aumenta potencialmente la satisfacción que los desarrolladores obtienen de su trabajo, esto incrementa la urgencia de aceptar la dualidad producto y proceso. Pensar en un mecanismo reutilizable como el único producto o el único proceso, bien oscurece el contexto y la forma de utilizarlo, o bien el hecho de que cada utilización elabora el producto que a su vez se utilizará como entrada en alguna otra actividad de desarrollo del software. Elegir un método sobre otro reduce enormemente las oportunidades de reutilización y de aquí que se pierda la oportunidad de aumentar la satisfacción ante el trabajo.

Cita:
Cualquier actividad se vuelve creativa si el autor se preocupa de hacerlo bien o de hacerlo mejor.
John Updike

Las personas obtienen tanta satisfacción (o más) del proceso creativo que del producto final. Un artista disfruta con las pinceladas de la misma forma que disfruta del resultado enmarcado. Un escritor disfruta con la búsqueda de la metáfora adecuada al igual que con el libro final. Un profesional creativo del software debería también obtener tanta satisfacción de la programación como del producto final.

El trabajo de los profesionales del software cambiará en los próximos años. La dualidad de producto y proceso es un elemento importante para mantener ocupada a la gente creativa hasta que se finalice la transición de la programación a la ingeniería del software.

1. Genere una crítica constructiva de este apartado y justifica tu respuesta.

2. ¿Qué aplicabilidad le daría a la Calidad en un proyecto de Software?

4. La calidad es:

- Satisfacer plenamente las necesidades del cliente.
- Cumplir las expectativas del cliente y algunas más.
- Despertar nuevas necesidades del cliente.
- Lograr productos y servicios con cero defectos.
- Hacer bien las cosas desde la primera vez.
- Diseñar, producir y entregar un producto de satisfacción total.
- Producir un artículo o un servicio de acuerdo a las normas establecidas.
- Dar respuesta inmediata a las solicitudes de los clientes.
- Sonreír a pesar de las adversidades.
- Una categoría tendiente siempre a la excelencia.
- Calidad no es un problema, es una solución.

Con base a la anterior, responda las siguientes preguntas:

a) ¿Quiénes están familiarizados con el concepto de calidad de Software?
b) ¿Cómo una empresa puede calificar el nivel de calidad del Software que producen?
c) ¿Qué creen que es lo deben mejorarla las empresas de Hoy con base a la calidad del Software?
d) ¿Usted Cómo lo haría?
e) ¿Qué propuesta presentaría a la empresa para generar políticas de calidad?

5. Escriba en el cuadro la aplicabilidad en la calidad en el desarrollo de software en cuanto a:

Calidad Interna	Calidad Externa

6 PISTAS DE APRENDIZAJE

Tener en cuenta:

- Que las actividades y los conceptos son la base fundamental para generar una buena partica en el desarrollo de software con calidad, para la mayoría lo más importante es la práctica.
- Que si el análisis de requisitos es la base fundamental para garantizar una calidad en el desarrollo.
- Que las pruebas permiten evidenciar la congruencia de los requisitos con la satisfacción del cliente con el fin de garantizar valor agregado a lo que se solicitó al Inicio de lo pactado.
- Que en el desarrollo software es complejo más no imposible, si se hace bien desde el principio con un buen equipo de trabajo.
- Que toda la información que se pueda levantar durante la etapa de análisis de requisitos es importante definir el límite del proceso con el fin de alcanzar la solución.

Tenga presente:

- La calidad es la evidencia de un producto de software con Q.
- Que el ciclo de vida del desarrollo de un proyecto permite definir proceso para un desarrollo con calidad con el fin de alcanzar los objetivos definidos.
- Que existen muchos desarrollos de software en el medio que no satisfacen las necesidades del cliente.
- Glosario de término.

Traer a memoria:

- La calidad define y evidencia lo que se definió en los requisitos para saber con qué se cuenta durante todas las etapas del proyecto.
- Que la calidad de un desarrollo de software no es algo que se lleva a cabo en pocos días, esto requiere de un proceso y se debe evidenciar en todo el ciclo de vida.

7 GLOSARIO

Análisis
Es la parte más importante del proceso de desarrollo de software cuya finalidad principal realizar un modelo para la comprensión del dominio del problema.

Calidad de Software
Es la congruencia definido en los requisitos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo explícitamente documentados, teniendo presente las características implícitas que se esperan de todo software desarrollado.

Casos de Uso
Es aquello que describe la interacción de los Actores con el sistema para lograr un objetivo.

Ciclo de vida
Especifica los pasos para la definición, implementación, documentación y mantenimiento de un desarrollo de software.

Completitud
Es el grado en que se ha conseguido la totalmente la implementación de las funciones de los requisitos.

Confiabilidad
Es la probabilidad de operación libre de fallas de un programa de computadora en un entorno determinado y durante un tiempo específico.

Consistencia
Es aquello que consiste en el uso de un diseño uniforme de técnicas de documentación a lo largo del proyecto de desarrollo de software.

Corrección
Es el grado en que un programa satisface sus especificaciones y consigue los objetivos pedidos por el cliente.

Defecto
Un fallo que se produce una vez que se ha entregado el software al usuario final.

Diagrama de Casos de Uso
Es el diagrama que muestra la relación entre los actores y los casos de uso dentro de un sistema.

