Ingeniería de software 2

Autor: Fredy Alonso León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Ingeniería de software 2 / Fredy Alonso León Socha, / Bogotá D.C.,
Fundación Universitaria del Área Andina. 2017

978-958-5459-12-0

Catalogación en la fuente Fundación Universitaria del Área Andina (Bogotá).

Edición:
Fondo editorial Areandino
Fundación Universitaria del Área Andina
Calle 71 11-14, Bogotá D.C., Colombia
Tel.: (57-1) 7 42 19 64 ext. 1228
E-mail: publicaciones@areandina.edu.co
http://www.areandina.edu.co

Primera edición: noviembre de 2017

Corrección de estilo, diagramación y edición: Dirección Nacional de Operaciones virtuales
Diseño y compilación electrónica: Dirección Nacional de Investigación

Hecho en Colombia
Made in Colombia

Ingeniería de software 2

Autor: Fredy Alonso León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

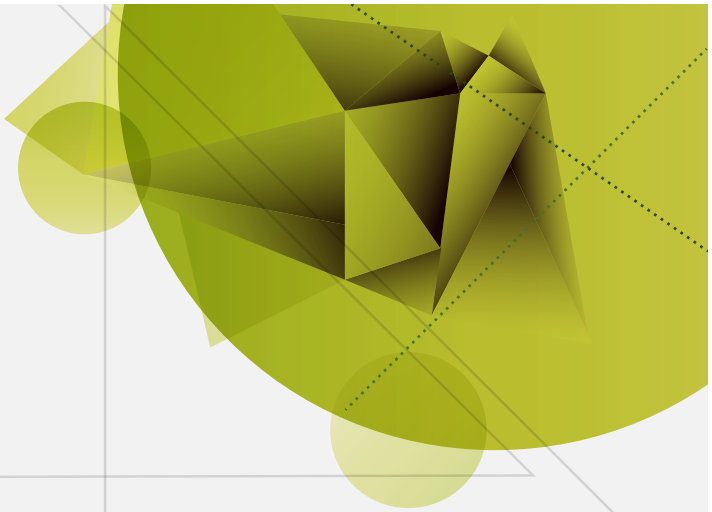
Índice

Índice

1

Unidad 1

Ingeniería de requerimientos



Ingeniería de software II

Autor: Fredy León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

El presente modulo tiene como objetivos realizar una introducción a la ingeniería de requisitos e identificar los tipos de requisitos utilizados en la etapa de requerimientos.

Se inicia con una introducción a la ingeniería de requerimientos, relacionando los conceptos de requisitos, objetivos, actividades básicas que se desarrollan y finalmente se relacionan los diferentes tipos de requisitos.

Estas temáticas le permitirán realizar las actividades que se deben seguir durante el proceso de análisis de requerimientos para un proyecto de software, así como comprender la importancia de la Ingeniería de Requerimientos en la creación de software que cumpla con las expectativas del cliente.

U1 Metodología

El módulo se basa en la lectura individual de la presente cartilla y posterior desarrollo de actividades planteadas en la plataforma online.

Se espera una participación y acceso a plataforma a su consideración, como mejor pueda organizarse, para cubrir los objetivos de aprendizaje y participación. Por mi experiencia, recomiendo la asistencia diaria para aprovechar al máximo el curso. Es interesante seguir los hilos de participación 2-3 veces al día y dedicar uno momento para analizar y participar inteligentemente.

La comunicación conmigo la pueden realizar abiertamente a través del Foro y personalmente a través del email.

U1

Desarrollo temático

Ingeniería de requerimientos

Introducción

¿Qué es un requisito?

Es una funcionalidad que un sistema debe realizar para que cumpla con el objetivo para el cual fue desarrollado.

- Es diferente un requisito a un deseo del cliente.
- La misión del Ingeniero de software es transformar las solicitudes del cliente o usuarios en requerimientos.
- Deben ser verificables mediante análisis, inspección, demostración o pruebas.
- Deben ser necesarios, por lo que omitirlos causarían un mal funcionamiento del sistema.
- Deben ser consistentes en el sentido que no debe contradecir otro requerimiento.

Ejemplo: interfaces graficas, control de errores, estándares utilizados, capacidad de rendimiento, seguridad de los datos, que sea portable, usabilidad, etc.

La siguiente imagen refleja lo que muchas veces ocurre en el desarrollo de cualquier sistema. Lo que el cliente tiene en mente no es lo que recibe como producto final, toda por la falta de comunicación entre los actores del proceso de desarrollo.

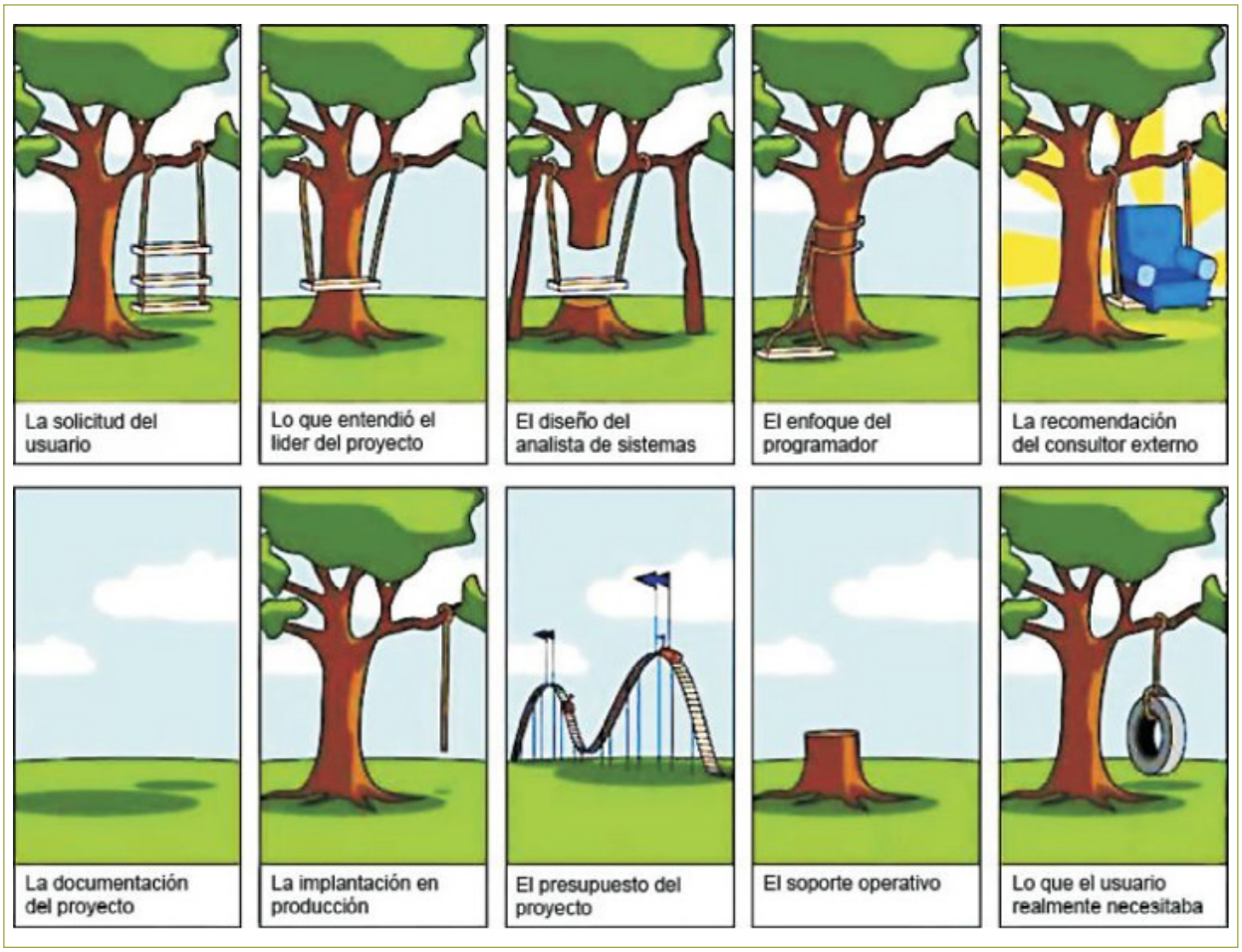


Imagen 1
Fuente: http://2.bp.blogspot.com/_jppV2Yd1MPk/Sw8J4BZYfmI/AAAAAAAABcw/N-oK0uihIFY/s1600/project_sw.jpg

A continuación se muestra una grafica que representa como se encuentran organización los requisitos según (Pohl, 1997).

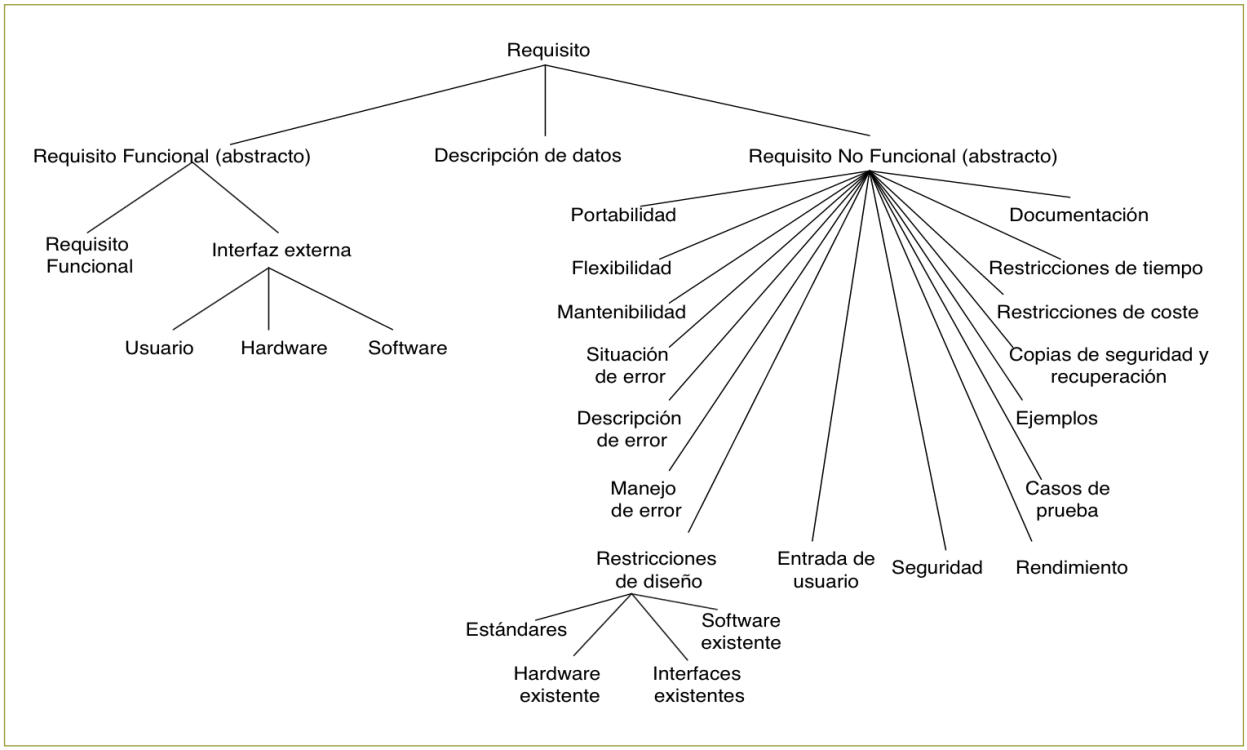


Imagen 2
Fuente: http://ocw.usal.es/ensenanzas-tecnicas/ingenieria-del-software/contenidos/Tema3-IntroduccionalaIR-1pp.pdf

A continuación se relacionaran algunas definiciones planteadas por especialistas en esta rama:

Según (Reifer, 1994), "Es el uso sistemático de procedimientos técnicas, lenguajes y herramientas para obtener con un coste reducido el análisis, documentación, evolución continua de las necesidades del usuario y la especificación del comportamiento externo de un sistema que satisfaga las necesidades del usuario".

Según (Loucopoulus y Karakostas, 1995), "Es un proceso sistemático de desarrollo de requisitos mediante un proceso cooperativo consistente en analizar el problema, documentar las observaciones resultantes en una variedad de formatos de representación, y comprobar la exactitud de la comprensión conseguida".

Objetivos

- Identificar y documentar los que los clientes y usuarios necesitan.
- Crear la documentación necesaria donde se describa el comportamiento externo y restricciones de un sistema, de tal forma que cumpla las necesidades identificadas.

- Analizar y validar la documentación de requisitos para determinar que sean viables, que estén completos y consistentes.
- Realizar la devolución de necesidades.

Actividades básicas

Para completar el proceso adecuadamente es necesario realizar una especificación y administración de cada uno de los requisitos de los clientes o usuarios. Es por esto que para lograr este objetivo se siguen cuatro etapas fundamentales:

Extracción

Se enfoca al descubrimiento de requisitos del sistema.

Análisis

Se analiza el documento levantado en la etapa de extracción. Se realiza una lectura, conceptualización, investigación de los requisitos, se identifican problemas y soluciones. También se programan reuniones con el equipo de trabajo y con el cliente para discutirlos.

Especificación

Se escribe el documento formal de requisitos en forma detallada. Aquí se hace uso de técnicas como UML y otros estándares de documentación.

Validación

Se enfoca en la verificación de los requisitos planteados en el documento final, de tal forma que realmente hagan una representación valida de lo que el sistema debe realizar. Las validaciones son internas porque debe verificarse lo que se hace y externas porque deben ser aprobadas por el cliente o usuario.

Tipos de requisitos

Requisitos de información

Hacen una descripción de los datos que el sistema debe almacenar y procesar para poder cumplir los objetivos para los cuales fue desarrollado.

Ejemplo: Se deberá almacenar información referente a préstamos realizados en el banco, tales como nombre, tipo de préstamo, cantidad, fecha del préstamo, fecha finalización, domicilio del cliente, etc.

Requisitos de interfaz

Definen la forma en que interactuara con otros sistemas.

Ejemplo: El sistema deberá establecer comunicación bluetooth con la impresora portátil para realizar la impresión del recibo. Los datos deben ser enviados en formato XML.

Requisitos funcionales

Representan aquellas tareas especificas que el sistema debe realizar y como deberá ser el comportamiento frente entradas o situaciones particulares.

Ejemplo: El usuario deberá poder buscar los artículos por nombre o código.

Requisitos de funcionalidad

Permiten especificar como deberá ser el comportamiento de un sistema.

La herramienta más utilizada para el levantamiento de este tipo de requisitos son los Casos de Uso. Estos describen simbólicamente como es la interacción de los diferentes actores que intervienen en el sistema y las acciones que realizan sobre el mismo sistema. Todas las acciones deben obtener

un resultado que sea observable y de valor para cada uno de los actores.

Los casos de uso hacen parte del Lenguaje Unificado de Modelado UML. Este tema será tratado más profundamente en la semana 4. La siguiente tabla resume los elementos que componen un diagrama de casos de uso:

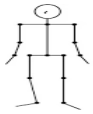
Nombre	Función	Representación grafica
Actor	Role que realiza un usuario dentro de un sistema.	
Casos de Uso	Acción específica que se realiza después de una orden de una entrada externa.	
Relaciones de Uso, Comunicación y Herencia	Asociación. Indica la invocación desde un actor o caso de uso a otra operación (caso de uso).	
	Dependencia o Instanciación. Indica cuando una clase depende de otra, es decir, se instancia (se crea).	
	• Generalización. Es exclusivo para casos de uso. Puede ser de Uso (<<uses>>) o de Herencia (<<extends>>). • extends: Se usa cuando un caso de uso tiene características similares a otro. • uses: Se usa cuando varios caso de uso tienen un conjunto de características similares.	

Tabla 1
Fuente: Propia.

Para comprender un poco más el tema, a continuación encontrara una imagen se muestra un modelo de casos de uso para un sistema de compra de obras de arte.

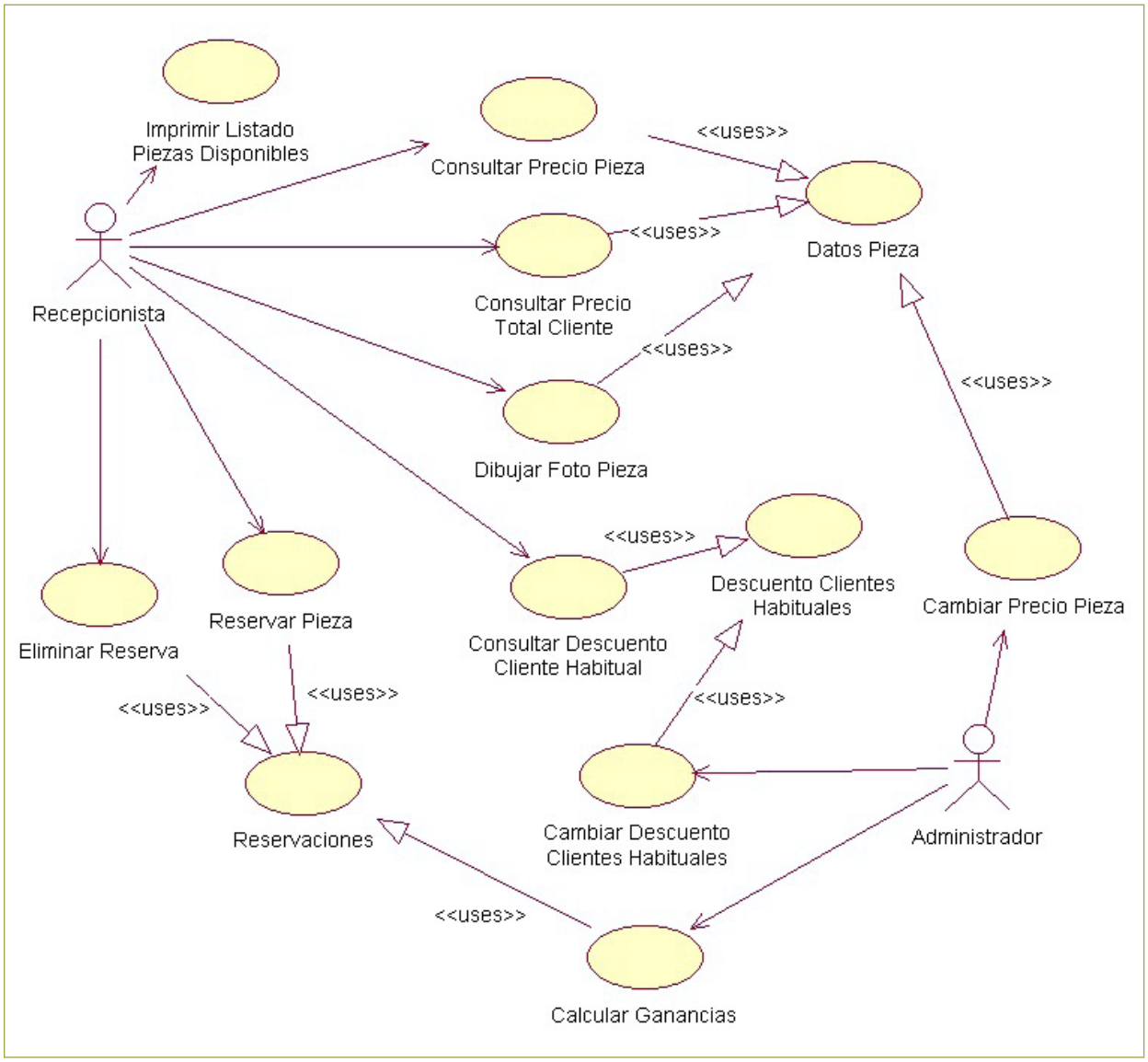


Imagen 3
Fuente: http://www.geocities.ws/vidalreyna/usecase.jpg

Requisitos de datos

Se refieren a los tipos de datos que manejara el sistema con el fin de determinar la forma en que deben ser procesados.

Ejemplo: fechas, enteros, float, string.

Requisitos no funcionales

A continuación se relacionan algunas características que tienen los requisitos no funcionales:

- Se encuentran por fuera de la forma en que debe funcionar el sistema.
- Se asocian con restricciones que tendrá un sistema, ya sean internas o externas. Una restricción es una limitante que hace que el problema se pueda resolver de una forma determinada ej.; que al final de un proceso haya que imprimir automáticamente un documento en una impresora con unas características especificas.
- La tarea de verificación normalmente es difícil de realizar.
- Hacen una definición de las funcionalidades globales que tendrá el sistema.
- Se relacionan con aquellas propiedades adicionales de un sistema.

A continuación se muestra un mapa conceptual referente a requisitos no funcionales:

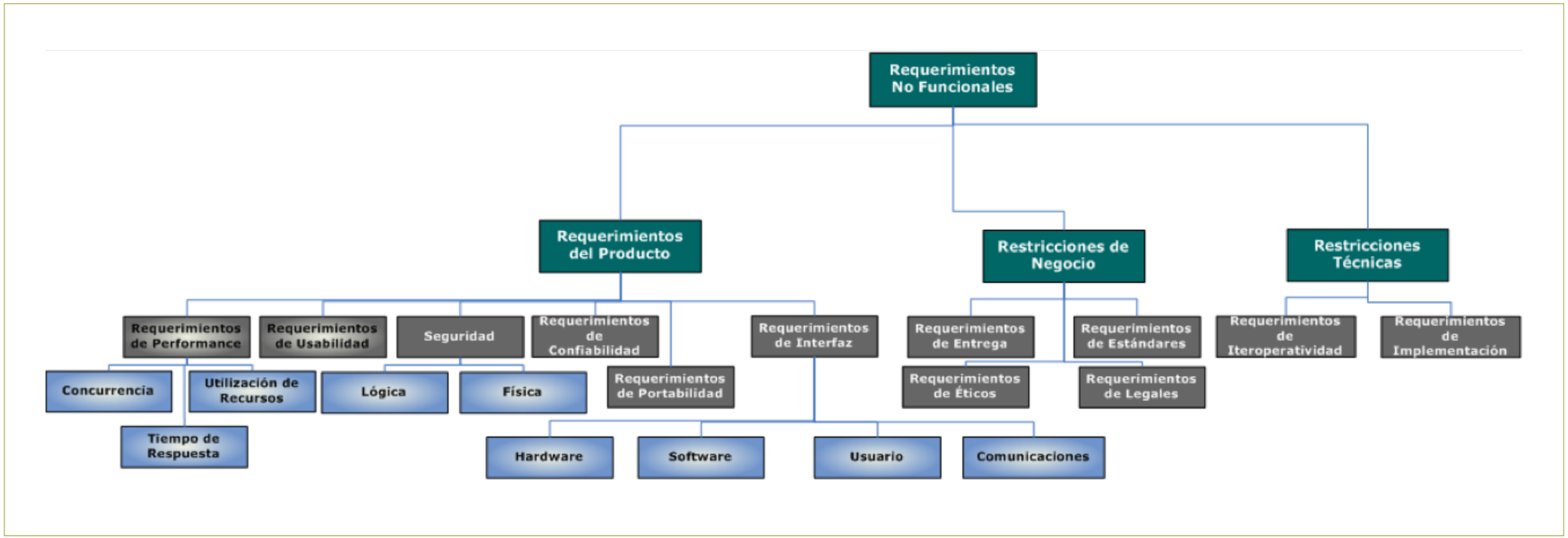


Imagen 4
Fuente: Definiciones de Ian Sommerville & Ian Gordon y la ISO 9126/1991

La siguiente tabla resume los requisitos no funcionales y sus características.

Requisitos no funcionales	
Requisitos de presentación	Forma grafica que se presentara al usuario.
Requisitos de usabilidad	La forma en que puede ser usado por el usuario. Ejemplo: se usará por personas con discapacidad auditiva. Para medir la usabilidad se toman algunos puntos de referencia tales como: • Tiempo de capacitación para que un usuario puedan manipularlo correctamente. • Tiempos para realizar tareas específicas. • Interfaces amigables para el usuario. • Documentación de uso, tipo y forma de presentación, configuraciones, ayudas, etc.
Requisitos de entorno	De acuerdo al lugar donde se ejecute el software. Describen el comportamiento externo del software. Ej: El sistema mantendrá la luminosidad del cultivo entre 70 y 80 lúmens.
Requisitos culturales	De acuerdo a creencias o la comunidad en que se utilice.
Requisitos legales	La normatividad que debe cumplir para su uso.

Tabla 2. Requisitos no funcionales
Fuente: Propia.

Requisitos de seguridad

Nivel de protección de los datos que almacena el sistema, para uso no autorizado, perdidas de información, accesos no autorizados etc.

Requisitos de mantenimiento

Reflejan la facilidad, periodicidad, y costos que deben tenerse en cuenta para realizar las tareas de mantenimiento, con el fin de corregir defectos, hacer mejoras, etc.

Requisitos de comprobabilidad

El nivel en el que un sistema permite ser comprobado, verificado o medido, en cuanto a las funciones que realiza.

Requisitos de disponibilidad

Relacionan el tiempo total que el software está disponible y operable para su uso, tomando como referencia un patrón de tiempo.

Ejemplo: El servidor de hosting debe tener el 99.9% de disponibilidad online.

Requisitos de escalabilidad

Grado en el que el sistema puede aumentar sus capacidades. Por ejemplo: Aumentar el número de conexiones o usuarios conectados.

Requisitos de extensibilidad

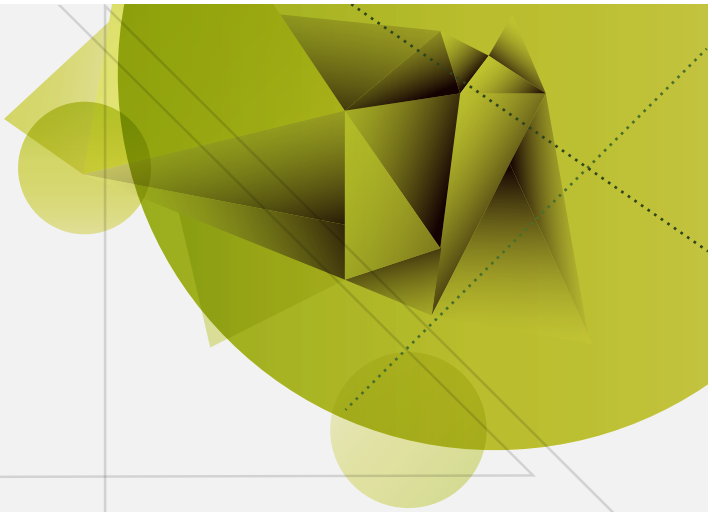
Miden la capacidad del sistema para incrementar las funcionalidades.

Ejemplo: Que adicional a conectarse a otros dispositivos mediante wifi, que pueda conectarse vía bluetooth.

1

Unidad 1

Análisis de requerimientos



Ingeniería de software II

Autor: Fredy León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

Muchos proyectos de software tienen altas posibilidades de fracaso si no se realiza un análisis previo y detallado de las necesidades del cliente o usuario, no se hace una proyección clara del alcance del proyecto y el analista de software no se involucra y comprende a fondo el problema a solucionar, haciendo que los requisitos que fueron identificados no correspondan a las necesidades del usuario o cliente final.

Por esta razón, el análisis de requerimientos tiene su enfoque en la generación de las especificaciones adecuadas que permitan describir de forma clara y concisa las necesidades del cliente, para de esta forma minimizar los problemas relacionados con la gestión de dichos requerimientos.

En este módulo aprenderá todo lo relacionado acerca del Análisis de requerimientos, tales como técnicas, herramientas, procesos de validación, gestión de cambios y otros temas que servirán de referencia para obtener requerimientos de calidad que le permitan culminar con éxito los proyectos de software que se presenten en su vida profesional.

U1 Metodología

El módulo se basa en la lectura individual de la presente cartilla y posterior desarrollo de actividades planteadas en la plataforma online.

Se espera una participación y acceso a plataforma a su consideración, como mejor pueda organizarse, para cubrir los objetivos de aprendizaje y participación. Por mi experiencia, recomiendo la asistencia diaria para aprovechar al máximo el curso. Es interesante seguir los hilos de participación 2-3 veces al día y dedicar uno momento para analizar y participar inteligentemente.

La comunicación conmigo la pueden realizar abiertamente a través del foro y personalmente a través del email.

Al final de cada sección enviaré un breve mensaje resumen destacando los mejores en sus intervenciones individuales y en los trabajos de grupo de esa sección.

Aquellos que hayan llamado mi atención negativamente por su falta de participación o por lo inadecuado de las mismas les enviaré un mensaje privado.

U1

Desarrollo temático

Análisis de requerimientos

Objetivos del análisis de requisitos

- Realizar una detección temprana de problemas.
- Construir un modelo de software que condense los requisitos del sistema.
- Asegurar viabilidad técnica, de costes y planificación.

Tareas del análisis de requisitos

Reconocimiento del problema

Evaluar las situaciones planteadas por el cliente a fin de identificar la necesidad existente y se realiza un plan de trabajo. También se inicia un proceso de comunicación con el equipo de trabajo a fin de reconocer y analizar grupalmente el problema a solucionar.

Evaluación y síntesis

Analizar, evaluar y condensar el documento de requerimientos, detallando las funciones del sistema, interfaces, diseño, etc.

Modelado

Utilizar técnicas de modelado como UML, Casos de uso, para definir roles y funciones de cada uno de los actores que intervienen en el sistema, partiendo de los objetivos y requisitos documentados.

Especificación

Para dar una representación del programa que pueda ser revisada y aprobada por el cliente. Se realiza la documentación.

En esta etapa normalmente se generan cambios de comportamiento, funciones establecidas para el sistema, la forma en que se va a presentar la información, criterios de validación y se reevalúa globalmente el plan de proyecto a fin de reafirmar las proyecciones realizadas durante la etapa de análisis.

El siguiente es el procedimiento normal que se debe ser a la hora de hacer la revisión del documento de especificación de requisitos:

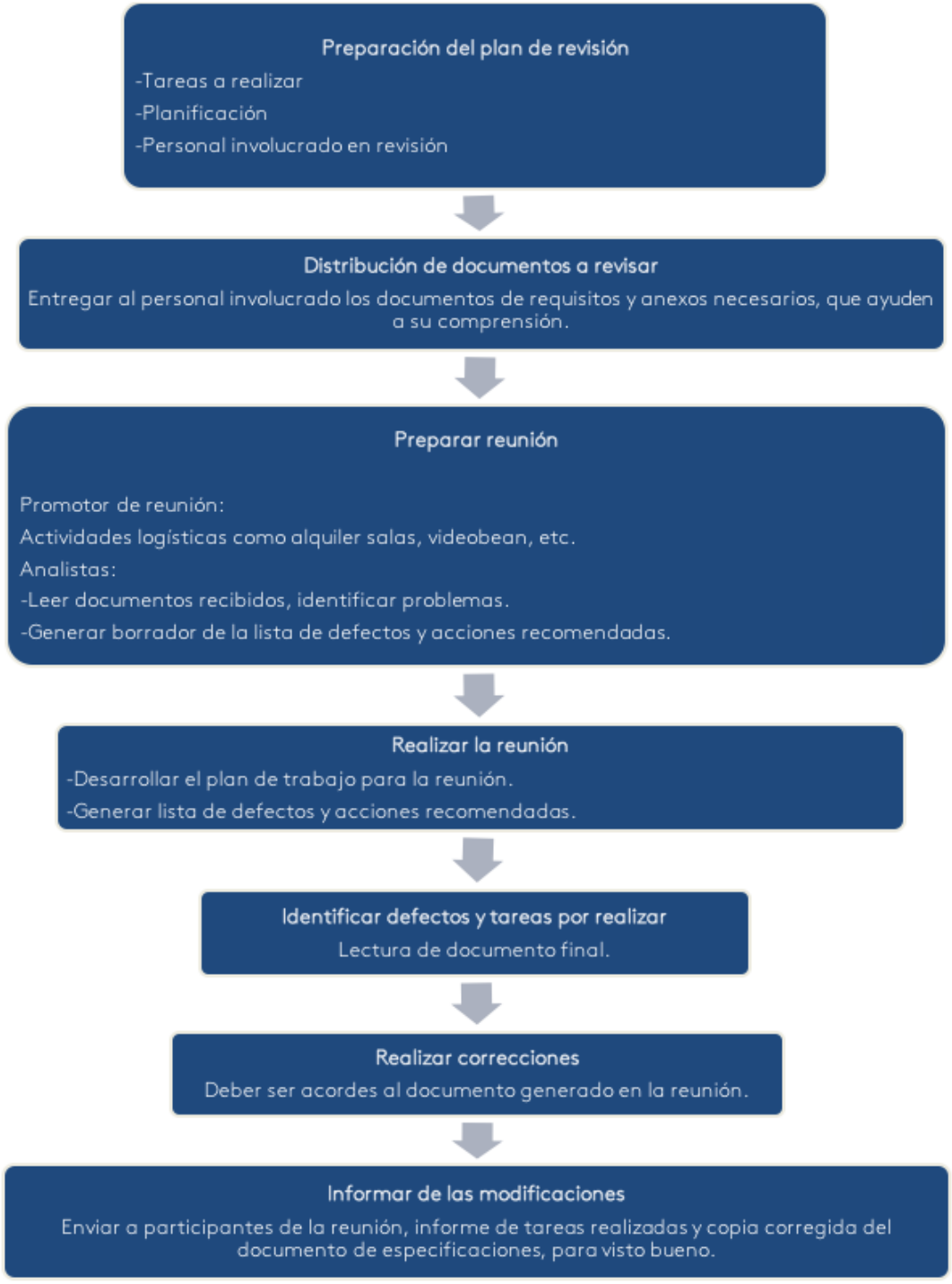


Imagen 1
Fuente: Propia.

