**Unidad 1: Ingeniería del Software**

Ingeniería: aplicación de conocimiento científico para resolver problemas, de significado práctico inmediato, que tienen requerimientos y restricciones conflictivas. Conocimientos y técnicas cuya aplicación permite la utilización racional de los materiales y de los recursos, mediante invenciones, construcciones u otras realizaciones provechosas para el hombre.

Software: programas de computadoras, procedimientos, posible documentación asociada y datos pertinentes a la operación de un sistema de computadora. El software está compuesto por (IEEE):

* Las instrucciones (programas) que proporcionan las características y funciones.
* Las estructuras de datos que le permiten manipular información.
* Los documentos que describen la operación y uso.

Mitos del Software

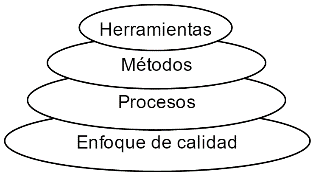
Mitos del Administrador: los procedimientos y estándares proporcionan todo el conocimiento necesario. Cuando el cronograma está retrasado, siempre es posible contratar nuevos programadores para terminar a tiempo. Si se decide subcontratar el proyecto de software, es posible relajarse y esperar a que se construya.

Mitos del Cliente: un enunciado general de los objetivos es suficiente para comenzar a escribir programas, los detalles se pueden afinar más adelante. Los cambios pueden ajustarse con facilidad porque el software es flexible.

Mitos del Desarrollador: una vez que el programa fue escrito y puesto a funcionar, el trabajo está terminado. Mientras un programa no se está ejecutando, es imposible evaluar su calidad. El único producto del trabajo que puede entregarse para tener un proyecto exitoso es el programa en funcionamiento. La ingeniería del software obligará a generar una documentación voluminosa e innecesaria y hará más lento el proceso.

**Ingeniería de Software**

Disciplina de ing., donde se aplican herramientas, métodos y procesos para generar un producto de software de calidad.

Establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales.

La ingeniería de software es (1) la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software; es decir, la aplicación de la ingeniería al software. (2) El estudio de enfoques según el punto 1.

Herramientas: instrumentos o sistemas automatizados que dan soporte a las actividades de producción de software.

Métodos: indican cómo construir técnicamente el software. Descripción del modelo de sistema, reglas, recomendaciones, guías en el proceso.

Procesos: secuencia de actividades para la producción de software. Combinación de métodos y herramientas que dan un resultado particular. Definen:

* + Secuencia en la que se aplican los métodos.
  + Entregas requeridas.
  + Controles: ayudan a asegurar la calidad y coordinar los cambios.
  + Directrices: ayudan a los gestores del software a evaluar el progreso.

Actividades: especificaciones del software, desarrollo del software, validación del software y evolución del software.

¿Por qué surge la IS? Crisis del Software

Entre la década del 60 y el 80 se comienzan a producir los primeros problemas en el proceso de desarrollo, debido a la creciente demanda, complejidad y desafíos, y la utilización de la misma fuerza de trabajo, métodos y herramientas.

Críticas a los desarrollos del software: retrasos imprevistos, costos, software no acorde, errores, dificultad de puesta en marcha, evolución y mantenimiento.

No silver bullet - Paper de Brooks - abril de 1987. Reconoce dos tipos de problemas:

* Esenciales: inherentes a la naturaleza del software.
* Accidentales: relacionadas a la producción de software, pero que no son inherentes a él.

La complejidad del Software

*“La complejidad del software es una propiedad esencial, no una propiedad accidental” (Brooks)*

Motivos que llevan a que el software sea complejo

* Complejidad del dominio del problema (imagen que tiene el cliente, la del desarrollador y el dominio en sí).
* Dificultad de la gestión del proceso de desarrollo.
* Flexibilidad del desarrollo de software (necesidad de grandes labores de abstracción y falta de estándares).
* Problemas en la caracterización del comportamiento de sistemas discretos (cantidad de variables e interacciones).

Particularidades del Software

* Producto enteramente conceptual.
* No tiene propiedades físicas ni está sujeto a leyes físicas o eléctricas.
* Distancia intelectual entre el software y el problema que el software resuelve.
* Es difícil para una persona que entiende el problema entender el sistema software que lo resuelve.
* Para probar es necesario disponer de un sistema físico.
* El mantenimiento no es sólo una substitución de componentes.

Productos de Software

* Genéricos: producidos por una organización para ser vendidos al mercado.
* Hechos a medida: sistemas desarrollados a clientes específicos bajo pedido.

La mayor parte del gasto del software es en productos genéricos, pero hay más esfuerzo en el desarrollo de los otros.

Características del Software

* Se desarrolla, **no** se fabrica/manufactura. Los costos se concentran en la ingeniería.
* No se desgasta, pero sí se “deteriora”.
* La mayoría del software se construye a medida. La reutilización de componentes recién ha empezado.

Características de los productos de Software

* **Mantenible**: debe ser posible que el software evolucione y que siga cumpliendo con sus especificaciones.
* **Confiable**: que el software funcione libre de fallas en un entorno determinado y durante un tiempo específico.
* **Eficiente**: el software debe usar los recursos del sistema adecuadamente, sin desperdiciarlos.
* **Interfaz adecuada**: el software debe contar con una interfaz de usuario adecuada y su documentación.

Tipos de Software (Pressman)

* Por función
  + Software de sistemas: SO.
  + Software de aplicación: ofimática, correo electrónico.
  + Software científico y de ingeniería: autocad, geogebra.
  + Software empotrado: software que está incluido en productos que no son normalmente de software.
  + Software de línea de productos: software para la generación de líneas de productos.
  + Aplicaciones Web: desarrollo web.
  + Software de Inteligencia Artificial: robótica, sistemas expertos, redes neuronales.
  + Juegos.
* Por origen
  + Software nuevo: aquel que se desarrolla para cumplir con una función.
  + Software heredado
  + Software de integración: software que comunica diferentes softwares.
* Por arquitectura: monolíticos, cliente-servidor, aplicaciones basadas en web.

Necesidad de un proceso: un proceso de desarrollo de software:

* Conjunto estructurado de actividades requeridas (y resultados asociados) para desarrollar o producir un sistema software (producto). Planificación, especificación, diseño o desarrollo, validación, evolución, gestión.
* Conjunto de procedimientos organizado para construir productos que satisfacen objetivos y estándares.
* Organiza las actividades. Impone consistencia y estructura.
* Contribuye a la calidad del software y a la velocidad con que se desarrolla.
* Define el enfoque que se adopta mientras el software está en desarrollo.
* Define un marco de trabajo para realizar SW de alta calidad.
* Brinda estabilidad, organización y control.
* Define las interacciones: Usuarios – Diseñadores, Usuarios – Herramientas, Diseñadores – Herramientas.

Características del proceso de desarrollo

* Entendible: ¿se encuentra el proceso bien definido y es entendible?
* Visible: ¿El proceso es visible al exterior?
* Soportable: ¿Puede el proceso ser soportado por herramientas CASE?
* Aceptable: ¿El proceso es aceptado por los involucrados en él?
* Confiable: ¿Los errores del proceso son descubiertos antes de que se conviertan en errores del producto?
* Mantenible: ¿Puede el proceso evolucionar para cumplir con los objetivos organizacionales?
* Rapidez: ¿Cuánto más rápido puede producirse el producto?

**Modelos del proceso**

Estrategia de desarrollo. Se selecciona de acuerdo a la naturaleza del proyecto y aplicación, los controles y entregas requeridas y las características del equipo u organización de desarrollo.

El objetivo es determinar el orden de las etapas involucradas en el desarrollo del software y establecer el criterio de transición para progresar de una etapa a la siguiente.

Ciclo de Vida: sucesión de etapas por las que atraviesa un producto de software a lo largo de su existencia.

Las principales diferencias entre distintos modelos de ciclo de vida están divididas en tres grandes visiones:

* Alcance del ciclo de vida (hasta dónde se desea llegar con el proyecto).
* Cualidad y cantidad de las etapas en que será divido el ciclo de vida.
* Estructura y la sucesión de las etapas, si hay realimentación entre ellas y si hay libertad para repetirlas.

Modelo en Cascada

* Enfoque sistemático y secuencial. El inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior. Cada fase genera entradas y documentación a la siguiente. Se obtiene el producto operativo al final.
* Se usa cuando los requerimientos están claros y no son volátiles.
* Los proyectos reales raramente siguen un flujo secuencial. Es costoso resolver problemas.

Modelos incrementales

* Incrementos: proporcionan un subconjunto de funcionalidad. Se entrega un producto operacional.
* El cliente se involucra más, se usan los incrementos como prototipos y generan experiencias de validación.
* Difícil de aplicar a sistemas transaccionales que tienden a ser integrados y a operar como un todo.
* Los errores en los requisitos se detectan tarde.
* Se utiliza cuando no se dispone del personal para una instalación completa, cuando se pueden definir los incrementos acotados en tamaño, con funcionalidades bien definidas y cuando no se está seguro de cumplir con plazos de tiempo o se tiene una fecha imposible de cambiar.
* No es recomendable cuando se tienen atributos de calidad críticos.

Modelos Evolutivos

* Iterativos y evolutivos 🡪 En cada iteración se obtienen versiones más completas del software.
* Se adaptan más fácilmente a los cambios introducidos a lo largo del desarrollo.

