Introducción

- **Ingeniería de SW**: Aplicación de un proceso sistemático, cuantificable y disciplinado a la creación, desarrollo, la operación y mantenimiento de software.
- **Software: evolución** de la artesanía a la ingeniería
    - Desarrollo artesanal:
        - Virtuosos con talento
        - Progreso al azar
        - Intuición
        - Manufactura para uso más que para ventas
        - Uso no planificado y exagerado de recursos
        - Sello personal, no repetible
    - Explotación comercial: Artesanal + Producción
        - Artesanos con habilidades
        - Procedimientos establecidos
        - Refinamiento pragmático
        - Entrenamiento
        - Preocupación por costos
        - Manufactura para venta
    - Ingeniería Profesional: Comercialización + Ciencia
        - Profesionales educados
        - Progreso basado en la ciencia
        - Análisis y teoría
        - Segmentación de mercado
        - Repetible
        - Medible
- **Ingeniero vs Científico**
    - Un científico construye para aprender, un ingeniero aprende para construir
    - El ingeniero trabaja con restricciones que un científico no tiene en cuenta. Ej: Costos, tiempos, presupuestos.
    - Un científico busca el conocimiento, un ingeniero utiliza el conocimiento para crear.
- **The mythical man month** (Addison Wesley)
    - Existen distintos productos de SW dependiendo del usuario final. Requiriendo más o menos diseño, integración o pruebas, implicando diferentes grados de esfuerzos.
        - Producto de programación (usado por otro), requiere tres veces más esfuerzo que un programa de uso privado. Ej: test, doc, mantenimiento
        - Componente de un sistema, requiere tres veces más esfuerzo que un programa de uso privado. Ej: integración, interface. (Ejecutar varios programas para lograr un objetivo)
        - Producto de Sistema de Programación: Requiere nueve veces más esfuerzo que un programa de uso privado.

- Problemas habituales en el desarrollo y manteniendo de SW: Se retrasan, no se terminan, excede el costo estimado, no cumple con las expectativas del cliente, no cumple con los requerimientos de calidad mínimos.
- Razones de los problemas: No hay administración y control de proyectos, el éxito depende de los gurúes, se confunde la solución con los requerimientos, estimaciones empíricas, no hay historia sobre proyectos realizados, calidad no medible, no se consideran riesgos, se asumen compromisos imposibles de cumplir, se comienzan proyectos con requerimientos no claros.
- Reacciones: Se busca al gurú, agregando gente a un proyecto retrasado, recortado / eliminando cualquier actividad que genere documentación, recortando etapa de Testing, Recortando cualquier actividad que no sea codificar, asumiendo definiciones por el usuario, recurriendo a una "bala de plata" (Nuevo paradigma, nueva herramienta, nuevo lenguaje).
- **Técnica que podemos utilizar para los problemas**: Diagrama Causa-Efecto (Espina de pescado) permite identificar las posibles causas relacionadas a un problema, con el objetivo de descubrir sus raíces. (Concentra la atención en las causas y no en los síntomas)
- **Los problemas con que nos encontramos en el desarrollo y mantenimiento de SW mayormente no dependen de la tecnología.**
- **Classic Mistakes** (Errores clásicos)
    - Es una lista de errores comunes que debemos tener presente para no repetir errores conocidos.
    - TOP 10 frecuentes
        - Expectativas irrealistas,
        - cronograma optimista
        - recortar QA
        - ilusión (Wishful thinking)
        - Confundir estimaciones con objetivos
        - Excesivo multitarea
        - Aumento de los requerimientos (por parte de los desarrolladores)
        - Oficinas ruidosas y concurridas
        - Abandonar la planificación
        - Insuficiente manejo de riesgos
- SWEBOK: Guia al conocimiento presente en el área de la ingeniera del software.