Gestión de cambios de los requerimientos

Busca evaluar y planificar los cambios al proyecto de una forma eficiente, de tal forma que se pueda asegurar la calidad y continuidad del servicio. Debe existir una comunicación con asertiva con los demás procesos.

Roles que interactúan en la gestión de cambios

La siguiente tabla resume los roles principales que intervienen en el procedimiento de control de cambios.

Rol	Descripción
Comité de control de cambios (GCC)	Se encarga de aprobar o rechazar solicitudes de cambios. Conformado por clientes y desarrolladores.
Promotor del cambio	Quien lidera y hace genera la solicitud para un cambio de requisitos.
Evaluador	Se encarga de analizar el impacto que puede causar la petición de cambio en el sistema, ya sea a nivel técnico, de cliente, de marketing.
Modificador	Se encarga de realizar el cambio solicitado, de acuerdo al análisis que previamente ha sido revisado y aprobado.
Verificador	Se encarga de verificar si los cambios se han hecho en forma correcta.
Validador	Normalmente el cliente final, quien es el que hace una validación del cambio realizado.

Tabla 1
Fuente: Propia.

Diagrama del proceso

La siguiente figura muestra las actividades que se desarrollan durante la gestión de un cambio de requerimientos de un proyecto de software.

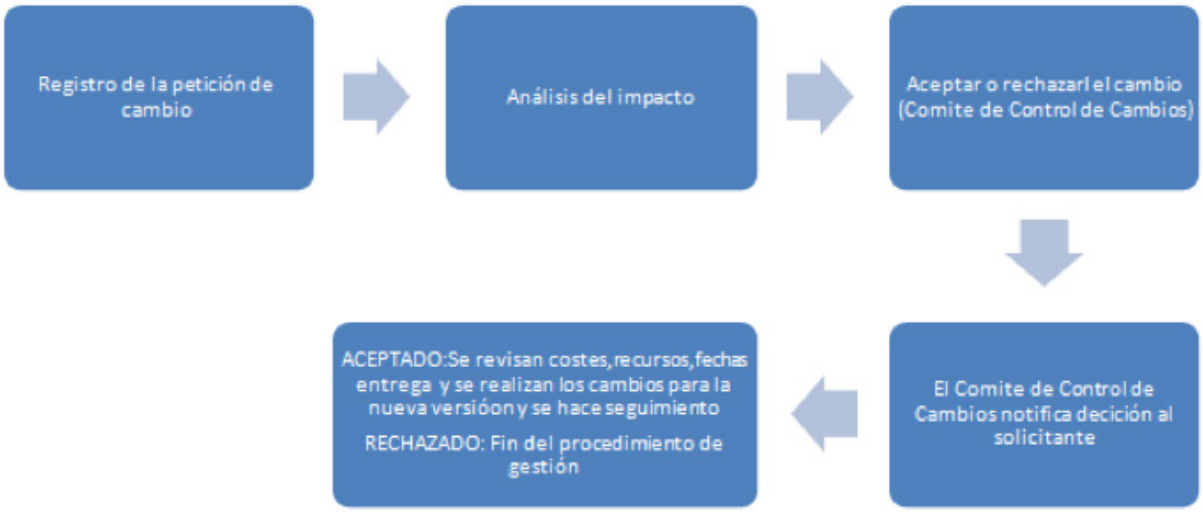


Imagen 2
Fuente: Propia.

Flujos de entrada y salida

A continuación se muestra las diversas entradas y correspondientes salidas que se requieren para la gestión de cambios por parte del Comité de Control de Cambios.

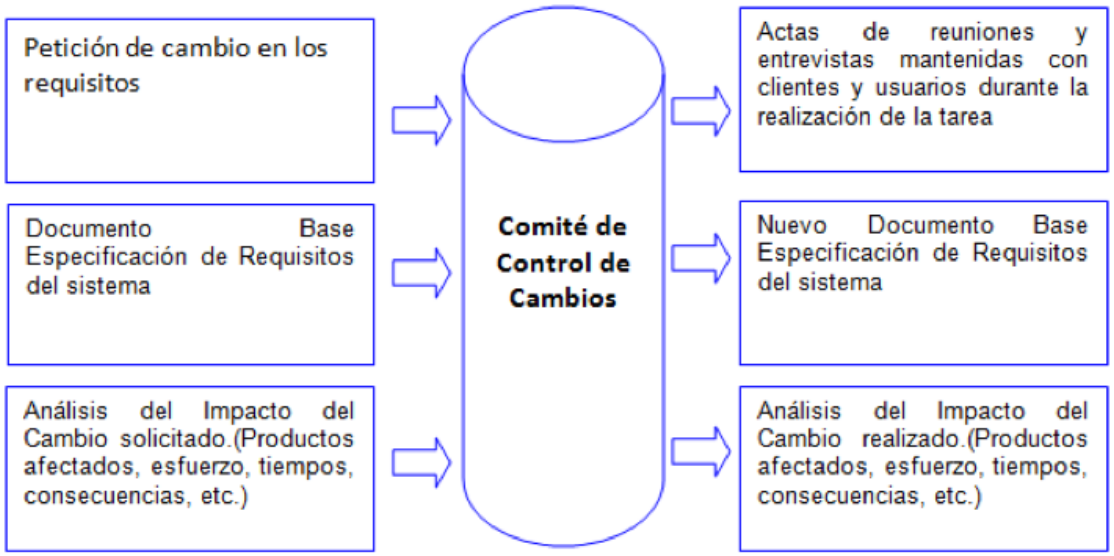


Imagen 3
Fuente: Propia.

Proceso de validación de requisitos

Busca garantizar que los requisitos cuenten con las características de calidad suficientes para poder continuar a la siguiente fase del proceso de desarrollo. Algunas de estas características son:

- Que sean consistentes.
- Completitud.
- Precisos.
- Acordes con la realidad.
- Que puedan ser verificados.

El proceso también busca disminuir el riesgo de tener que corregir después que el proyecto ha avanzado, lo que generaría altos costos.

- Como se muestra en la imagen 4, la validación de requisitos recibe como entradas: El documento de requisitos (creado durante la fase de especificación).
- Los estándares o lineamientos por los que deben regirse.
- El conocimiento que el equipo de trabajo encargado de realizar la validación.

Una vez realizado el proceso, como productos finales se tendrán: Un listado de problemas y un listado de acciones que permiten dar solución a dichos problemas y mejorar el documento de requisitos.

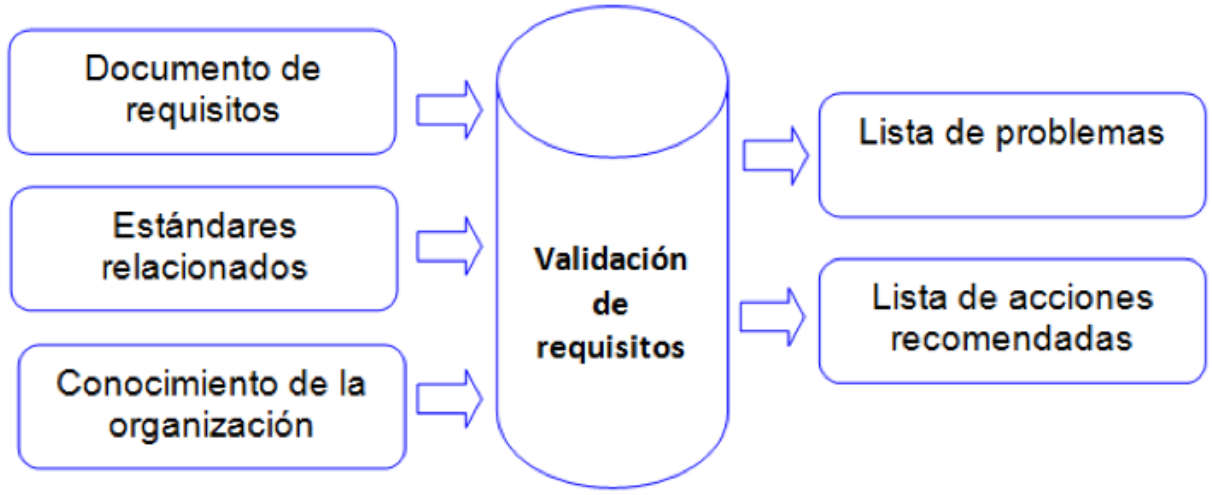


Imagen 4. Entradas y salidas de la validación de requisitos
Fuente: Propia.

Técnicas de validación de requisitos

Prototipado

El prototipado busca construir un modelo usable de un producto de software, de tal que clientes o usuarios puedan realizar pruebas y determinar las correcciones o mejoras que se deberán realizar, teniendo en cuenta la especificación de requisitos que dicho producto debe tener.

Existen diferentes tipos de prototipos, los cuales se muestran a continuación:

Mock-ups. Se enfoca en hacer una representación de la interfaz grafica de la aplicación y las pruebas se limitan a este contexto.

Imagen 5

Fuente: Propia.

Storyboards. Muestra gráficamente diferentes tareas que debe realizar la aplicación.

Imagen 6
Fuente: Propia.

Maquetas. Se crea un prototipo funcional, con las pantallas propuestas para la interfaz gráfica, botones, conexiones entre pantallas. Dependiendo de la fidelidad que se requiera, se puede programar acciones básicas para mostrar resultados concretos del sistema.

Independientemente del tipo de prototipo que se utilice, se deben seguir una serie de pasos para realizar el proceso de validación de requisitos, como se muestra en la siguiente figura.

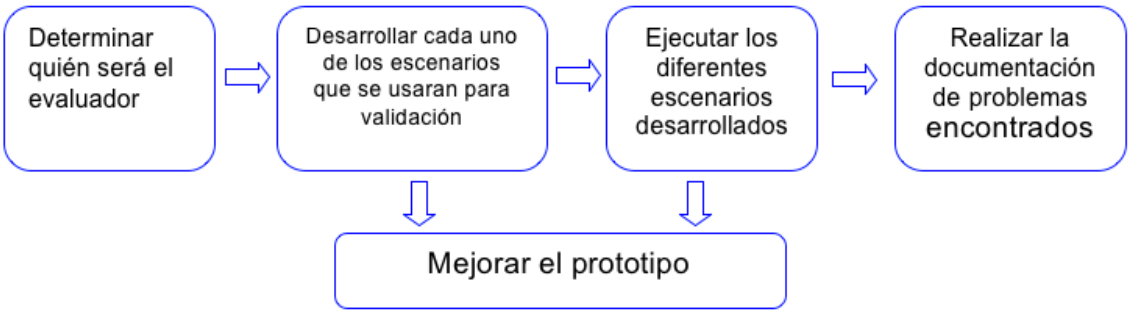


Imagen 7
Fuente: Propia.

Generación de casos de prueba

También se le llama Revisión de requisitos o Test de requisitos. Este método se centra en la verificabilidad, una de las características de la calidad del software. Se definen un caso de prueba de tal forma que se pueda comprobar cada uno de los requisitos funcionales del software.

Para que se pueda cumplir la verificabilidad de un requisito debe ser posible definir uno o varios casos de prueba.

A continuación, se desarrollará un ejemplo de desarrollo de un caso de prueba para el siguiente requisito:

X. El sistema deberá generar un informe.

Para este caso se define la tarea que debe realizar el software, mas no el cómo lo debe realizar, ni que resultados deberían generarse. Esto pasa porque el requisito no está bien definido y algunas razones son las siguientes:

- No se indica cuales son los datos base para generar el informe.
- El informe debe ser semanal, entre fechas, de un pedido, por un cliente, de ventas, etc.
- ¿En qué momento del proceso se debe generar?
- ¿Quién lo genera, automáticamente o manualmente?

Teniendo en cuenta lo anterior, el caso de prueba seria:

1. Se seleccionarán los datos A, B, C.… para generar el informe. Estos datos ya deben estar bien definidos, con sus respectivos datos (cantidades, fechas, detalles, etc.).
2. Se deberá invocar la función de generar informe (automática o manual).
3. Se generará un informe con los datos A, B, C.… indicando además fecha y hora de generación. Se debe incluir en la parte superior, el titulo del informe, el cual es ingresado por el usuario, etc.

Creación de manuales de usuario

Lo que busca es que se pueda crear el manual de usuario, partiendo de las especificaciones de requisitos. De no poderse lograr, seguramente dicha especificación no es suficientemente detallada.

Animación y validación de modelos

Consiste en realizar animaciones de los modelos de pantallas del sistema y demás especificaciones, para que el cliente o usuario pueda visualizar como quedara el producto final.

Herramientas

Rational Requisite Pro

- Desarrollado por IBM.
- Los requisitos son almacenados en forma de documentos.
- Gestiona el control de cambio de requisitos, para especificaciones de software y pruebas.
- Usa Oracle como motor de base de datos, corriendo sobre Unix o Windows.
- También soporta SQL Server sobre Windows.
- Página Oficial: http://www.ibm.com/developerworks/ssa/library/IBM_Rational_RequisitePro.html

CaliberRM

- Permite la captura y comunicación de las necesidades del usuario.
- Permite adaptarse a procesos de negocio.
- Ofrece variedad de soportes para clientes finales: Navegadores Web, Eclipse, Microsoft Visual Studio y Windows.
- Ofrece un repositorio seguro y centralizado de acuerdo a las necesidades del proyecto.
- Facilita el trabajo colaborativo.
- Permite el análisis de impacto Trazabilidad en tiempo real.
- Página Oficial: http://www.borland.com/en-GB/Products/Requirements-Management/Caliber

IRqA (Integral Requisite Analizer)

- Brinda soporte para Captura, Análisis, Especificación y validación de requerimientos.
- Gestiona el estado de arte.
- Los requisitos son almacenados en documentos MS Word.

Telelogic Doors

- Está diseñado para realizar tareas de captura y enlace de requisitos.
- Analizar y gestionar cambios a la información.
- Ligado a especificaciones técnicas de proyectos a los requisitos y normas especificas.
- Permite mejorar los procesos de comunicación, colaboración y validación.
- Página oficial: http://telelogic-doors.software.informer.com/

Visure Requirements

- Soporta captura manual de requisitos, servicios/soluciones y casos de prueba.
- Permite importar requisitos desde MS Word, Excel o XRI.
- Permite definir tipos de requisitos funcionales y no funcionales.
- Gestiona los cambios durante las fases del ciclo de vida.
- Gestiona la trazabilidad de requisitos y su respectivo ciclo de vida.
- Página oficial: http://www.visuresolutions.es/

CASE Spec

- Gestión, especificación y análisis de requisitos.
- Trabajo colaborativo.
- Análisis de trazabilidad.
- Diseño de especificaciones y documentación.
- Testeo y validación.
- Página Oficial: http://analysttool.com/

REM (Requisite Management)

- Brinda el soporte necesario para las fases de Ingeniería de Requisitos de proyectos de software.
- Creado por el profesor Amador Durán Toro, de la universidad de Sevilla.
- Permite agregar objetivos y actores.
- Crear casos de uso.
- Crear diferentes tipos de requisitos: Funcionales, no funcionales, de restricción, de almacenamiento de información.
- Genera matrices de rastreabilidad.
- Página oficial: https://www.lsi.us.es/descargas/descarga_programas.php?id=3

DRES

- Basado en PHP.
- La administración del proyecto se hace mediante navegadores web.
- La documentación ese genera en formato HTML.
- Usa MySQL como motor de base de datos y CORBA como servidor para almacenamiento y gestión de requerimientos.
- Soporte para extensiones en XML, directorios jerárquicos, reportes, etc.
- Página oficial: http://sourceforge.net/projects/dres/

OSRMT (Open Source Requeriments Management Tool)

- Permite la trazabilidad completa del Ciclo de vida de desarrollo de software: características, gestión de requisitos, diseño, implementación y pruebas.
- Control de versión de documento de requisitos.
- Página oficial: http://sourceforge.net/projects/osrmt/

Heler

- Monousuario.
- Soporte para Windows y licencia GPL (GNU, 2008).
- Creada bajo el Modelo Vista Controlador.
- Escrito en JAVA.
- Usa PostgreSQL como motor de base de datos.
- Usa Poseidon para modelado de diagramas UML.
- Módulos contenidos: Proyecto, stakeholder, actores y casos de uso, requisitos.

Gather Space

- Gestor de requerimientos.
- Casos de uso e historial de usuario.
- Jerarquía y Asociaciones.
- Casos de prueba.
- Seguimiento de validaciones.
- Página oficial: http://www.gatherspace.com

ITestMan

- Permite generar un repositorio de la información que se genera duran te el proceso de gestión de requisitos.
- Puede generar automáticamente las especificaciones de requisitos así como la documentación de pruebas de validación.
- Generar informes en formatos diferentes y para cada una de las áreas creadas para el proyecto.
- Acceso a la información en tiempo real, respecto a avances de requisitos, pruebas, problemas etc.
- Está ligado con estándares de desarrollo como MIL-STD 498, ISO 12207, ED109, CMMI nivel 3, etc.
- Página oficial: http://www.polar-consultores.es

Accept Software

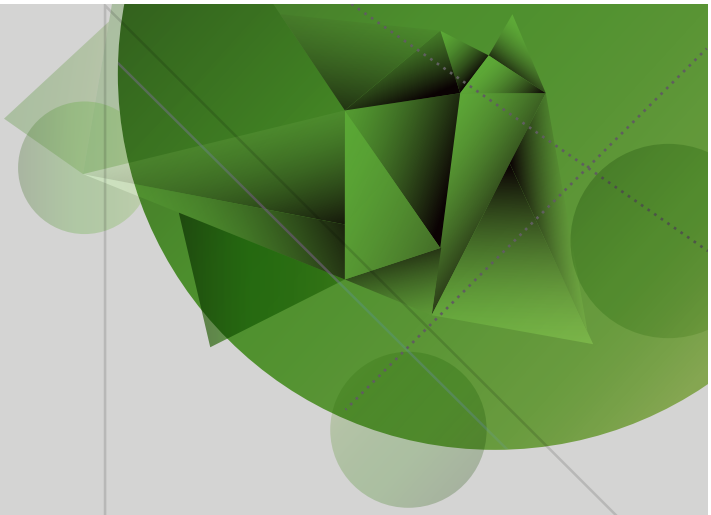
- Permiten recopilar, organizar y requisitos del producto, ideas, estrategias y directorios.
- Uso de componentes modulares.
- Anticipación de problemas y conflictos.
- Priorización y reducción de tiempos basada en información real de mercado.

- Toma de decisiones en los diferentes niveles.
- Página oficial: http://www.accept360.com/

2

Unidad 2

Arquitecturas de software



Ingeniería de software II

Autor: Fredy León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

La formación del Ingeniero en sistemas involucra el conocimiento de diferentes arquitecturas que permiten soportar el procesamiento y transmisión de la información, los cuales se convierten en elementos fundamentales al momento de desarrollar una solución de software. Esto le permite plantear diferentes alternativas para una solución informática, usando como soporte las soluciones computacionales existentes.

En el presente modulo aprenderá los fundamentos de la arquitectura de software, Clasificación de Estilos Arquitectónicos, Tipos y Modelos de Arquitectura de software, Niveles de diseño de software, entre otros temas, teniendo como enfoque la implementación en soluciones informaticas.

U2

Metodología

Para el desarrollo de los temas teóricos del curso se asignarán a los estudiantes algunas actividades de investigación y exposición, actividades que serán guiadas y apoyadas por el tutor y complementarán los contenidos publicados en plataforma. Las evaluaciones de las actividades serán utilizadas como retroalimentación para los expositores y para el mismo grupo.

El tutor realizará una teleconferencia semanal de un tema especifico, en el cual se presentarán ejemplos para reforzar dicho tema. En cada semana se publica el material de estudio que apoyara el desarrollo de los contenidos temáticos correspondientes a dicha semana, ya sea en recursos bibliográficos, archivos digitales o referencias electrónicas (páginas WEB).

Se recomienda el ingreso diario a la plataforma para aprovechar al máximo el curso.

U2

Desarrollo temático

Arquitecturas de software

La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un software, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores del software compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación. Es considerada el nivel más alto en el diseño de la arquitectura de un sistema puesto que establecen la estructura, funcionamiento e interacción entre las partes del software.

Componentes

Clientes	Es un aplicación informática que hace uso de un servicio remoto instalado en un servidor.
Servidores	Es una aplicación de software que está ejecutándose constantemente y cuya función es la de recibir peticiones de clientes y generar respuestas a las mismas.
Bases de Datos	Almacenamiento de información.
Filtros	Niveles de seguridad implementados.
Niveles en sistemas jerárquicos	Acceso a ciertas funcionalidades de acuerdo a los roles de los actores que intervienen en el sistema.

Tabla 1
Fuente: Propia.

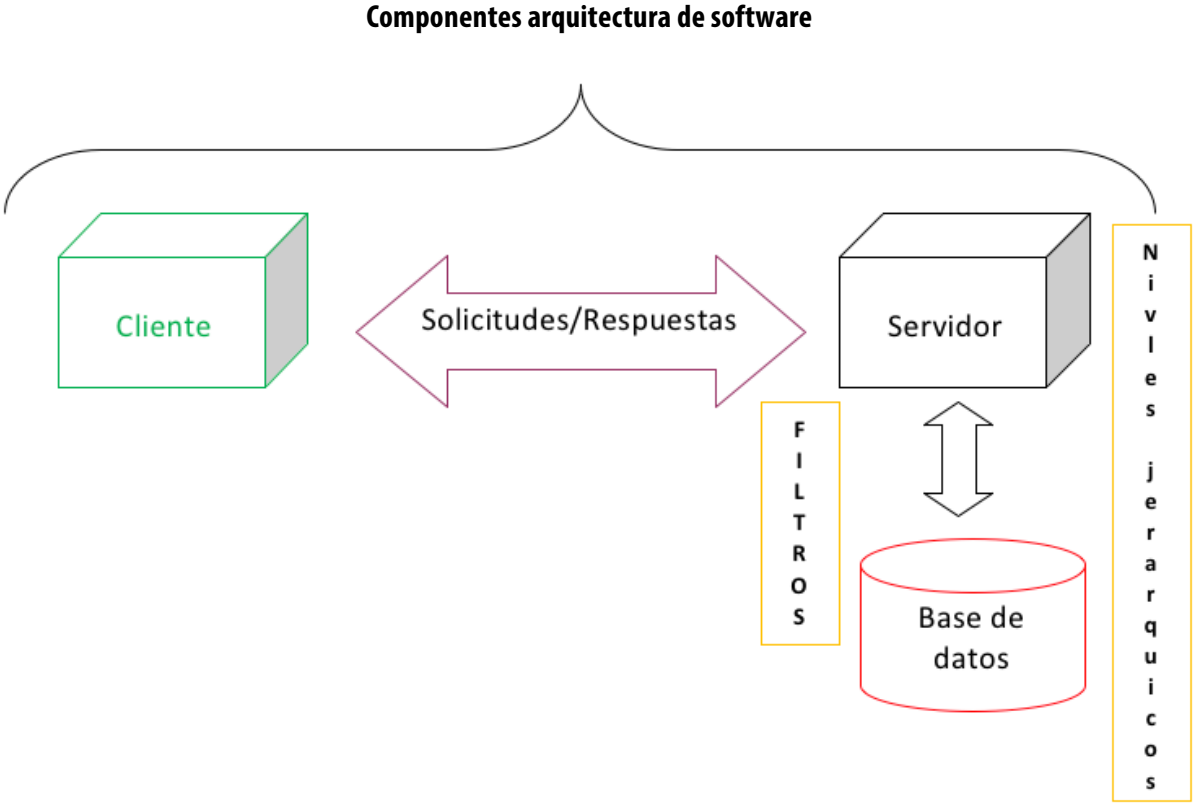


Imagen 2
Fuente: Propia.

Interacciones

Son las acciones que se realizan entre los componentes de la arquitectura de software.

Llamadas a procedimientos	Los procedimientos son estructuras de control que realizan la abstracción de una acción y esta se realiza a través de una función. Las funciones pueden tener argumentos con base en los cuales se realiza la acción. Existen llamadas a procedimientos en el mismo sistema (procedimiento convencional) y llamadas a procedimientos remotos (RPC- Remote Procedure Call), los cuales se alojan en otro servidor.
Comportamiento de variables	Son los estados de una variable, de acuerdo a los argumentos recibidos en una función. Las variables son gestionadas a través de las funciones.
Protocolos cliente servidor	Son el conjunto de reglas que permiten la transferencia de información entre las entidades que conforman el sistema. Algunos de los protocolos más conocidos son: -Protocolo TCP: utilizado en conexiones a internet y está orientado a conexiones. -El protocolo FTP: protocolo de transferencia de archivos, utilizado para procesos de transferencia de archivos. Aquí puede encontrar una lista de otros protocolos y su descripción: http://www.internetmania.net/int0/int133.htm
Transmisión asíncrona de eventos	El proceso de sincronización entre quien emite y quien recibe, se realiza en cada palabra de código transmitido, de acuerdo a unos bits especiales que contienen las especificaciones de cada código.

Tabla 2
Fuente: Propia.

Características:

- Hace parte del diseño del software.
- En este nivel se definen la estructura y propiedades globales del sistema tiene sus propios componentes, así como sus propiedades y relaciones entre cada uno de ellos.
- Representa la estructura del sistema y describe cada una de las partes que integran el mismo.
- Se rige por patrones que monitorean la forma en que están compuestos los componentes del sistema y las restricciones que se aplicarán a dichos patrones.
- Incluye parte del diseño y desarrollo no incluidos en otros módulos que conforman el sistema.

Tipos de arquitecturas

La siguiente infografía resume las características más importantes de las arquitecturas de software: Monolítica, Cliente-Servidor y 3 Capas.

Imagen 3
Fuente: http://2.bp.blogspot.com/-ZUdH7ytMTXg/UEZK48Pyz2I/AAAAAAAABWU/04N8adzSZAc/s1600/Tipos-de-Arquitecturas-de-Software_Blog.jpg

Niveles de diseño de software

Se definen 3 niveles: Arquitectura, Diseño de código y Diseño de la ejecución.

El nivel de Arquitectura, se compone de subsistemas, los cuales a su vez incluyen componentes(módulos) con las interconexiones con otros componentes.

El nivel de Código, incluye el diseño de los algoritmos y las estructuras de datos necesarias para la codificación. (números, caracteres, procedimientos, registros, arreglos, etc.).

El nivel de Ejecución, incluye los arreglos de datos, los mapas de memoria, asignaciones de cada registro, etc.

En el siguiente diagrama se muestra la composición general de los 3 niveles de diseño de software.

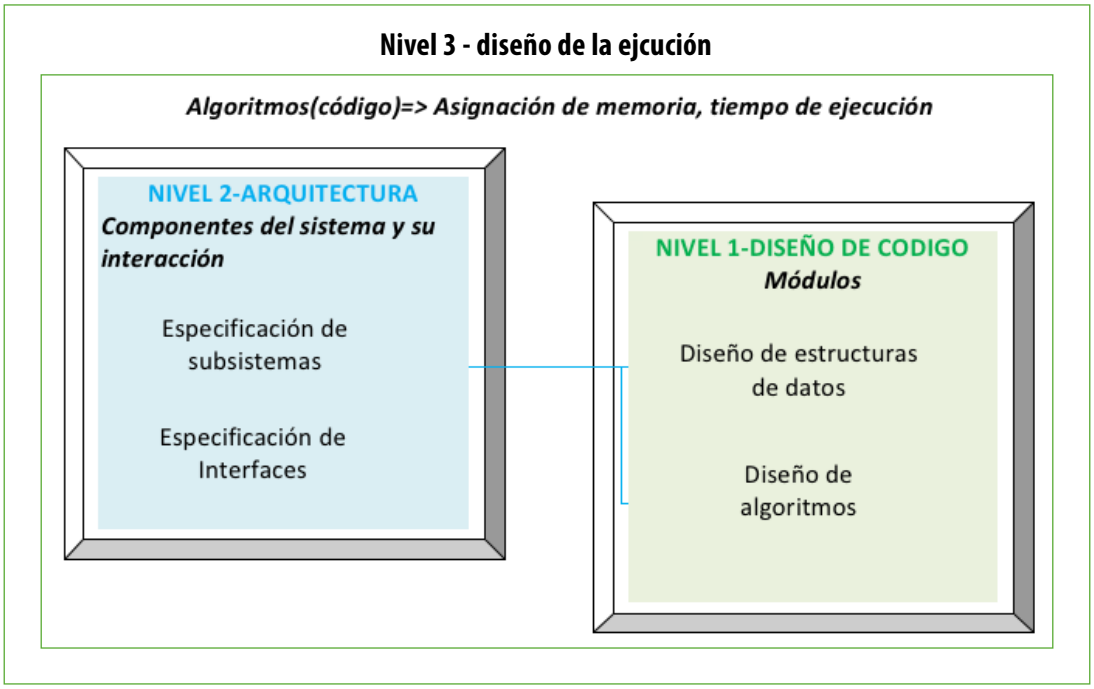


Imagen 4
Fuente: Propia.

Modelos de la arquitectura de software

Modelos estructurales

Se enfoca en que la estructura general del sistema tenga coherencia. No se concentra en la composición. Algunos modelos desarrollados bajo esta estructura son CORBA o PRISM, que son repositorios de componentes específicos.

Modelos dinámicos

Se refiere a los cambios que pueden ocurrir en la configuración o estructura del sistema, de acuerdo a los valores cambiantes de los datos y a eventos externos.

Modelos de proceso

Pasos o procesos relacionados directamente con la construcción de la arquitectura, la cual resulta de hacer el seguimiento de un script de proceso.

Modelo de interfaz

Define las interfaces de los subsistemas.

Modelo de relaciones

Representa el flujo de datos entre componentes del sistema.

Clasificación de estilos arquitectónicos

Sistemas Basados en Flujos de Datos

También denominado como "Tuberías y Filtros". Una tubería (pipeline) es una arquitectura que interconecta componentes computacionales llamados "filtros" a través de unos conectores denominados "pipes", y las interacciones se realizan en forma de flujo. Los datos son transportados a través de las "pipeline" pasando por los "filtros", de tal forma que transforman las entradas en salidas. No hay retención acerca de los estados en cada componente.

El siguiente es un ejemplo basico,que permite aclarar el concepto de la los Sistemas Basados en Flujo de Datos.

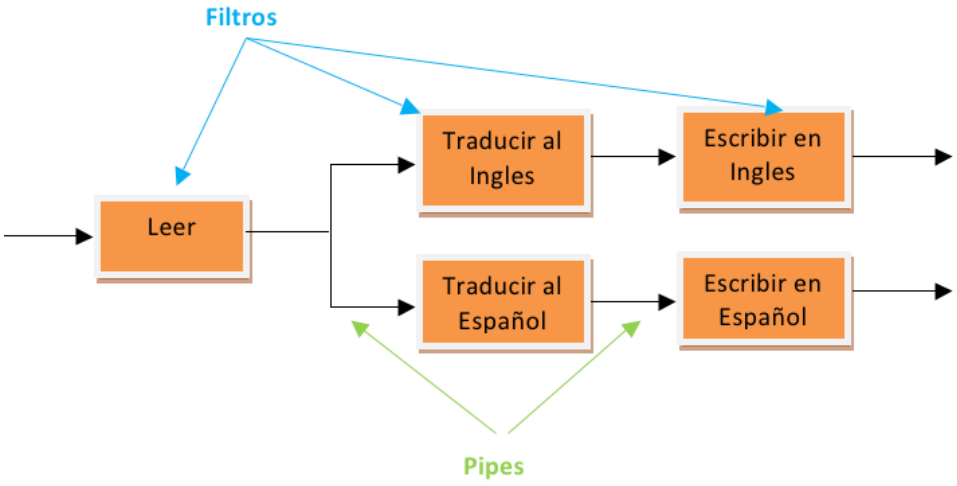


Imagen 5
Fuente: Propia.

Para tener en cuenta...

- Cada componente tiene un conjunto de entradas y de salidas y se encarga de transformar las entradas en salidas.
- Los filtros son independientes y no comparten el estado con otros filtros. Su labor la realizan en forma independiente del flujo de entrada.

Sistemas Call/Return (Llamada y Respuesta)

- Este sistema se centra en que pueda ser Escalable y Modificable.
- Los procedimientos o funciones proporcionan la abstracción de una acción.
- La acción es invocada cuando se realiza el llamado a la función.
- Los argumentos se utilizan para usar diferentes variables en una misma.
- llamada y el valor de dichos argumentos se pasa a la función.

Las arquitecturas más conocidas en este tipo de sistemas son: Modelo-Vista –Controlador (MVC) y Llamada de Procedimiento Remoto (RPC).

Modelo-Vista-Controlador

- MVC-Model-View-Controller.
- Se presenta en 3 clases diferentes y separadas, el Dominio, la presentación y las acciones basadas en los datos que ingresa el usuario.
- Se basa en la idea de reutilizar el código y separar conceptos, para facilitar las tareas del proceso de desarrollo y el mantenimiento posterior.

La siguiente grafica muestra la relación entre los 3 bloques del MVC y las principales operaciones que realiza cada uno.