Prototipos

* Se utilizan cuando no están claros los requerimientos al inicio debido a que se tiene poco conocimiento del dominio o porque hay inseguridades del lado cliente y/o desarrollador.
* Reduce el riesgo.
* Es un medio excelente para recoger el ‘feedback’ del usuario final.

**Métodos Ágiles**

Alternativa a los procesos de desarrollo tradicionales.

Principios fundamentales

* Valorar al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.
* Desarrollar software que funciona más que conseguir una buena documentación.
* La colaboración con el cliente más que la negociación de un contrato.
* Responder a los cambios más que seguir estrictamente un plan.

Tipos de Metodologías Ágiles

* Programación Extrema (XP)
  + Centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de SW.
  + Promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores.
  + Realimentación continua entre el cliente y el equipo de desarrollo.
  + Comunicación fluida entre todos los participantes.
  + Simplicidad en las soluciones implementadas.
  + Flexibilidad para enfrentar los cambios.
  + Se usa cuando los requerimientos son imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.
* SCRUM
  + Está especialmente indicada para proyectos con un rápido cambio de requisitos.
  + El desarrollo se realiza mediante iteraciones (sprints) con una duración máxima de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente.
  + Promueve reuniones diarias a lo largo del proyecto.

¿Por qué es difícil la Ing. del Software?

El software es uno de los objetos de mayor complejidad hecho por humanos. Es una disciplina joven víctima de su propio éxito: la teoría sobre la que se debe apoyar no está terminada, y la tecnología y la capacidad de construir sistemas complejos crece rápidamente.

Demandas Comerciales

* Sistemas que nunca deben fallar y que siempre deben alcanzar sus plazos de entrega.
* Sistemas que deben ser seguros, confiables, livianos y extensibles.
* Sistemas de redes abiertas de performance crítica, y que son muy costosos de apagar.

IS: Problemas Principales

* La planificación y la estimación de costos son frecuentemente imprecisas.
* La productividad no se corresponde con la demanda.
* La calidad del software no llega a ser a veces ni aceptable.

Mediante la combinación de:

* Métodos completos para todas las fases del desarrollo del software,
* Mejores herramientas para automatizar estos métodos,
* Bloques de construcción más potentes para la implementación del software,
* Mejores técnicas para garantizar la calidad del software,
* Y una filosofía predominante para coordinación, control y gestión,

podemos conseguir una disciplina para el desarrollo del software: “Ingeniería del Software”.

**Estándares de la IS**

* Estándar: conjunto de criterios aprobados, documentados y disponibles para determinar la adecuación de una acción o conjunto de acciones (estándar de proceso) o de un objeto (estándar de producto).
* Guía: criterios bien definidos y documentados que encaminan una actividad o tarea. Más flexible que un estándar.

Utilidad de los Estándares

* Agrupan lo mejor de las buenas prácticas y usos. Engloban los conocimientos de una organización.
* Marco para implementar procedimientos de aseguramiento de la calidad.
* Proporcionan continuidad entre el trabajo de distintas personas.

Tipos de Estándares en IS: para datos, de codificación, de documentación, de proceso, estructurales. Otras actividades: seguridad, auditoría, planificación, seguimiento, control de calidad, etc.

Estándares relacionados con el proceso

* Necesidad establecer un enfoque disciplinado y sistemático
* Objetivos:
  + Conseguir un marco común para “hablar el mismo lenguaje” en el desarrollo y gestión de SW.
  + Definir los procesos de desarrollo, mantenimiento y gestión de software de forma genérica y abstracta.

**Estándares**

SEI’s CMM/CMMi (Capability Maturity Model)

Modelo de evaluación de procesos, que proporciona una medida de la eficacia de las prácticas de IS de una compañía. Los resultados son numéricos y proporcionan una indicación de la madurez del proceso en la organización. Cinco niveles:

* 1 – Inicial: el éxito depende de esfuerzos personales.
* 2 – Repetible/Gestionado: se establecen políticas y procedimientos para llevar a cabo un proyecto. Se planifica y se revisa para evaluar objetivos. Se obtienen niveles de calidad parecidos a proyectos anteriores.
* 3 – Definido: se adopta un proceso de software estándar y se adapta a cada proyecto. Se ajusta a las políticas de la organización.
* 4 – Gestionado/Cuantitativamente: la calidad del producto y del proceso es evaluada, predecible y cuantificable.
* 5 – Optimizado: el proceso es continuamente mejorado usando las medidas obtenidas de procesos anteriores.

KPA (Área de procesos clave): conjunto de prácticas relacionadas ejecutadas de forma conjunta para conseguir un conjunto de objetivos. Dos representaciones:

* Por niveles: nivel de madurez considerando todas las KPAs (define un nivel de madurez organizacional).
* Continua: nivel de madurez en cada KPA independientemente de los restantes.

ISO 9000

Familia de estándares para la gestión de la calidad de cualquier proceso de producción.La organización debe tener un proceso de calidad que supervise todas las fases de la producción y entrega del producto.Para cada proyecto se define un plan de calidad.Fomenta la producción de documentación e informes.

Objetivos:

* Estandarizar las actividades del personal por medio de la documentación.
* Incrementar la satisfacción del cliente.
* Medir y monitorear el desempeño de los procesos.
* Incrementar la eficacia y/o eficiencia de la organización en el logro de sus objetivos.
* Mejorar continuamente en los procesos, productos, servicios, eficacia, etc.

IEEE 1074 - Standard for Developing Software Life Cycle Processes

* Define
  + Las actividades que constituyen los procesos necesarios para el desarrollo y el mantenimiento de software (parte de un sistema o autónomo)
  + Los procesos de gestión y soporte a lo largo de toda la vida del software.
* Ciclo de vida “una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento de software”.
* Requiere la definición de un ciclo de vida, pero no implica ninguno determinado.
* Cada organización debe asociar las actividades definidas en el estándar a su propio ciclo de vida de software.
* Prescribe los procesos del ciclo de vida, no los productos del mismo.
* Implementación: establecer los requerimientos, seleccionar, desarrollar, establecer y validar el proceso de ciclo de vida del proyecto de software.
* Conformidad con el estándar: implica la realización de todas las actividades obligatorias.

Estándar ISO/IEC 12207: Information Technology / Software Life Cycle Processes

Marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso.

Objetivos principales: proporcionar una estructura de trabajo para que todos los involucrados en el desarrollo de software usen un lenguaje común (procesos bien definidos).

* Modularidad: procesos con mínimo acoplamiento y máxima cohesión.
* Responsabilidad: busca establecer un responsable para cada proceso, facilitando la aplicación del estándar.

Características:

* Proporciona procesos para definir, controlar y mejorar los procesos de ciclo de vida software.
* El marco descrito por el estándar está diseñado para ser adaptado a cada organización y proyecto.
* El proceso de adaptación consiste en la eliminación de procesos, actividades y tareas no aplicables.

Procesos:

* Procesos Principales: útiles a las personas involucradas en la adquisición, suministro, desarrollo, explotación o mantenimiento del software.
  + Adquisición: tareas que el comprador realizan para adquirir un producto de software.
  + Suministro: tareas que realiza el proveedor.
  + Desarrollo: actividades y tareas realizadas por el desarrollador: implementación del proceso, análisis de requisitos del sistema, diseño de la arquitectura del sistema, análisis de los requisitos del software, diseño detallado del software, codificación y prueba del software, instalación, apoyo a la aceptación.
  + Explotación u Operación: el sistema debe ser operado de acuerdo con la documentación de usuario, en su entorno previsto.
  + Mantenimiento: el software o la documentación necesita ser modificado, debido a problemas, necesidades de mejora o adaptación.
* Procesos de Soporte: sirven de apoyo al resto; se aplican en cualquier momento del ciclo de vida.
  + Documentación: registrar la información producida por cualquier proceso o actividad del ciclo de vida.
  + Gestión de configuración: se encarga de gestionar las modificaciones de los elementos de configuración del software de un sistema y las versiones de los mismos.
  + Aseguramiento de calidad: aporta confianza en que los procesos y los productos de software del ciclo de vida cumplen con los requisitos especificados y se ajustan a los planes establecidos.
  + Verificación: verifica si los requisitos están bien recogidos (verificación horizontal), y si los productos de cada fase del ciclo de vida cumplen los requisitos impuestos (verificación vertical).
  + Validación: indica si el sistema o software final cumple con las necesidades del usuario.
  + Revisión conjunta: evaluar el estado del software y sus productos en una actividad del ciclo de vida o fase del proyecto. Se realiza durante todo el ciclo de vida.
  + Auditoría: evaluación exhaustiva que produce un documento de recomendaciones para enmendar o mejorar los aspectos débiles que se detecten.
  + Resolución de problemas: analizar y eliminar los problemas descubiertos durante el proceso.
* Procesos Generales o de la Organización: se realizan a nivel organizativo para gestionar y mejorar la organización.
  + Gestión: implica planificación, seguimiento y control, revisión y evaluación.
  + Infraestructura: establece la infraestructura necesaria para el resto de procesos: hardware, software, herramientas, normas, instalaciones.
  + Mejora: establecer, valorar, medir, controlar y mejorar los procesos del ciclo de vida del software.
  + Formación: mantener el personal formado, desarrollando un plan de formación.
* Proceso de Adaptación: Permite adaptar el estándar a cada proyecto y organización. Factores que influencian la forma de adquirir, desarrollar, explotar o mantener un sistema:
  + Contexto organizacional,
  + Tamaño y complejidad del proyecto,
  + Requerimientos regulatorios,
  + Culturas corporativas, etc.