Modelo de Calidad

- **Calidad**: Grado en que un sistema, componente o proceso satisface las necesidades o expectativas del cliente o del usuario ó cumple con los requisitos especificados.
- **Calidad SW**: Grado en que el SW posee una combinación desea de atributos
- Modelos de Calidad en SW
    - **Calidad de Producto**: (No se tiene en cuenta el precio)

- Modelo de Mc Call: Organiza los factores en tres ejes (Operación, Revisión, Transición) con 11 factores de calidad y cada factor se desglosa en otros criterios (23), obteniendo así las métricas.
- Modelo de Barry Boehm: Empieza por la utilidad general y se va abriendo en subcategorizas (Mantenibilidad, Usabilidad, portabilidad) para obtener las métricas.
- ISO 9126: Estándar internacional para definir y medir calidad en productos software. Expone características internas y externas de la calidad, en 6 grandes grupos, que se subdividen en 27 categorías para dar las métricas externas e internas.
    - Grupos: **Funcionalidad**, **Confiabilidad** (Capacidad del software de mantener su nivel de prestación bajo condiciones establecidas. Ej: tolerancia a fallos, recuperabilidad), **Usabilidad** (Esfuerzo necesario para su uso), **Eficiencia**, **Mantenibilidad** (extender, modificar, corregir errores), **Portabilidad** (Ser transferido y adaptado desde una plataforma a otra)
    - Pasos
        - Designar pesos a los atributos y subatributos
        - Crear métricas para cada atributo para pasar los cualitativo a cuantitativo
    - Usos: Medir, Comparar, Guía en las decisiones
    - Se diferencia del resto de los modelos por ser un estandar internacional
    - Hay atributos de calidad que compiten entre ellos. Ej: Seguridad vs Funcionalidad
    - -2: Calidad externa ; -3 Calidad Interna; -4: Calidad en uso (aceptación por parte del usuario)
- Los modelos relación factores de calidad externos de alto nivel (Usuarios, clientes, gerentes) con subfactores de calidad de menor nivel que representan atributos del sw o proceso de producción (Técnicos, programadores). Y se definen métricas para cada subfactor, pudiendo ser posible también la definición de métricas para los factores de alto nivel.

- **Calidad de Proceso**:
    - CMMI
        - No es un proceso para desarrollar software, no es un ciclo de vida, no es una metodologia
        - Es un modelo para la mejora y evaluación (madurez) de procesos para el desarrollo, mantenimiento y operaciones de sistemas de software (Conjunto de modelos, métodos de evaluación y entrenamientos).
        - Nos dice QUE hacer, pero no COMO ni QUIEN
        - Institucionalizado: Inconscientemente competente.
        - Proceso maduro: Soportado por la gerencia, documentado, infraestructura adecuada para soportarlo, medido y controlado,

presupuestos y plazos creíbles, riesgos administrados, organización proactiva.
    - Proceso Inmaduro: Improvisado, informal, no hay entrenamiento ni herramientas para sustentarlo
    - Representaciones
        - Continua: Máxima flexibilidad para enfocarse en un área de proceso especifica de acuerdo los objetivos del negocio.
        - Por niveles: Da una hoja de ruta a implementar: grupo de aéreas de proceso, secuencia de implementación.
        - El contenido de ambas es el mismo, cambia su organización. Hay que analizar cuál es la representación que más se adapta a los objetivos de la organización.
    - Por cada metas hay practicas especificas las cual se evalúan y se le dan un puntaje según su madurez.
    - Consistencia entre lo que hacen, dicen que hacen y está escrito.
- **Visiones de la Calidad**
    - Visión Manufactura (Desarrollo del SW):
        - Se compara con los requerimientos, su cumple con ellos => calidad
        - Mayor calidad implica menor costo porque no hay desperdicios, merma.
        - Busco optimizar el proceso (En SW intento que cada paso sea consistente con los demás). (Objetiva y cuantificable)
    - Visión Producto (Diseño del SW):
        - Por las características del producto (Escalable, portable, ect)
        - Mas características => más calidad => mayor costo
    - Visión Usuario (Relevamiento):
        - Depende para que lo quiera
        - Que tanto se adapta a las necesidades del usuario
    - Visión basada en Valor (Post-Implementación):
        - Valor de uso (Que yo le doy) vs Precio
        - Balanza entre valor y precio
        - Algo tiene buena calidad si alguien me compra algo a cierto precio
        - Mientras mayor es el costo, su calidad percibida baja (Si no representa un buen valor para el cliente, su calidad baja).
        - Ej: Percibes la calidad de un par de zapatos, pero al ver el precio pierdes interés en comprarlos. No representa un buen valor para el cliente.
        - Ej: Windows Pago vs Linux Free
    - Visión trascendental (Post-Implementación / Diseño):
        - Visión filosófica, abstracta
        - Sabemos que es de calidad por los sentidos humanos, pero no de forma racional.
        - Ej: Es tal marca
- Mayor calidad no implica mayor costo, depende de la visión con la que se trabaja. Ej: Visión manufactura, mayor calidad implica menor costo.
- Costo de Calidad: Compuesto por los costos de conformidad (Prevención y evaluación) y costos de no conformidad (Fallas internas y Fallas externas)