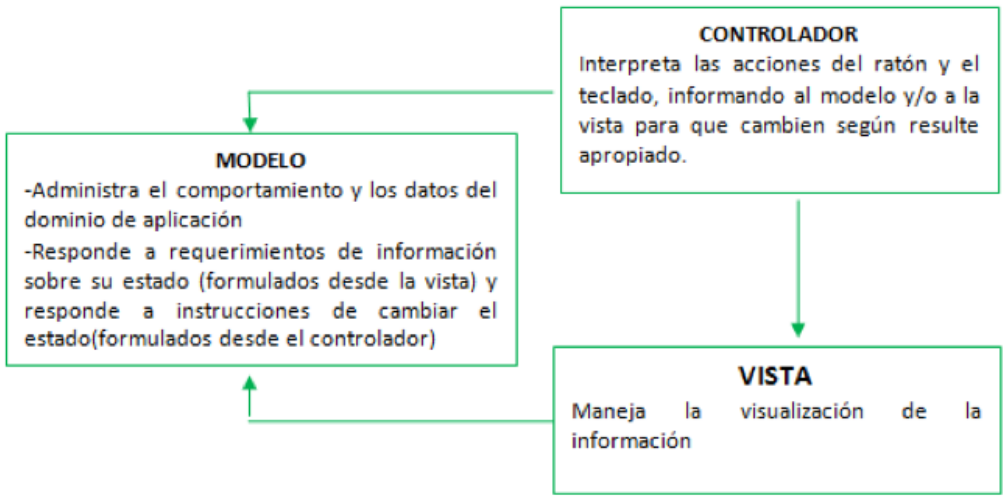


Imagen 6
Fuente: Propia.

Arquitectura de llamada de procedimiento remoto

- RPC - Remote Procedure Call.
- PRC se basa en el modelo Cliente-Servidor.
- Los componentes de una arquitectura programa principal y subrutina, se distribuyen entre varias computadoras de una red.
- Cada servicio se identifica por un numero de programa y de versión.
- Utiliza argumentos para pasar datos entre procedimientos ubicados en las maquinas.

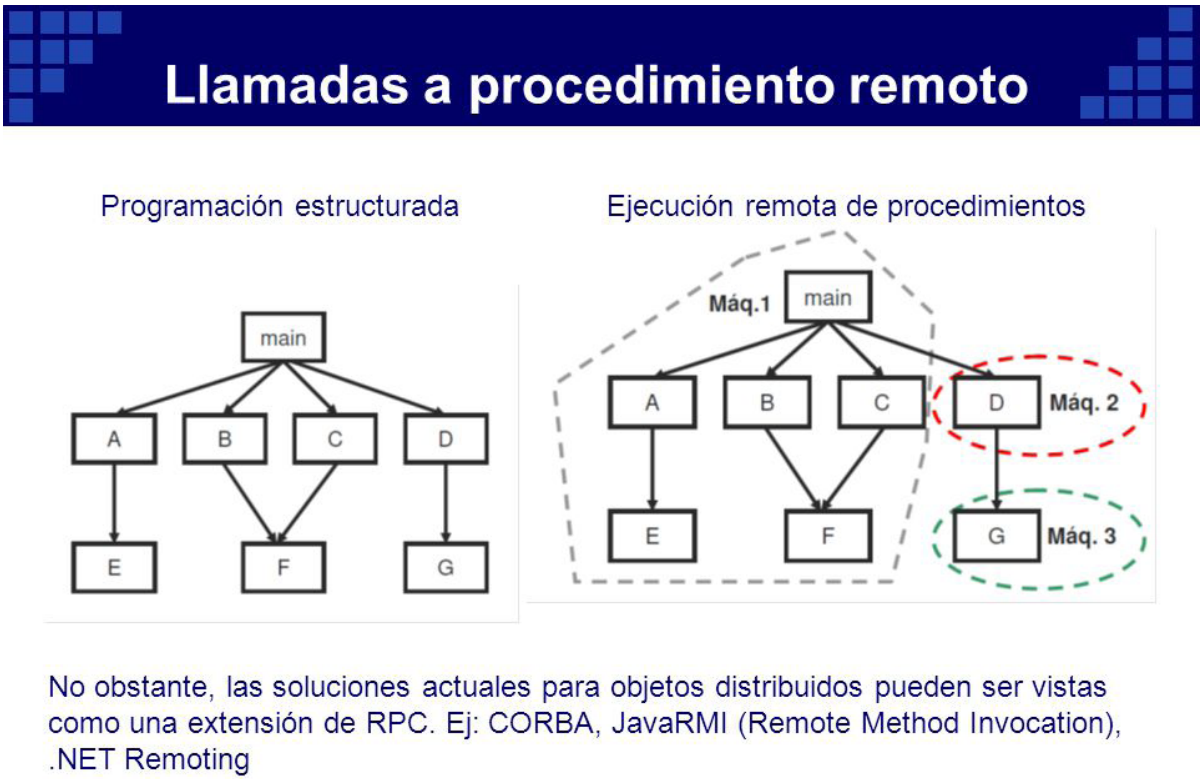


Imagen 7
Fuente: http://images.slideplayer.es/17/5411401/slides/slide_37.jpg

Arquitecturas de programa principal y subrutina

Existe un nivel jerárquico de subrutinas, lideradas por el programa principal, el cual hace el llamado a subrutinas inferiores y estas a su vez llaman a otras subrutinas. No es un modelo estructural, por lo que por ejemplo; la RUTINA 2.1.1, no necesariamente hace parte de la RUTINA 2.1

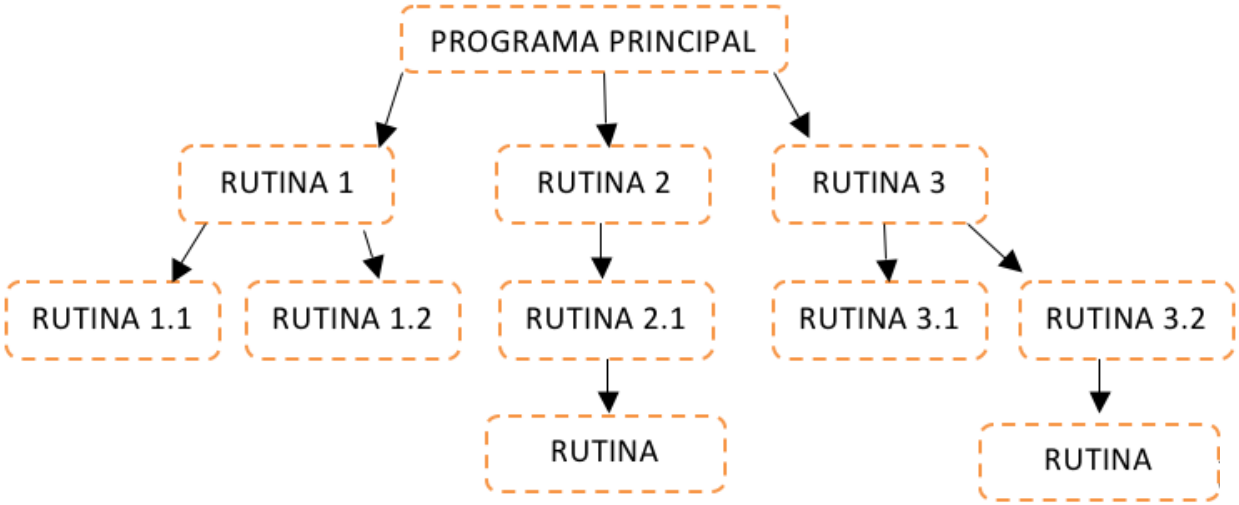


Imagen 8
Fuente: Propia.

Arquitecturas orientadas a objetos

- Se basa en los principios OO: Encapsulamiento, Herencia y Polimorfismo.
- Se realiza una encapsulación de los datos y operaciones por parte de los componentes (objetos).
- Se utiliza el paso de mensajes para que los componentes puedan comunicarse y coordinarse.
- Las interfaces y las implementaciones se encuentran separadas.

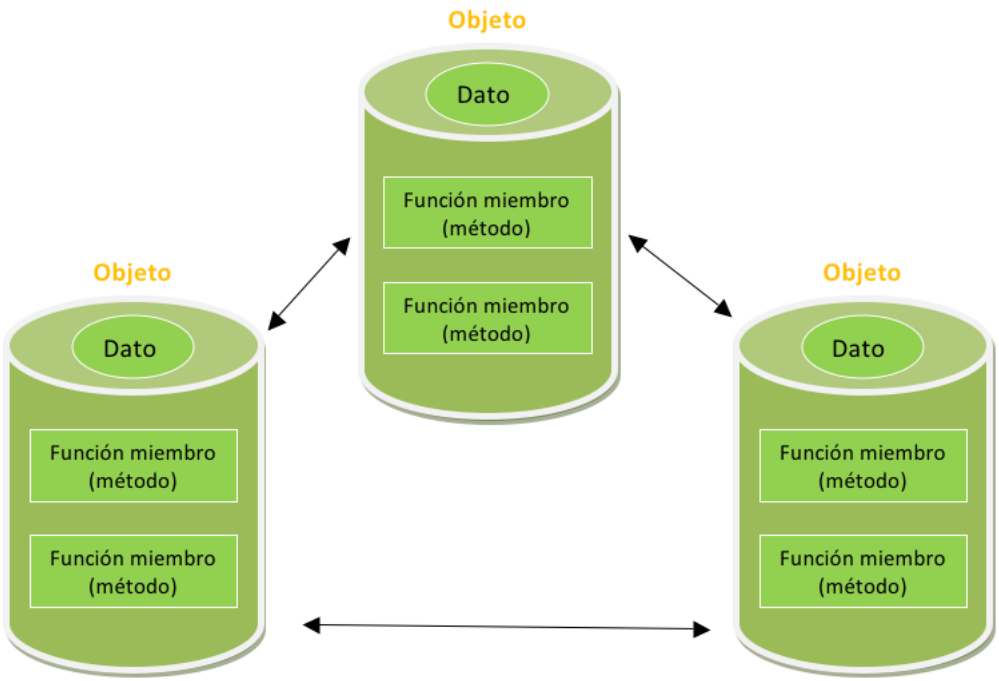


Imagen 9
Fuente: Propia.

Arquitectura de sistemas jerárquicos en capas

Se soporta sobre 3 capas:

- Una capa externa en la cual los objetos interactúan con la interfaz de usuario.
- Una capa intermedia que proporciona los servicios de utilidades y de software de aplicaciones.
- Una capa interna en la cual los objetos sirven de interfaz con el sistema operativo donde se ejecute el programa.

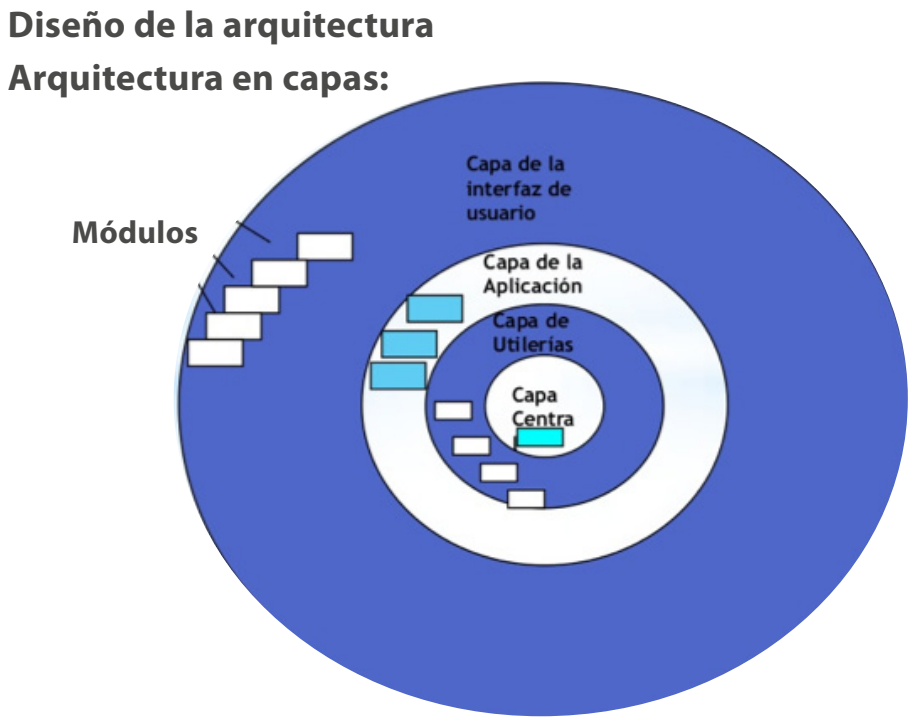


Imagen 9
Fuente: http://es.slideshare.net/landeta_p/2-2-estilos-arquitectonicos

Sistemas centrados en datos

- También llamado repositorios.
- Se basa en el almacenamiento de datos, ya sean archivos o bases de datos.
- Los clientes acceden al repositorio principal.
- Existen repositorios pasivos, en los que los clientes acceden a los datos independientemente de los cambios o acciones que haya realizado otro cliente.
- Existen repositorio de pizarrón, en el que el repositorio realiza el envío de notificaciones a un cliente especifico, de acuerdo a si los datos asociados a dicho cliente han cambiado. Varios subsistemas especializados, pueden integrarse para construir una solución total.

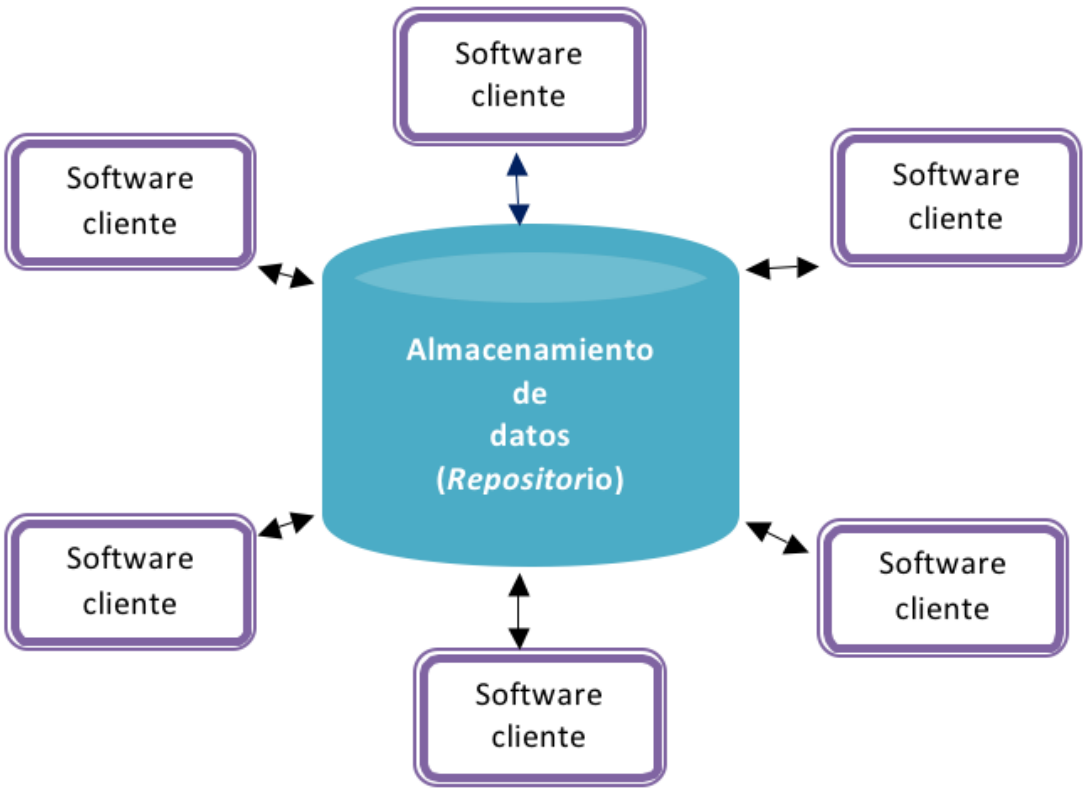
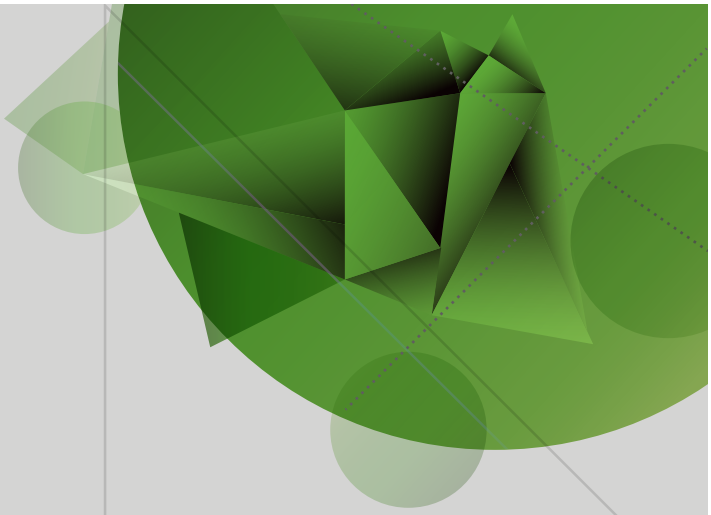


Imagen 10
Fuente: Propia.

2

Unidad 2

Arquitecturas de software



Ingeniería de software II

Autor: Fredy León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

Toda organización o profesional enfocado al desarrollo de sistemas debe generar productos que cumplan cabalmente los requisitos exigidos por el cliente, lo que implica generar un producto de calidad. Esto implica utilizar metodologías, procedimientos, normas, estándares que permitan analizar, diseñar, programar y probar el software a fin de alcanzar un máximo de confiabilidad, mantenibilidad y facilidades de pruebas, aumentar la productividad, ya sea para efectos de desarrollo o para las labores de control de calidad del producto de software.

En la presenta cartilla encontrara la fundamentación teórica asociada a la calidad: Principios, Modelos, Estándares, Métodos y otros temas, que usted como Ingeniero de Sistemas podrá aplicar en los procesos de desarrollo de software a los que se enfrente en su vida profesional.

U2 Metodología

El módulo se basa en la lectura individual de la presente cartilla y posterior desarrollo de actividades planteadas en la plataforma online.

Se espera una participación y acceso a plataforma a su consideración, como mejor pueda organizarse, para cubrir los objetivos de aprendizaje y participación. Por mi experiencia, recomiendo la asistencia diaria para aprovechar al máximo el curso. Es interesante seguir los hilos de participación 2-3 veces al día y dedicar uno momento para analizar y participar inteligentemente.

La comunicación conmigo la pueden realizar abiertamente a través del Foro y personalmente a través del email.

Al final de cada sección enviaré un breve mensaje resumen destacando los mejores en sus intervenciones individuales y en los trabajos de grupo de esa sección.

Aquellos que hayan llamado mi atención negativamente por su falta de participación o por lo inadecuado de las mismas les enviaré un mensaje privado.

U2

Desarrollo temático

Aseguramiento de la calidad

Conceptos

Se conoce como SQA (Software Quality Assurance) o GCS (Gestión de la Calidad del Software).

A continuación, se relacionan algunas definiciones de calidad, propuestas por diferentes organizaciones:

Organización	Concepto
ISO	"El conjunto de características de una entidad que le confieren su aptitud para satisfacer las necesidades expresadas y las implícitas".
Real Academia Española	"Propiedad o conjunto de propiedades inherentes a una cosa, que permiten apreciarla como igual, mejor o peor que las restantes de su especie".
IEEE Institute of Electrical and Electronics Engineers	"Grado con el cual el cliente o usuario percibe que el software satisface sus expectativas".

Tabla 1
Fuente: Propia.

Algunos conceptos en el tema de calidad:

Gestión de la calidad del software	Determina y aplica las políticas de calidad.
Aseguramiento de la calidad del software	Conjunto de actividades que de una manera planificada y sistemática, permiten evaluar el proceso de desarrollo del producto y aportan para que el producto cumpla los objetivos establecidos para la calidad.
Control de calidad del software	Técnicas y actividades operativas que se utilizan para cumplir los requisitos de calidad; manteniendo un proceso bajo control y eliminando las causas de defectos que se puedan presentar durante el ciclo de vida del desarrollo
Verificación	Comprobar si los productos finales por cada etapa del ciclo de vida cumplen con los requisitos que fueron establecidos para la etapa inmediatamente anterior
Validación	Si el producto de software satisface los requisitos que ha dispuesto el usuario o cliente

Tabla 2
Fuente: Propia.

Principios de la calidad de software

- Se requiere que la dirección del proyecto esté involucrada.
- Es un proceso que se desarrolla durante todo el tiempo que dure el ciclo de vida del proyecto.
- Es alcanzable siempre y cuando las personas involucradas contribuyan y trabajen conjuntamente para cumplir los objetivos.
- Prevenir los defectos implica una inversión de recursos.
- En las primeras fases del proyecto se debe priorizar y poner especial atención en la detección y eliminación de defectos.

Modelos

Modelo de McCall

De acuerdo al Modelo de McCall (1977), los siguientes son los factores sobre los que se sustenta la calidad de un producto software:

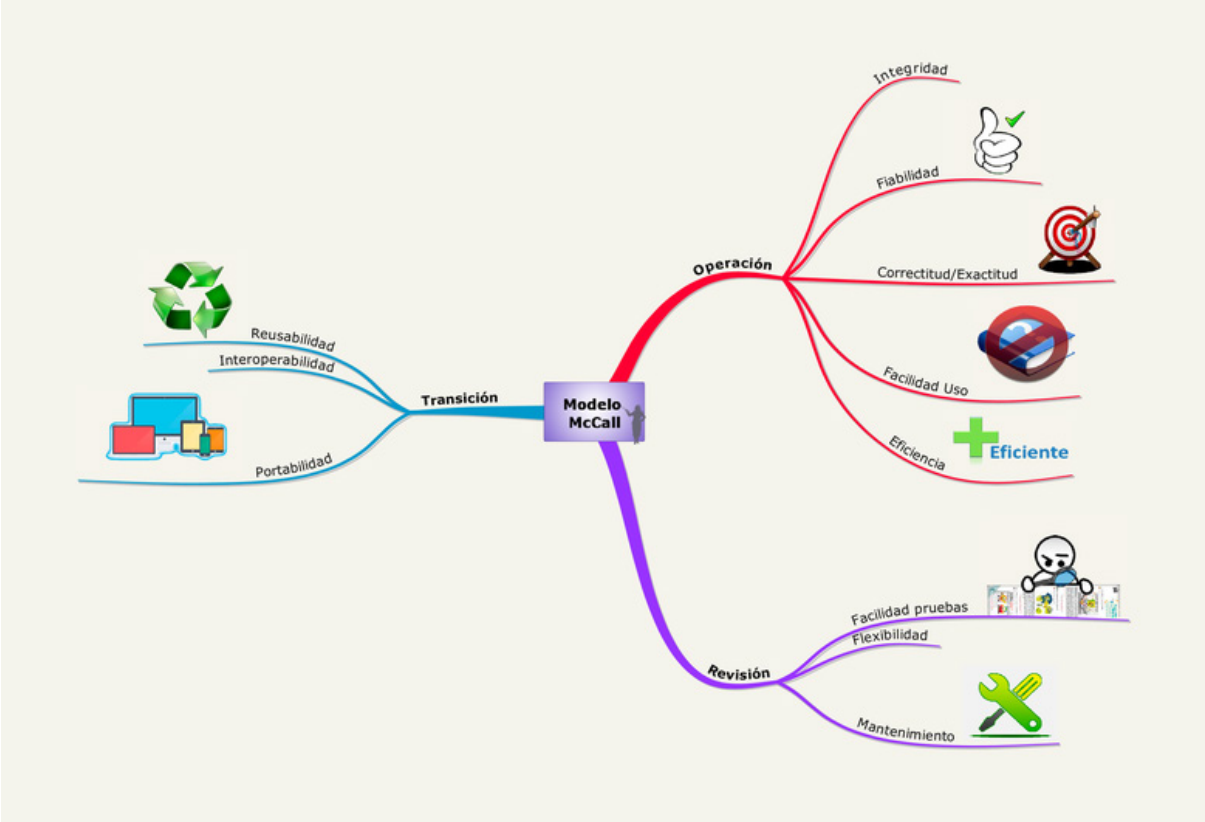


Imagen 1
Fuente: https://goo.gl/S2Odxv

Este modelo se organiza en 3 ejes: Transmisión, Operación del producto y Revisión. A continuación encontrara una explicación de cada uno de los criterios, los cuales se miden sobre una escala de 0 a 10.

Ejes	Factores	Criterios
Operación del producto	Facilidad de uso	• Facilidad de comunicación: que las entradas y salidas se puedan asimilar fácilmente. • Facilidad de aprendizaje: que el usuario se familiarice fácilmente con el software y su operación. • Formación: el nivel en el que nuevos usuarios incorporen el software dentro de sus actividades.
	Corrección	• Completitud: implementar completamente todas las funciones que fueron establecidas. • Consistencia: que sea uniforme con las técnicas y notaciones utilizadas para realizar el diseño y la implementación. • Trazabilidad o rastreabilidad: que se pueda hacer un seguimiento desde el proceso de requisitos hasta la implementación final.
	Integridad	• Control de accesos. Restricciones de accesibilidad a ciertas funciones del software y datos almacenados. • Facilidad de auditoría: que pueda ser auditado fácilmente. • Seguridad: protección y control de los programas o los datos que maneja.
Transmisión del producto	Fiabilidad	• Precisión: nivel de precisión que se requiere para realizar cálculos y generar. • Tolerancia a fallos: que pueda continuar funcionando en un entorno con condiciones diferentes sobre las cuales fue diseñado • Modularidad: que su estructuración se realice a través de módulos independientes. • Simplicidad: nivel de comprensibilidad al implementar nuevas funcionalidades. • Exactitud: nivel de precisión de cálculos que realiza y del control.
	Eficiencia	• Eficiencia en ejecución: minimización del tiempo de procesamiento. • Eficiencia en almacenamiento: minimización de espacio de almacenamiento que requiere.
Revisión del producto	Facilidad de mantenimiento	• Concisión: implementar funciones con menos cantidad de código. • Auto descripción: explicaciones (Popups, insitu, etc.) sobre la funcionalidades.
	Facilidad de prueba	• Instrumentación: que permita fácilmente la observación del comportamiento durante su ejecución, de tal forma que sea fácil realizar mediciones e identificar errores.
	Reusabilidad	• Independencia entre sistema y software: Funcionamiento independiente del entorno operativo. • Independencia del hardware: Funcionamiento independiente del hardware sobre el que se ejecute.
	Flexibilidad	• Capacidad de expansión: expandir a nuevas funcionalidades y almacenamiento de datos. • Generalidad: ampliar las funciones que han sido implementadas.
	Portabilidad	• Independencia entre sistema y software. • Independencia del hardware.
	Interoperabilidad	• Compatibilidad de comunicaciones: utilización de protocolos de comunicación e interfaces estandarizadas. • Compatibilidad de datos: utilización de estándares en la representación de los datos. • Estandarización en los datos: utilización de estándares para estructuras y tipos de datos.

Tabla 3
Fuente: Propia.

Modelo CMMI

CMMI (Capability Maturity Model Integration) o modelo de madurez y capacidad integrado.

Este modelo permite:

- Hacer una descripción de los componentes del modelo y las relaciones existentes entre los mismos.
- Realizar una comprensión general de las áreas que compone el proceso.
- Identificar y Localizar información relevante en el modelo.
- Aplicar en forma práctica los conocimientos sobre el entorno de trabajo y en el equipo encargado de la evaluación de componentes y relaciones del modelo.

La siguiente imagen muestra los niveles durante el proceso evolutivo del proyecto, siguiendo el modelo CMMI.

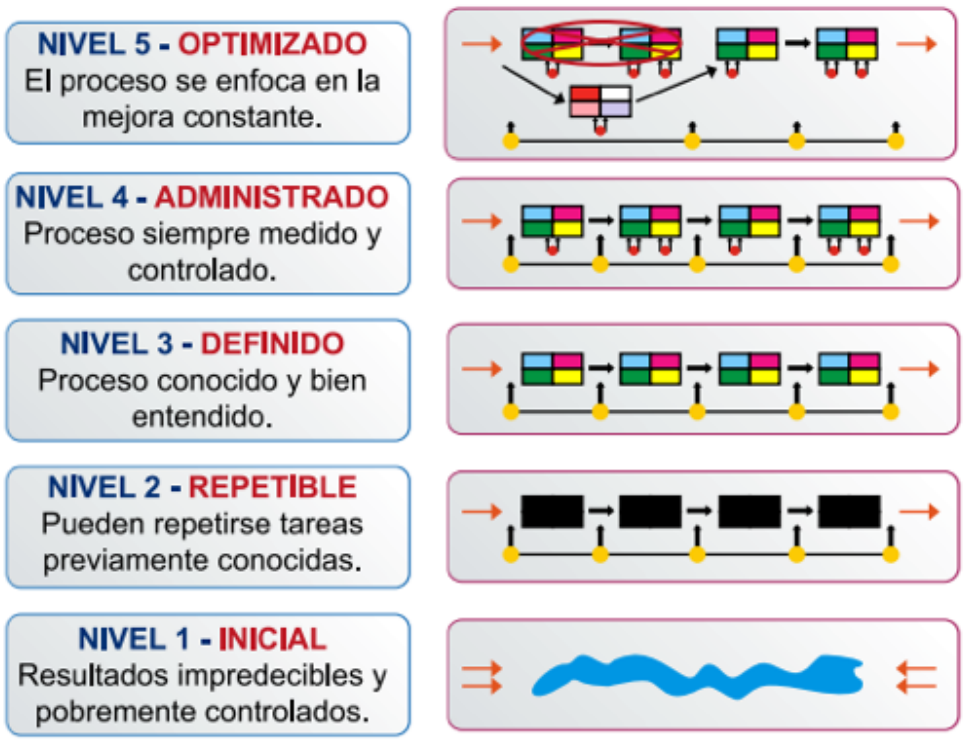


Imagen 2
Fuente: https://goo.gl/lWIDof

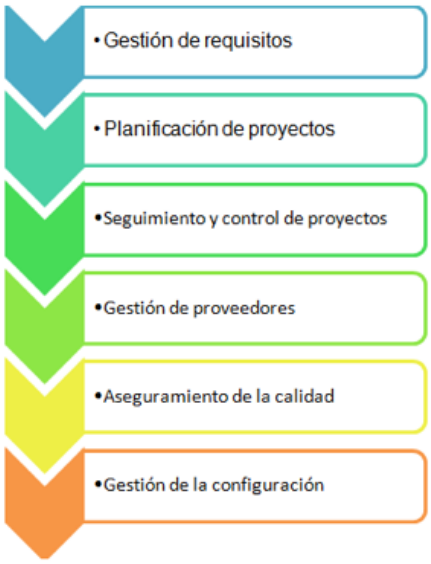
A continuación se muestra la descripción de cada uno de los niveles:

<table>
<tr><th>Nivel</th><th>Descripción</th></tr>
<tr><td>Nivel 1</td><td>

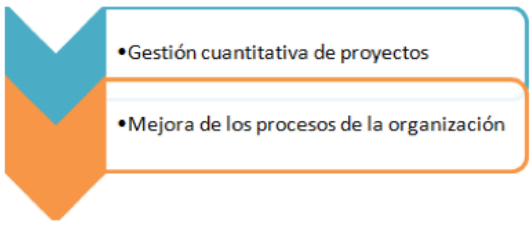
Empresas si procesos.
Presupuestos disparados.
Trabajos en tiempos extra para poder entregar el proyecto.
Entregas extemporáneas del proyecto.
Sin control acerca del estado del proyecto.
Futuro incierto del proyecto.

</td></tr>
<tr><td>Nivel 2</td><td>

El proyecto ya es gestionado y controlado durante el proceso de desarrollo.
Se conoce del estado del proyecto en todo momento.
Se pueden repetir tareas ya realizadas.

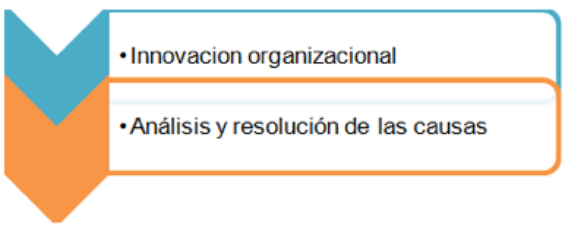
Procesos a implantar para alcanzar este nivel:


</td></tr>
<tr><td>Nivel 3</td><td>
Se encuentra establecida y documentada la forma en que se deben desarrollar los proyectos.
Existen métricas que permiten alcanzar los objetivos.
Procesos a implantar para alcanzar este nivel:


</td></tr>
<tr><td>Nivel 4</td><td>
Se usan objetivos medibles en los proyectos, enfocados a cumplir con las necesidades de cliente y la organización.
Buen nivel en uso de métricas que permiten gestionar la organización.
Procesos a implantar para alcanzar este nivel:


</td></tr>
<tr><td>Nivel 5</td><td>

Existen mejoras evolutivas y con innovación en los procesos.
Procesos del proyecto y la organización enfocados a mejorar las actividades.
Las métricas se han identificado, evaluado y puesto en marcha.

Procesos a implantar para alcanzar este nivel:


</td></tr>
</table>

Tabla 4
Fuente: Propia.

Modelo ISO/ IEC 15504 (SPICE)

- Se conoce como Software Process Improvement Capability Determinación (SPICE) o Determinación de la Capacidad de Mejora del Proceso de Software.
- Se enfoca en evaluar y mejorar la capacidad y calidad de los procesos de desarrollo y mantenimiento de sistemas de información y productos de software.
- Establece un marco de referencia y una serie de requisitos que deben ser implementados en los procesos de evaluación.
- Brinda información acerca de los requisitos necesarios para los modelos de evaluación de los procesos y de organizaciones.
- Brinda unas guías de referencia para definir las competencias que debe tener un evaluador de procesos.