**Unidad 2: Ingeniería de Requerimientos**

Se relaciona con la búsqueda de la situación futura y el cambio asociado, con encontrar información y considerar posibles opciones, y con la identificación de lo que debería ser diseñado para satisfacer una necesidad futura percibida.

Ideas fundamentales: la parte más difícil es el inicio, impacto perjudicial del error, dificultad de rectificar posteriormente.

Costo de reparar errores: varía de acuerdo al momento en que se esté dentro del ciclo de vida del software.

Problemática de la IR: progresar sobre requerimientos no consolidados es la principal razón de problemas en el desarrollo.

Ingeniería de Requerimientos

Requerimientos: descripción de los servicios proporcionados por el sistema y sus restricciones operativas. El proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones, se denomina ingeniería de requerimientos.

Es una subtarea de la IS. Proporciona métodos, técnicas y herramientas para la definición de lo que debe hacer el software.

Rol fundamental en la definición del problema, el estudio de factibilidad y el análisis. Una etapa de requerimientos es necesaria porque ayuda al cliente a entender y expresar las nuevas necesidades e identificar cómo puede satisfacerlas.

Es importante asegurar que todos los stakeholders del proyecto llegan a un entendimiento compartido de los términos usados para describir los requerimientos. Los errores en los requerimientos se detectan tarde.

Requerimiento: cualquier cosa que un cliente necesite. Punto de vista del diseñador: cualquier cosa que deba diseñarse.

El énfasis está en lo que irá en el producto, sus características. No cómo se diseñará o construirá. Especificación de lo que debería ser implementado.

Definición de la IEEE:

* Una condición o capacidad necesaria para un usuario para resolver un problema o alcanzar un objetivo
* Una condición o capacidad que debe ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación, u otro documento formalmente impuesto
* Una representación documentada de una condición o capacidad dada en 1 o 2.

Clasificación de los requerimientos

En cuanto al contenido:

* **Funcionales**: describen las entradas, salidas y funciones de transformación del sistema
* **No funcionales**: definen atributos de calidad, como confiabilidad, performance, etc.
* **Inversos**: definen cómo el sistema de software nunca se debe comportar.

En cuanto al origen:

* **Del negocio**: objetivos de alto nivel de la organización. Documentados en la visión o alcance del proyecto.
* **Del usuario**: tareas que los usuarios deben hacer con el producto. Documentados en los casos de uso o escenarios.
* **Funcionales**: funcionalidades del software a construir.
* **No funcionales**: responden a estándares, regulaciones, contratos, interfaces y atributos de calidad.

Requerimientos y Calidad

Los requerimientos están vinculados con la calidad del sistema. Calidad según IEEE: grado en que un sistema, componente o proceso satisface los requerimientos especificados y las necesidades y expectativas de un cliente o usuario.

Beneficios de buenos requerimientos

* Acuerdo con stakeholders (comportamiento esperado y criterio de aceptación).
* Base para estimaciones de recursos y tiempos.
* Mejoras en la usabilidad, mantenibilidad, etc.
* Menor retrabajo y omisiones.

Las ventajas crecen con el tamaño y complejidad del sistema.

Rol de los Requerimientos

* Acuerdo desarrolladores-clientes-usuarios finales (aspecto contractual).
* Base para el diseño de software.
* Soporte para verificación y validación.
* Soporte a la evolución del sistema (cambio del sistema viejo al sistema nuevo y cambio de requerimientos).

Alcance de la IR

* Procesos que concluyen en la formulación de requerimientos.
* Generación de productos relacionados a esos procesos.
* Administración de los requerimientos en el desarrollo, operación y mantenimiento de software.

IR: El rol de las técnicas y las fallas

La falla o éxito de un proyecto siempre será definido con relación a un grupo particular, con sus objetivos, intereses, roles y expectativas, definidos en el contexto global de una organización y su medioambiente político y social y no sólo con relación a la tecnología.

Tipos de fallas en el Desarrollo de Software

* Fallas de correspondencia: los objetivos del proyecto no han sido alcanzados.
* Fallas de procesos: el presupuesto, tiempo u otro recurso ha sobrepasado el punto en donde se esperaba algún beneficio respecto del sistema propuesto, o los recursos gastados no resultaron en un sistema implementable.
* Fallas de interacción: un bajo nivel de uso de un sistema puede ser interpretado como una falla.
* Fallas de expectativas: fallo en dar con las expectativas de al menos un grupo de interés del sistema.

La gestión de requerimientos, como proceso, debe minimizar la ocurrencia de fallas.

IR como proceso

Objetivo: estandarizar el proceso y los productos para:

* Permitir el seguimiento y control de cada actividad.
* Medir la efectividad del procedimiento y formas de mejorarlo a través de la evaluación de los productos parciales.
* Posibilitar el uso de herramientas automáticas aumentar la productividad y reducir los costos administrativos.

Proceso de la IR

* Identificación de requisitos (Elicitar).
* Representación de requisitos (Modelar).
* Análisis de requisitos.
* Comunicación de requisitos.

IR: Subcomponentes

* Desarrollo de Requerimientos: elicitación, análisis, especificación, validación.
* Gestión de requerimientos: cambios.

Documento de Requerimientos: especificación de lo que se requiere que haga un sistema (no de cómo hacerlo).

Características de una buena definición de requerimientos

* Correcta: todo lo que está es lo que el cliente necesita. No tiene nada que no necesite.
* No ambigua: interpretación única.
* Completa: la ERS es completa si incluye todos los requerimientos significativos, todas las respuestas del software a todas las entradas posibles, todas las referencias a las figuras, tablas y diagramas de la ERS y las definiciones.
* Consistente: pueden ocurrir tres tipos de conflictos: diferentes términos para el mismo objeto, características de objetos conflictivas (contradictorias), fallas temporales o lógicas.
* Ranqueable: cada requerimiento tiene una identificación que indica su importancia y/o estabilidad.
* Verificable: solo si todo requerimiento es verificable. Debe existir un proceso con el que se pueda chequear que se cumple con el requerimiento. Los requerimientos ambiguos no son verificables.
* Modificable: la ERS deberá estar organizada en forma coherente y fácil de usar.
* Trazable: el origen de cada requerimiento debe ser claro, facilitando el rastreo de decisiones anteriores.
* Usable: diseñada de modo que sea una referencia y si es necesario se modifique durante la vida del producto.

Beneficios de una buena ERS

* Base de acuerdo entre los clientes y los proveedores sobre las características del producto de software.
* Base para estimar costos y cronogramas.
* Reduce el esfuerzo de desarrollo. Línea base para la validación y verificación. Base para la mejora posterior.
* Facilita la transferencia (nuevos usuarios, máquinas, etc.).

Requerimientos: Niveles de Abstracción

* Los requerimientos de usuario: declaraciones en lenguaje natural y diagramas de los servicios y restricciones.
* Los requerimientos del sistema: funciones, servicios y restricciones operativas del sistema.

**Requerimientos Funcionales**

* Describen lo que el sistema debe hacer. Se convierten en algoritmos, la lógica y gran parte del código del sistema.
* Declaraciones de los servicios que debe proporcionar el sistema, manera en que éste debe reaccionar a entradas particulares y cómo se debe comportar en situaciones particulares. Pueden indicar lo que no debe hacer.
* La especificación de requerimientos funcionales de un sistema debe estar completa (todos los servicios solicitados deben estar definidos) y ser consistente (los requerimientos no deben tener definiciones contradictorias).

**Requerimientos No Funcionales**

* No se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes, como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento.
* A menudo son más críticos que los requerimientos funcionales particulares.
* Pueden restringir el proceso que se debe utilizar para desarrollar el sistema.
* A veces son difíciles de verificar. Para esto se definen métricas que indican el cumplimiento de los requerimientos no funcionales:
  + Rapidez: cantidad de transacciones por segundo, tiempo de respuesta al usuario o a eventos.
  + Tamaño: Kbits, RAM, etc.
  + Facilidad de uso: tiempo de formación, cantidad de cuadros de ayuda.
  + Fiabilidad: tiempo medio entre fallas, probabilidad de no disponibilidad, tasa de ocurrencia de fallos.
  + Robustez: tiempo de reinicio tras fallos, probabilidad de corrupción de los datos después de fallos.

Requerimientos del Producto

* Especifican o restringen el comportamiento del software.
* Indican requerimientos de eficiencia (rendimiento o espacio), de seguridad, de usabilidad y de confiabilidad.

Requerimientos de la organización

* Requerimientos derivados de políticas y procedimientos en la organización del cliente y del desarrollador.
* Incluyen requerimientos operacionales (cómo se usará el sistema), requerimientos del proceso de desarrollo y requerimientos ambientales (definen el entorno de operación del sistema).

Requerimientos Externos

* Requerimientos derivados de factores externos al sistema y a su proceso de desarrollo.
* Incluyen requerimientos regulatorios, legislativos (para que funcione de acuerdo con la ley) y éticos (garantizan que el sistema será aceptable para sus usuarios y público en general).

