

RESUMEN 1ER PARCIAL

IEEE (Institute of Electrical and Electronics Engineers)

La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software

Se ocupa de la producción sistemática y el mantenimiento de productos de software que son desarrollados en tiempo y forma

SEI Software Engineering Institute



Ingeniería es la aplicación sistemática del conocimiento científico en la creación y construcción de soluciones '**cost effective**' para resolver problemas prácticos al servicio del hombre

Ing en SW es a parte de la ing. que aplica los principios de las ciencias de la computación y las matemáticas para alcanzar las soluciones cost effective a problemas de software

Body of Knowledge BoK - SWEBoK Software Engineering Body of Knowledge

Cuerpo de conocimiento es un requisito para calificar cualquier área de ingeniería como profesión.

Describe el conocimiento en Ing de SW. Comenzó en 1998 liderado por la IEEE para convertir la ing del SW en una disciplina legítima.

SWEBoK 3.0 es la más actual del 2014. Tiene reconocimiento internacional

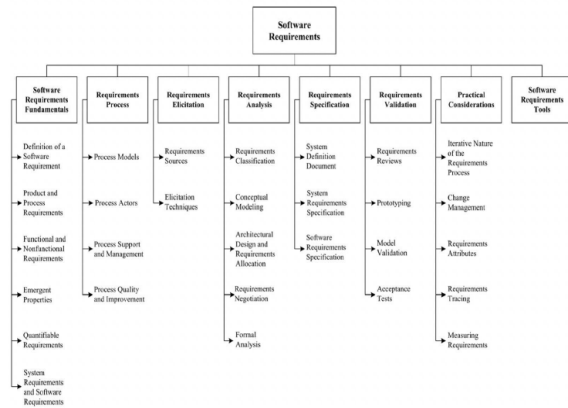
En general esas definiciones comparten los siguientes puntos:

- Crear soluciones cost-effective
- A problemas prácticos (concretos)
- Aplicando conocimiento científico / codificado
- Construyendo software que sirva al hombre

La ingeniería permite a una persona común hacer cosas que antes requería de gente virtuosa

The 15 SWEBOK Requirements Key Areas

- 1 - Software Requirements
- 2 - Software Design
- 3 - Software Construction
- 4 - Software Testing
- 5 - Software Maintenance
- 6 - Software Configuration Management
- 7 - Software Engineering Management
- 8 - Software Engineering Process
- 9 - Software Engineering Models and Methods
- 10 - Software Quality
- 11 - Software Engineering Professional Practice
- 12 - Software Engineering Economics
- 13 - Computing Foundations
- 14 - Mathematical Foundations
- 15 - Engineering Foundations



Problemas habituales en el desarrollo y mantenimiento de SW

- Los proyectos se terminan con mucho atraso, o no se terminan
- Los proyectos se exceden de costo estimado
- El producto final del proyecto no cumple con las expectativas del cliente
- El producto final no cumple con los requisitos de calidad mínimos esperados

Causas comunes de los problemas

- No hay administración y control de los proyectos
 - No se puede controlar lo que no se mide

- El éxito depende de los gurúes
- Se confunde la solución con los requerimientos
- Las estimaciones son empíricas y no hay historias sobre proyectos realizados
- La calidad es una noción netamente subjetiva que no se puede medir
- No se consideran los riegos
- Se asumen compromisos imposibles de cumplir
- Las actividades de mantenimiento van degradando el software
- Se comienzan los proyectos con req no claros ni estables

Reacciones frecuentes

- Buscando al guru que nos saque del problema
- Agregando gente al proyecto ya atrasado
- Recortando/eliminando las actividades que generan documentación
- Recortando/eliminando el testing
- Recortando/eliminando lo que no sea codear
- Asumiendo definiciones en lugar de usuario

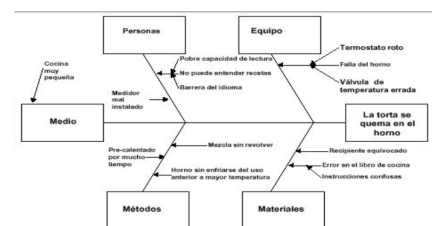
Diagrama causa y efecto como solución repetir errores

La representación de varios elementos (**causas**) de un sistema que pueden contribuir a un problema (**efecto**)

Es útil para la recolección de datos y es efectivo para estudiar los procesos y situaciones.



Un diagrama causa y efecto bien hecho es un vehículo para ayudar a los equipos de forma continua a tener un concepto común de un problema complejo



Con el diagrama de espina saco información para después resolver el problema.

Se divide en

- Personas - Hombre
- Medio - Entorno
- Métodos
- Maquina
- Materiales

Cuanto tengo que resolver el problema, miro las causas y ordeno por distintas variables

ejemplo: que tan fácil de resolver es, que tanto impacto tiene, o el porcentaje de relevancia como causante del problema general.