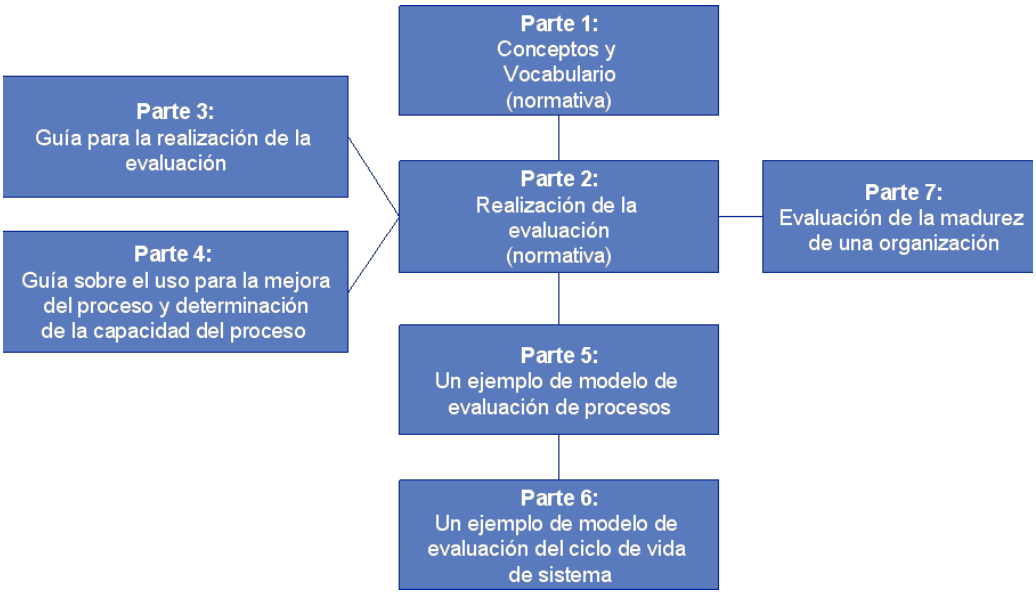


Imagen 3
Fuente: http://www.kybeleconsulting.com/wp-content/uploads/2011/11/estructura15504.gif

Este modelo plantea 6 niveles de madurez:

- Nivel 0. Madurez 0.
- Nivel 1. Se realiza.
- Nivel 2. Se planifica.
- Nivel 3. Procesos definidos y desplegados.
- Nivel 4. Procesos medidos y analizados.
- Nivel 5. Mejora continua, innovación y experiencia.

A continuación, se muestra una infografía en la que se explican cada uno de los niveles de madurez existentes en el Modelo ISO/ IEC 15504.

Imagen 4
Fuente: http://www.it360.es/i/infografia-iso15504-550.jpg

Modelo ISO/IEC 9126

- Software Product Evaluation, (Evaluación de los productos de Software).
- Indica características en cuanto a calidad, así como los lineamientos que se deben seguir para su implementación.
- El marco conceptual sobre el que se fundamenta, tienen en cuenta la Calidad del Proceso, Calidad del Producto de software y la Calidad en el uso del producto de software. Los 3 se evalúan a nivel interno y externo.

El modelo ISO/IEC 9126 se cimenta sobre 7 indicadores: Funcionalidad, confiabilidad, utilidad, eficiencia, capacidad de mantenimiento, portabilidad y calidad en uso. Cada uno de ellos está compuesto por una serie de criterios, que permiten definir el nivel de calidad del software.

Indicador	Criterio	Descripción
Funcionalidad	Adecuación	Suministrar las funciones correctas que puedan cumplir las tareas y objetivos que ha especificado el usuario.
	Exactitud	Realizar proceso procesos y generar resultados en forma precisa o de acuerdo a lo esperado.
	Interoperabilidad	Interactividad con otros sistemas específicos.
	Seguridad	Protección de información y datos. Niveles de acceso de acuerdo a roles y funciones.
	Conformidad de la funcionalidad	Cumplimiento de estándares de funcionalidad.
Confiabilidad	Madurez	Sortear fallas al encontrar errores. Ej.; Cuando no hay espacio suficiente, notificaciones al usuario acerca de operaciones indebidas.
	Tolerancia a errores	Seguir funcionando aun cuando se presenten errores.
	Recuperabilidad	Recuperarse y restaurar datos afectados después de una falla.
	Conformidad de la fiabilidad	Cumplir estándares o normas enfocadas a la fiabilidad.
Usabilidad	Entendimiento	Que el usuario pueda comprender fácilmente su uso y funcionalidades, teniendo como soporte la documentación del software.
	Aprendizaje	Como el usuario puede aprender a manejar el software, teniendo como soporte la documentación del mismos.
	Operabilidad	Como el software puede ser operado y controlado por el usuario.
	Atracción	Cualidades que hacen que el software sea atractivo y agradable para el usuario(diseño gráfico).
	Conformidad de uso	Cumplimiento de estándares de usabilidad.
Eficiencia	Comportamiento de tiempos	Los tiempos de respuesta, procesamiento y rendimiento deben ser adecuados.
	Utilización de recursos	Optimización de cantidad y tipos de recursos cuando el software funciona bajo los estándares y requerimientos sobres los cuales fue diseñado.
	Conformidad de eficiencia	Cumplimiento de estándares relacionados a la eficiencia.
Capacidad de mantenimiento	Capacidad de ser analizado	Que permita la realización de diagnósticos acerca de deficiencias, causas de fallas o identificar secciones modificadas.
	Cambiabilidad	Poder implementar una modificación, incluyendo el diseño, codificación y documentación de los cambios realizados.
	Estabilidad	Evitar los efectos inesperados.
	Facilidad de prueba:	Proteger los datos cuando se realizan pruebas a modificaciones.
	Conformidad de facilidad de mantenimiento	Cumplimiento de estándares de facilidad de mantenimiento.
	Portabilidad	Cambiar de entorno.
Calidad en uso	Eficacia	Que los usuarios puedan realizar procesos de forma exacta e integral.
	Productividad	Utilizas recursos adecuados de tal forma que no afecte la productividad del empleado.
	Seguridad:	Que no ponga en peligro la integridad de las personas, instituciones, software, propiedad intelectual o entorno.
	Satisfacción	Grado de satisfacción del usuario frente a la interacción con el software.

Tabla 5
Fuente: Propia.

El equipo de trabajo SQA

Su función principal es la del aseguramiento de la calidad del software. Para lograr este objetivo, el equipo de trabajo debe realizar una serie de funciones especificas, las cuales se relacionan en la siguiente grafica:

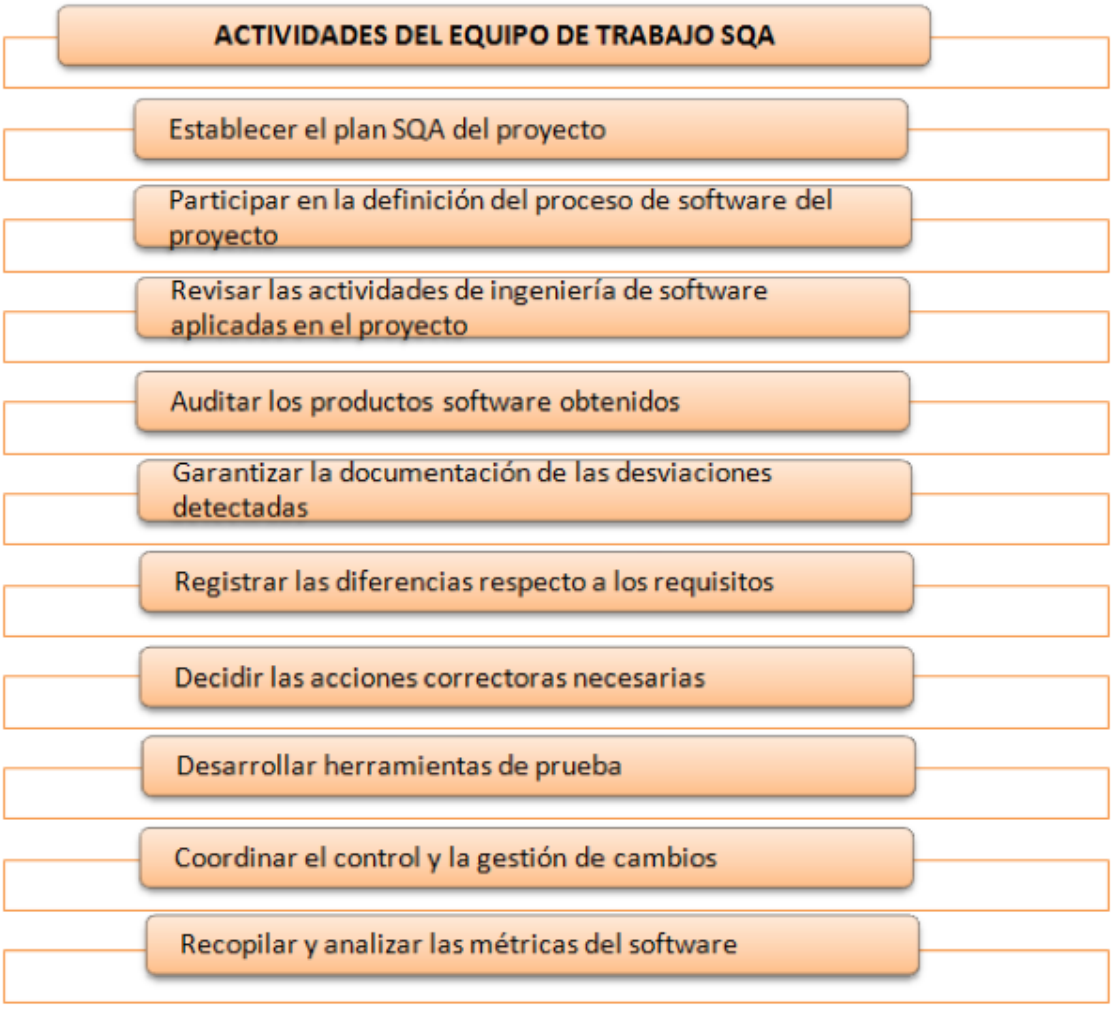


Imagen 5
Fuente: Propia.

Para poder realizar las funciones anteriormente relacionadas, cada integrante del equipo debe contar con ciertas habilidades técnicas, académicas y personales, que le permitan cumplir dichas funciones dentro del equipo de trabajo. La siguiente grafica muestra un resumen detallado de cada una de las características.

Imagen 6
Fuente: Propia.

Planes de calidad

Para el establecimiento de un plan para el aseguramiento de la calidad del proyecto, se deben seguir las siguientes pautas:

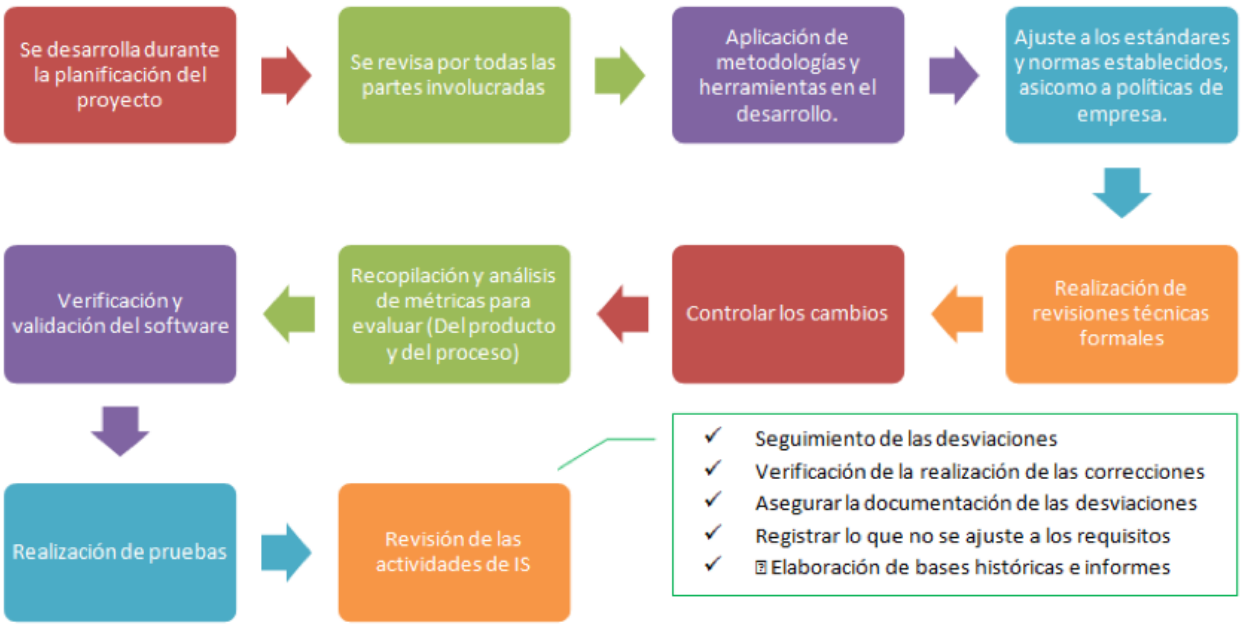


Imagen 7
Fuente: Propia.

Organismos de normalización

Dentro de los organismos más reconocidos Se encuentran la IEEE, ANSI y ISO.

IEEE

Institute of Electrical and Electronics Engineers o Instituto de Ingeniería Eléctrica y Electrónica.

La siguiente imagen, resume los estándares IEEE orientados al aseguramiento de la calidad.

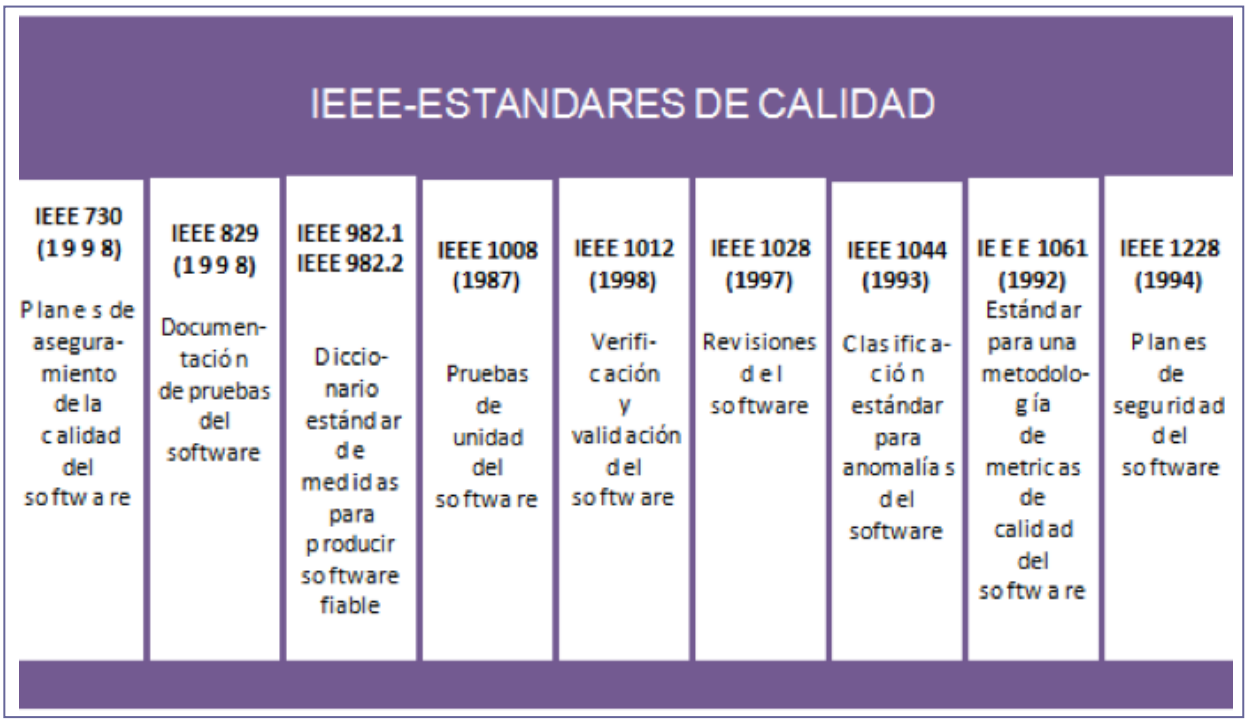


Imagen 8
Fuente: Propia.

ISO

Organización Internacional para la Estandarización o International Organization of Standardization.

Dentro de sus normas, la ISO9000 es la que reúne las directrices para la gestión de la calidad. A continuación se muestra la descripción de la misma y las normas que la componen:

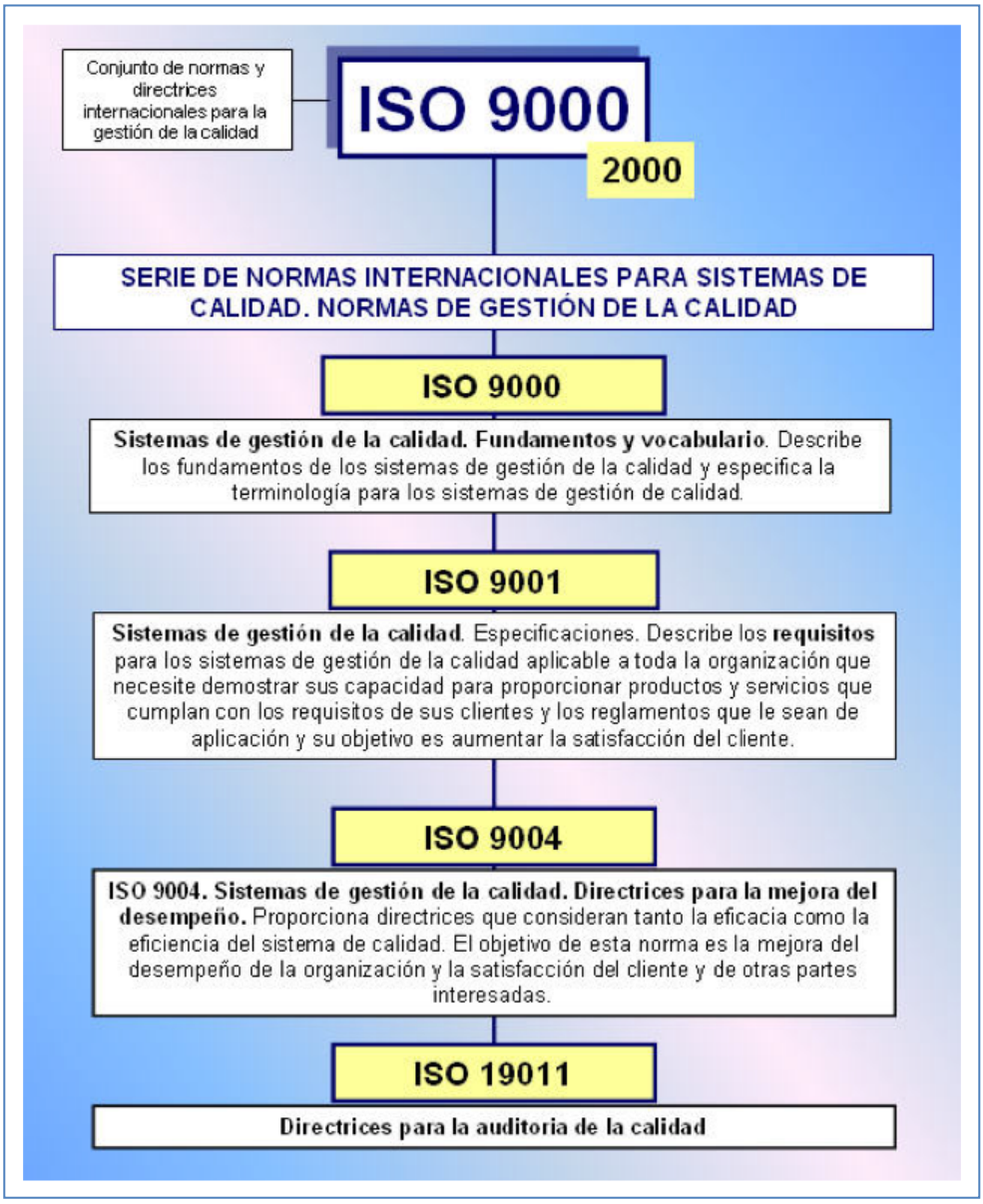


Imagen 8
Fuente: http://endrino.pntic.mec.es/jhem0027/calidad/normalizacion/iso9000dos.jpg

Otra de las normas comúnmente utilizadas como referencia para la implementación de planes de calidad es la norma ISO 25010.A continuación se relaciona su estructura general:

Imagen 10
Fuente: http://iso25000.com/images/figures/iso25010_b.png

En el siguiente link, encontrará una descripción más detallada de esta norma:

http://iso25000.com/index.php/normas-iso-25000/iso-25010

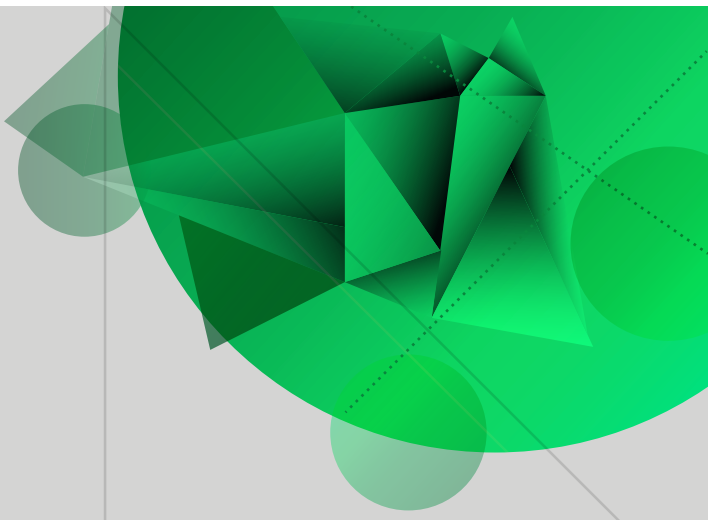
Métodos utilizados

- Auditorías PPQA (Process and Product Quality Assurance): para garantizar que el proceso y el producto generado, se ajusten según el plan establecido.
- Pruebas de Validación: ingreso de datos por parte del tester para identificar errores.
- Comparación de datos: comparación de resultados entre la aplicación de prueba con una aplicación patrón, introduciendo los mismos parámetros.
- Prueba de esfuerzo (Stress Testing): hacer uso del software en condiciones de alto nivel de carga durante un periodo de tiempo, para verificar su comportamiento.
- Pruebas de uso: uso del software por parte del usuario a fin de obtener retroalimentación. También es una manera para mejorar la interfaz.
- Revisiones por pares (Peer Reviews): revisiones hechas por los desarrolladores y un conjunto de pares que buscan evaluar el contenido técnico y/o de calidad. Pueden aplicarse al análisis, diseño y codificación.
- Revisión Técnica formal (RTF): incluye recorridos, inspecciones y revisiones a través de ciclos de inspecciones.

3

Unidad 3

Documentación de software



Ingeniería de software II

Autor: Fredy León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

La base de todo proyecto de software tiene su fundamentación y su apoyo en la documentación generada para que a partir de ahí se comience a realizar la construcción del producto solicitado por el cliente final. Esta empieza al tiempo que inicia el desarrollo, finalizando poco antes que se realice la entrega del programa o aplicación al cliente.

En la presente cartilla encontrará la fundamentación teórica relacionada con la documentación que se debe generar en cada una de las etapas del proceso de desarrollo, para que el Ingeniero de Sistemas en su rol de Project Manager pueda definir las bases de cómo elaborar , organizar y compartir toda la documentación necesaria para el proyecto, de una forma efectiva para optimizar el tiempo invertido o en su rol como Desarrollador, pueda generar la documentación necesaria y requerida para el soporte de cualquier proyecto de software.

U3

Metodología

El módulo se basa en la lectura individual de la presente cartilla y posterior desarrollo de actividades planteadas en la plataforma online.

Se espera una participación y acceso a plataforma a su consideración, como mejor pueda organizarse, para cubrir los objetivos de aprendizaje y participación. Por mi experiencia, recomiendo la asistencia diaria para aprovechar al máximo el curso. Es interesante seguir los hilos de participación 2-3 veces al día y dedicar uno momento para analizar y participar inteligentemente.

La comunicación conmigo la pueden realizar abiertamente a través del Foro y personalmente a través del email.

Al final de cada sección enviaré un breve mensaje resumen destacando los mejores en sus intervenciones individuales y en los trabajos de grupo de esa sección.

U3

Desarrollo temático

Documentación de software

En forma general, la documentación es el proceso de reunir y procesar documentos sobre un tema específico con el fin de difundir los resultados. Teniendo en cuenta lo anterior, la documentación de software, es el proceso de generar información asociada a cada una de las etapas de desarrollo, con el fin de ofrecer un soporte de consulta por parte de los usuarios finales del producto.

El conjunto de documentos que se deben generar para cualquier sistema, dependerá de las siguientes circunstancias:

- Del contrato con el cliente para el sistema (el cliente).
- Del tipo de sistema que se está desarrollando.
- De la vida útil prevista.
- De la cultura de la empresa.
- Del tamaño de la empresa que desarrolla el sistema.
- Del calendario de desarrollo.

Reglas para una buena documentación

Clements et al (2002) relaciona siete reglas que se deben tener en cuenta para una buena documentación. Estas reglas son:

1. Escribirse desde el punto de vista del lector: debe escribirse pensando en quienes van a ser uso de la misma, de forma clara, eficiente y cortes. No debe escribirse desde el punto de vista del escritor.
2. Evitar ideas repetitivas: esto con el fin de evitar confusiones al usuario. En esta regla se incluye la de evitar ambigüedades, la cual consiste en explicar en forma clara, para evitar diferentes interpretaciones.

3. Explicar las notaciones: hace mas fácil al lector comprender el significado de gráficos, diagramas y si se usa algún lenguaje visual, remitir al lector a la fuente semántica.
4. Organizar y estandarizar la información: permite al lector una mejor navegación, mejorando el entendimiento de la misma. También se puede referenciar más fácilmente.
5. Registros fundamentados: cuando se documenten resultados de decisiones, se debe explicar las razones de rechazo o aceptación de las rutas alternativas, con el fin de tener fundamentos futuros cuando se tengan presiones de cambios.
6. Documentación actualizada: actualización de los cambios realizados a través del tiempo con el fin de reflejar la realidad y la consistencia interna de la documentación.
7. Revisión de la documentación: obtener una retroalimentación por parte de los lectores a los cuales está dirigida, a fin de realizar ajustes.

Clasificación

Documentación interna

Es la que viene asociada directamente con el código.

Puede ser:

- Complementaria: cuando permite realizar una mejor comprensión acerca del funcionamiento y útil para estudiar el código.
- Ayudas interactivas: consideradas como una función mas del software y permiten que el usuario pueda realizar acciones de una forma más rápida y sencilla (Ej.; atajos de teclado).

Documentación externa

- No hace parte del código.
- Describe le forma de instalar y hacer uso del sistema, el diseño y los requisitos, las funciones y pruebas para realizar el mantenimiento.
- Debe poder utilizarse todo el tiempo en que el sistema esté operando.
- Debe contar con una buena indexación para que la información pueda ser encontrada con facilidad.
- Puede ser física o digital (bases de datos, texto).

Documentación del proyecto de software

La documentación del proyecto se lleva a cabo durante todo el proceso de desarrollo y en cada una de las fases se debe generar ciertos documentos. La siguiente imagen muestra un resumen de los documentos generados en cada una de las fases:

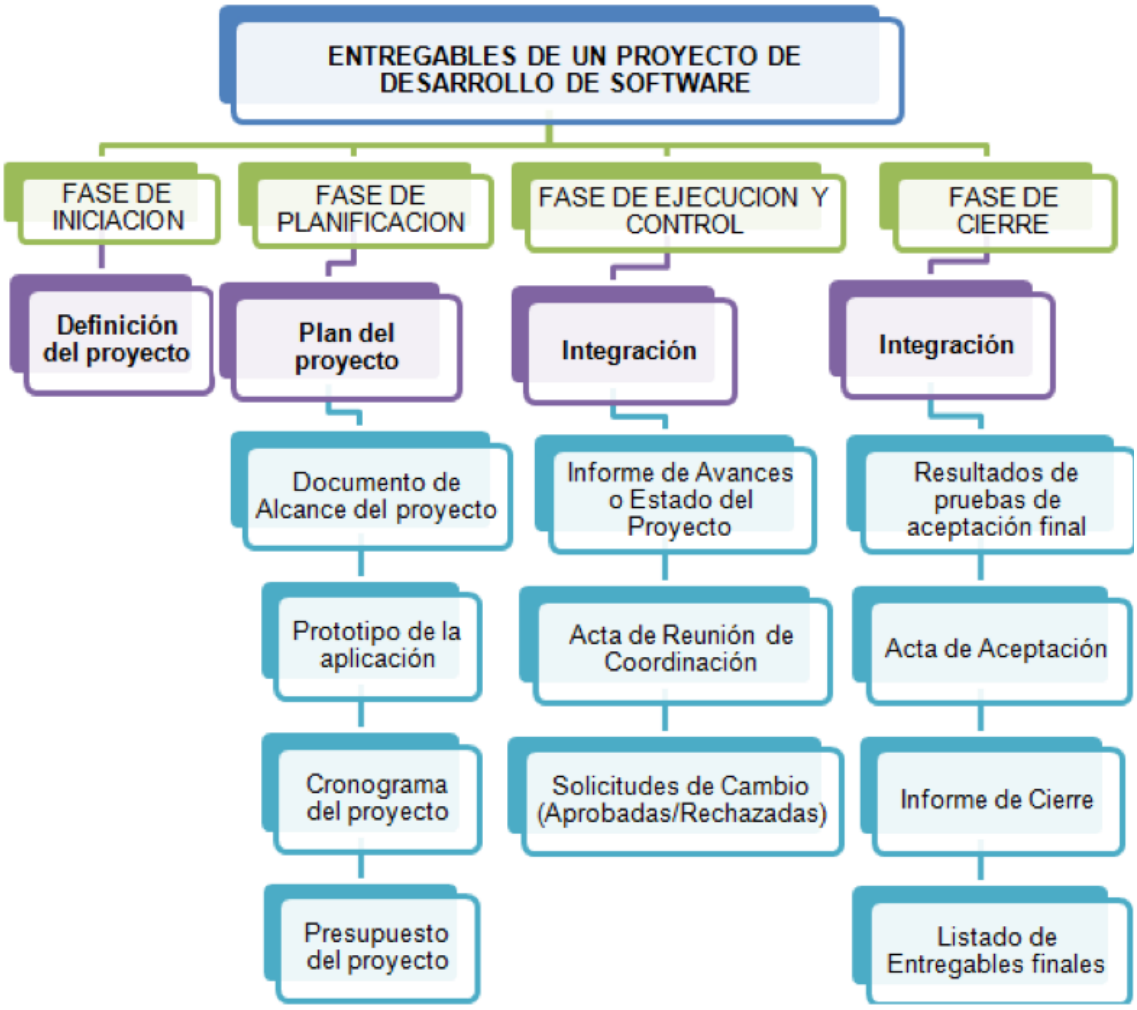


Imagen 1
Fuente: Propia.

Para una descripción detallada de los entregables relacionados en cada una de las fases, favor remitirse al siguiente link:

https://prezi.com/jmtgeyzapgju/entregables-por-etapa-de-desarrollo-de-un-proyecto-basado-en/

En la siguiente imagen muestra un resumen de los entregables en las diferentes etapas del ciclo de desarrollo de software:

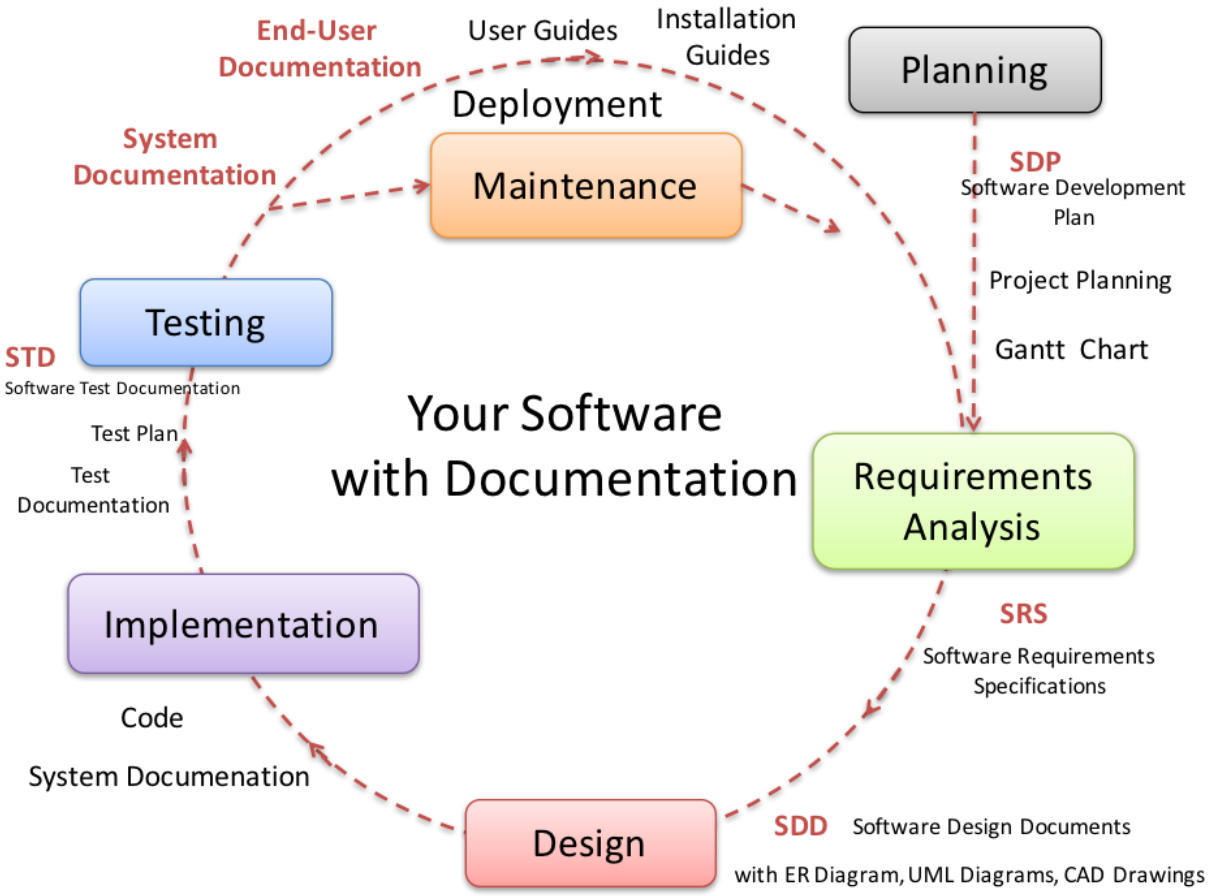


Imagen 2
Fuente: http://home.hit.no/~hansha/documents/software/software_development/topics/resources/Software%20Documentation%20Overview.pdf

Documentación del proceso

- Este tipo de documentación se genera para que se pueda gestionar y administrar el desarrollo del sistema.
- En los enfoques ágiles, se busca minimizar la Cantidad de documentación del proceso.