**Requerimientos y sus características**

* **Necesario:** su omisión provoca una deficiencia en el sistema a construir y no puede ser reemplazado por otro.
* **Conciso:** fácil de leer y entender. Redacción simple y clara.
* **Completo:** se proporciona la información suficiente para su comprensión.
* **Consistente**: no es contradictorio con otro requerimiento.
* **No** **ambiguo**: tiene una sola interpretación. El lenguaje usado en su definición no debe causar confusiones.
* **Verificable**: puede ser cuantificado de manera que permita hacer uso de los métodos de verificación.

Procesos de IR: 4 actividades de alto nivel

* Estudio de factibilidad: valorar si el sistema es útil para la empresa.
* Adquisición y análisis: descubrir requerimientos.
* Especificación: convertir los requerimientos en alguna forma estándar.
* Validación: comprobar que los requerimientos definan el sistema que requiere el cliente.

**Adquisición y Análisis de Requerimientos**

Se trabaja en conjunto con el cliente y usuarios finales. Actividades: descubrimiento, clasificación y organización, priorización y negociación, especificación de requerimientos.

Adquisición de Requerimientos

* Recopilar información sobre el sistema requerido y separar los requerimientos de usuario de los de sistema.
* Fuentes de información: documentación, stakeholders, especificaciones de sistemas similares.
* Técnicas más utilizadas: material escrito, entrevistas, cuestionarios, observación y muestreo.

Entrevistas: se clasifican en: iniciales o preliminares, recolección de datos y seguimiento. Pueden ser estructuradas o no. Se debe entrevistar a diferentes personas, en distintos niveles, para obtener un panorama completo del entorno.

Cuestionarios: serie de preguntas escritas que se responden por escrito.

Observación Directa: se usa para entender procesos operacionales y ayudar a derivar requerimientos de apoyo para ellos.

Muestreo: reunir solo un conjunto representativo de los datos. Apropiado cuando existe gran volumen de datos.

Material Escrito: manuales de procedimientos, de usuarios, de operaciones, formularios, y descripción de trabajos.

Escenarios: bosquejo de la interacción del sistema.

Casos de Uso: identifica actores e interacciones.

**Validación de Requerimientos**

* Proceso de verificar que los requerimientos definan realmente el sistema que el cliente quiere.
* Comprobaciones: de validez, de consistencia, de totalidad, de realismo, verificabilidad.

Técnicas de Validación de Requerimientos

* Revisión de los requerimientos: un equipo de revisores verifica errores e inconsistencias.
* Creación de prototipos: se muestra un modelo a los usuarios finales para constatar si cubre las necesidades reales.
* Generación de casos de prueba.

Administración de Requerimientos

* Proceso de comprender y controlar los cambios en los requerimientos del sistema.
* Debe seguirse cada requerimiento (trazabilidad), y mantener los vínculos entre requerimientos dependientes.

Planeación de la Administración de Requerimientos

* Debe hacerse:
  + Identificación de los requerimientos.
  + Proceso de administración del cambio (efectos y costo del cambio).
  + Políticas de seguimiento.
  + Herramientas de apoyo.
* Deben seleccionarse herramientas automatizadas para el almacenamiento de los requerimientos, la administración del cambio y del seguimiento.

Administración del Cambio en los Requerimientos

* Debe aplicarse a todos los cambios propuestos a los requerimientos del sistema.
* Es fundamental conocer los costos de un cambio, para saber si debe llevarse a cabo o no.
* Tres etapas fundamentales para la administración del cambio:
  + Análisis del problema y especificación del cambio.
  + Análisis del cambio y estimación del costo.
  + Implementación del cambio.

**Unidad 3: Medidas e indicadores de SW**

Medición:

* Proporciona una visión más objetiva y profunda del proyecto.
* Proporciona información para una evaluación fundamentada.
* Se utiliza para mejorar el proyecto:
  + Ayudar en la estimación, control de calidad, evaluación de productividad y control de proyectos.
  + Evaluar la calidad de los resultados de trabajos técnicos
  + Ayudar en la toma de decisiones tácticas a medida que el proyecto evoluciona.

Medir para: caracterizar, evaluar, predecir, mejorar.

* Comprender mejor los procesos, los productos, los recursos y los entornos
* Establecer las líneas base para las comparaciones con evaluaciones futuras.
* Determinar el estado con respecto al diseño.
* Valorar la consecución de los objetivos de calidad
* Evaluar el impacto de la tecnología y las mejoras del proceso en los productos y procesos.
* Planificar: establecer objetivos alcanzables para el coste, planificación, y calidad de manera que se puedan aplicar los recursos apropiados.
* Mejorar el relevamiento de información cuantitativa que ayuda a identificar obstáculos, problemas de raíz, ineficiencias y otras oportunidades para mejorar la calidad del producto y el rendimiento del proceso.

Medida: Proporciona una indicación cuantitativa de la extensión, cantidad, dimensiones, capacidad o tamaño de algunos atributos de un proceso o producto. La medición es el acto de determinar una medida.

Métrica: medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Relata las medidas individuales sobre algún aspecto (ej.: número medio de errores encontrados por revisión o el número medio de errores encontrados por persona y hora en revisiones)

Indicador: es una métrica o una combinación de métricas que proporcionan una visión profunda del proceso del SW, del proyecto de SW o del producto en sí.

Los indicadores de proyecto permiten al gestor de proyectos del SW:

* Evaluar el estado del proyecto en curso.
* Seguir la pista de los riesgos potenciales.
* Detectar las áreas de problemas antes de que se conviertan en críticas.
* Ajustar el flujo y las tareas del trabajo.
* Evaluar la habilidad del equipo del proyecto en controlar la calidad de los productos de trabajo del SW.

Calidad del SW

* Medición de SW: necesidad de obtener datos cuantitativos que ayuden a mejorar la calidad. A partir de Métricas.
* Creación de Modelos de Calidad: útiles para discutir, planificar y obtener indicadores de calidad.
* Aplicación de Estándares de calidad: directrices para el aseguramiento externo e interno de la calidad.

Medición del SW

Se utiliza a lo largo de un proyecto de SW como apoyo en la estimación, control de calidad, valoración de la productividad y control del proyecto.

Se aplica para auxiliar la evaluación de la calidad de los productos y para apoyar la toma de decisiones.

Medición de atributos: identificarlos, desarrollar métricas, utilizarlas para obtener indicadores y planificar e implementar el proceso de mejora.

Beneficios de las Métricas

* Comunicación efectiva basada en hechos y datos y no en opiniones o cuestiones subjetivas.
* Identificación temprana de problemas y riesgos.
* Seguimiento y control de los objetivos.
* Soporte a las decisiones.

Se debe garantizar el buen uso de las métricas, de lo contrario, pueden ocasionar más problemas que beneficios. A veces, las métricas son usadas como medidas de persecución personal.

**Tipos de métricas**

* Las medicas directas del proceso de ingeniería del SW incluyen el costo y el esfuerzo aplicados.
* Las métricas indirectas son más difíciles de evaluar, como la funcionalidad, calidad, eficiencia, mantenimiento.

Ejemplos de métricas

* Orientadas al Tamaño: líneas de código. KLDC
* Orientadas a la Función: funcionalidad entregada por la aplicación. PF
* Orientadas al tamaño y a la función: indicadores del esfuerzo y costo del desarrollo.
* Orientadas a Objetos: tamaño de la aplicación, nº de casos de prueba, componentes independientes, subsistemas.
* Orientadas a CU: describen funciones y características visibles al usuario que son requerimientos básicos.
* Orientadas a Ingeniería Web: Evaluación de complejidad relativa y esfuerzo de desarrollo.

**Dimensiones de las métricas**

Métricas del proceso:

* Evalúan si los procesos de la organización se están llevando adelante según lo esperado, asegurar que los procesos definidos están siendo considerados, e implementar mejoras de manera de lograr los objetivos de negocio.
* Mediciones del Proceso: única forma racional de mejorar procesos: medir sus atributos específicos, desarrollando y empleando métricas.
* Métricas del Proceso:
  + Proporcionan Indicadores de Proceso
  + Se recopilan de varios proyectos
  + Se utilizan a largo plazo
  + Permiten tener visión de la eficiencia de un proceso existente (actividades, productos de trabajo, hitos, esfuerzos, etc.)
* Normas básicas de aplicación de las métricas:
  + Utilizar el sentido común al interpretar los datos.
  + Proporcionar una realimentación regular a personas y equipos.
  + No utilizar métricas para evaluar a personas.
  + Establecer métricas claras y objetivos para alcanzarlas.
  + No utilizar métricas para amenazar a particulares o equipos.
  + Si una métrica identifica un área problemática, no se debería considerar como negativa.
  + Interpretar todas las métricas en su conjunto.

Métricas del Proyecto:

* Planificar objetivamente.
* Se focalizan en cuestiones clave que comprometen el proyecto (tiempo, tamaño, esfuerzo, costo, calidad, estado).
* Se utilizan para adaptar el flujo de trabajo del proyecto y las actividades técnicas.
* Utilización:
  + Evaluar estado del proyecto actual y supervisar y controlar el avance.
  + Analizar riesgos potenciales.
  + Detectar áreas o cuestiones problemáticas.
  + Ajustar flujos y tareas de trabajo.
  + Evaluar habilidad del equipo en mantener bajo control la calidad de los productos.
  + Optimizar la planificación para evitar retrasos y problemas y riesgos potenciales.
  + Evaluar la calidad de los productos en el momento actual.
* Primera aplicación de las métricas del proyecto: desde la planificación. A medida que avanza el proyecto, las medidas del esfuerzo y el tiempo se comparan con las de la planificación.