Diagrama de Clases
Es el diagrama que muestra una colección de elementos del modelo tales como las clases, tipos y sus contenidos y relaciones.

Diagrama de Colaboración
Es un diagrama que muestra interacciones entre objetos organizadas alrededor de los objetos y sus vinculaciones. A diferencia de un diagrama de secuencias, un diagrama de colaboraciones muestra las relaciones

entre los objetos. Los diagramas de secuencias y los diagramas de colaboraciones expresan información similar, pero en una forma diferente.

Diagrama de Componentes
Es un diagrama que muestra la organización de los componentes y sus dependencias.

Diagrama de Entidad / Relación
Es una descripción conceptual de las estructuras de datos y sus relaciones.

Diagrama de Estado
Es el diagrama que muestra el estado de la máquina.

Diagrama de actividad
Es una descripción informal del sistema de información, por medio de actividades definidas en los requisitos.

Diagrama de Interacciones
Es un término genérico que se aplica a diversos tipos de diagramas que enfatizan la interacción entre objetos. Incluye: diagrama de colaboraciones, diagrama de secuencias, diagrama de actividades.

Diagrama de Secuencia
Es el diagrama que muestra los objetos que participan en la interacción y la secuencia de mensajes que intercambian.

Diseño
Es la parte del proceso de desarrollo de software cuyo propósito principal es decidir cómo se construirá el sistema. Durante el diseño se toman decisiones estratégicas y tácticas para alcanzar.

Eficiencia
Es la cantidad de recursos de computadoras y de código requeridos por un programa para llevar a cabo sus funciones.

Escenario
Es la descripción de modelo de un Caso de Uso.

Especificación
Es un informe detallado del analista, desarrollador y cliente.

Error
Fallo en un producto que se descubre antes de entregar el SW al usuario final.

Indicador
Una métrica o combinación de métricas que proporcionan una visión profunda del proceso de software.

Métricas
Calcula los resultados por medio de un estándar para la evaluación de producto de software y sus servicios
Es una medida del grado en que un sistema, componente o proceso posee un atributo dado.

Medida
Proporciona una indicación cuantitativa de la cantidad, dimensiones o tamaño de algunos atributos de un producto.

Medición
Acto de determinar una medida.

Modelo dinámica
Es una variación semántica de la generalización en la que un objeto puede cambiar de tipo o rol.

Modelo estática
Es una variación semántica de la generalización en la que un objeto no puede cambiar de tipo, o de rol.

Modelo múltiple
Es una variación semántica de la generalización en la cual un objeto puede pertenecer a más de una clase en forma directa.

Tolerancia de errores
Consiste en el daño que se produce cuando el programa encuentra un error.

Trazabilidad
Consiste en poder seguir el "rastro" de cada uno de los requerimientos a medida que son transformados de un modelo a otro.

UML "Unified Modeling Language"
Es un lenguaje que permite especificar, construir, visualizar y documentar los elementos que componen un sistema de software intensivo.

8 BIBLIOGRAFÍA

Abran, A., Bourque, P., Dupuis, R., & Moore, J. W. (2001). Guide to the software engineering body of knowledge-SWEBOK. IEEE Press.

Anaya, R. (2012). Una visión de la enseñanza de la ingeniería de software como apoyo al mejoramiento de las empresas de software. Revista Universidad EAFIT, 42(141), 60-76.

Bourque, P., & Fairley, R. E. (2014). Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press.

CHIDAMBER, S. y KEMERER, C. (1994). "A Metrics Suite for Object Oriented Design", IEEE Transactions on Software Engineering vol. 20, no. 6, pp. 476-493.

CRUZ-LEMUS, J., GENERO, M., y PIATTINI, M. (2005). "Metrics for Software Conceptual Models", Capítulo 7: Metrics for UML Statechart Diagrams, Imperial College Press, United Kingdom

IEEE Computer Society (2004). SWEBOK - Guide to the Software Engineering Body of Knowledge, 2004. (Capítulos 4 y 5) http://www.swebok.org/

Jacobson,g, ( ) I., Booch, G., and Rumbaugh, J. (2000): El Proceso Unificado de Desarrollo. Addison-Wesley.

Muñoz Periñán, I. L., & Gómez Arenas, L. D. S. (2011). Vista ampliada para Gerencia de Proyectos usando mejores prácticas del PMBok® cuarta edición y CMMI®-SVC V. 1.2 nivel de capacidad o madurez.

Piattini (2007) (Cap 10)

Pressman, R. (2005): Ingeniería del Software: Un Enfoque Práctico. 6º Edición. McGraw Edición. McGraw-Hill.

Pfleeger (2002). (Caps. 7, 8 y 9)

Sommerville, I., & Galipienso, M. I. A. (2005). Ingeniería del software. Pearson Educación.

Rogers, P. (2005). Ingeniería de Software un Enfoque Práctico. Editorial McGraw-Hill, Madrid.

Cibergrafia

- http://swnotes.wordpress.com/2009/11/28/el-metodo-scampi/
- http://www.vereau.org/index.php/2007/07/mi-resumen-del-standar-de-evaluacion-scampi/
- http://www.ub.edu.ar/catedras/ingenieria/ing_software/ubftecwwwdfd/glossary/glosary.htm
- http://slideplayer.es/slide/3929857/