- Costos de conformidad: Costos asociados con el logro de la calidad. A menor costo obtenemos mayor número de fallas.
  - Prevención: Costos asociados con la prevención de defectos. (Ej: capacitación, revisiones tempranas, planificación, análisis)
  - Evaluación: Costos asociados con analizar y probar el producto. (Ej: Test, inspecciones, control de calidad)
- Costos de no conformidad: Costos de lidiar con defectos. A menor número de fallas, menor costo. (Varia si el producto es crítico o no)
  - Internos:
  - Externos:
- Nos ayuda a entender el costo-beneficio de la tener calidad vs no tener calidad mediante un modelo métrico, y determinar un rango de operación.

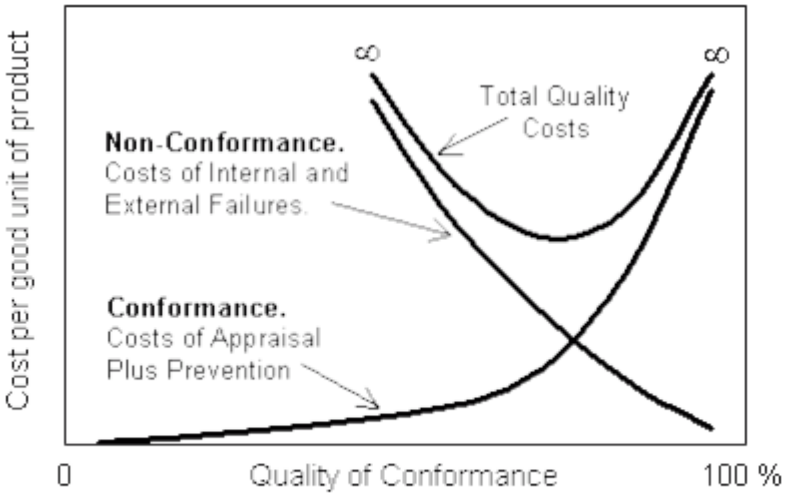
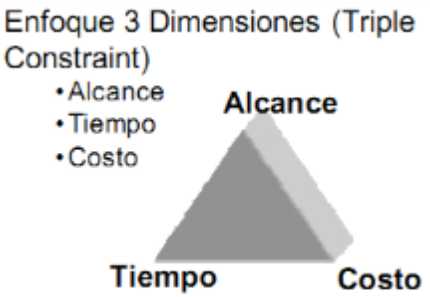


Figure 18.a. Traditional Cost of Quality Model

- **Pirámide Calidad**: (Base -> Punta) Exactitud (requerimientos troncales), Fiabilidad y disponibilidad, usabilidad ;;; Capacidad (Ej: Hacer la tarea de forma eficiente y rápida), Disfrute (Ej: estético)

Planificación de Proyectos

- Proyecto: Esfuerzo temporal emprendido para crear un producto o servicio único para lograr un objetivo. (Con sucesión de actividades o fases en las que se coordinan los distintos recursos).
- Gestión de Proyectos: disciplina que consiste en planificar, organizar, obtener y controlar recursos, utilizando herramientas y técnicas para lograr que el proyecto logre sus objetivos en tiempo y forma.
- Los recursos son limitados, la relacion entre los elementos pueden verse en el siguiente grafico