Métricas del producto:

* Asegurar la aceptación del cliente.
* Las cuestiones de mayor peso son las relativas a los atributos físicos y dinámicos del producto (arquitectura, productividad, confiabilidad, usabilidad, estabilidad, performance, etc.).
* Proporcionan la visión interna necesaria para crear modelos efectivos de análisis y diseño, código sólido y pruebas minuciosas.

Comparación

* Muchas métricas se utilizan en ambos dominios.
* Las métricas del producto se combinan para desarrollar métricas del proyecto.
* Las métricas del proceso pueden concebirse como recopilaciones de métricas del proyecto.
* Las métricas del proceso son estratégicas: determinan el curso del proceso de desarrollo de SW.
* Las métricas del proyecto son tácticas: determinan el curso del proyecto actual.
* Métricas del producto: privadas para un individuo. Métricas del proyecto: públicas para un equipo de SW. Estas se consolidan con el fin de crear métricas del proceso que sean públicas para la organización como un todo.

**Métricas Orientadas a la Función**

Valor de normalización: medida de la funcionalidad entregada por la aplicación. Ya que la funcionalidad no se puede medir directamente, se debe derivar indirectamente mediante otras medidas.

Medida: Punto de Función.

Características de los dominios de información: número de entradas de usuario, de salidas, de peticiones, de archivos y de interfaces externas. A dichas características se les asigna un valor de complejidad.

Técnica de PF

Propósito: medir el SW cualificando la funcionalidad que proporciona externamente, basándose en el diseño lógico del sistema.

Objetivos de los puntos de función:

* Medir lo que el usuario pide y recibe.
* Medir independientemente la tecnología utilizada en la implantación del sistema.
* Proporcionar una métrica de tamaño que dé soporte al análisis de la calidad y la productividad.
* Proporcionar un medio para la estimación del SW.
* Proporcionar un factor de normalización para la comparación de distintos SW.

Métricas Ampliadas de PF

Los puntos de función fueron diseñados originalmente para sistemas de información y por eso les da más importancia a los datos y menos a los aspectos de control y funcionales. Para subsanar esto, se han propuesto los puntos de característica y los puntos de función 3D.

* Puntos de característica: agrega la cuenta de algoritmos al PF.
* Puntos de función 3D: tiene en cuenta las estructuras internas de datos, los datos externos, las entradas/salidas y peticiones de usuario, las transformaciones de datos y las transiciones de estados.

**Métricas para la calidad**

* **Corrección**: grado en el que el SW lleva a cabo la función requerida. (defectos por KLDC, un defecto se define como una falta verificada de conformidad con los requisitos).
* **Facilidad de mantenimiento**: facilidad de corrección, de adaptación si cambia el entorno, o mejoramiento. Se deben utilizar medidas indirectas. (métrica orientada al tiempo: tiempo medio de cambio (TMC)).
* **Integridad**: capacidad para resistir ataques contra su seguridad. Amenaza es la probabilidad de que un ataque ocurra en un tiempo determinado. La seguridad es la probabilidad de que se pueda repeler el ataque.
* **Facilidad de uso**: intento de cuantificar lo amigable que puede ser el SW con el usuario y se puede medir en función de la habilidad intelectual y/o física requerida para aprender el sistema, el tiempo requerido para usar eficiente el sistema, el aumento neto en productividad y la valoración subjetiva de la disposición de los usuarios hacia el sistema.

Métrica EED (eficacia de la eliminación de defectos)

* Proporciona beneficios tanto a nivel del proyecto como del proceso.
* Mide la habilidad de filtrar las actividades de la garantía de calidad y de control al aplicarse a todas las actividades del marco de trabajo del proceso.

Métricas del modelo de análisis: calidad de la especificación

* Corresponden a la calidad de la especificación de requerimientos.
* Incluyen las siguientes características: especificidad, compleción, corrección, comprensión, verificación, consistencia, logro, concisión, trazabilidad, modificación, exactitud, reutilización, validación.

Métricas de mantenimiento

Índice de Madurez del SW (IMS): indicación de la estabilidad del producto de SW, basada en los cambios de cada versión del producto.

Métricas de documentación

Índice de Fog: evalúa la dificultad de lectura de un texto. Mide la longitud promedio de las palabras y declaraciones en los documentos.

**Cocomo** (Constructive Cost Model)

* Modelo de estimación de costes compuesto por 4 submodelos: básico, intermedio y avanzado.
* Está basado en modelos de estimaciones matemáticas y en la cantidad de líneas de código del proyecto.
* Está orientado al producto final, no a fases intermedias.

Inconvenientes del modelo

* Comentarios en líneas de código.
* Estimaciones sobre un nº de líneas de código variable.
* No se le da importancia a la productividad, referente a los hábitos de trabajo.
* Dificultad para contemplar costes de revisiones, reuniones.

Modos – Clasificación de los proyectos

* **Orgánico**: proyectos relativamente sencillos, menores de 50.000 líneas de código. Se tiene experiencia en proyectos similares y se encuentra en un entorno estable.
* **Semiorgánico**: proyectos intermedios en complejidad y tamaño. La experiencia en este tipo de proyectos es variable, y las restricciones intermedias.
* **Empotrado**: proyectos bastante complejos, en los que apenas se tiene experiencia y en un entorno de gran innovación técnica. Se trabaja con unos requisitos muy restrictivos y de gran volatilidad.

Estimación por Cocomo

* Para estimar costos, esfuerzo y tiempo requerido para desarrollar el proyecto.
* Para calcular el Esfuerzo, es necesario hallar la variable KDLC, donde es necesario conocer las líneas por cada PF según el lenguaje utilizado.
* Las formulas a aplicar son
  + E = Esfuerzo =
  + T = Tiempo de duración del desarrollo =
  + P= Personal =
* También, se requiere conocer el valor de la variable FAE (Factor de Ajuste de Esfuerzo), la cual se calcula teniendo en cuenta los atributos del producto, del ordenador, del personal y del proyecto. Se debe justificar cada uno de los valores seleccionados.

Modelo Intermedio

Añade al modelo básico 15 factores de ajuste o guías de coste, los cuales proporcionan mayor precisión en la estimación.

Atributos del modelo: SW, HW, personal, proyecto.

**Programa de Métricas de SW dirigido por Metas**

* Identificar objetivos de la empresa.
* Identificar lo que se quiere conocer o aprender.
* Identificar los subobjetivos.
* Identificar entidades y atributos relacionados con los objetivos secundarios.
* Formalizar los objetivos de la medición.
* Identificar preguntas cuantificables e indicadores relacionados que se usarán como apoyo para lograr las metas.
* Identificar elementos de datos que se recopilarán para construir los indicadores para responder las preguntas.
* Definir las medidas e identificar acciones para implementarlas medidas.
* Preparar un plan para implementar las medidas.

Paradigma GQM (Goal – Question – Metric)

* Fundamento: basar la medición en objetivos medibles dirigidos por el modelo apropiado.
* Se utiliza a nivel de proceso y proyecto. Provee un marco de trabajo para interpretar los datos y entender el enfoque sobre los objetivos.
* **Metas**: aquello que la organización o el proyecto intenta alcanzar. Se define un objetivo para una entidad.
* **Preguntas**: refinamientos de las metas en las que se identifican áreas específicas de incertidumbre.
* **Métricas**: para responder a las preguntas y confirmar si las mejoras cumplieron su objetivo.

GQM - Pasos Básicos

1. Desarrollar el conjunto de objetivos corporativos, de división y proyecto.
2. Construir programas para cada objetivo que lo definen en la forma más completa posible.
3. Especificar las métricas requeridas para contestar las preguntas.
4. Construir los procesos de recolección de datos.
5. Recolectar, validar y analizar los datos obtenidos.

GQM – Propuesta de Pasos

1. Identificar diversas entidades que necesitan ser evaluadas para lograr la meta planteada: productos, recursos, actividades, código fuente, casos de prueba, solicitudes de cambio, tareas de ingeniería de SW, planificación, etc.
2. Identificar preguntas asociadas a cada una de las entidades definidas relacionadas con el objetivo planteado.
3. Agrupar las preguntas relacionadas e identificar para cada grupo una submeta relacionada con la original.
4. Identificar subentidades y atributos necesarios a evaluar para las submetas.
5. Definir de manera completa métricas e indicadores para los atributos identificados.

**Métricas Privadas vs Métricas Públicas**

Métricas Públicas: Engloban información originalmente privada para particulares y equipos. Ej.: esfuerzo, tiempo, etc.

Algunas métricas son privadas para el equipo de un proyecto y públicas para la organización. Ej.: índice de defectos por proyecto y general.

Privacidad de las métricas

Propone clasificar las métricas directas o básicas en algún nivel de privacidad:

* **Individual**: sólo los individuos que recolectan los datos sobre su propio trabajo conocen su información. Ej.: horas semanales individuales dedicadas a actividades de desarrollo.
* **Equipo de proyecto**: Los datos son privados para el equipo de proyecto. Ej.: horas semanales totales dedicadas a actividades de desarrollo.
* **Organización**: los datos son compartidos a lo largo de la organización. Ej.: horas semanales promedio de desarrollo a lo largo de todos los proyectos.

ISO / IEC 15939 – 2007

Define un proceso de medición para el desarrollo de SW e ingeniería de sistemas.