Algunas de las categorías no formales que hacen parte de la Documentación del proceso son:

1. Planes, estimaciones y horarios: utilizados para predecir y controlar el software proceso.
2. Informes: informan de cómo se utilizaron los recursos durante el proceso de desarrollo.
3. Normas: establecen el modo en que deber ser implementadas. Pueden ser desarrolladas a nivel de la organización, o bajo estándares nacionales o internacionales.
4. Documentos de trabajo (working papers): Graban las ideas y pensamientos de los ingenieros que trabajan en el proyecto, son versiones temporales de la documentación del producto, a menudo describen estrategias de implementación y la justificación de las decisiones de diseño.
5. Mensajes de correo electrónico, wikis, whatsapp, etc. Son un soporte de los detalles de comunicaciones entre gerentes e ingenieros de desarrollo.

Software Development Plan (SDP)

El Plan de Desarrollo de software, normalmente incluye las siguientes secciones:

1. Introducción: describe brevemente los objetivos del proyecto, el alcance y las restricciones que puedan afectar la gestión del proyecto (Por ejemplo; presupuesto, tiempo, etc.).
2. Organización del proyecto: describe cómo es la estructura organizacional del equipo de desarrollo, las personas involucradas, la metodología de desarrollo (Scrum, XP, Waterfall, etc.).
3. Análisis de riesgos.
4. Requisitos de recursos de hardware y software.
5. Desglose del trabajo: dividir el proyecto en actividades.
6. Calendario del proyecto: muestra las dependencias entre las actividades, el tiempo estimado para finalizarlas, la asignación de personas a las actividades. Los puntos 5 y 6 normalmente se realizan mediante un gráfico de Gantt (creado en, por ejemplo, Microsoft Project, Microsoft Excel, etc.).
7. Mecanismos de seguimiento y presentación de informes: Definición del informe de gestión. Que, cuando y como se deben producir.
8. Herramientas que está utilizando.

Software Requirements Specifications (SRS)

- Documento de Requerimientos de Software describe el QUE debería hacer el sistema, haciendo uso de palabras y figuras.
- Es el documento oficial que requieren los Desarrolladores de sistemas.
- Debe incluir tanto una definición Requisitos de usuario y una especificación de Requisitos del sistema.

- Es un medio de comunicación entre los miembros del Equipo de Desarrollo.
- Es el documento guía utilizado por los ingenieros de mantenimiento.
- Sirve de ayuda para que los administradores del proyecto, Planear, Presupuestar y programar el proceso de desarrollo de software.
- Sirve para indicar a los usuarios cómo utilizar y administrar el sistema.
- Sirve de soporte para cuando se realizan proceso de Control de Calidad, Certificación de Sistemas, etc.

Estructura básica del Documento de Requerimientos:

Sección	Descripción
Prefacio	Describe en forma general lo que se busca con el documento; el historial de versiones, incluyendo la justificación del porque de la creación de cada nueva versión y un resumen de los cambios realizados en cada versión.
Introducción	Describe la necesidad del sistema, incluyendo en forma breve las funciones y como trabajara con otros sistemas. También describe cómo el sistema se ajusta a los objetivos empresariales o estratégicos generales de la organización que requiere del software.
Glosario	Define los términos técnicos utilizados en el documento. No se debe hacer suposiciones sobre la experiencia o experticia del lector.
Definición de requisitos de usuario	Describe los servicios proporcionados para el usuario. Los requisitos funcionales y no funcionales del sistema. La descripción debe realizarse en lenguaje natural, diagramas u otras anotaciones que sean comprensibles para los clientes. Los estándares de producto y proceso que deben seguirse específicamente.
Arquitectura del sistema	Presenta una visión general de alto nivel de la arquitectura del sistema, mostrando la distribución de funciones a través de módulos del sistema. Los componentes arquitectónicos que se reutilizan deben ser resaltados e identificados.
Especificación de requisitos de sistema	Debe describir los requisitos funcionales y no funcionales con más detalle. Se pueden definir interfaces a otros sistemas.
Modelos de sistema	Incluye modelos de sistemas gráficos que muestren las relaciones entre los componentes del sistema y el sistema y su entorno. Por ejemplo; modelos de objetos, modelos de flujo de datos o modelos de datos semánticos.
Evolución del sistema	Describe los supuestos fundamentales en los que se basa el sistema, y cualquier cambio previsto debido a la evolución del hardware, necesidades cambiantes de los usuarios, etc. Les ayuda a los diseñadores de sistemas, para evitar decisiones de diseño que puedan limitar los posibles cambios futuros en el sistema.
Apéndices	Describe en forma detallada y especifica la información relacionada con la aplicación que se está desarrollando; Por ejemplo, requisitos de hardware y descripciones de bases de datos. Los requisitos de hardware definen las configuraciones mínimas y óptimas para el sistema. Los requerimientos de base de datos definen la organización lógica de los datos utilizados por el sistema y las relaciones entre los datos.
Índice	Incluye un índice alfabético normal, un índice de diagramas, un índice de funciones, entre otros.

Tabla 1
Fuente: Propia.

Software Design Documents (SDD)

- El Documento de diseño de Software, describe él COMO se podría hacer, haciendo uso de GUI Diseño, UML, Modelo Entidad-Relación, etc.
- El SDD muestra cómo será el sistema de software, por lo que describe la estructura del software, los componentes de software, las interfaces y los datos necesarios para la fase de implementación.
- Cada requisito en el SRS debe ser rastreable a una o más entidades de diseño en el SDD.
- Es la referencia principal para el código Desarrollo y, por lo tanto, debe contener toda la información requerida por Programador para escribir código.
- El SDD se realiza en dos etapas. El primero es un Diseño preliminar en el que se define la arquitectura general del sistema y la arquitectura de datos. La segunda es un Diseño detallado, en el que se proporcionan datos más detallados Se definen estructuras y se desarrollan algoritmos para la arquitectura definida.

Sección	Descripción
Propósito	Explica el propósito de la SDD y especificar la audiencia deseada para ella.
Alcance	Relaciona el documento de diseño con el SRS y con el software que se va a desarrollar.
Definiciones, Siglas y abreviaciones	Define todos los términos, definiciones. abreviaciones y demás simbología utilizada.
Referencias	Proporciona una lista completa de todos los documentos mencionados en el SDD, relacionándolos por título, número de informe, fecha y organización editorial.
Atributos de las entidades de diseño	Los siguientes son atributos comunes a todas las entidades, independientemente del enfoque utilizado, ya sea procedimental o orientado a objetos: Identificación: Nombre de la entidad. Tipo: Ej.; Subprograma, módulo, procedimiento, proceso, elemento de datos, objeto, etc. Propósito: Proporciona la justificación para la creación de la entidad, designando los requisitos funcionales y de desempeño específicos para los que se creó. Función: Indicar la transformación aplicada por las entradas de la entidad para producir la salida deseada. Subordinados: Identifica las entidades que componen esta entidad. Esta información se utiliza para rastrear requisitos para diseñar entidades e identificar las relaciones estructurales padre / hijo a través de un sistema de software descomposición. Dependencias: Identifica la relación de la entidad con otras entidades. Describe la naturaleza. de cada interacción que puede implicar iniciación, orden de ejecución, intercambio de datos, creación, duplicación, uso, Almacenamiento o destrucción de otras entidades. Interfaz: Realiza una descripción de como otras entidades interactúan con esta entidad, indicando los métodos de interacción y reglas que rigen esas interacciones. Proporcionando una descripción de los rangos de entrada, el significado de Entradas y salidas, el tipo y formato de cada entrada o salida, y códigos de error de salida. Recursos: Identifica y describe todos los recursos externos al diseño que son necesarios para la entidad realice su función. Proporciona información sobre elementos como dispositivos físicos (impresoras, discos, Memoria), servicios de software (bibliotecas matemáticas, servicios del sistema operativo, bibliotecas gráficas de interfaces de usuario) y recursos de procesamiento (ciclos de CPU, asignación de memoria). Procesamiento: Describe las reglas utilizadas por la entidad para lograr su función. Describe el algoritmo utilizado por la entidad para realizar una tarea específica. Incluye secuenciación de eventos o procesos, pasos del proceso, condiciones, criterios de terminación, etc. Datos: Describe el método de representación, valor inicial, uso, formato y valores aceptables de datos internos.
Descripción de la descomposición	Estructura General: Esta sección del SDD debe registrar la división del sistema de software en entidades de diseño. Describe la Forma en que el sistema está estructurado y el propósito y la función de cada entidad. Para cada entidad, proporciona una referencia A la descripción detallada. Utiliza los atributos de identificación, tipo, propósito, función y subordinados. Enfoque de procedimiento: Incluye una descripción de los módulos básicos del sistema y cómo se refieren a otros módulos (a qué módulos llama, etc.) También se deben proporcionar descripciones del módulo en el que se descompone el sistema.
Enfoque orientado a objetos	En caso de que se utilice un enfoque orientado a objetos, debería haber diagramas de clase (herencia) que muestren las clases en el sistema. También se deben proporcionar descripciones textuales para cada clase / objeto que el sistema se descompone dentro. Se hace uso de UML para describir los siguientes diagramas: • Diagramas de casos de uso. • Diagramas de clase. • Diagramas de secuencia. • Diagramas de estado. • Diagramas de actividad.

Tabla 2
Fuente: Propia.

Software Test Documents (STD)

Especifica los elementos o componentes se van a ser probados, de tal forma que el equipo de trabajo realice el proceso de Validación y Verificación de los requerimientos funcionales y no funcionales. El Documento de pruebas de software incluye:

- Software Test Plan (STP) o Plan de pruebas de software.
- Software Test Documentation (STD) o Documentación de las pruebas de software.

A través del Plan de Pruebas se hace una trazabilidad de los requerimientos, logrando que el equipo de trabajo identifique el porcentaje de avance del proyecto.

Del STP se pueden identificar errores, defectos o fallas que tiene el prototipo y de esta forma realizar las correcciones necesarias, asegurando de esta forma la calidad del producto final. El Plan de pruebas de software incluye los siguientes ítems:

- Identificador del plan de prueba.
- Introducción.
- Artículos de prueba.
- Características a probar.
- Características no probadas.
- Enfoque.
- Criterios de aprobación / rechazo de los elementos.
- Criterios de suspensión y requisitos de reanudación.
- Resultados de la prueba.
- Tareas de prueba.
- Necesidades ambientales.
- Responsabilidades.
- Necesidades de personal y formación.
- Horario.
- Riesgos y contingencias.
- Aprobaciones.

Los resultados de las pruebas se registran normalmente en un documento Excel, que incluye las pruebas unitarias y de integración. Un ejemplo de este documento lo puede descargar de:

http://pegasus.javeriana.edu.co/~CIS1010IS05/Documentos/Diseño/Reporte%20de%20pruebas.xlsx

Documentación del producto

- El propósito es describir el producto de software generado.
- Debe evolucionar de acuerdo al producto que describe.

La documentación del producto incluye:

- Documentación de usuario: indica a los usuarios cómo deben usar el producto de software.
- Documentación del sistema: destinada principalmente a Ingenieros de mantenimiento.

Documentación del sistema

- La documentación del sistema incluye todos los documentos que describen el sistema, desde la especificación de requisitos hasta el Plan final de pruebas.
- Los documentos que describen el diseño, la implementación y las pruebas de un sistema son esenciales si se busca que el software sea entendible y mantenido.
- La documentación debe estar estructurada, desde vistas generales hasta descripciones más formales y detalladas de cada aspecto del sistema.
- Hace una descripción general del diseño, análisis, implementación y pruebas del software. Incluye todos los documentos asociados a la especificación del sistema(requisitos), diseño, último plan de pruebas; de forma tal que se facilite el mantenimiento.
- Debe indicar relaciones y dependencias de archivos, con el fin de evitar posibles pérdidas de información o desconfiguración del sistema y se muestra en forma estructural, de lo general a lo particular.

Para sistemas grandes que se desarrollan según las especificaciones del cliente, el sistema Documentación debe incluir:

1. El documento de requisitos.
2. Un documento que describe la arquitectura del sistema.
3. Para cada programa en el sistema, una descripción de la arquitectura de ese programa.
4. Para cada componente del sistema, una descripción de su funcionalidad e interfaces.
5. Listado de programas del código fuente. Deben estas comentados, de tal forma que explique las secciones complejas de código y proporcionar una justificación para el método de codificación usado.
    - Si se utilizan nombres significativos y se utiliza un buen estilo de programación estructurada, gran parte del código debe auto documentación sin la necesidad de comentarios adicionales.

- Esta información normalmente se mantiene en formato digital. La impresión física se realiza de acuerdo a la solicitud del lector.

6. Documentos de validación. Describe cómo se realiza la validación de cada programa y cómo la información está relacionada con los requisitos. Sirve de soporte para procesos de aseguramiento de calidad en la organización.
7. Guía de mantenimiento del sistema. Describe los problemas conocidos con el sistema, dependencias de hardware y del software y como ha sido la evolución del sistema en cuanto a su diseño.

La siguiente imagen muestra un resumen de los diagramas utilizados comúnmente en la documentación del sistema.

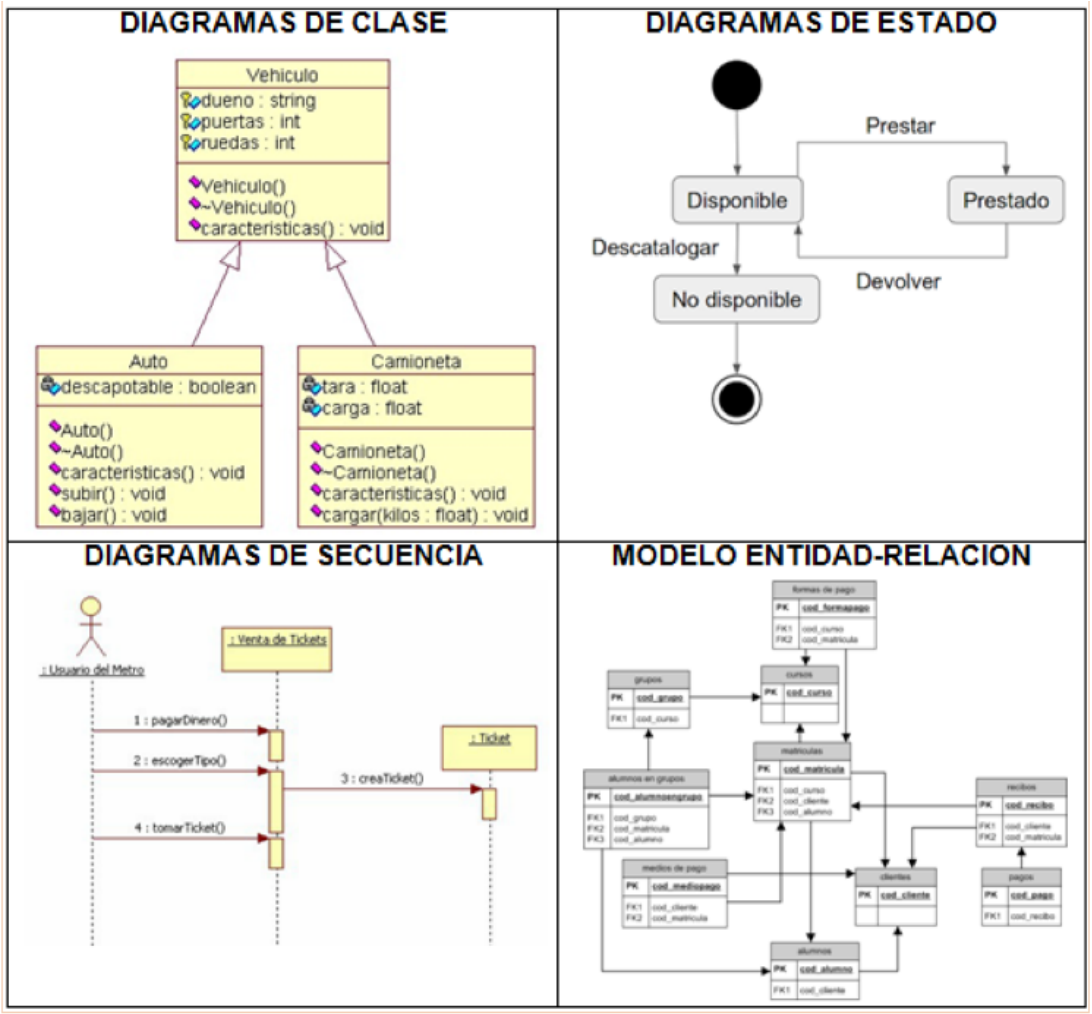


Imagen 3
Fuente: Propia.

La siguiente imagen muestra los tipos de documentos generados y sus respectivos lectores:

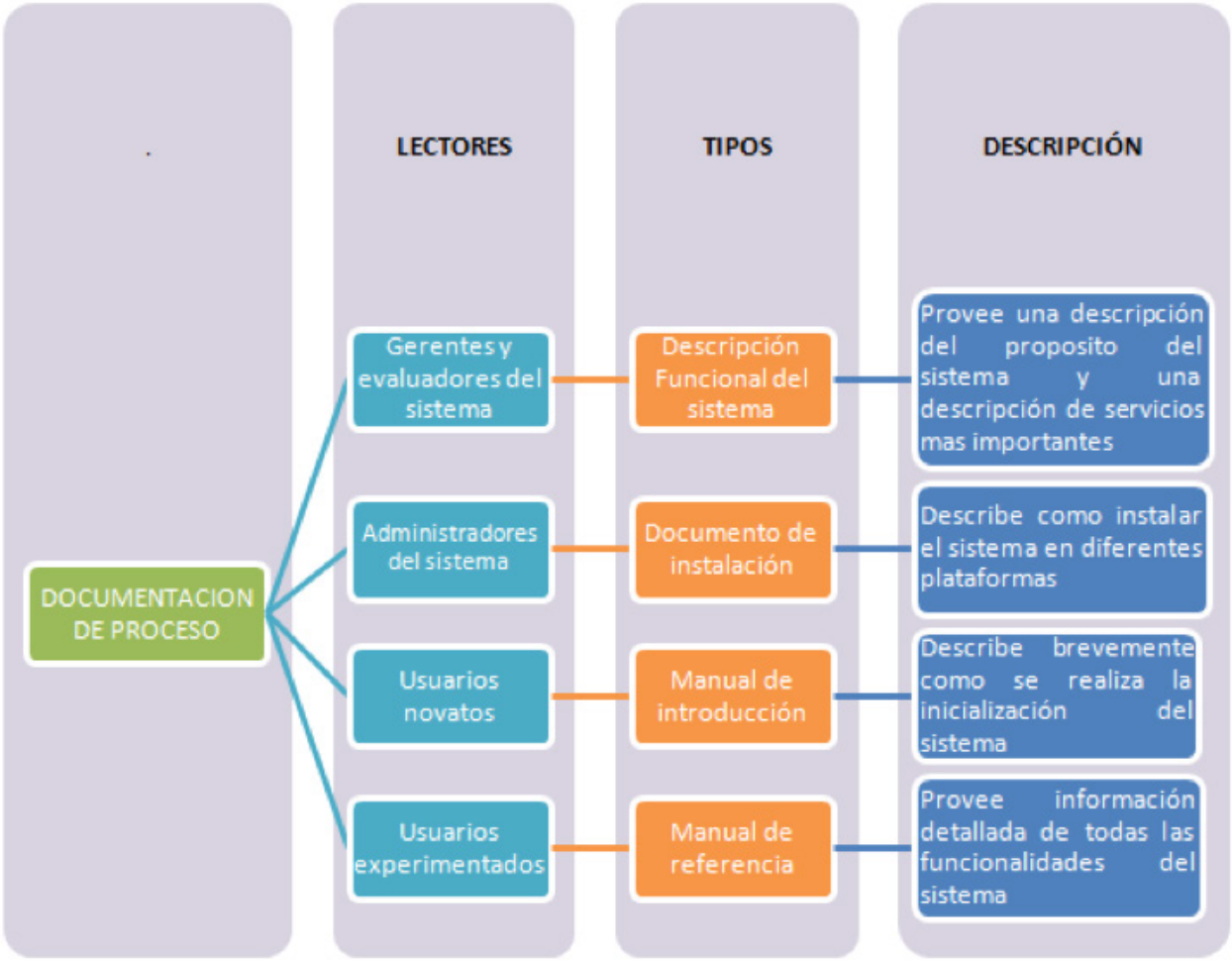


Imagen 4
Fuente: Propia.

Documentación del usuario

- Se orienta a las personas que harán uso del sistema e indica las funciones que puede realizar el sistema.
- Debe mostrar una visión clara y realista del sistema.
- Se estructura en detalle, según el estado del usuario, de forma tal que se puede acceder a un punto especifico, sin necesidad de leer toda la documentación.
- Los manuales pueden estar separados o unidos (cada uno marcado de acuerdo al volumen).

- Puede incluir documentación complementaria como tarjeta de referencia rápida, ayuda en línea.
- La documentación debe ser elaborada por el equipo de desarrollo y el documentalista.

La documentación debe estructurarse teniendo en cuenta las diferentes tareas de usuario y diferentes niveles de experiencia. Se debe distinguir entre los Usuarios finales y Administradores del sistema:

1. Los usuarios finales utilizan el software para ayudar con alguna tarea y quieren saber como el software puede ayudarlos. No les interesan los detalles de la computación o de la administración.
2. Los administradores del sistema son responsables de software utilizado por los usuarios finales. Algunos de los posibles roles son:
   - Operador sin el sistema es de tipo mainframe.
   - Gestor de red es el sistema implica un Red de estaciones de trabajo.
   - Gurú técnico que corrige a los usuarios finales, respecto a problemas de software.
   - Quién hace la conexión entre los usuarios y el software proveedor.

A continuación, se relacionan los tipos de documentos destinados al usuario:

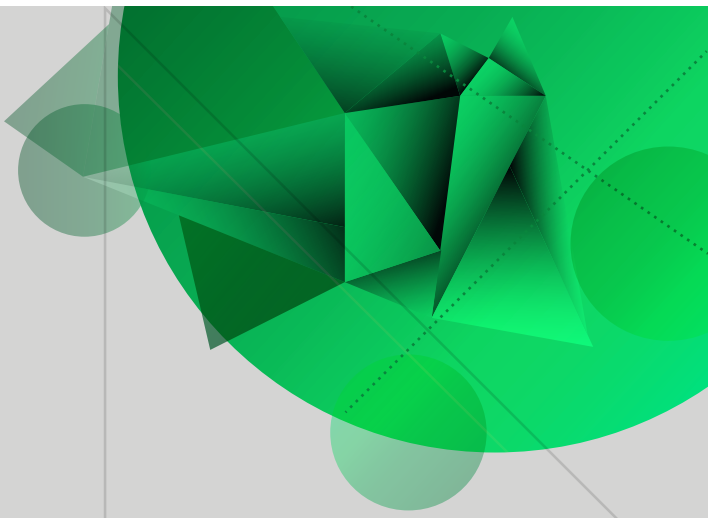
Tipo documentación	Descripción	Contenido
Descripción funcional	Descripción acerca de lo que el sistema puede realizar.	• Hace una relación de los requisitos. • Describir claramente y en forma general las funcionalidades y sus alcances. • Muestra ejemplos de uso. • Permite que el usuario decida si el sistema es apropiado o no, de acuerdo a sus necesidades.
Manual de instalación	Documento de explicación acerca del proceso de instalación y configuraciones de hardware.	• Brinda los detalles que permiten la instalación en diferentes entornos. • Describe el formato del código, los conjuntos de caracteres usados, archivos y el modo en que se muestra la información. • Relaciona la configuración mínima de hardware que requiere el sistema para funcionar. • Contiene la lista de archivos que adicionalmente se instalaran. • Debe mostrar como que hace la inicialización del sistema y los cambios que se puedan requerir en los archivos relacionados directamente con la configuración.
Manual de introducción	Realiza una introducción al uso del sistema.	• Describir el uso "normal" del sistema. • Explicar cómo se inicia el trabajo en el sistema. • Explicar las funcionalidades más comunes. • Mostrar ejemplos de uso. • Describir cómo solucionar problemas comunes.
Manual de referencia	Indica las ventajas para el usuario y como puede aprovecharlas.	• Documento final acerca del uso completo del sistema. • Debe usarse técnicas formales para su descripción. • Debe permitir al usuario comprender los conceptos y terminología utilizada del sistema. • Deberá mostrar situaciones de error e informes generados (Logs).
Guía de operación o guía del operador	Indica la forma de contrarrestar situaciones que puedan surgir durante el uso del sistema.	• Se elabora solo si se requiere de un operador. • Explicar los mensajes que se visualizan en la consola del operador, así como las respuestas a cada uno de ellos. • Explicar cómo se ha de llevar a cabo el mantenimiento del hardware.

Tabla 3
Fuente: Propia.

3

Unidad 3

Estándares para documentación de software



Ingeniería de software II

Autor: Fredy León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

La base de todo proyecto de software tiene su fundamentación y apoyo en la documentación generada para que a partir de ahí se comience a realizar la construcción del producto solicitado por el cliente final. Esta empieza al tiempo que inicia el desarrollo, finalizando poco antes que se realice la entrega del programa o aplicación al cliente.

Por otro lado, una vez finalizado el producto de software y puesto a disposición del cliente, es necesario realizar una serie de acciones que permitan garantizar el buen funcionamiento tanto del mismo software, como del entorno donde se ejecute. Sin esto, cualquier desperfecto informático puede provocar graves problemas para la empresa, tales como perdida de información, costos adicionales, etc.

Teniendo en cuenta los puntos anteriores, en esta cartilla aprenderá acerca de los estándares y herramientas utilizadas para Documentación de software y finalmente se relacionan los principios y conceptos del mantenimiento de software para que usted como Ingeniero de sistemas, pueda adquirir las bases teóricas y habilidades necesarias para definir, realizar y gestionar actividades asociadas a la documentación y mantenimiento de sistemas de software.

U3

Metodología

El módulo se basa en la lectura individual de la presente cartilla y posterior desarrollo de actividades planteadas en la plataforma online.

Se espera una participación y acceso a plataforma a su consideración, como mejor pueda organizarse, para cubrir los objetivos de aprendizaje y participación. Por mi experiencia, recomiendo la asistencia diaria para aprovechar al máximo el curso. Es interesante seguir los hilos de participación 2-3 veces al día y dedicar uno momento para analizar y participar inteligentemente.

La comunicación conmigo la pueden realizar abiertamente a través del Foro y personalmente a través del email.

Al final de cada sección enviaré un breve mensaje resumen destacando los mejores en sus intervenciones individuales y en los trabajos de grupo de esa sección.

Aquellos que hayan llamado mi atención negativamente por su falta de participación o por lo inadecuado de las mismas les enviaré un mensaje privado.

Estándares para documentación de software

Estándares
IEEE STD 1063-2001

Estándar para la documentación de usuario de software.

Brinda el marco de referencia para establecer qué partes deben conformar cualquier documento que deba ser utilizado por un usuario del sistema o programa en cuestión. Las partes a incluir son:

Imagen 1
Fuente: Propia.

ISO / IEC 26514:2008

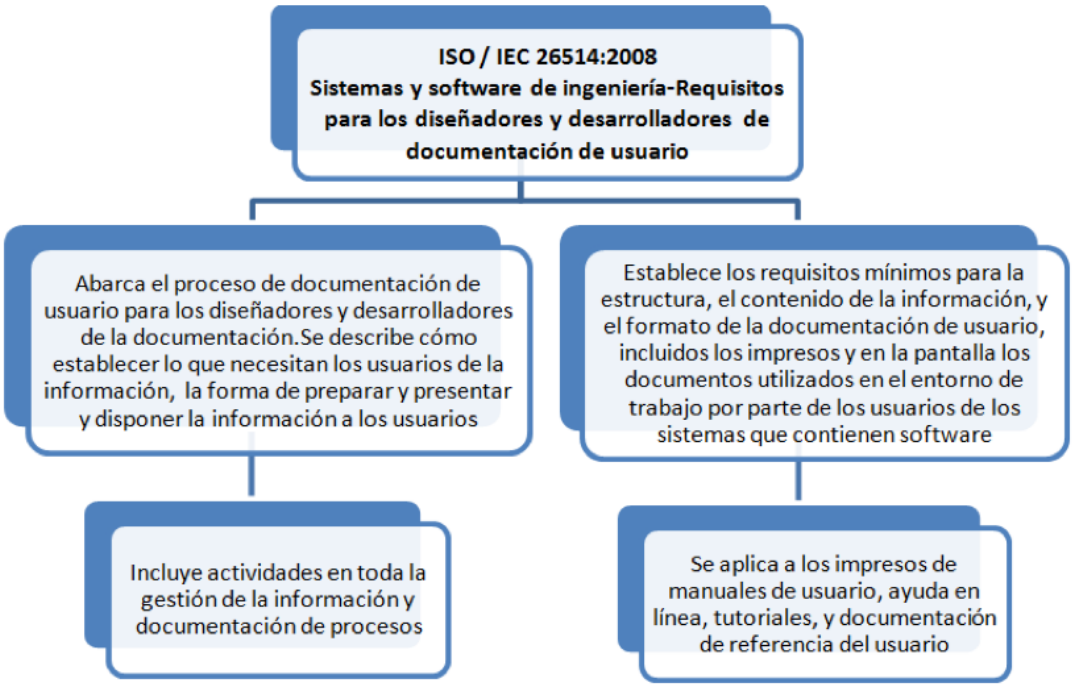


Imagen 2
Fuente: Propia.

Es especialmente útil para el desarrollo de los siguientes tipos de documentación:

- Documentación de diferentes productos de software.
- Contenidos Multimedia (uso de animación, vídeo y sonido).
- E-learning-Capacitación especializada (Materiales del curso).
- Documentación producida por administradores de sistemas, instaladores, operadores de equipo, que no son usuarios finales.
- Documentación de mantenimiento de los sistemas de software.
- Documentación incorporada en la misma interfaz de usuario.

ISO/IEC 18019:2004

- Software e ingeniería de sistema - Directrices para el diseño y preparación de documentación de usuario para software de aplicación.
- Describe la forma de establecer el requerimiento la información que necesitan los sus usuarios, la forma en que debe ser presentada, la forma de preparar y ponerla a disposición.

- Dirigido a organizaciones que cuenten con un departamento de documentación, responsable de especificar, diseñar y preparar la documentación del usuario para el software de aplicación y las personas que administran estas actividades, incluyendo desarrolladores de herramientas para crear documentación impresa, diseñadores de productos, desarrolladores de aplicaciones, Programadores, traductores y personal de localización.

Estándar IEEE-829

Estándar para documentación de pruebas de software.

El objetivo del estándar es proporcionar un conjunto estandarizado de documentos para la documentación de pruebas de software.