- Composición de un proyecto: Resumen Ejecutivo, Objetivos (de negocios / proyecto), Alcance, riesgos del proyecto, calendario, estimaciones, recursos, estructura y roles, procedimientos de control y seguimiento.
    - Objetivo: (No confundir con solucion), describe los resultados de negocio esperados.
        - SMART (Especifico, medible, alcanzable, realista, acotado en el tiempo)
    - Alcance: Define los límites del sistema (Que incluye y que no incluye)
- **Riesgos**
    - Un riesgo es un problema que todavía no llegó e impacta en alguno de los objetivos del proyecto
    - Un problema es un riesgo que se manifestó
    - Nunca puede ser eliminado por completo
    - Riesgo: Probabilidad de sufrir una perdida (Calidad, incremento costo, incremento tiempo)
    - Exposición de un riesgo: (probabilidad de ocurrencia) * (impacto negativo)
    - Administrador de Riesgos
        - Involucra todas las tareas relacionadas con la identificación, la resolución y comunicación de los riesgos.
        - No involucra decisiones futuras, incluye todas las decisiones presentes que tienen incidencia en el futuro
        - Toma de decisiones bajo niveles de incertidumbre
        - -> Identificación -> Análisis -> Planificación -> Seguimiento -> Control ->
            - **Identificación**: Dado que CONDICION entonces (posiblemente) CONSECUENCIA
            - **Análisis**: Convertir la información de riesgos que se identifico en información que permita tomar decisiones. (Probabilidad de impacto, causas y acciones correctivas, identificar causas comunes, identificar tiempos de ocurrencia).
            - **Planificacion**
                - Evitar el riesgo: Cambiar las condiciones originales de ese evento para eliminar totalmente el riesgo identificado.
                - Transferir el riesgo: Trasladar el impacto negativo a un tercero. Ej: Seguros
                - Mitigar el riesgo: Bajar probabilidad de ocurrencia
                - Plan de contingencia: Establecer un disparador para bajar el impacto. "trigger".
                - Aceptar el riesgo: No cambiar el plan original.

- **Seguimiento:** Monitoreo, aplicar métricas sobre presupuesto, calendario y consideraciones técnicas, informar desvíos respecto de los objetivos e identificar nuevos riesgos.
- **Control**: (SQA) Realizar las correcciones de las desviaciones
- **Comunicación**: Feedback sobre las actividades sobre los riesgos. Para ser analizados y administrados comunicarlos a los niveles adecuados.

  - El plan del riesgo debe ser más barato que el costo del riesgo
- **WBS**: Método para representar en forma jerárquica los componentes de un proceso (en nuestro caso el plan de proyecto) o un producto. (No determina dependencias, solo jerarquía, tiene que tener todas las tareas o entregables para cumplir con el alcance)
  - **Nodo raiz** (Nombre del proyecto), dividirlo en los **componentes principales**, **subdividir** los componentes de forma jerárquica hasta que se llegue a un nivel que se pueda trabajar con el nivel de granularidad alcanzado (En nuestro caso hasta que para cada tarea **podamos estimar esfuerzo, su duracion y asignarle recursos**)
  - Tipos de WBS
    - A nivel conceptual: Por **proceso** (analisis / desarrollo / construccion / testing / implementacion) ; **Producto** (funcionalidades) ; Hibridas.
    - Por formato: Jerárquico o tabla indentada (dot list numeric)
- **Determinación de las dependencias**
  - WBS -> Dependencias
  - Cada tarea tenga un entregable preciso (Que se pueda definir cuando la tarea finalizó)
  - Identificar camino critico
  - Las precedencias puede ser
    - Recursos (Ej: tengo un solo programador y lo necesito para varias tareas)
    - Tareas: (Ej: necesito terminar una para empezar la siguiente)
- **Asignación de Recursos**
  - Debemos conocer los recursos con los que podemos contar y cual es su disponibilidad real para asignarlos al proyecto
- **Tipos de planificación**
  - Ciclos de vida agiles (RUP, Scrum Puro, XP) tienen más adaptabilidad
  - Ciclos de vida basados en planificación (Spiral, Cascada por fases, Cascada puro) tienen más predictibilidad.
  - Panificación Adaptativa (Agil)
    - Plantea la naturaleza cambiante de la realidad y los sistemas, tratando de acomodarse a esos cambios.
    - Se adaptan facilmente a los cambios de alcance y prioridades
    - Planifica a corto plazo (se hace dificil determinar el fin del proyecto)
    - Esquema iterativo (En cada iteración se define que funcionalidades se completaran en la misma)
    - Complejo de utilizarlo en proyectos grandes o críticos
    - Involucramiento constante del usuario / cliente
    - Se recomienda usar: baja criticidad, equipo con experiencia, requerimientos volatiles, equipo pequeño a mediano, cultura que acepta el caos, front-end