Actividades:

1. Establecer y Mantener el Compromiso de Medición
2. Planificar el Proceso de Medición
3. Realizar el Proceso de Medición
4. Evaluar la Medición

**Unidad 4: Estimación del proyecto de SW**

Gestión de proyectos de SW

* La gestión de un proyecto de SW comienza con la planificación del proyecto. Antes de que el proyecto comience, el gestor y el equipo de SW deben realizar una estimación del trabajo a realizar, de los recursos necesarios y del tiempo que transcurrirá desde el comienzo hasta el final de su realización.
* Las métricas del proyecto y del proceso proporcionan una perspectiva histórica y una potente introducción para generar estimaciones cuantitativas. La experiencia anterior puede ayudar en gran medida al desarrollo y revisión de las estimaciones.
* La estimación de recursos, costos y planificación temporal de un esfuerzo en el desarrollo de SW requiere experiencia, acceder a una buena información histórica y el coraje de confiar en predicciones cuantitativas cuando todo lo que existe son datos cualitativos.
* La estimación conlleva un riesgo inherente y es este riesgo el que lleva a la incertidumbre.

Factores que afectan la fiabilidad de la estimación

* Complejidad del proyecto
  + Tiene un gran efecto en la incertidumbre. Sin embargo, la complejidad es una medida relativa que se ve afectada por la familiaridad con esfuerzos anteriores.
  + Existen medidas cuantitativas de la complejidad del SW, las cuales se aplican en el nivel de diseño y de codificación. Al comienzo del proceso de planificación se pueden establecer otras valoraciones de complejidad más subjetivas (por ejemplo, los factores de ajuste de la complejidad del punto de función).
* Tamaño del proyecto
  + Puede afectar a la precisión y a la eficiencia de las estimaciones.
  + A medida que el tamaño aumenta, crece rápidamente la interdependencia entre varios elementos del SW. El problema de la descomposición se hace más difícil porque los elementos descompuestos pueden ser todavía excesivamente grandes.
* Grado de incertidumbre estructural
  + La estructura se refiere al grado en el que los requisitos se han definido, la facilidad con la que pueden subdividirse funciones y la naturaleza jerárquica de la información que debe procesarse.
  + El riesgo se mide por el grado de incertidumbre en las estimaciones cuantitativas establecidas por recursos, coste y planificación temporal.
  + Si no se entiende bien el ámbito de proyecto o los requisitos del proyecto están sujetos a cambios, la incertidumbre y el riesgo son peligrosamente altos. El planificador del SW debería solicitar definiciones completas de rendimiento y de interfaz.
  + El planificador y el cliente deben tener presente que cualquier cambio en los requisitos del SW significa inestabilidad en el coste y en la planificación temporal.

Factores críticos de éxito

* Soporte ejecutivo.
* Involucramiento del cliente.
* Gerente de Proyecto experimentado.
* Objetivos de negocios claros.
* Alcances claros y acotados.
* Infraestructura tecnológica probada.
* Requerimientos consolidados.
* Proceso formal y maduro de desarrollo.
* Estimaciones realistas.

Estimaciones

* Una organización solo mejora en sus estimaciones cuando las usa correctamente. No existe una estimación exacta.
* Se crean, usan y refinan durante el planeamiento estratégico, factibilidad/propuesta, evaluación de sub-contratistas y planeamiento básico de proyecto (iterativo).
* La calidad de las estimaciones obviamente hace a la calidad del proyecto, pero las estimaciones se deben hacer en un momento donde la cantidad y calidad de la información es limitada.
* Es importante capturar las estimaciones al final del proyecto para tenerla en cuenta en proyectos futuros. Las mejores estimaciones se basan en la historia.
* Sobre-estimación:Ley de Parkinson (los seres humanos tienden a ocupar todo el tiempo asignado para una tarea), Costo inviable.
* Infra-estimación: escasez de recursos.
* Cambios: tecnología, nuevas funciones.

**Objetivo de la planificación**

Proporcionar un marco de trabajo que permita al gestor hacer estimaciones razonables de recursos, costo y planificación temporal. Se hacen al comienzo de un proyecto, y deberían actualizarse regularmente a medida que avanza el proyecto.

Las estimaciones deberían definir los escenarios del peor y mejor caso para limitar los resultados del proyecto.

Actividades de la planificación:

1. Determinar el **ámbito** del SW. Se debe delimitar su declaración.

Describe el control y los datos a procesar, la función, el rendimiento, las restricciones, las interfaces y la fiabilidad.

Te trabaja con las técnicas de recolección de información de los stakeholders.

1. Estimación de los **recursos** requeridos para acometer el esfuerzo de desarrollo de SW.

Cada recurso queda especificado mediante cuatro características: descripción, informe de disponibilidad, fecha cronológica en la que se requiere, tiempo que será aplicado.

* 1. Recursos Humanos: número de personas requeridas. Solo puede ser determinado después de hacer una estimación del esfuerzo de desarrollo.

Especificar la posición dentro de la organización y la especialidad.

* 1. Recursos de SW Reutilizables: creación y la reutilización de bloques de construcción de SW.

Pueden ser componentes ya desarrollados, componentes ya experimentados, componentes con experiencia parcial o componentes nuevos.

* 1. Recursos de Entorno: HW y SW donde se apoya el proyecto de SW.

**Estimaciones Seguras**

* Dejar la estimación para más adelante: no es factible. No se puede organizar un proyecto sin estimar previamente.
* Basar las estimaciones en proyectos similares ya terminados: puede funcionar bien si el proyecto actual es bastante similar a los pasados. Suele ser muy difícil encontrar otro proyecto similar en cuanto a características básicas, cliente, personal, etc.
* Utilizar técnicas de descomposición relativamente sencillas para generar las estimaciones de coste y de esfuerzo del proyecto: método viable. Enfoque “divide y vencerás”.

La precisión de una estimación del proyecto se predice basándose en:

* + El grado en el que el planificador ha estimado adecuadamente el tamaño del producto a construir.
  + La habilidad para traducir la estimación del tamaño en esfuerzo humano, tiempo y dinero (función de métricas fiables de proyectos anteriores).
  + El grado en el que el plan del proyecto refleja las habilidades del equipo de SW.
  + La estabilidad de los requisitos del SW y el entorno que soporta el esfuerzo de la ingeniería del SW.
* Utilizar uno o más modelos empíricos para la estimación del coste y esfuerzo del SW: método viable. Se pueden usar como complemento de las técnicas de descomposición. Estos modelos se basan en la experiencia.

Las opciones viables para la estimación serán buenas si los datos históricos utilizados como base para estimar son buenos.

Ideal: aplicar conjuntamente las técnicas indicadas; usando cada una de ellas como comprobación de las otras.

Herramientas Automáticas de Estimación

Las técnicas de descomposición y los modelos empíricos de estimación se pueden implementar con SW.

Todos realizan las siguientes funciones genéricas:

* Dimensionamiento de las entregas del proyecto. Se estima el tamaño de uno o más productos de SW.
* Selección de las actividades del proyecto. Se selecciona el marco de trabajo del proceso adecuado.
* Predicción de los niveles de la plantilla. Se especifica el número de personas disponibles para realizar el trabajo.
* Predicción del esfuerzo del SW.
* Predicción del coste del SW.
* Predicción de la planificación del SW.

**Tamaño del SW**

* En el contexto de la planificación de proyectos, se refiere a una producción cuantificable del proyecto de SW.
* Si se toma un enfoque directo, se puede medir en LDC. Desde un enfoque indirecto, se representa como PF.
* Diferentes enfoques del problema del tamaño:
  + Tamaño en lógica difusa: Para aplicar este enfoque, el planificador debe identificar el tipo de aplicación, establecer su magnitud en una escala cuantitativa y refinar la magnitud dentro del rango original.
  + Tamaño en punto de función.
  + Tamaño de componentes estándar: El planificador de proyectos estima el número de incidencias de cada uno de los componentes estándar, y utiliza datos de proyectos históricos para determinar el tamaño de entrega por componente estándar.
  + Tamaño del cambio: Se utiliza cuando un proyecto utiliza SW existente que se debe modificar. El planificador estima el número y tipo de modificaciones que se deben llevar a cabo.

Valor esperado para el tamaño del SW

* Es una media ponderada de las estimaciones optimistas, las más probables y las pesimistas.
* Se asume que existe una probabilidad muy pequeña en donde el resultado del tamaño real quedará fuera de los valores pesimistas y optimista.
* Una vez que se ha determinado el valor esperado de la variable de estimación, se aplican datos históricos de productividad de LDC y PF.
* No se puede estar seguro de que las estimaciones son correctas. Cualquier técnica de estimación se debe volver a comprobar con otro enfoque, aunque siempre va a prevalecer el sentido común y la experiencia.

Estimación basada en el proceso

* La técnica más común para estimar un proyecto es basar la estimación en el proceso que se va a utilizar. El proceso se descompone en un conjunto relativamente pequeño de actividades o tareas, y en el esfuerzo requerido para llevar a cabo la estimación de cada tarea.
* La estimación comienza con un esbozo de las funciones del SW obtenidas a partir del ámbito del proyecto. Para cada función se debe llevar a cabo una serie de actividades del proceso de SW.
* Si la estimación basada en el proceso se realiza independientemente de la estimación de LDC o PF, se tienen dos o tres estimaciones del coste y del esfuerzo que se pueden comparar y evaluar. Si ambos tipos muestran una concordancia razonable, se puede creer que las estimaciones son fiables. Si no, se debe investigar y analizar más.