Existen 8 tipos de documento que pueden usarse en 3 etapas distintas de las pruebas de software:

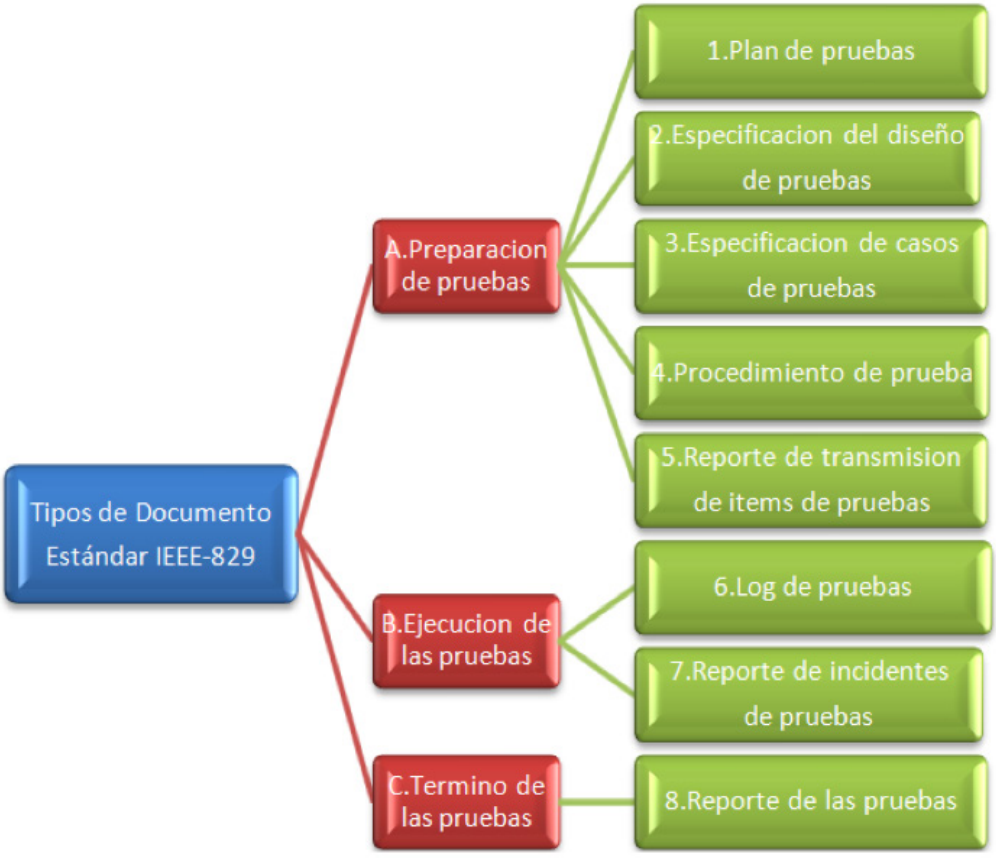


Imagen 3
Fuente: Propia.

Por favor, visite el siguiente link para ampliar información del estándar IEEE-829:

http://artemisa.unicauca.edu.co/~cardila/CS_08_Estandares_para_pruebas_software.pdf

IEEE 1016-2009

- IEEE Standard for Information Technology—Systems Design—Software Design Descriptions o Estándar IEEE para tecnología de la información-Diseño de sistemas-Descripciones de diseño de software.
- Especifica el contenido de información requerido y la organización para el documento de Diseño de Software (SDD).
- Aplica para bases de datos automatizadas y lenguajes de descripción de diseño, pero puede utilizarse para documentos en papel y otros medios de descripción.
- Define los siguientes Puntos de Vista de Diseño para su uso:
    - Contextual
    - Composición
    - Lógico
    - Dependencia
    - Información
    - Uso de patrones
    - Interfaz
    - Estructura
    - Interacción
    - Perspectiva dinámica del estado
    - Algoritmos
    - Recursos

Herramientas para documentación

Dokuwiki

- Su uso se enfoca a grupos de desarrolladores, grupos de trabajo en general y pequeñas empresas.
- Escrito en lenguaje de programación PHP y distribuido en código abierto bajo la licencia GPL.

- La información se almacena en archivos de texto planos (no requiere de base de datos).
- Creado por Andreas Göhr en junio de 2004.
- Para su instalación se requiere de un Servidor de páginas web con soporte de PHP(Apache),PHP versión 5.1.2 o superior con extensiones gráficas GD2 incluidas.
- Página Oficial: https://www.dokuwiki.org/es:dokuwiki

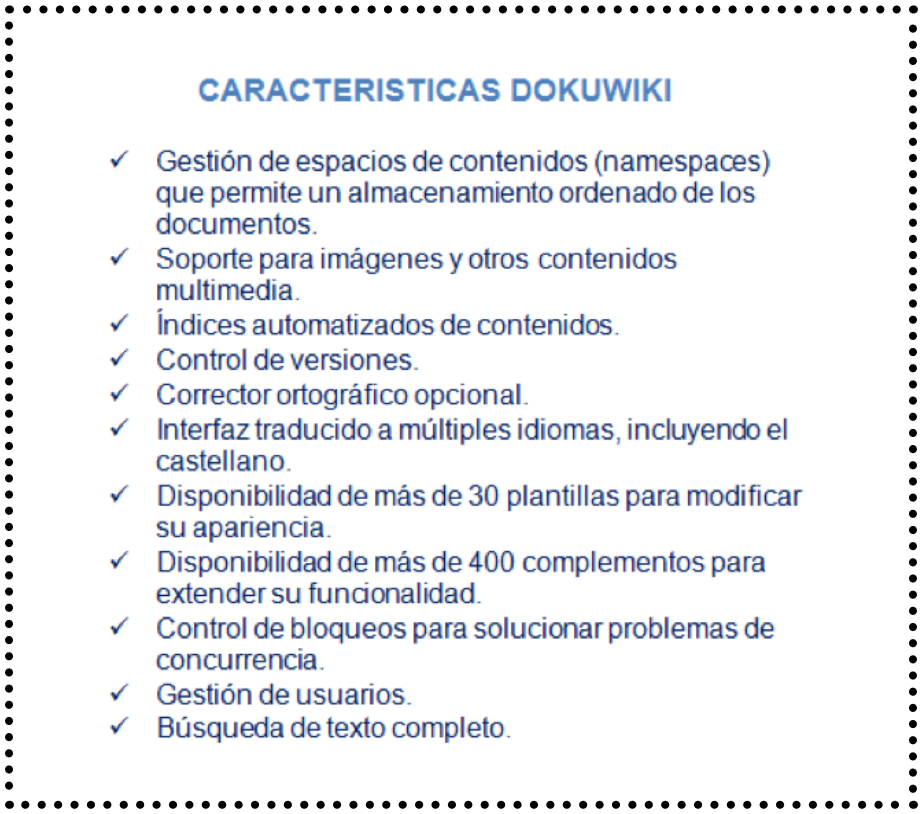


Imagen 4
Fuente: Propia.

Alfresco

- Plataforma de gestión de contenido empresarial (ECM) basada en Software Libre y arquitectura Cliente-Servidor.
- Permite crear repositorios de archivos y contenidos.
- Almacena toda la información en un mismo Sistema.
- Basado en servicios Cloud.
- Página oficial: https://www.alfresco.com/es/

A continuación se relacionan algunas de sus funcionalidades:

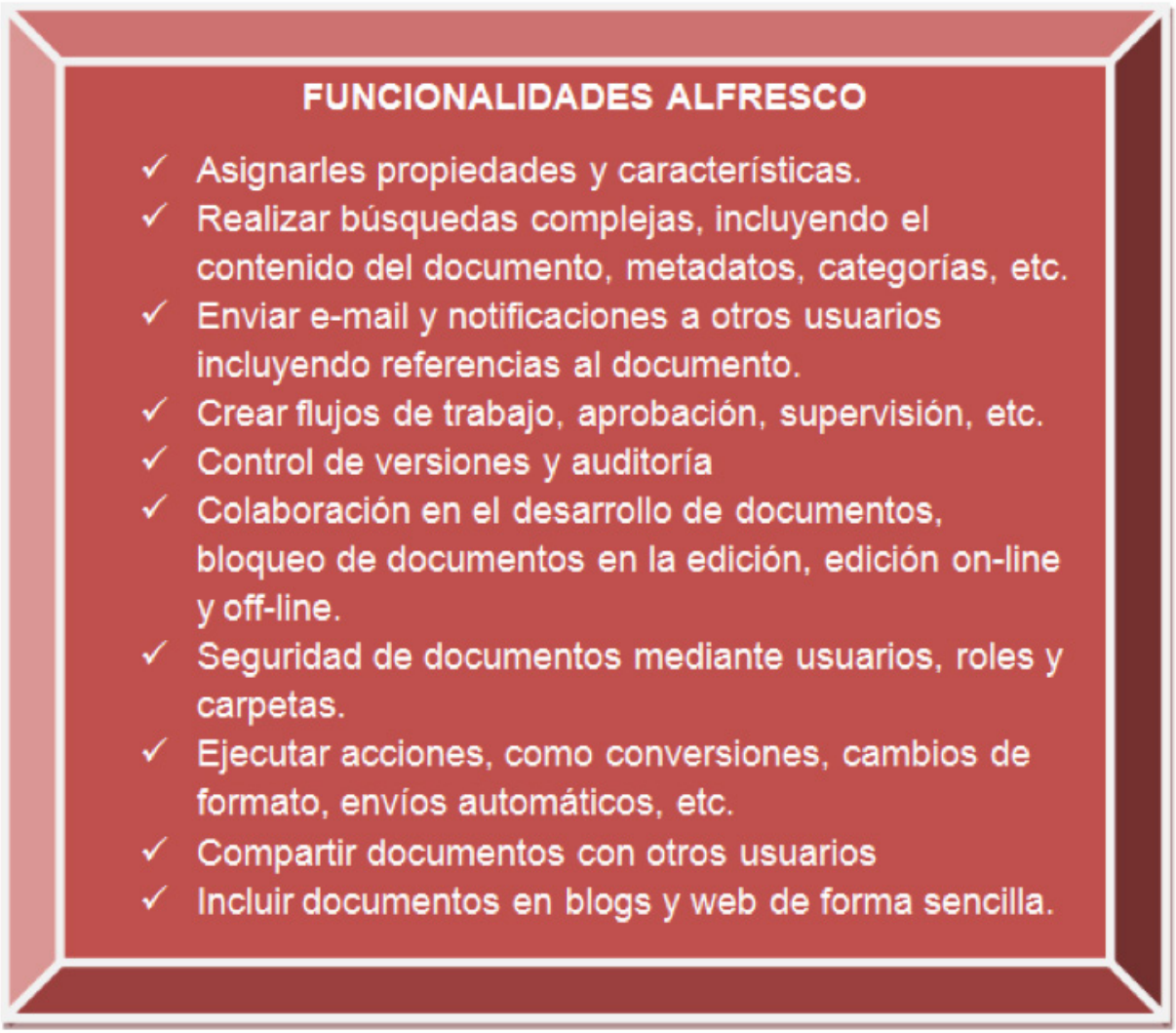


Imagen 5
Fuente: Propia.

Maneja 3 versiones: Alfresco One, Alfresco Cloud y Alfresco Community Edition. A continuación se relacionan sus principales características:

Imagen 6
Fuente: https://www.alfresco.com/es/products/enterprise-content-management

Gitbook

- Herramienta para crear documentación de proyectos, además de libros técnicos.
- Haciendo de Markdown, es posible realizar la maquetación de documentos y generar los archivos en formatos pdf, ebook o web.
- Haciendo uso de Git /Github es posible realizar la publicación del documento técnico y manejar las actualizaciones en forma transparente, para que sea fácilmente editable y abierta a nuevas contribuciones o actualizaciones.
- Permite crear las publicaciones digitales en Internet, haciendo uso de un sistema de control de versiones Git.
- Página oficial: https://www.gitbook.com/

Doxygen

Sus funcionalidades están enfocadas a:

- Generar documentación de código fuente para C++, C, Java, Objective-C, Python, IDL (versiones Corba y Microsoft), VHDL, así como para PHP, C# y D.
- Extraer la estructura del código fuente de un conjunto de ficheros diferentes al formato de Doxygen.
- Generar documentación a partir de un conjunto de ficheros de código que tienen el «estilo Doxygen.

- Página Oficial: http://www.stack.nl/~dimitri/doxygen/
- Soporta los siguientes formatos de salida:
  - HTML.
  - LATEX.
  - RTF.
  - Postscript.
  - PDF.
  - Unix man pages.
  - Windows help compressed HTML.
  - XML.

Dr Explain

Es un software que permite generar archivos de ayuda, manuales online, guías de usuario y documentación de aplicaciones.

La documentación se exporta en formatos HTML (manuales en línea), CHM (archivos de ayuda MS Windows®), RTF y PDF.

Ofrece una plataforma llamada TiWri , una plataforma en la nube enfocada a que varios usuarios puedan crear, editar y actualizar la documentación.

A continuación, se relacionan las características más importantes de Dr Explain:

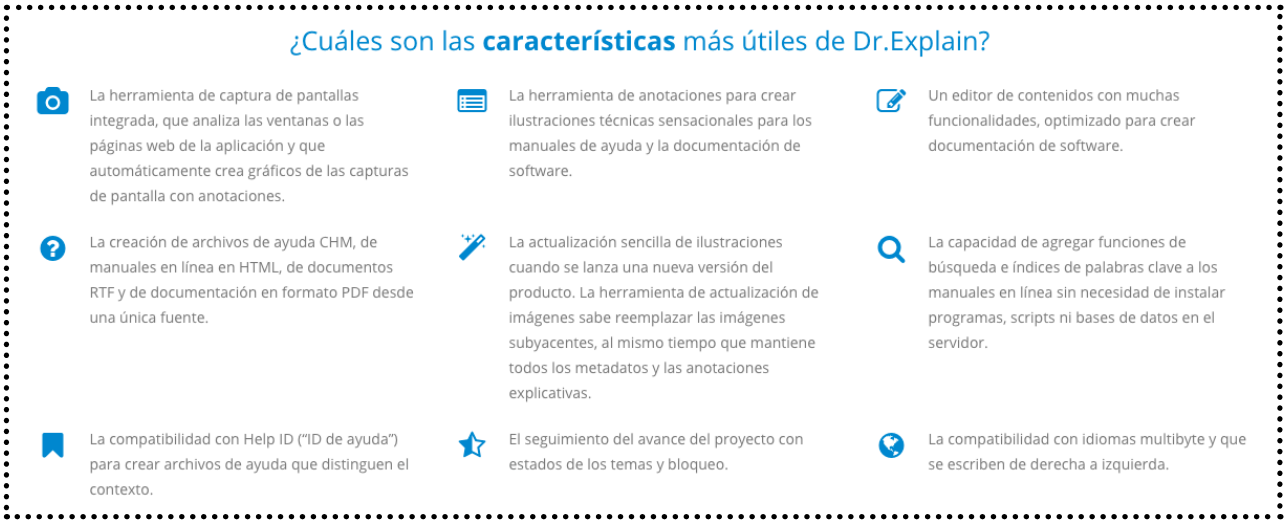


Imagen 7
Fuente: http://www.drexplain.es/

Mantenimiento de software

- En el proceso de mantenimiento de software se desarrollan actividades y tareas que son requeridas para poder modificar un producto de software, de tal forma que pueda mantenerse la integridad del mismo.
- Las tareas y actividades son desarrolladas por el encargado del mantenimiento, comúnmente llamado mantenedor.
- El proceso concluye cuando un producto software es retirado completamente.
- Las salidas son los datos u objetos generados por las actividades de mantenimiento.

Conceptos

Autor	Concepto
IEEE 1219	"la modificación de un producto software después de haber sido entregado [a los usuarios o clientes] con el fin de corregir defectos, mejorar el rendimiento u otros atributos, o adaptarlo a un cambio en el entorno".
ISO 12207	"El Proceso de Mantenimiento contiene las actividades y tareas realizadas por el mantenedor. Este proceso se activa cuando el producto software sufre modificaciones en el código y la documentación asociada, debido a un problema o a la necesidad de mejora o adaptación. El objetivo es modificar el producto software existente preservando su integridad. Este proceso incluye la migración y retirada del producto software. El proceso termina con la retirada del producto software".
Pressman [1998]	"la fase mantenimiento se centra en el cambio que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software, y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente".

Tabla 1
Fuente: Propia.

Tipos

Existen 4 tipos principales de mantenimiento de software, cada uno se enfoca en tareas especificas sobre el sistema y se generan de acuerdo a los actores que interactúan con el sistema. Estos son: Corrrectivo, Adaptativo, Perfectivo y Preventivo. El siguiente diagrama refleja la relación de cada uno sobre el sistema.

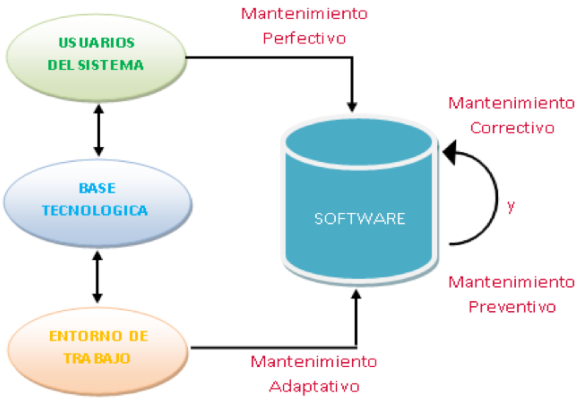


Imagen 8
Fuente: Propia.

Mantenimiento correctivo

Se enfoca en la localización y eliminación de características del sistema que tienen potencial de causar fallos en el producto de software. Los fallos se generan cuando el sistema tiene un comportamiento diferente al que se estableció en la definición de requerimientos. A continuación se describe los principales tipos de fallos en un producto de software:

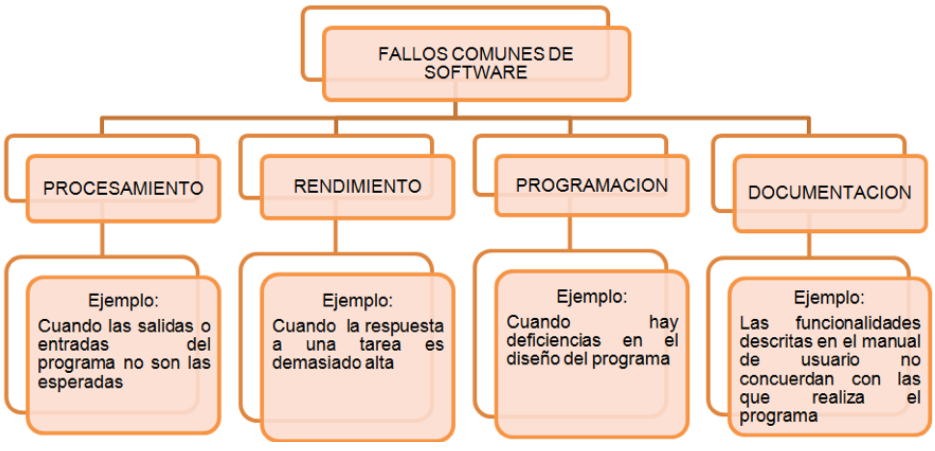


Imagen 9
Fuente: Propia.

Mantenimiento adaptativo

Son modificaciones que se le deben realizar al programa, debido a cambios que ocurren en el entorno del software (hardware o software sobre el cual se ejecuta). Por ejemplo; cambio del sistema operativo, cambio de una arquitectura de red LAN a Cloud, incorporar herramientas ODBC al entorno de desarrollo del software, etc. Hay 2 clases principales de cambios en el entorno del software:

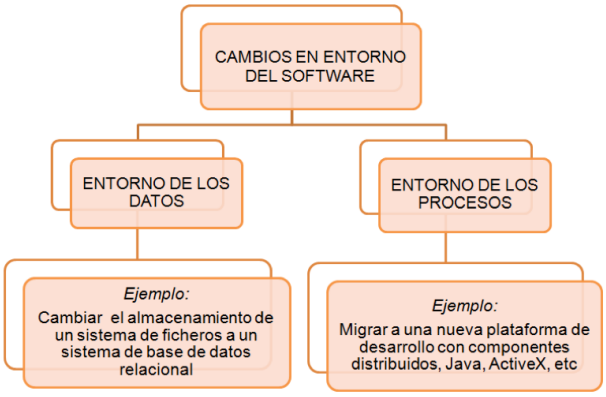


Imagen 10
Fuente: Propia.

Mantenimiento perfectivo

Son actividades enfocadas en el mejoramiento e implementación de nuevas funcionalidades al producto de software, debido a nuevos cambios en los requisitos de software. El mantenimiento perfectivo puede ser de Ampliación y de Eficiencia:

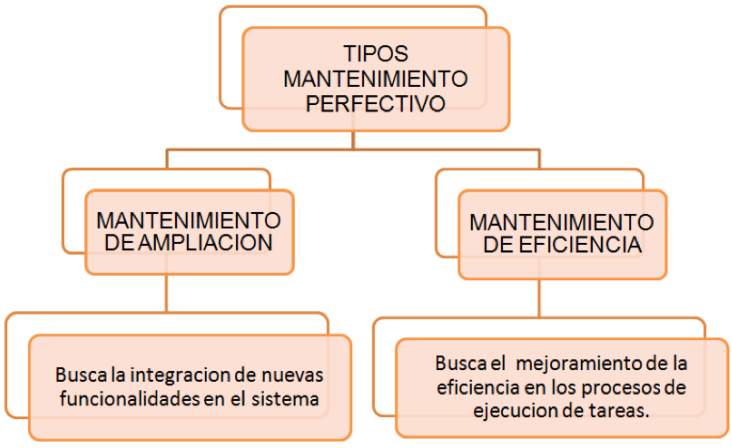


Imagen 11
Fuente: Propia.

Mantenimiento preventivo

Se enfoca en facilitar el mantenimiento del software. Algunas de las tareas comunes son:

- Realizar verificaciones de las precondiciones de utilización.
- Adecuar y reestructurar el código para que sea más legible.
- Identificar posibles problemas y fallos.
- Validación de datos de entrada.
- Agregar comentarios para mejorar la comprensión futura.

Tipos de actividades

Actividad	Descripción
Análisis de impacto y de costes/beneficios	Analizar diferentes alternativas de implementación y/o a comprobar su impacto en la planificación, coste y facilidad de operación.
Comprensión del cambio	Identificar errores y causas o realizar análisis de requisitos solicitados para mejora de funcionalidades.
Diseño del cambio	Rediseño del sistema.
Codificación y pruebas unitarias	Codificar y probar el funcionamiento de los componentes que han sido modificados.
Inspección, certificación y consultoría	Realizar una inspección de los cambios ,comprobación de diseños, programar reuniones para inspecciones, etc.
Pruebas de integración	Realizar comprobaciones respecto a cómo se integran con el sistema, los componentes que han sido modificados.
Pruebas de aceptación	En compañía del equipo de mantenimiento, los usuarios hacen comprobaciones entre los cambios realizados y sus necesidades
Pruebas de regresión	Volver a realizarle pruebas que ya han sido validadas sobre el software.
Documentación del sistema, usuario, etc.	Analizar, revisar y reescribir la documentación con el fin que se ajuste a las nuevas modificaciones realizadas en el software.

Tabla 2
Fuente: Propia.

El plan de mantenimiento de software

Los debe preparar el mantenedor mientras se encuentre activa la etapa de desarrollo y en términos generales debe incluir:

- La justificación del porque debe realizarse el mantenimiento.
- Quienes serán los responsables de las actividades, así como sus roles y tareas.
- Que recursos tecnológicos, físicos, logísticos se requieren.
- En donde se debe realizar.
- Cronograma de actividades (inicio, finalización, fechas, etc.).

A continuación se describen las secciones principales que componen un Plan de Mantenimiento de Software:

Sección	Descripción
Introducción	• Descripción general del sistema al cual se le realizara el mantenimiento. • Identificar el estado inicial del software. • Argumentar las razones por las cuales se debe realizar el mantenimiento. • Identificación de la organización encargada del proceso. • Descripción de los protocolos acordados entre el cliente y el desarrollador del software.
Concepto de mantenimiento	Describir el concepto, nivel de soporte y cronograma del mantenimiento.
Organización y actividades	Antes de la entrega: • Implementar el proceso. • Establecer la infraestructura logística, tecnológica, etc. • Establecer los procesos requeridos de formación. • Establecer el Proceso de Mantenimiento. Después de la entrega: • Implementar el proceso. • Analizar los problemas y modificaciones. • Realizar las modificaciones. • Implementar los procesos de revisión y aceptación del mantenimiento. • Migraciones. • Resolver problemas (soporte online). • Realizar proceso de formación para usuarios y quienes mantienen el sistema. • Mejoramiento del proceso. Actividades del usuario • Realizar las pruebas de aceptación. • Referencia con otras organizaciones.
Recursos	Personal • Tamaño del equipo del proyecto. Software • Identificar el software requerido para el proceso. Hardware • Identificar el hardware requerido para el proceso. Instalaciones • Identificar los requerimientos de infraestructura. Documentación • Plan de calidad del software. • Plan de gestión del proyecto. • Plan de gestión de la configuración. • Documentos del desarrollo. • Manuales de mantenimiento. • Plan de verificación. • Plan de validación. • Plan de pruebas, procedimientos e informes de pruebas. • Plan de formación. • Manuales de usuario. Datos • Otros requerimientos de recursos (si los hubiera)
Proceso	La forma en que se realizará el trabajo. Esto incluye un resumen del proceso que realiza el encargado del mantenimiento y un proceso personalizado.
Formación	Identificar necesidades de formación que se requieren para los encargados del mantenimiento y para los usuarios.
Registro e informes	• Listado de solicitudes de modificación, ayuda o informes de problemas encontrados. • Organizar por categorías las solicitudes realizadas, indicando el estado actual y priorización de las mismas. • Métricas recogidas durante el proceso de mantenimiento.

Tabla 3
Fuente: Propia.

Estándares utilizados

ISO 14764

- Define un marco de referencia para planificar, ejecutar, controlar, revisar evaluar, adaptar y cerrar planes de mantenimiento.
- Provee los conceptos, terminologías y procedimientos para la implementación de técnicas, métodos y herramientas en el mantenimiento de software.
- Hace una definición de las tareas, actividades y requerimientos para planificar el mantenimiento de software.
- Define los distintos tipos de mantenimiento de software.

El estándar ISO 14764 propone el establecimiento de una guía que permita el desarrollo del mantenimiento, siguiendo los requisitos relacionados a continuación:

- Describir detalladamente el sistema que recibirá el mantenimiento.
- Identificar cual es el estado actual del software e identificar cuáles serán los cambios que deben realizarse.
- Describir la forma en que será llevado a cabo el proceso de mantenimiento.
- Identificar la organización encargada de hacer el soporte técnico o mantenimiento y de esta forma definir cuál es el objetivo del mismo.
- Describir y dejar por escrito los acuerdos generados entre el vendedor del software y el cliente final, indicando con claridad las condiciones, modificaciones e información adicional que se requiera.

El estándar también propone las siguientes etapas que son fundamentales para la implementación del proceso de mantenimiento:

1. Implementación del proceso.
2. Análisis de modificaciones y problemas.
3. Implementación de modificaciones.
4. Revisión y aceptación del mantenimiento.
5. Migración.
6. Retiro.

La siguiente imagen muestra como es la relación entre las diferentes etapas:

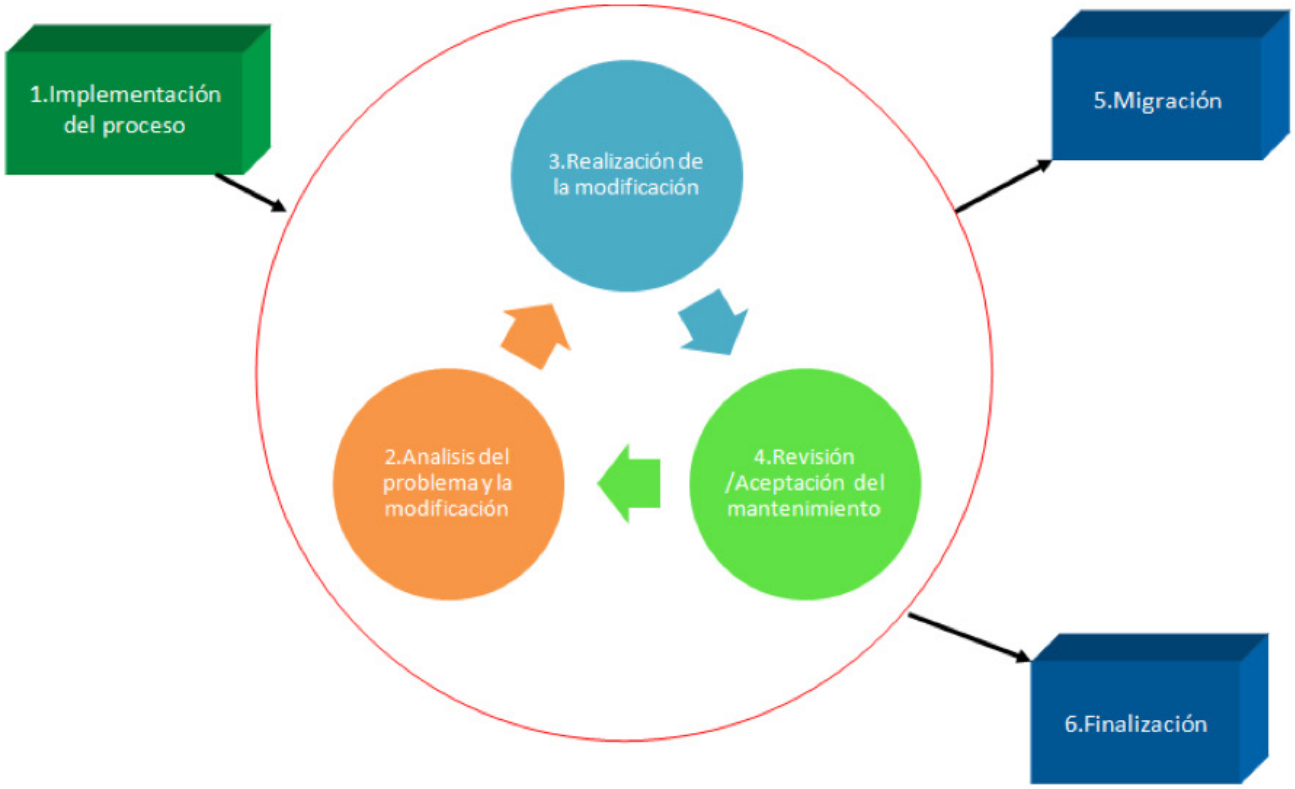


Imagen 12
Fuente: Propia.

Cada una de las 6 etapas incluye una serie de actividades que debe realizar el mantenedor, las cuales se explican a continuación:

Fase	Actividades del mantenedor
Implementación del proceso	• Establecer el plan y procedimientos de mantenimiento. • Establecer los procedimientos requeridos para recepcionar, registrar y hacer un seguimiento respecto a informes de problemas y solicitudes de modificaciones realizadas por los usuarios. • Implementar y/o definir cómo será la vinculación de la organización con el proceso de gestión de la configuración.
Análisis del problema y la modificación	• Realizar un análisis del problema o la modificación solicitada con el fin de establecer que impacto generará en la organización, sobre el sistema e interfaces asociadas al mismo. • Socializa el problema o modificación con el equipo de mantenimiento y la organización. • Definir las diferentes alternativas que permitan la implementación de la modificación solicitada. • Realizar la documentación e informe acerca del problema o modificación, resultados generados, alternativas de implementación identificadas. • Finalmente se obtiene el aval que permite continuar a la etapa de Realización de la modificación.
Realización de la modificación	• Determinar qué elementos del software requieren ser modificados. • Solicitar al equipo de desarrollo la modificación solicitada y las pruebas requeridas después de la modificación.
Revisión y aceptación del mantenimiento	• Socializar con el usuario o cliente final, la modificación realizada con el fin de verificar la integralidad. • Se obtiene la aprobación por parte del usuario o cliente final, soportándola mediante un documento o contrato de aceptación entre las partes.
Migración	Esta etapa se incluye únicamente cuando la modificación implique el funcionamiento en un nuevo entorno operativo. • Diseñar el plan que registra la migración. • Realizar las notificaciones a los usuarios o clientes finales respecto al inicio y finalización de la migración. • Realizar proceso de capacitación a los usuarios respecto al funcionamiento del sistema en el nuevo entorno operativo. • Realizar una evaluación de los impactos generados por el nuevo entorno operativo. • Archivar el antiguo producto de software.
Retirada	Esta etapa solo existirá si el producto de software ha concluido su vida útil y ha sido reemplazado por otro nuevo. Las actividades a realizar son similares al de la etapa de Migración.

Tabla 4
Fuente: Propia.

IEEE 1219

- Standard for Software Maintenance.
- Hace una descripción del proceso para gestionar y ejecutar las actividades del mantenimiento.

El estándar IEEE1219 hace una definición de los cambios en un producto de software, mediante una serie de fases realizadas iterativamente y en cascada, las cuales se relacionan a continuación:

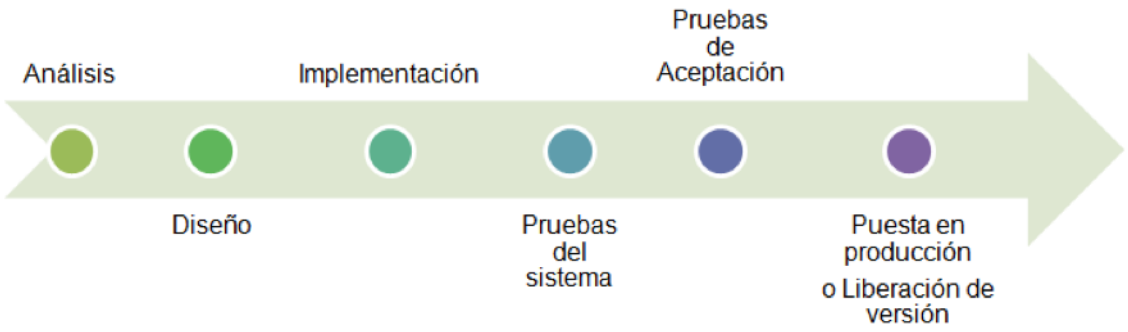


Imagen 13
Fuente: Propia.

4 Unidad 4

Software seguro

Ingeniería de software II

Autor: Fredy León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

En los proyectos de software, es común que durante su desarrollo se encuentren muchos problemas o imprevistos, como retrasos en los plazos de entrega, sobre costos no estipulados, personal no apto para el desarrollo, lo cual genera productos de mala calidad o que no cumplan las expectativas del cliente.