Así elijo por donde empezar a resolver el problema identificado.

multitasking → puede ser que pierdas tiempo, por lo general no conviene. sumado al CONTEXT SWITCH que suma mas perdida de tiempo

Condiciones para que agregar mas gente no sea un problema

1. Tareas deben ser particionables
2. Que las tareas no necesiten mucha comunicación
3. Personal capacitado (curva de aprendizaje)

Características de software como producto ingenieril

Producto lógico/mental, no físico. No se rige por las leyes de la física

Se desarrolla pero **no tiene etapa de producción**

Es fácil de modificar, pero predecir las consecuencias de un cambio no lo es.

La curva de envejecimiento es distinta, ya que no se desgasta con el uso pero sí con los cambios sucesivos.

Classic Mistakes

Mistake Exposure Index → La frecuencia de ocurrencia multiplicada por la severidad de la ocurrencia

1. Expectativas poco realistas
2. Cronogramas muy optimistas
3. Quality Assurance shortchanged → se recortan los testeos, el proyecto termina lleno de bugs
4. Wishful Thinking → It's closing your eyes and hoping it works when you have no reasonable basis for thinking it will
5. Confusing Estimates with targets → schedules based on the desirability of business targets w/o creating analytically-derived cost or schedule estimates. **Some organizations refer to the target as the 'estimate.'**
6. Feature creep
7. Noisy, crowded offices
8. Abandoning planning under pressure
9. Insufficient risk management
10. Heroics

F. Classic Mistakes by Mistake Exposure Index (MEI)

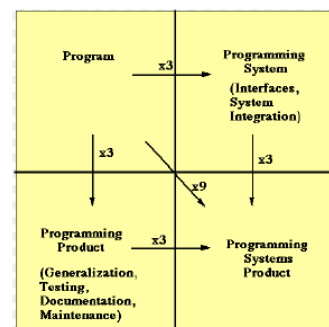
Table F-1 Complete List of Mistake Exposure Indices

Mistake	MEI
Unrealistic expectations	9.9
Overly optimistic schedules	9.6
Shortchanged quality assurance	9.0
Wishful thinking	8.9
Confusing estimates with targets	8.8
Excessive multi-tasking	8.7
Feature creep	8.1
Noisy, crowded offices	7.8
Abandoning planning under pressure	7.8
Insufficient risk management	7.8
Heroics	7.7
Shortchanged upstream activities	7.6
Inadequate design	7.6
Lack of user involvement	7.6
Weak personnel	7.4

The Tar Pit

Programa → Completo en si mismo, listo para ejecutarse por el autor **en el sistema en el que se desarrollo.**

Producto de programación → Puede ser ejecutado, probado, reparado y extendido por cualquiera. Esta disponible para usarse en muchos ambientes operativos y para multiples conjuntos de datos.



Debe estar escrito en un estilo generalizado y probado minuciosamente para que sea confiable. A su vez debe tener documentación que permita que cualquiera lo use, corrija y extienda.

Sistema de programación → Colección de programas interactivos cuyas actividades están coordinadas y sus formatos son estrictos. Su montaje constituye un infraestructura completa para tareas grandes. Toda entrada y salida se debe ajustar en sintaxis y semántica con interfaces bien definidas. Debe diseñarse de forma que solo use un presupuesto terminado de recursos (espacio de memoria, dispositivos I/O, tiempo de computo). También se debe probar el programa junto con otros componentes del sistema en todas la combinaciones esperadas.

Producto de sistemas de programación → es un producto de programación y un sistema de programación a la vez.

Calidad de Software

Ecuación de costo y beneficio (trade off) → Depende la finalidad el software, la importancia de su calidad. un software de aviones es muy importante.

Existe un costo de calidad (prevención y medición) contra un costo de no calidad (fallas que afectan al cliente y al equipo)

Calidad → algo de valor que cumple con los requerimientos func y no func

Los req de calidad deben ser establecidos al principio del proyecto junto con los otros requerimientos. **la calidad se discute en cada paso**

DEFINICIONES

Crosby → Cumplir con los requerimientos

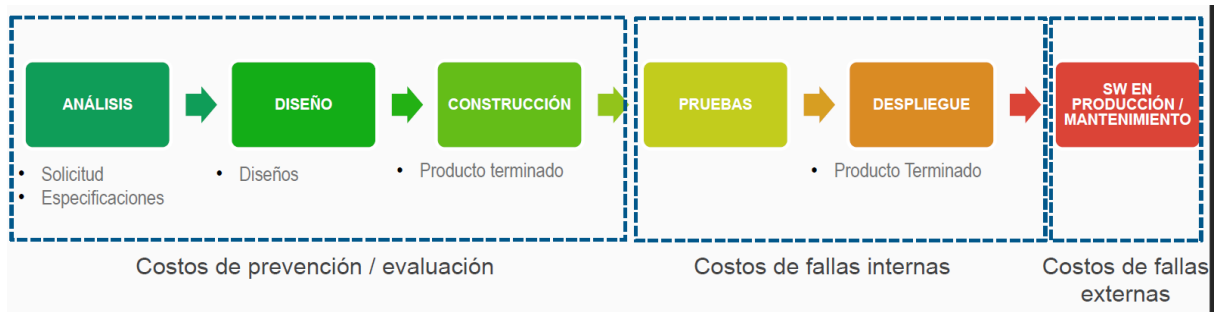
Weinberg → Cumplís con los requerimientos de alguna persona. **Calidad como valor para una persona. Valor es lo que uno esta dispuesto a pagar para obtener esos req**

Juran → Adecuación al uso. Satisfacer las necesidades de cliente + ausencia de deficiencias



ISO 8402-1986 → La totalidad de aspectos y características de un producto o servicio que se sustenta en su capacidad de cumplir las necesidades especificadas o implícitas.