Divergencias entre estimaciones:

Se deben a:

* No se entiende adecuadamente el ámbito del proyecto o el planificador lo interpretó mal.
* Los datos de productividad usados para técnicas de estimación basados en problemas no son apropiados para la aplicación, están obsoletos o se han aplicado erróneamente.

Modelos Empíricos de Estimación

* Un modelo de estimación para el SW de computadora utiliza fórmulas derivadas empíricamente para predecir el esfuerzo como una función de LDC o PF.
* Los datos empíricos que soportan la mayoría de los modelos de estimación se obtienen de una muestra limitada de proyectos. Por esta razón, el modelo de estimación no es adecuado para todas las clases de SW ni todos los entornos de desarrollo. Por ello, los resultados obtenidos de dichos modelos se deben utilizar con prudencia.

Estructura de los modelos de estimación

Un modelo común de estimación se extrae utilizando el análisis de regresión sobre los datos recopilados de proyectos de SW anteriores. Existen muchos modelos de estimación orientados a LDC y PF, pero cada uno producirá un resultado diferente para los mismos valores.

**Desarrollar para comprar**

A menudo es más rentable adquirir el SW que desarrollarlo. Los gestores de ingeniería del SW se enfrentan a la decisión de desarrollar o comprar que se puede complicar aún más con las opciones de adquisición:

* Se puede comprar con licencia ya desarrollado
* Se pueden adquirir componentes totalmente experimentados o parcialmente experimentados.
* Se puede construir de forma personalizada por una empresa externa.

Para productos de SW más caros, se pueden aplicar las directrices siguientes:

* Desarrollo de una especificación para la función y rendimiento del SW deseado. Definición de las características medibles, siempre que sea posible.
* Estimación del coste interno de desarrollo y la fecha de entrega.
  + Selección de tres o cuatro aplicaciones candidatos que cumplan mejor las especificaciones.
  + Selección de componentes de SW reutilizables que ayudarán en la construcción de la aplicación requerida.
* Desarrollo de una matriz de comparación que presente la comparación una a una de las funciones clave. Alternativamente, realizar el seguimiento de las pruebas de evaluación para comparar el SW candidato.
* Evaluación de cada paquete de SW o componente según la calidad de productos anteriores, soporte del vendedor, dirección del producto, reputación, etc.
* Contacto con otros usuarios de dicho SW y petición de opiniones.

Fuentes externas

* La decisión de contratar fuentes externas puede ser estratégica o táctica. Nivel estratégico: los gerentes consideran si una porción significativa de todo el trabajo de software puede contratarse a otros. Nivel táctico: un gerente de proyecto determina si parte o todo un proyecto puede lograrse mejor al subcontratar el trabajo.
* Ventajas: los ahorros se pueden lograr reduciendo el número de personas y las instalaciones que los apoyan.
* Desventajas**:** se pierde control sobre el SW que se necesita. Riesgo de poner el destino en manos de un tercero.

**Unidad 5: Gestión de la calidad**

La calidad está asociada con la satisfacción de alguien.

Gestión de la configuración del SW

Actividad de autoprotección. Las actividades de GCS sirven para identificar el cambio, controlarlo, garantizar que se implementa adecuadamente e informar del mismo a todos los interesados.

No hay calidad cuando

* El sistema cancela.
* No hace algo que el usuario pidió.
* Hace mal algo.
* Hace algo que el usuario aún no pidió.
* Es lento.
* Es difícil de usar.
* Cuesta mucho hacerlo.
* Cada cambio es difícil de introducir.
* Es feo.
* No sé cómo probar que funciona bien.
* Sólo yo puedo arreglarlo.
* Reinventa la rueda (empezar de nuevo).
* Es más complejo que lo necesario.

SW de calidad

Satisface expectativas del cliente y usuario, necesidades del equipo de desarrollo y mantenimiento, y de otros interesados en el producto o proceso. La calidad es para el cliente, el desarrollador, el usuario, el operador y el líder del proyecto.

Calidad: totalidad de aspectos y características de un producto o servicio que se sustentan en su capacidad de cumplir las necesidades especificadas o implícitas.

Visiones de la calidad

* Trascendente: la calidad es algo que se puede reconocer, pero no definir.
* De Usuario: adecuación para su uso, confiabilidad, performance, facilidad de uso. Depende de cada persona.
* De Construcción: mira la calidad del producto mientras se lo construye y luego de entregarlo.
* De Producto: usabilidad, integridad, eficiencia, correctitud, confiabilidad, facilidad de mantenimiento, facilidad de prueba, flexibilidad, portabilidad, interoperabilidad, facilidad de aprendizaje, facilidad de modificaciones.
* De Valor Agregado

**Aseguramiento de la calidad**

Diseño y planificación de acciones sistemáticas que se requieren para asegurar la calidad del software.

Responsables: ingenieros de software, gerentes y líderes de proyecto, clientes, comerciales, grupo de SQA.

Propósito: suministrar a la gerencia una visibilidad del proceso que está siendo usado por el proyecto y de los productos que se construyen. Implica la revisión y auditoría de los productos y actividades para verificar que cumplen con los procedimientos y estándares aplicables.

Aseguramiento de la calidad: proceso, estándares, control de cambios, verificación y validación, revisiones y auditorías, gestión de proyectos, métricas.

Propiedades

Las visiones permiten definir modelos de calidad (conjunto de características), que permiten definir propiedades del producto y el proceso. En base a estas, se definen métricas según las cuales se realiza la gestión de calidad para asegurarla.

Características

* Funcionalidad
* Confiabilidad
* Usabilidad
* Eficiencia
* Mantenibilidad
* Portabilidad

**Proceso**

Conjunto de herramientas, métodos, y prácticas que usamos para construir un producto.

La calidad del software está fuertemente determinada por la calidad del proceso usado para desarrollarlo y mantenerlo.

Proceso de desarrollo del SW

Aplicación de un método para el desarrollo de software por parte de las personas con el entrenamiento adecuado. No se debe confundir la definición de un proceso (deseo) con el proceso real (modo en que efectivamente se hace).

Cuando no hay proceso no es posible planear y controlar las actividades, ni asegurar sus resultados.

Cómo es un proceso para un proyecto

MODELOS

Ingeniería de Requerimientos

Desarrollo

Prueba de Aceptación

Instalación

Plan de Prueba

MODELOS

SISTEMA

SISTEMA PROBADO

SISTEMA FUNCIONANDO

CONDICIONES Y CASOS DE PRUEBA

Soporte al proceso

* Administración y control de proyectos
* Administración de configuraciones
* Aseguramiento de la calidad

Estructura del aseguramiento de la calidad: se lleva a cabo durante todo el ciclo de desarrollo

Costo de no tener calidad

* Del usuario: mayor costo de administración, no poder hacer ciertas cosas, no llegar a tiempo al mercado, perder imagen con el cliente, no poder confiar en la herramienta.
* De sistemas: mantenimiento fuera de hora, sobrecarga, incumplimiento de plazos, quejas, menor credibilidad.

Economía de la calidad: un producto de calidad requiere de menos tiempo y esfuerzo (tiene menos errores, se tarda menos en hacer que funcione, tiene menos fallas, se tarda menos en arreglarlo). Provoca satisfacción en el usuario.

SW de calidad: es aquel que cumpla con los requisitos (que al ser usado no tenga fallas).

Error: diferencia entre un valor o condición computada y el valor o condición verdadera o teóricamente correcta.

Definiciones de error

* Equivocación: acción humana que produce un resultado incorrecto.
* Defecto: paso, proceso, o definición de dato incorrecto. Ausencia de cierta característica.
* Falla: resultado de ejecución incorrecto. Es el producido por el software distinto al resultado esperado.

Verificación y Validación

* Validación: ¿Estamos haciendo el producto correcto? Prueba el cliente.
* Verificación: ¿Estamos haciendo el producto correctamente? Test, casos de prueba.

**Gestión de la calidad**

Pruebas de Software: ejecutar un programa con el objetivo de encontrar defectos en él. Las pruebas sólo pueden mostrar la presencia de defectos, no su ausencia. Una prueba es exitosa si encuentra defectos (o fallas, no sabemos).

Proceso de prueba: las pruebas siempre deben realizarse contra un resultado esperado.

Economía de la prueba

Se puede invertir mucho esfuerzo en probar

Probar es el proceso de establecer confianza en que un programa hace lo que se supone que tiene que hacer, ya que nunca es posible demostrar que un programa es correcto.

Continuar probando es una decisión económica. Muchas organizaciones concentran sus esfuerzos en arreglar los errores en vez de prevenirlos.

**Unidad 6: Gestión de la Configuración**

**Gestión de Configuración del SW**

Es el arte de identificar, organizar y controlar las modificaciones que sufre el SW que se construye.

Es una actividad de autoprotección. La meta es maximizar la productividad minimizando los errores.

El Cambio: en cualquier momento y por cualquier razón el sistema cambiará y el deseo de cambiarlo persistirá a lo largo de todo el ciclo de vida. La GCS sirve para identificar y controlar el cambio, garantiza que se implemente adecuadamente e informa a todos sus interesados.

Fuentes del Cambio: nuevas condiciones en el negocio, nuevas necesidades del cliente, restricciones presupuestarias y la reorganización, crecimiento o reducción del negocio.