Aunque dichos problemas o imprevistos no se pueden eliminar en su totalidad, si es posible realizar controlarlos, mitigarlos o minimizar los efectos adversos; haciendo uso de técnicas de gestión y análisis de riesgos. Esto permite aumentar la probabilidad e impacto de eventos positivos, disminuir la probabilidad e impacto de eventos negativos para el proyecto, mejorar los niveles de motivación del equipo de trabajo y lograr un mayor índice de satisfacción laboral, ya que se minimizarán las presiones o trabajo extra y se reducirán los niveles de estrés.

En este modulo adquirirá los conocimientos que le permitan establecer un plan para la gestión de riesgos en un proyecto de software, las fases que componen dicho plan, métodos para identificación de riesgos, clasificación de riesgos, entre otros aspectos.

U4 Metodología

El módulo se basa en la lectura individual de la presente cartilla y posterior desarrollo de actividades planteadas en la plataforma online.

Se espera una participación y acceso a plataforma a su consideración, como mejor pueda organizarse, para cubrir los objetivos de aprendizaje y participación. Por mi experiencia, recomiendo la asistencia diaria para aprovechar al máximo el curso.

La comunicación conmigo la pueden realizar abiertamente a través del Foro y personalmente a través del email.

Se realizará una teleconferencia semanal de un tema especifico, en el cual se presentarán ejemplos para ampliar la temática semanal.

En cada semana se publica el material de estudio que apoyara el desarrollo de los contenidos temáticos correspondientes a dicha semana, ya sea en recursos bibliográficos, archivos digitales o referencias electrónicas (páginas WEB).

U4 Desarrollo temático

Software seguro

Conceptos básicos

Bien informático	Elementos que componen un sistema informático.
Sistema informático	Conjunto de bienes informáticos que ejecutan unas tareas específicas.
Amenaza	Posible riesgo que puede causar daño a un bien informático.
Riesgo	Probabilidad que una amenaza se materialice.
Análisis de riesgo	Proceso que permite determinar vulnerabilidades de un sistema, las posibles amenazas y su probabilidad de ocurrencia e impacto. Es conocido como Evaluación de Riesgo o PHA por sus siglas en inglés: Process Hazards Analysis. Es un método que permite recopilar, evaluar, registrar y difundir toda información que permita una formulación de recomendaciones enfocadas a adoptar medidas frente a una amenaza determinada. Estudia las causas de las posibles amenazas, así como los daños y consecuencias que éstas puedan generar.
Riesgo residual	Riesgo después de haber implementado controles de seguridad para minimizarlo.
Impacto	Daño producido o causado por una amenaza.
Seguridad	Minimizar los riesgos de un bien informático a niveles adecuados.
Vulnerabilidad	Debilidades o aspectos atacables de un sistema informático y que permiten calificar el nivel de riesgo.
Seguridad en aplicaciones	Es el uso de principios y buenas prácticas de seguridad durante el ciclo de desarrollo de software.
Perdida	Si el riesgo se convierte en una realidad, ocurrirán consecuencias no deseadas o pérdidas.

Gestión de riesgo	Busca identificar, dirigir y eliminar las fuentes de riesgo para que esto no afecte la adecuada finalización de un proyecto software. Esta gestión continuada conduce al mejoramiento de la eficiencia, lo que incluye una evaluación continua de lo que pueda ir mal, determinar la relevancia de los riesgos, la implementación de estrategias para controlarlos y garantizar que dichas estrategias sean eficientes.
Incertidumbre	El acontecimiento que caracteriza al riesgo puede o no puede ocurrir.
Riesgo técnico en software	La probabilidad de generarse efectos negativos asociados al desarrollo de software a raíz del incumplimiento de los requerimientos funcionales y/o no funcionales.

Tabla 1
Fuente: Propia.

Clasificación de riesgos

En función de lo que afecta:

Riesgos del proyecto	Afectan el proceso de desarrollo, planificación temporal costos y calidad del proyecto. Permite Identificar aquellos posibles problemas respecto a presupuesto, cronograma, recursos humanos y logísticos, cliente, etc. Por ejemplo, que el líder o algún integrante del proyecto renuncie.
Riesgos del producto	Afectan directamente la calidad del software que se va a desarrollar. Permiten identificar problemas potenciales en cuanto inseguridad técnica, especificaciones, diseños, implementación, técnica o tecnología obsoleta, interfaz grafica, verificabilidad y mantenimiento, etc. Por ejemplo, adquirir un plugin de cual no se conoce su desempeño o no se conoce la compatibilidad con el software sobre el cual va a funcionar.
Riesgos del negocio	Afectan directamente a la organización que lidera el desarrollo del proyecto de software. Por ejemplo, al cambiar un directivo que tiene otras prioridades u otro enfoque del proyecto. Los principales riesgos de negocio son: • Riesgo de mercado: producto con demasiada calidad. • Riesgo estratégico: producto que no se ajusta a las necesidades. • Riesgo de ventas: producto con pocos clientes. • Riesgo de presupuesto: producto que no está acorde al presupuesto.

Tabla 2
Fuente: Propia.

En función de su facilidad de detección:

Riesgos conocidos	Si después de evaluar el plan de proyecto, el área técnica y otras áreas del proyecto ,los riesgos pueden ser predecidos.
Riesgos predecibles	Son extraídos de experiencias en anteriores proyectos o proyectos similares.
Riesgos impredecibles	Aquellos que son difíciles de identificar con anticipación.

Tabla 3
Fuente: Propia.

Proceso de Gestión de Riesgos

Describe las responsabilidades y actividades relacionadas con la Gestión

de Riesgos y define los siguientes aspectos:

- Organigrama para la gestión de riesgos.
- Proceso para identificar y analizar riesgos.
- La técnicas y herramientas a utilizar.
- La taxonomía de riesgos.
- Registro de riesgos para su identificación y gestión, utilizando plantillas previamente estandarizadas.
- Actividades y frecuencia de control de riesgos.

El proceso de gestión de riesgos se compone de las siguientes etapas:

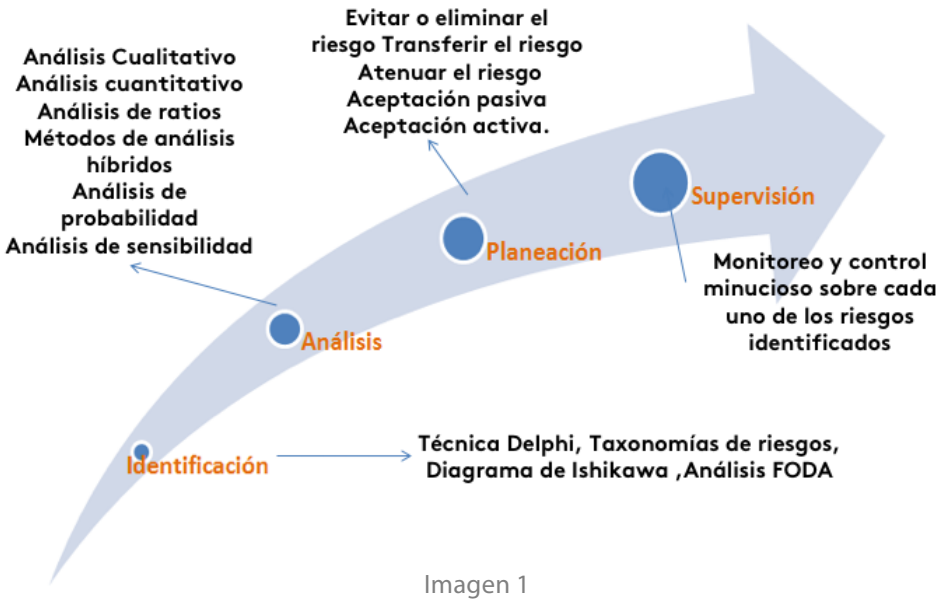


Imagen 1
Fuente: Propia.

Identificación del riesgo

En la etapa de Identificación del riesgo es de vital importancia la experiencia y punto de vista del personal encargado de la identificación. Generalmente se crea una lista de comprobación de elementos de riesgo, los cuales son categorizados de la siguiente forma:

Riesgos específicos del producto: se hace una exanimación profunda del plan de proyecto y del entorno del mismo con el fin de identificarlos.

Riesgos genéricos: Son los que comúnmente se generan en todos los proyectos de software. Su identificación se realiza bajo estas características:

- Tamaño del producto.
- Impacto en el negocio.
- Características del cliente.
- Definición del proceso.
- Entorno de desarrollo.
- Tecnología a construir.
- Tamaño y experiencia del personal.

De acuerdo al análisis de las categorías anteriores, se pueden identificar tipos de riesgos mucho más específicos, dentro de los que están:

- Riesgos tecnológicos: estos se derivan del tipo de hardware y software que se utilizan.
- Riesgos personales: directamente relacionados con el capital humano que se encuentra vinculado al proyecto de software.
- Riesgos organizacionales: de acuerdo a la organización. que lidera el proyecto.
- Riesgos de herramientas: se generan a partir del tipo de herramientas de software que son utilizadas en el desarrollo del proyecto.
- Riesgos de requerimientos: surgen de los cambios en los requerimientos.
- Riesgos de estimación: proceden de las proyecciones de costos y recursos.

Algunas técnicas utilizadas en la identificación de riesgos son:

Juicio Experto: Técnica Delphi. Una técnica prospectiva que se usa para con el propósito de realizar pronósticos y predicciones. La información obtenida es principalmente de tipo cualitativa.

Taxonomías de riesgos. La Estructura de Desglose de Riesgo(RBS) especifica en forma jerárquica las diferentes fuentes de donde podrían surgir riesgos para un determinado proyecto. Cada proyecto tiene su propia RBS.

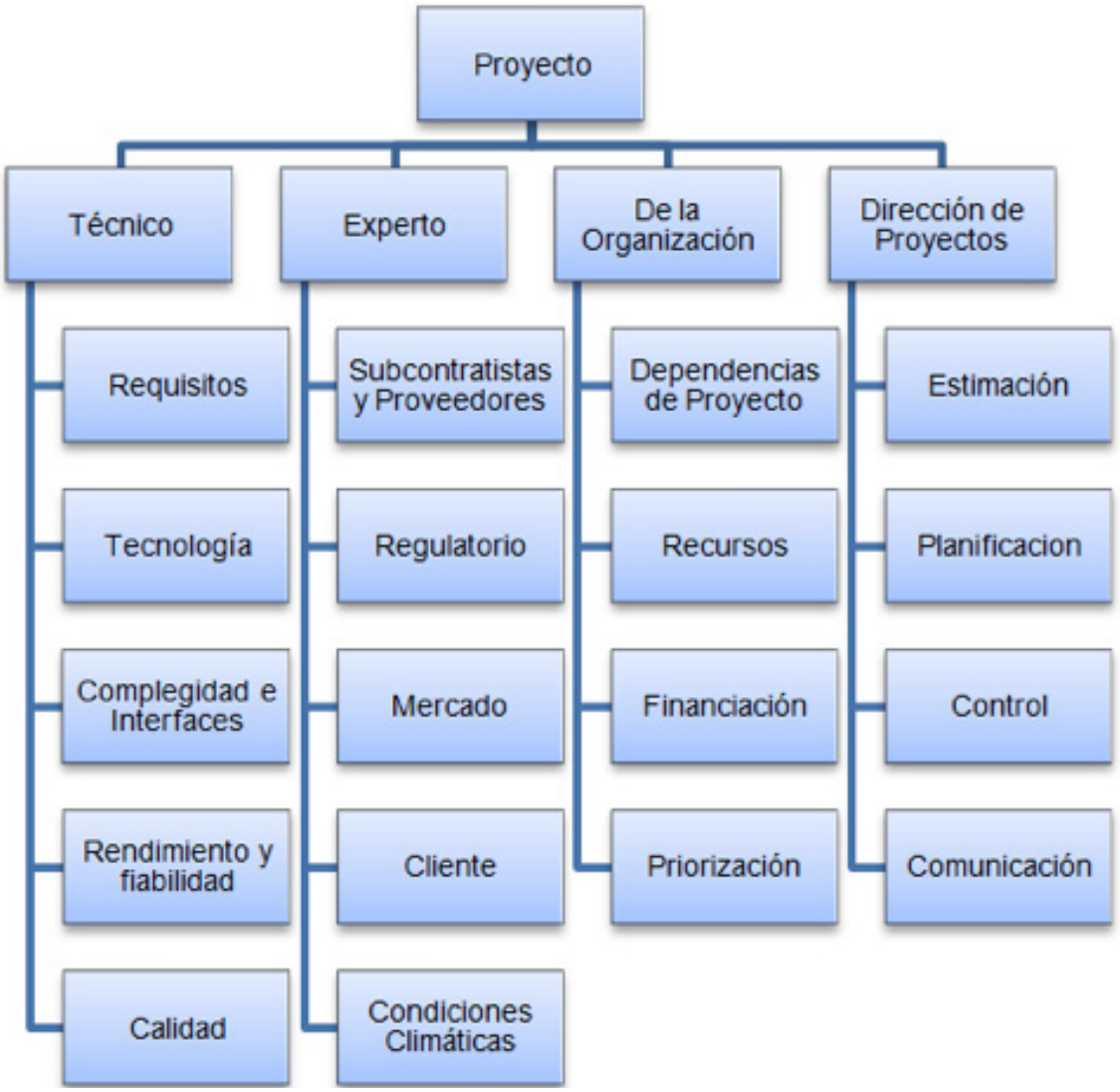


Imagen 2
Fuente: Propia.

Análisis SWOT o FODA. (Strengths, Weakneses, Oportunities, Threatens) o FODA (Fortalezas, Oportunidades, Debilidades y Amenazas). Permite una identificación de fortalezas, debilidades, oportunidades y amenazas relacionadas con el proyecto, la situación, el contexto o programa determinado.

Diagrama de Ishikawa. Más conocido como Diagrama de Causa Efecto o Diagrama de Espina de Pescado. Es una representación gráfica para visualizar las causas que explican un determinado problema, lo que permite orientar la toma de decisiones.

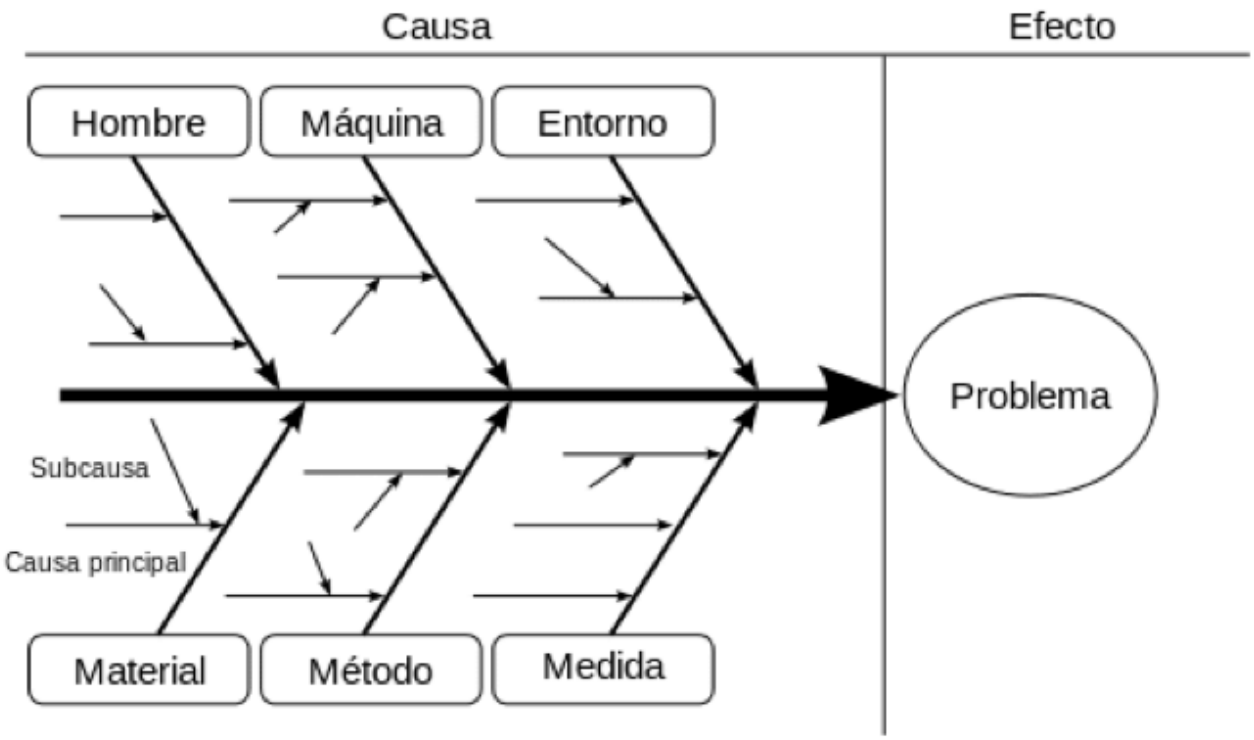


Imagen 3
Fuente:
http://www.gestiondeoperaciones.net/wp-content/uploads/2014/12/diagrama-causa-efecto.png

Adicionalmente existen otras técnicas que también son utilizadas en la fase de identificación de riesgos:

- Revisión de la documentación existente.
- Revisión, planificación y estimaciones.
- Tormenta de ideas. Todos los integrantes del equipo de trabajo hacen su aporte.

Análisis de Riesgos

En esta etapa se examinan detalladamente los riesgos con el objetivo de determinar el impacto, probabilidad de ocurrencia y el periodo de tiempo en el que se puede controlar o disminuir el riesgo. La siguiente tabla muestra cómo podría realizarse un análisis de los riesgos, de acuerdo a los puntos mencionados:

ATRIBUTO	VALOR	DESCRIPCIÓN
Impacto	Insignificantes	No merecen ser tenidos en cuenta.
	Tolerables	Están dentro de un margen de aceptación, por lo cual no comprometen ni el proyecto, ni el producto, ni la organización.
	Graves	Comprometen gravemente el proyecto o el producto o la organización.
	Catastrófica	Amenazan la supervivencia del proyecto o del producto o de la organización.
Probabilidad	Muy baja	<10%
	Baja	del 10 al 25%
	Moderada	del 25 al 50%
	Alta	del 50 al 75%
	Muy alta	>75%
Marco de tiempo	Corto plazo	30 días
	Medio plazo	1 a 4 meses
	Largo plazo	Más de 4 meses

Imagen 4
Fuente: Propia.

Lo que sigue es implementar alguna técnica de análisis la cual se aplicara a los riesgos identificados. Algunas de las técnicas utilizadas se relacionan a continuación:

Análisis Cualitativo	Está basada en la experiencia y punto de vista del analista de riesgo, por lo que intuitivamente se determina el impacto y la probabilidad que un riesgo ocurra o afecte el proyecto de software. Algunas técnicas utilizadas para este tipo de análisis son: • Juicio por panel de expertos. • Tablas de impacto. • Matrices de probabilidad e impacto. • Agrupación por causas y por prioridad temporal.
Análisis cuantitativo	Se centra en cuantificar en forma precisa el impacto y la probabilidad que ocurra un determinado riesgo. Algunas técnicas utilizadas para este tipo de análisis son: • Obtención descriptiva de datos estadísticos. • Distribuciones de probabilidad. • Juicio Experto. • Uso de diagramas de Tornado para Análisis de Sensibilidad. • Uso de Arboles de Decisión para Análisis de Valor Esperado. • Modelaje y Simulación (Montecarlo).
Análisis de ratios	Se usa la comparación entre el proyecto actual y proyectos anteriores o similares, de acuerdo a los datos generados por dichos proyectos. Según el criterio del analista, se genera un ratio que permita calcular los valores que se utilizaran como bases en el nuevo proyecto. Por ejemplo, se necesita implementar un servidor para almacenamiento de información el cual debe estar funcionando las 24 horas al día los 365 días al año y no se cuenta con un generador de electricidad. Teniendo en cuenta que para un proyecto anterior existieron fallos eléctricos semanales, y para la época en la que se desarrolla el proyecto actual hubo un incremento de compra de aires acondicionados, se puede aplicar un ratio de un 50% al riesgo de fallos de electricidad semanal, es decir 1,5. Entonces: 2 (fallos eléctricos semanales) * 1,5 (ratio) = 3 fallos eléctricos semanales.
Métodos de análisis híbridos	Es una combinación de la toma de decisiones a partir de cálculos y estadísticas con la experiencia y conocimiento del analista, los cuales sirven de complemento a la decisión.
Análisis de probabilidad	Hacen uso de estudios probabilísticos con el fin de hacer la estimación de los riesgos.
Análisis de sensibilidad	Se hace una definición de las variables que pueden afectar al proyecto y a través de herramientas de cálculo matemático se evalúan para de esta forma establecer las posibles consecuencias de los cambios a todo el proyecto. Por lo que se puede decir que se hace un análisis de la sensibilidad del proyecto frente a los cambios en las variables que han sido definidas, permitiendo así cuantificar los riesgos.

Tabla 4
Fuente: Propia.

definir una matriz de riesgos, en la cual se detalla el impacto, probabilidad y respectiva prioridad.

Riesgo	Probabilidad de ocurrencia	Impacto	Momento	Dificultad de detección
Rotación de personal	Moderada	Tolerable	Durante el proyecto	Media
Reducción de presupuesto	Moderada	Grave	Inicio/Durante	Alta
Quiebra de la empresa	Muy baja	Catastrófico	Inicio/Durante	Alta
...	...	...	...	...

Imagen 5
Fuente: Propia.

Planeación de Riesgos

Partiendo del análisis de riesgos, se realiza una planificación que le permita al equipo del proyecto estar preparado para responder adecuadamente a los mismos. Este plan debe indicar:

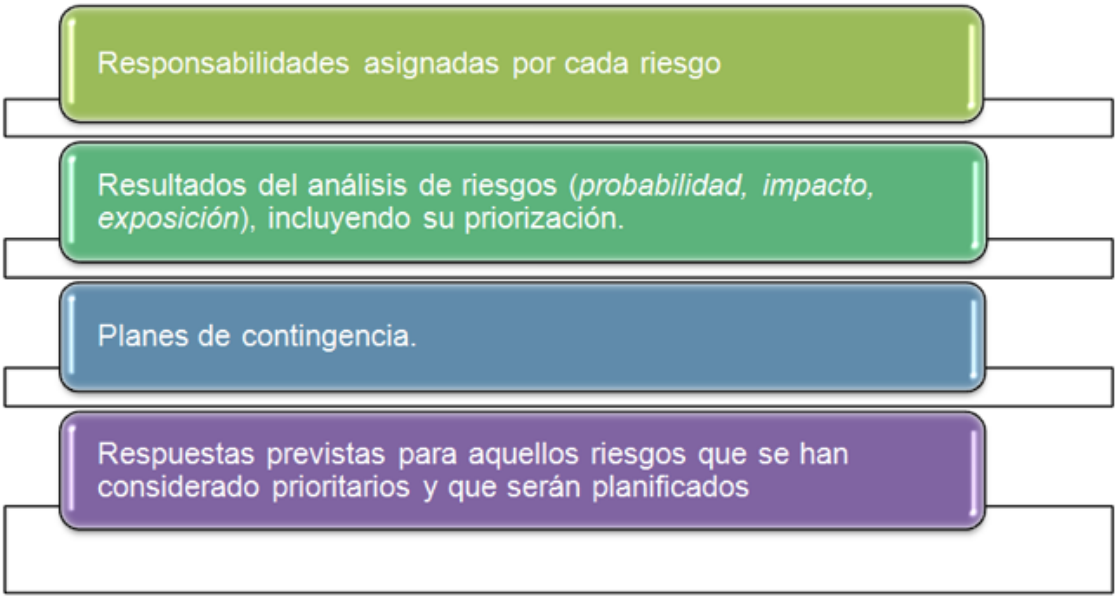


Imagen 6
Fuente: Propia.

A continuación se relacionan algunas estrategias para contrarrestar el riesgo:

Evitar o eliminar el riesgo	Por ejemplo, Evitar la compra de herramientas de software que no conoce o que presente algún riesgo de retraso en el desarrollo del proyecto.
Transferir el riesgo	Por ejemplo, si en una empresa existe alta rotación de personal y con problemas de juicios laborales, indemnizaciones, etc., una alternativa es transferir ese riesgo a las empresas de servicios temporales, quienes brindaran el personal calificado a las empresas y asumirán las condiciones legales de contratación de personal.
Atenuar el riesgo	Por ejemplo, si se presume que la cantidad de usuarios que se conectan a un servidor está creciendo exponencialmente, entonces se debe adquirir uno con mayores prestaciones y con esto evitar posibles fallas por sobrecarga de dicho servidor.
Aceptación pasiva	Implica no realizar ninguna acción frente al riesgo, siempre y cuando tenga muy baja probabilidad de ocurrencia.
Aceptación activa	Se definen planes de contingencia que permitan actuar en aquellos casos en los que el riesgo se haga realidad. Por ejemplo, en casos en los que el servidor exceda su capacidad de almacenamiento, haciendo backups diarios. Los planes de contingencia deben incluir: El Nivel de riesgo residual que se espera una vez se apliquen las acciones respectivas. Acciones específicas, documentadas, compartidas y conocidas por todos los participantes del plan de contingencia y que permitan ejecutar la estrategia para mitigar el riesgo. Presupuesto y tiempos para cada una de las respuestas.

Tabla 5
Fuente: Propia.

Supervisión de Riesgos

Los objetivos de esta etapa son:

- Actualizar periódicamente el registro de riesgos de acuerdo al avance del proyecto, identificando y analizando los posibles nuevos riesgos que puedan generarse, y elaborando respuestas para los mismos.
- Comprobar si los riesgos identificados se materializaron; y de ser así, activar los correspondientes planes de respuesta.
- Realizar el seguimiento de los planes de respuesta en ejecución.
- Gestionar y administrar el fondo de reserva para contingencias.

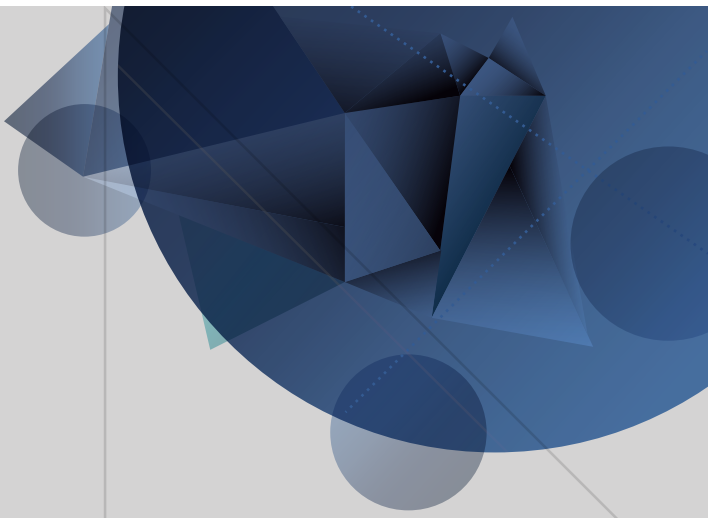
También se centra en realizar un control minucioso sobre cada uno de los riesgos identificados con el fin de determinar:

- Si ha habido cambios en el impacto o probabilidad de los mismos.
- Si se han generado nuevos riesgos y no han sido identificados
- Si se generaron nuevos riesgos como resultado de otros que ya habían sido.
- identificados (riesgos residuales).
- Si un riesgo se presentó con la probabilidad y el impacto que se esperaban.
- Si el plan de contingencia se realizó de acuerdo a las políticas y procedimientos establecidos y si estas si fueron efectivas.

4

Unidad 4

Ciberseguridad



Ingeniería de software II

Autor: Fredy León Socha

AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO

Introducción

El auge y expansión que ha tenido la red de internet en los últimos años ha logrado que muchas más personas estén conectadas al mundo online. En esa misma medida los ataques y delitos informáticos han aumentado, razón por la cual se han activado las alarmas de los países, para crear entidades y políticas que se enfoquen en los temas de Ciberseguridad y Ciberdefensa entre o otros.

Con el avance de nuevas tendencias tecnológicas, como por ejemplo el internet de las cosas o la computación en la nube, redes sociales, etc.; las personas, entidades gubernamentales y privadas, toman más conciencia respecto a lo importante de proteger su información personal y confidencial; pues ser vulnerables en la red puede implicar perdida de sus datos, sistemas, etc.

En este modulo conocerá los marcos referenciales relacionados con Ciberseguridad, tipos y fases de los Ciberataques, estrategias de Ciberdefensa, los ciberdelitos mas comunes, cuales son las instituciones gubernamentales colombianas especializadas en Ciberseguridad y Ciberdefensa y finalmente aprenderá acerca de las normas y políticas que el gobierno colombiano ha definido para estos temas.

U4

Metodología

El módulo se basa en la lectura individual de la presente cartilla y posterior desarrollo de actividades planteadas en la plataforma online.

Se espera una participación y acceso a plataforma a su consideración, como mejor pueda organizarse, para cubrir los objetivos de aprendizaje y participación. Por mi experiencia, recomiendo la asistencia diaria para aprovechar al máximo el curso.

La comunicación conmigo la pueden realizar abiertamente a través del Foro y personalmente a través del email.

Se realizará una teleconferencia semanal de un tema especifico, en el cual se presentarán ejemplos para ampliar la temática semanal.

En cada semana se publica el material de estudio que apoyara el desarrollo de los contenidos temáticos correspondientes a dicha semana, ya sea en recursos bibliográficos, archivos digitales o referencias electrónicas (páginas WEB).

Ciberseguridad

Conceptos generales

Ciberespacio	Es un entorno no físico(virtual) que ha sido creado por un sistema informático con el objetivo de operar en una Red (Real Academia Española, 2001).
Ciberguerra	Conflicto en el Ciberespacio.
Ciberataque	Es el conjunto de acciones que se desarrollan sobre la red de internet con el objetivo de inhabilitar ,destruir o alterar sistemas informáticos ;o copiar, robar o borrar información confidencial de dicho sistema.
Ciberdefensa	Conjunto de acciones de defensa activas pasivas, proactivas, preventivas y reactivas para asegurar el servicio o fin de recursos informáticos, equipos ,redes o sistemas; de asegurar el uso libre de su ciberespacio y negarlo a fuerzas enemigas o en oposición; de mantener la integridad, disponibilidad, y confidencialidad de la información e implementar la gestión de riesgos o amenazas de posibles ciberataques (Departamento Nacional de Prosperidad, 2016).
Ciberseguridad	Conjunto de acciones de carácter preventivo que tienen por objeto el asegurar el uso de las propias redes y negarlo a terceros.
Ciberdelito	Acción criminal en el ciberespacio.
Ciberterrorismo	Acción terrorista en el ciberespacio.
Ciberinteligencia	Actividades de la Ciberseguridad enfocadas al análisis de intenciones de ciberatacantes, identificación, prevención, localización y atribución de ataques o amenazas que se desarrollan en el ciberespacio.

Tabla 1
Fuente: Propia.

Ciberataque

Tipos de Ciberataques

A continuación se relacionan los tipos de ciberataques mas comunes:

- Bombardear constantemente las redes con malware. Los proveedores de firewall normalmente emiten actualizaciones de protección contra malware una vez al día. Esta baja frecuencia de actualización hace que el sistema quede expuesto a posibles ataques continuos a cualquier hora del día.
- Infectar las redes con diferentes tipos de malware. Los ciberdelincuentes utilizan diferentes tipos de malware para realizar ataques a redes. Estos son los 5 tipos de malware más comunes:
  - Los virus informáticos. Transmitidos a través de archivos compartidos, descargas web, archivos adjuntos en correos electrónicos.
  - Troyanos. Se diseñaron principalmente para extraer información confidencial de la red. Pueden controlar el sistema infectado y abren un camino por el que el atacante accede en cualquier momento.
  - Spyware. Normalmente afecta a navegadores web, registrando en secreto en los patrones de uso y comportamiento del usuario en la red.
  - Adware. Utilizado principalmente para distribución de anuncios publicitarios cuando el usuario accede a internet; bombardeándolo con ventanas emergentes, barras de herramientas, secuestros de página inicial del navegador, entre otros.
- Encontrar y atacar las redes más débiles. Aquellas organizaciones que utilizan firewalls de baja calidad, los desactivan o limitan las medidas de seguridad a fin de mejorar el rendimiento de la red, son las más vulnerables a los ataques cibernéticos.
- Transformarse con frecuencia y atacar a escala global. Todos los días y en todos los continentes se generan, reinventan y comparten nuevos tipos de malware y por consiguiente nuevas amenazas. Los ataques cada vez son mucho más rápidos de tal forma que no hay tiempo para implementar un contraataque, pues al siguiente día el malware ha evolucionado.

Fases de un ataque informático

Imagen 1
Fuente: Propia.

FASE 1. Reconocimiento (Reconnaissance)

En esta fase se recolecta y analiza la información de la víctima. El atacante planea la estrategia para el ataque, haciendo uso de toda la información posible de la víctima y utilizando métodos como:

- Ingeniería Social. Busca que la persona por sus propios medios suministre la información a través de redes sociales (Twitter, Facebook, Instagram, etc.).
- Buscar en Papelera de reciclaje (Drumpster diving): El atacante busca información de que sistema operativo y aplicaciones utiliza la víctima, la ubicación de los rourters, host; direcciones IP; nombres de dominios, contactos, servidores de correos electrónicos, DNS (Domain Name Server), etc.