- Planificación predictiva (tradicional): Se recomienda usar en proyectos críticos, equipo con minoria de expertos, el alcance es conocido y estable, gran número de personas en el equipo para coordinar, cultura que requiere más orden, back-end

• Estimaciones

- Estimar: calcular o determinar el valor de algo.
- **Conceptos Confusos**
  - Objetivo de la organización (Target): ¿Cuando lo necesito?
  - Estimación: ¿Cuando puedo entregarlo?
  - Compromiso: ¿Cuando debo entregarlo?
- **Variables a estimar** (En orden): Tamaño, esfuerzo -> costo, Cronograma. Cuanto más refinada la definición, más exacta será la estimación.
- **Esfuerzo de una actividad**: Suma de los tiempos que le dedicaran los diferentes recursos a cierta actividad, medida en Hora/Hombres (Cantidad de horas hombre que realiza un solo hombre para realizar dicha actividad). El **tiempo de una actividad** es el tiempo calendario que conlleva dicha actividad.
- El proceso de estimar
  - Requerimientos [(Producto) => Tamaño (Producto)] => [Esfuerzo (Proceso) => Duración (Proceso)]
  - Primer grupo de corchete: Igual para todos; Segundo grupo: Depende del equipo
- Cuando las personas estiman buscan similitudes y diferencias con proyectos previos. Es decir, es necesario historias de proyectos pasados.
- Un método creíble debe ser caja blanco
- **Ciclo estimación**: -> Estimar (T/E/Cr) -> Medir (Valores reales) -> Registrar (Estimados y Reales) -> Comparar -> Analizar -> Calibrar ->
- **Métodos para estimar**
  - Métodos no paramétricos:
    - Juicio experto
      - Se basa en la experiencia personal (Netamente estomacal)
      - Por lo general se recurre a analogías
    - Pert (También conocido como Clark)
      - Se basa en el jucio experto
      - Dividir el sistema en módulos y estimar para cada modulo.
      - Incluye factores optimistas y pesimistas en su estimación
      - Estimación = (Optimista + 4*Medio + Pesimista) / 6
      - Se puede usar para estimar tamaño y esfuerzo
      - Top-Down Por analogías
    - Wideband Delphi
      - Juicio experto en grupo
      - Un grupo de expertos y un coordinador se juntan en una sala. El coordinador presenta a cada experto la especificación y un formulario de estimación, luego de que debatir, los expertos estiman y entregan de forma anónima. El coordinador hace un resumen de las estimaciones y las reparte. Se realiza esto en varios

ciclos y da como resultado una estimación media o un rango de estimación.
        - Se recomienda usar en proyectos pocos conocidos donde no se cuenta con historia.
        - Bottom-Up por descomposición
- Métodos paramétricos: Dados valores de entrada (parámetros), una función nos da la estimación
    - Puntos de Función
        - Miden el tamaño del SW en base a la funcionad capturada en los requerimientos
        - Los PFs son independientes de la tecnología
        - Requieren un análisis de requerimientos avanzados
        - Se identifican los elementos del sistema, se calculan los FP sin ajustar, se calculan los factores de influencia, se calcula los puntos de función.
    - Puntos de casos de usos
        - Basado en el método de puntos de función
        - Depende de: Cantidad y complejidad de los casos de uso, de los actores intervinientes en el sistema, factores técnicos y ambientales.
        - Requiere que sea posible contar el numero de transacciones encada caso de uso. Transacción: Evento que ocurre entre el actor y el sistema.
        - Se clasifican a los actores, se califica los casos de uso, se aplican factores de ajustes, se calculan los UCP.
    - Puntos de objeto
        - Se asigna a cada componente un peso, de acuerdo a la clasificación por su complejidad. Se utiliza como factor de ajuste el porcentaje de reutilización de código.
        - Se identifican los elementos del sistema, se calcula la complejidad, se aplica el factor de ajuste, se calcula el Objet points
    - CoCoMO
        - Modelo empírico
        - Toma en cuenta el proyecto, producto, el HW y la experiencia de los RRHH
        - Bien documentado y de dominio público