Mantenimiento vs GCS

Mantenimiento: actividades de IS que se producen despuésde que el SW se entregó al cliente y esté en funcionamiento.

GCS: actividades de seguimiento y control que comienzan cuando se inicia el proyecto de IS y termina sólo cuando el SW queda fuera de circulación.

Configuración del SW

Son los elementos que componen todala información producida como parte del proceso de ingeniería del SW.

Información producida:

* **Programas** de computadora (código fuente y ejecutable).
* **Documentos** que describen los programas de computadora (técnicos y de usuario).
* **Datos** (contenidos en el programa o externos).

Elementos de configuración del SW

Toda información creada como parte del proceso de ingeniería de SW es un elemento (documento, conjunto completo de casos de pruebas, componentes, etc.).

* Elementos componentes: Herramientas que permiten el acceso y gestión de cada ítem de CS.
* Elementos de proceso: Procedimientos y tareas para gestionar el cambio.
* Elementos de construcción: Herramientas que automatizan la construcción del SW.
* Elementos humanos: Conjunto de herramientas y procesos utilizados por el equipo de trabajo.

Cada elemento que está bajo la GCS se denomina ítem de configuración. No todas las entidades necesitan ser controladas todo el tiempo.

Debe decidirse cuándo un elemento pasa a estar bajo control.

* Demasiado temprano → Burocracia (costo y tiempo).
* Demasiado tarde → Caos.

**Línea Base**

* Es una especificación o producto que se ha revisado formalmente y sobre los que se ha llegado a un acuerdo, y que de ahí en adelante sirve como base para un desarrollo posterior. Solamente puede cambiarse a través de procedimientos formales de control de cambios.
* Un elemento de configuración se convierte en línea base si fue revisado y aprobado.
* Un cambioes el paso de una línea base a la siguiente.

Líneas bases típicas

* Especificación del sistema.
* Plan del Proyecto de SW.
* Especificación de Requisitos del SW.
* Especificaciones de diseño (módulos, interfaces).
* Código fuente.
* Planes/Procedimientos.
* Datos de prueba.
* Sistema en funcionamiento.

**Conceptos básicos**

* Visibilidad: Dada la naturaleza abstracta del SW, es fácil ignorar qué contiene al evolucionar desde los requisitos al código. La visibilidadse consigue mediante la identificación unívoca de todos los productos SW y sus relaciones.
* Trazabilidad: Todas las representaciones de un producto de SW a lo largo de su ciclo de vida deben ser consistentes.
* Versión: Versión es la forma particular que adopta un objeto en un contexto dado. Desde el punto de vista de evolución, es la forma particular de un objeto en un instante dado. Se suele denominar revisión.
* Configuración: Un sistema SW comprende distintos componentes (HW y SW), que evolucionan individualmente. Por lo tanto, hay que garantizar la consistencia del conjuntodel sistema.

Una configuración es una combinación de versiones particulares de los componentes que forman un sistema consistente. Desde el punto de vista de evolución, es el conjunto de las versiones de los objetos componentes en un instante dado.

* Gestión de la Configuración (GC): Es una disciplina para la identificación de la configuración de un sistema que aplica conceptos técnicos y administrativos para identificar y documentar características funcionales y físicas de un ítem de configuración, controlar cambios en las características, registrar y reportar procesamiento de cambios y estado de la implementación y verificar adecuación con requerimientos especificados.

Tiene el objetivo de controlar sistemáticamente los cambios a la configuración y mantener la integridad y trazabilidad de la configuración durante el ciclo de vida del sistema.

* GCS: Es la Gestión de la Configuración aplicada específicamente al SW. Permite sistematizar el control de cambios de la configuración y mantener la integridad y rastreabilidad de la configuración a través del ciclo de vida del SW.

Es una disciplina para gestionar la evolución de los productos de SW durante todo el ciclo de vida.

**Gestión de la Configuración del SW**

Es un elemento importante de garantía de calidad. Es una etapa de soporte en el proceso de producción de SW que cubre a todas las demás áreas. Su responsabilidad principal es el control de cambios. Ayuda en la identificación de las versiones de los artefactos que componen una versión específica del sistema.

Cinco tareas bases: identificación, control de versiones, control de cambios, auditorías de configuración y generación de informes.

Identificación

* Elementos que se van a controlar: todos los artefactos que son necesarios para el futuro mantenimiento del sistema 🡪 ítems de configuración
* Ejemplos: plan de proyecto, especificación de requerimientos, documento de arquitectura, diseño de BD, componentes del sistema (código), casos de prueba, manual de usuario, especificaciones de despliegue, HW requerido, etc.
* Cada elemento tiene nombre, descripción, lista de recursos y relaciones.

Control de versiones

* Combina procedimientos y herramientas para gestionar las versiones de los elementos de configuración. Asegura que los cambios hechos a dichas versiones no interfieran unos con otros.
* Los sistemas de control de versiones brindan, además, la facilitad de construir una versión específica del SW y el seguimiento de conflictos.
* Cada versión del SW es una colección de elementos de configuración y cada versión puede estar compuesta por distintas variantes.
* Modelo ortogonal de versiones
  + Codeline: secuencia de versiones de código fuente con versiones posteriores derivadas de versiones anteriores. Aplican regularmente a componentes de sistemas, de manera que existen diferentes versiones de cada uno.
  + Línea base: es una definición de un sistema específico. Incluye las versiones de los componentes más las librerías utilizadas, archivos de configuración, etc. Son importantes porque puede ser necesario volver a crear una versión específica de un sistema completo.
  + Línea principal: Secuencia de versiones de sistema desarrolladas desde una línea base original.
* Identificación de versiones: Cada organización define su nomenclatura para los números de versión.
  + <id><9>.<9>.<9>.<9> 🡪 Ejemplo: Id 1.2.3.4
    - 1: reléase: modificación importante del producto.
    - 2: nivel: modificación leve en el producto.
    - 3: versión: corrección de un fallo o problema.
    - 4: instalación: modificación debido al entorno donde se instalará.
  + Dos números - MAYOR.MENOR
    - Número mayor: grandes cambios.
    - Número menor, pequeños cambios.
  + MAYOR.MENOR.MICRO.Fase de desarrollo(alpha, beta, etc.). Ejemplo: 1.2.2-Beta
    - Micro: corrección de SW con pocos o ningún cambio.
  + MAYOR.MENOR.MICRO.BUILD
    - Build: revisión del código fuente sin cambios sustanciales.

Control de cambios

* Combinación de procedimientos humanos y herramientas automatizadas para proporcionar un mecanismo para identificar, controlar el acceso y manejar cambios tanto al producto como a la línea base.
* Sirve para establecer las versiones básicas del producto y del proyecto.
* Provee formas de controlar los requerimientos de cambio internos y externos que afecten las versiones básicas y asegura que los cambios son hechos a las versiones básicas de los productos una vez que se han aprobado.
* Un proceso de administración de cambios asegura que los costos y beneficios del cambio sean adecuados y que el cambio se implemente correctamente.
* Proceso:
  + Pedir del cambio.
  + Evaluar el cambio: esfuerzo técnico, efectos secundarios, impacto sobre otros componentes, costos.
  + Informar los resultados de la evaluación a la Autoridad de Control de Cambio, la cual lo aprueba o deniega.
  + Generar una orden de cambio de ingeniería para cada cambio: qué se cambiará, restricciones, criterios de revisión y auditoría.
  + Realizar el cambio.
  + Revisar el cambio (auditoría).

Auditoría de la configuración

* Verifica que, en un momento dado, el sistema en desarrollo es una colección de productos consistente y bien definidos.
* Determina que todos los elementos de configuración están presentes en la línea base del SW, estableciendo la correctitud de la versión de cada elemento de configuración.
* Tipos:
  + Auditoría Física: verifica que el SW y su documentación son internamente consistentes y están listos para ser entregados al cliente.
  + Auditoría Funcional: evaluación independiente de los ítems de configuración para verificar que la funcionalidad y la performance sean consistentes con la ERS.
  + Auditoría de Rastreabilidad: verifica si a lo largo del proceso de desarrollo se mantuvo la rastreabilidad entre los componentes.

Generación de informes

Los informes intentan responder las siguientes preguntas: ¿qué pasó? ¿quién lo hizo? ¿cuándo? ¿qué más se vio afectado?

* Informe de estado de la configuración: registro y reporte de la información necesaria para gestionar eficientemente la configuración.
  + Estado de cada cambio propuesto.
  + Momento en que se incorpora un cambio de configuración del SW.
  + Deficiencias detectadas durante la auditoria de configuración del SW.
  + Información descriptiva de cada cambio propuesto.

**Ventajas de GCS**

* Como una herramienta de control permite mantener la integridad de los elementos y evaluar y ejecutar los cambios en un ambiente controlado.
* Como una herramienta de visibilidad proporciona evidencia objetiva y concreta de la creación y evolución del producto y permite que las inspecciones y auditorías de la configuración establezcan el estado de avance real del proyecto.
* Como una herramienta de reducción de costos permite la reducción de los costos de desarrollo, ayudando a mantener el “orden” en el proyecto y a reducción de los costos de mantenimiento, asegurando la integridad del SW en la operación y la actualización y consistencia de toda la documentación.
* Como una herramienta de apoyo a la administración del proyecto incrementa la eficiencia y la efectividad de la administración y la efectividad de sus disciplinas.