FASE 2. Escaneo (Scanning)

El atacante hace uso de toda la información obtenida en la Fase 1 para poder identificar vulnerabilidades específicas. Por ejemplo:

- De acuerdo al sistema operativo que utiliza la víctima, entonces buscará todas las posibles vulnerabilidades para ese sistema operativo.
- Escaneo de los puertos de dicha computadora para encontrar un puerto abierto que le permita acceder al sistema.
- Arquitectura del sistema informático y su tipo de configuración.

FASE 3. Obtener Acceso (Gaining Access)

Se accede al sistema informático, de acuerdo a las vulnerabilidades encontradas en la Fase 2 y al nivel de destrezas, habilidades o conocimientos del atacante. El acceso puede ser Offline (al estar conectado en una red LAN) u Online (Cuando se está conectado a internet).

Las técnicas utilizadas pueden ser:

- Buffer overflows: lograr que se desborde la memoria temporal.
- Denial-of-service: superar el nivel de peticiones y/o denegaciones de servicio.
- Sesión hijacking: secuestro de sesión.
- Password cracking: descifrar claves usando métodos como: diccionary atack y brute forcé atack.

FASE 4. Mantener el Acceso (Maintaining Access)

El atacante utiliza los recursos del sistema atacado, así como sus propios recursos para lanzar nuevos ataques; escanear y acceder a otros sistemas informáticos; copiar, robar o alterar el funcionamiento de programas de software o datos del sistema.

Para realizar la captura del trafico de la red, sesiones de Telnet o FTP (File Transfer Protocol) utiliza programas de software o "Sniffers" y para que no pueda ser detectado y eliminar cualquier rastro del ataque, hace uso de técnicas como:

- Backdoor: accesos vulnerables encontrados en el sistema.
- Troyanos: disfrazados de programas útiles para el usuario, ganan acceso al sistema y dejan abierta las puertas (Backdoor) para que el atacante pueda acceder más adelante. No se auto replican ni se reproducen cuando infectan los archivos.

FASE 5. Cubrir las huellas (Covering Tracks)

El atacante hace un borrado de toda evidencia del acceso (log files, ids-Sistema de detección de intrusos, etc.) con el fin de no ser detectado por el administrador de la seguridad de la red.

Ciberdefensa

Como se mencionaba en el apartado Conceptos en Ciberseguridad, uno de los objetivos es contribuir a la seguridad de la información y datos confidenciales. Para esto se estableces e implementan acciones como:

- Análisis de riesgos y posibles ataques informáticos.
- Informe de vulnerabilidades.
- Informes de inteligencia cibernética.
- Apoyos y colaboración como partner certificado de Symantec.
- Simulacros de Ciberataques.

Otras medidas a tomar para mitigar los riesgos de ataques son:

Comprobar la veracidad de correos electrónicos y archivos adjuntos sospechosos.

- Realizar pagos en línea a través de fuente conocidas y confiables.
- Monitorear los procesos y establecer políticas de seguridad en la empresa.
- Leer detenidamente las condiciones antes de aceptar el acceso, descarga o instalación de un software, acceso a un sitio web, etc.
- Elegir contraseñas con altos niveles de seguridad, por ejemplo incluir en la contraseña mayúsculas, minúsculas, números caracteres especiales.
- Proteger y controlar el acceso de los niños a internet.
- Tener copias de seguridad de nuestra información.
- Descargar aplicaciones de sitios de confianza y activar medios de transmisión como Wifi, Bluetooth solo cuando se necesite.
- Hacer uso de antivirus, firewall y demás herramientas que apoyen la seguridad en nuestros sistemas informáticos.
- Conectarse a redes Wifi y Sitios seguros.

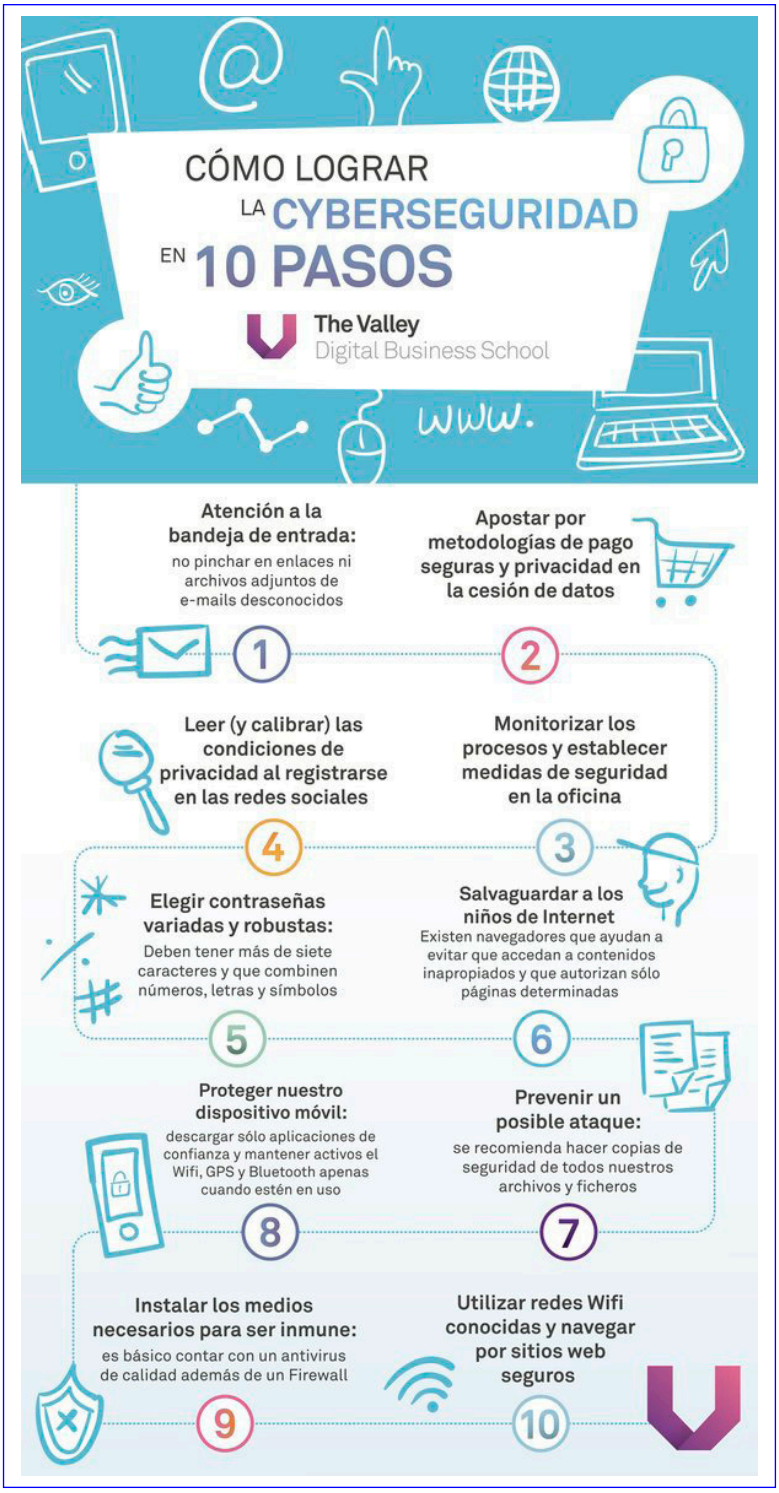


Imagen 2
Fuente: https://alfredovela.files.wordpress.com/2016/02/ciberseguridad-10-pasos-infografia.jpg?w=860

Ciberdelitos

Son acciones realizadas por las personas para atentar contra los derechos, bienes o libertades de otras personas y que tienen como común denominador el uso de medios informáticos. Dentro de los Ciberdelitos más comunes están:

Cyberbulling. También se conoce como Ciberacoso y se presenta cuando una persona amenaza, molesta, chantajea, insulta, humilla, atormenta a otra a través de internet, teléfono móvil u otro tipo de tecnologías.

Revenge Porn. El Porno vengativo es la publicación en internet de contenido sexual explicito sin el consentimiento de la persona directamente involucrada. El contenido es generado por la propia víctima y compartido al infractor usando medios digitales como WhatsApp o Correo electrónico. Este Ciberdelito es considerado como violencia sexual psicológica, debido a que la víctima es sometida a una exposición de su sexualidad, sin su consentimiento.

Grooming. Son acciones realizadas por un adulto para ganarse la confianza de un menor de edad y crear una relación emocional con él; y de esta forma abusar sexualmente y vincularlo a la prostitución o pornografía infantil.

Pharming. Es el proceso mediante el cual un atacante logra vulnerar la seguridad de un servidor para cambiar los DNS de un servidor para que un nombre de dominio redirija a la página web que dicho atacante haya especificado.

Phishing. Proceso en el cual el delincuente (Phiser) adquiere datos confidenciales de una persona u organización; suplantando la identidad de una entidad, haciendo uso de ingeniería social y propagando su ataque a través de medios digitales tales como email, sms, redes sociales, whastapp, etc. Lo información robada puede ser: Datos personales, información bancaria, datos de acceso, entre otros.

En la siguiente imagen se amplía un poco más el concepto de Phising, el tipo de información confidencial y medios de propagación del ataque.

Imagen 3
Fuente: https://www.infospyware.com/images/2012/Qu-es-el-Phishing_E4F5/Phising01.png

Un ataque de Phising sigue 4 pasos esenciales, los cuales se describen a continuación:

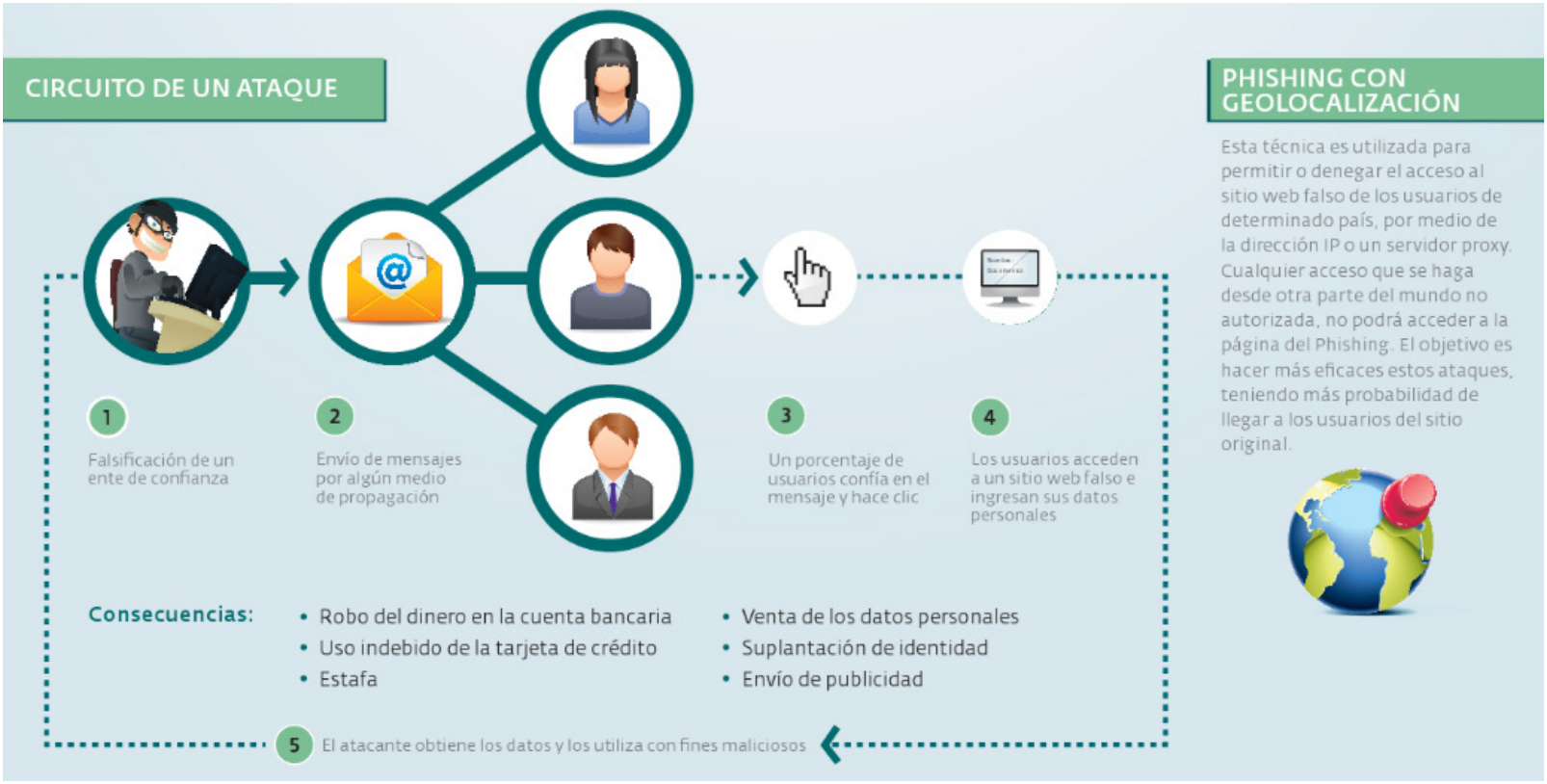


Imagen 4
Fuente: https://www.infospyware.com/images/2012/Qu-es-el-Phishing_E4F5/Phising02_3.png

Ahora que se conoce los ciberdelitos mas comunes, es importante saber cómo se pueden evitar. A continuación se relaciona una infografía de cómo hacer frente a tales delitos.

Imagen 5
Fuente: https://plus.google.com/+ministeriotic/posts/UwNxkb7pZQu

Entidades de Ciberseguridad en Colombia

COLCERT -Grupo de Respuestas a Incidentes Cibernéticos de Colombia

Encargado de coordinar las áreas de Ciberseguridad y Ciberdefensa colombiana, buscando proteger la infraestructura o información crítica y hacer frente a acciones cibernéticas que puedan atentar o comprometer la seguridad y defensa nacional.

El COLCERT es una entidad adscrita al mini Ministerio de Defensa Nacional.

A continuación se relacionan los objetivos de COLCERT:

- Coordinar y asesorar a CSIRT's y entidades tanto del nivel público, como privado y de la sociedad cívil para responder ante incidentes informáticos.
- Ofrecer servicios de prevención ante amenazas informáticas, respuesta frente a incidentes informáticos, así como a aquellos de información, sensibilización y formación en materia de seguridad informática.
- Actuar como punto de contacto internacional con sus homólogos en otros países, así como con organismos internacionales involucrados en esta técnica.
- Promover el desarrollo de capacidades locales/sectoriales, así como la creación de CSIRT's sectoriales para la gestión operativa de los incidentes de ciberseguridad en las infraestructuras críticas nacionales, el sector privado y la sociedad civil.
- Desarrollar y promover procedimientos, protocolos y guías de buenas prácticas y recomendaciones de ciberdefensa y ciberseguridad para las infraestructuras críticas de la Nación en conjunto con los agentes correspondientes y velar por su implementación y cumplimiento.
- Coordinar la ejecución de políticas e iniciativas público-privadas de sensibilización y formación de talento humano especializado, relativas a la ciberdefensa y ciberseguridad.
- Apoyar a los organismos de seguridad e investigación del Estado para la prevención e investigación de delitos donde medien las tecnologías de la información y las comunicaciones.
- Fomentar un sistema de gestión de conocimiento relativo a la ciberdefensa y ciberseguridad, orientado a la mejora de los servicios prestados por el colCERT.

CCOC -Comando Conjunto Cibernético de las Fuerzas Militares

Está conformado las fuerzas militares: fuerza aérea, armada, ejercito.

El CCOC fue creado bajo lineamientos de Ciberseguridad y Ciberdefensa del CONPES 3701, con el objetivo de:

- Atender amenazas y ataques cibernéticos al estado, gobierno y fuerzas militares.
- Prestar servicios Servicios de protección al sector defensa.

- Crear mesas de trabajo para atender al sector público y privado con el fin de determinar infraestructura critica y los planes de acción para atender las amenazas cibernéticas.
- Realizar proceso de sensibilización con la sociedad civil.

A continuación se describen las capacidades operativas del CCOC:

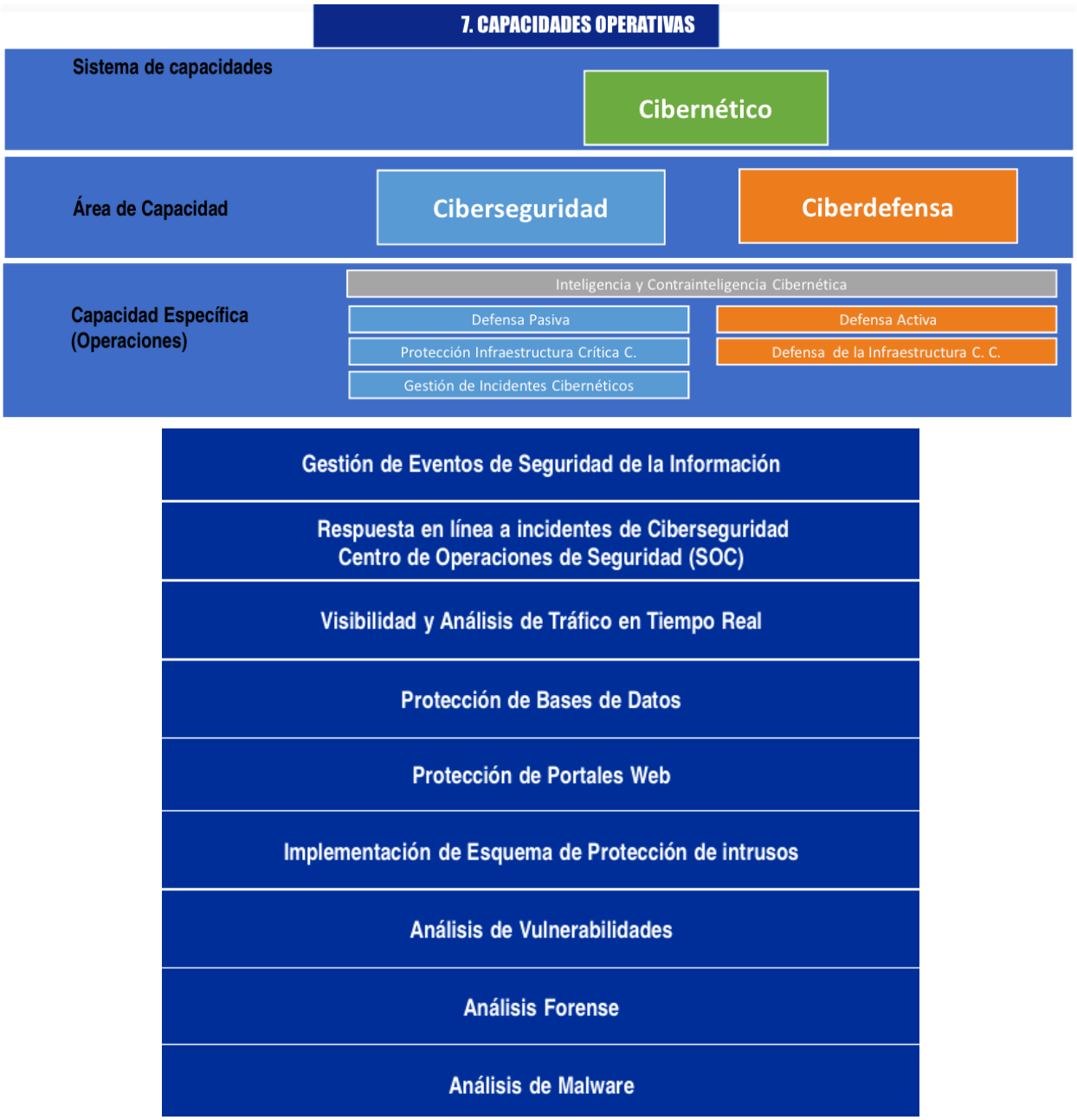


Imagen 6
Fuente: http://media.arpel2011.clk.com.uy/ciber/18.pdf

CCP-Centro Cibernético Policial

Plataforma virtual que la policía nacional ha creado para prevenir delitos por internet tales como pornografía infantil, suplantación sitios web, hurto de contraseñas, jaqueo de correos entre otros.

Hace parte de la Dirección de Investigación Criminal e INTERPOL.

Se encuentra divido en 3 áreas principales:

- Fraudes electrónicos y protección de datos.
- Ciberterrorismo.
- Pornografía infantil.

Como parte de la estrategia de lucha contra los ciberdelitos, el CCP ha desarrollado 3 aplicaciones móviles: PROTECTIO, CAI VIRTUAL y NUEVA FAMILIA DE BILLETES; las cuales están disponibles para su descarga en las tiendas de aplicaciones Google Play, App Store y Windows Store.

Imagen 7
Fuente: https://caivirtual.policia.gov.co/

Para llevar a cabo los objetivos que buscan cada una de las 3 entidades vistas anteriormente (COLCERT, CCOC y CCP) y bajo los lineamientos de la Política Nacional De Seguridad Digital, existe una articulación intersectorial que busca:

- Asistencia técnica.
- Coordinación en gestión de incidentes.
- Asistencia ante emergencias.
- Desarrollo de capacidades operativas.
- Proveer información de inteligencia cibernética.
- Asesoramiento y apoyo en ciberdefensa.

La siguiente imagen muestra como se lleva a cabo esta articulación.

Imagen 8
Fuente: https://isc.sans.edu/diaryimages/images/Instituciones.jpg

También existen otras entidades especificas para cada una de las fuerzas armadas de nuestro pais, estas son:

- Unidad Cibernética Ejército Nacional.
- Unidad Cibernética Armada Nacional.
- Unidad Cibernética Fuerza Aérea.

El Marco legal colombiano en Ciberseguridad

El Marco Legal se inicia en 2011 con la publicación del CONPES 3701- Lineamientos De Política Para Ciberseguridad Y Ciberdefensa, en 2012 se crea el CCOC- Comando Conjunto Cibernético de las Fuerzas Militares, en 2014 se definen los lineamientos de Ciberseguridad y Ciberdefensa; así, a través del tiempo se han definido políticas públicas, hasta la última en abril de 2016 que es la del CONPES 3854- Política Nacional De Seguridad Digital.

En la siguiente imagen encuentra una línea de tiempo de la evolución del marco legal que Colombia ha generado en cuanto Ciberseguridad.

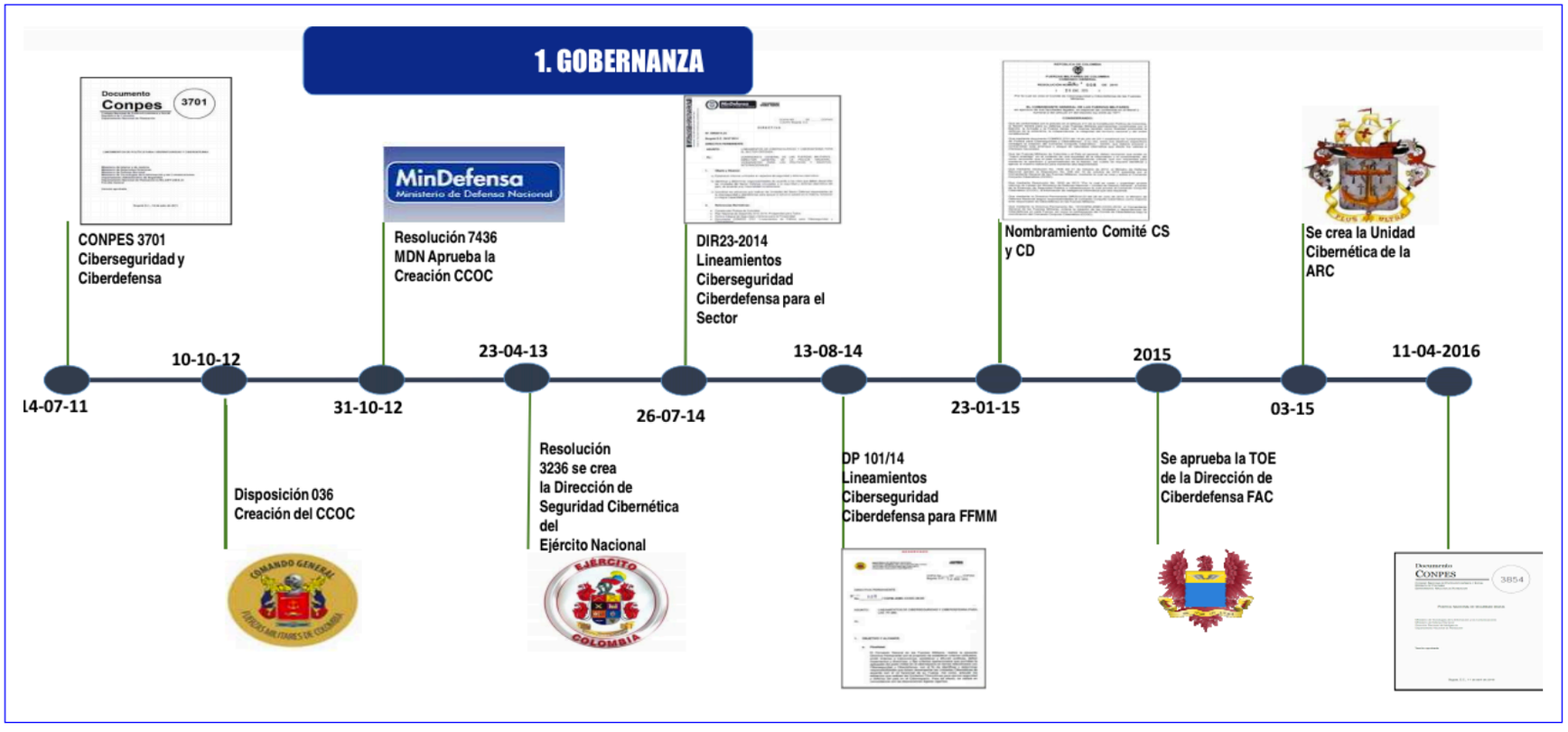


Imagen 9
Fuente: http://media.arpel2011.clk.com.uy/ciber/18.pdf

A continuación se describen las normas más importantes generadas bajo la política de Ciberseguridad en Colombia.

CONPES 3701 - Lineamientos de Política para Ciberseguridad y Ciberdefensa

Se definen los lineamientos, normatividad, antecedentes nacionales e internacionales en cuanto a Ciberseguridad y Ciberdefensa; con el fin de establecer una estrategia que permita hacer frente al incremento de las amenazas informáticas en el territorio nacional.

El documento CONPES 3701 lo puede descargar del siguiente link:

http://www.mintic.gov.co/portal/604/articles-3510_documento.pdf

CONPES 3854-Política Nacional de Seguridad Digital

En la política nacional de seguridad digital se incluye la gestión del riesgo como eje fundamental sobre el cual se cimenta la seguridad digital, ya que actualmente se centraba en hacer frente a las amenazas cibernéticas teniendo como objetivos la defensa del país y la lucha contra el crimen.

El objetivo principal de la Política Nacional de Seguridad digital es " Fortalecer las capacidades de las múltiples partes interesadas para identificar, gestionar, tratar y mitigar los riesgos de seguridad digital en sus actividades socioeconómicas en el entorno digital, en un marco de cooperación, colaboración y asistencia. Lo anterior, con el fin de contribuir al crecimiento de la economía digital nacional, lo que a su vez impulsará una mayor prosperidad económica y social en el país" (DNP,2016, p47)

El documento CONPES 385 lo puede descargar del siguiente link:

https://colaboracion.dnp.gov.co/CDT/Conpes/Econ%C3%B3micos/3854.pdf

Nodo de Ciberseguridad

El Ministerio de Tecnologías de Información y Comunicación de Colombia en su estrategia NDI-Nodos de Innovación, presenta unos vectores de desarrollo prioritarios y orientados a que se fortalezca el nivel del país en cuanto a Ciberseguridad. El nodo especifico para el tema es el Nodo de Ciberseguridad, el cual se explica a continuación:

Imagen 10
Fuente: http://www.mintic.gov.co/portal/604/articles-6120_recurso_1.png

Adicional a estas políticas, existen leyes especificas tales como:

Imagen 11
Fuente: Propia.

También se han generado los siguientes decretos:

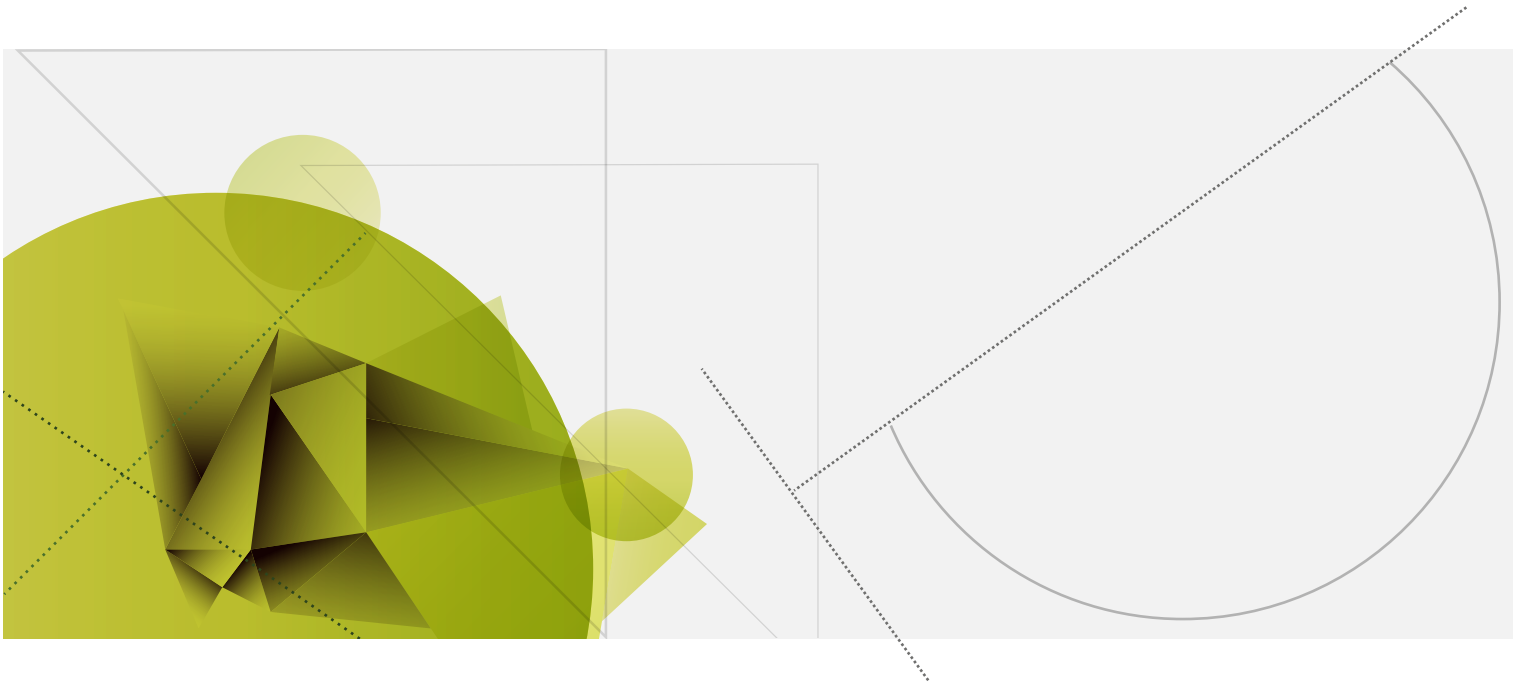
Decreto Ley 019 de 2012 – Racionalización de trámites a través de medios electrónicos. Criterio de seguridad.

- Ley 1581 de 2012 – Ley estatutaria de Protección de datos personales.
- Ley 1623 de 2013 – Ley de Inteligencia – Criterios de seguridad.
- Ley 1712 de 2014 – Transparencia en el acceso a la información pública.
- Decreto 2364 de 2012 – Firma electrónica.
- Decreto 2609 de 2012 – Expediente electrónico.
- Decreto 2693 de 2012 – Gobierno electrónico.
- Decreto 1377 de 2013 – Protección de datos personales.
- Decreto 1510 de 2013 – Contratación Pública electrónica.
- Decreto 333 de 2014 – Entidades de certificación digital.

Bibliografia

- Arias, A. & Durango, A. (2014). Ingeniería y arquitectura de software 2ed. Vigo, España: IT Campus Academy.
- Calero, C. & Piattini, M. (2010). Calidad del producto y proceso del software. Madrid, España: RA-MA.
- Kendall, K. & Kendall, J. (2005). Análisis y diseño de sistemas. Naucalpan de Juárez, México: Pearson Education.
- Osorio, O., Benjamin, Z. & Chávez, A. (2014). Mantenimiento y estimación ágil de proyectos software. Madrid, España: EAE.
- Pantaleo, G. (2012). Calidad en el desarrollo de software. Madrid, España: Marcombo.
- Pérez, R. (2015). Mantenimiento del software. Malaga, España: IC Editorial.
- Pressman, R., Murrieta, J., Pineda, E. & Campos, V. (2010). Ingeniería de Software: Un Enfoque Práctico. México, D.F.: McGraw-Hill.
- Sommerville, I. (2005). Ingeniería del Software. 7ed. Madrid, España: Pearson Addison Wesley.

Esta obra se terminó de editar en el mes de noviembre
Tipografá Myriad Pro 12 puntos
Bogotá D.C,-Colombia.



AREANDINA
Fundación Universitaria del Área Andina

MIEMBRO DE LA RED
ILUMNO