Costo de la No Calidad



Es mas caro lo que depende del usuario final que lo que depende de mi.

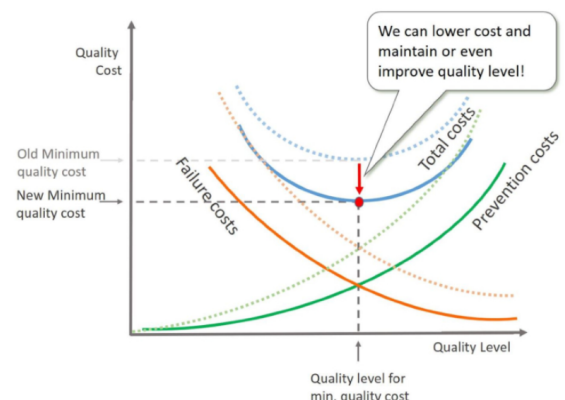
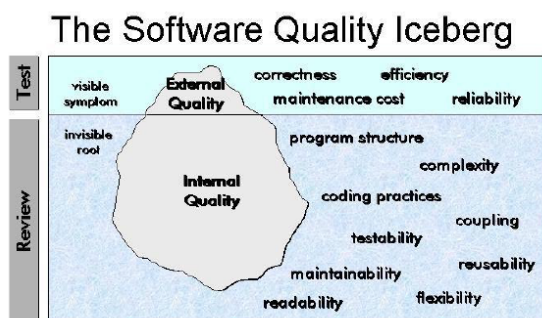
Cuanto mas tarde engancho un error (o sea cuanto mas cerca de la fecha de produccion), mas costoso es corregirlo. POR ESTO SE INVIERTE EN PRUEBAS.

Cada vez que hago algo hago de no calidad (=perder calidad) tengo costo.

Costo de calidad → la inversion en pruebas

Costos ocultos de la no calidad

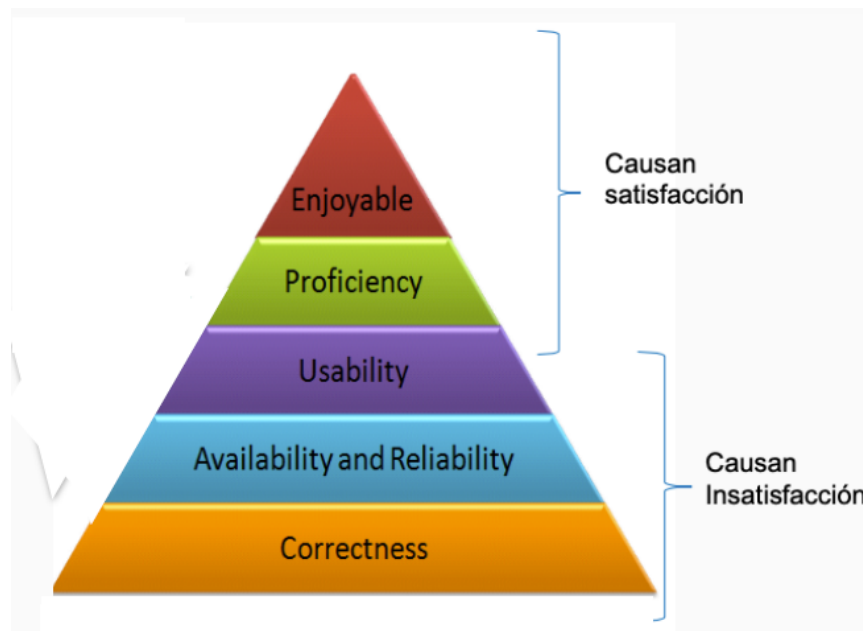
- Baja motivación de los equipos de trabajo, duplicación de esfuerzos porque las cosas hay que corregirlas y rehacerlas constantemente.
- Overtime constante
- Desgaste el equipo de trabajo
- Imagen negativa del cliente.



La parte de arriba del iceberg seria una vez ya ejecutado el software

Se utiliza la redundancia en softwares muy críticos

PIRAMIDE DE JERARQUIA DE FACTORES DE CALIDAD



Hay factores que generan INSTATISFACCION → Si no están, la borro. **Estos son los mas importantes**

Factores QUW GENERAN SATISFACCION → si no están, meh, pero si están es mejor

Visiones de Calidad

Vision trascendental → objetiva, no se puede medir ni explicar (coca vs pepsi). Se reconoce pero no se puede definir.

Vision del usuario → fit for use. Que haga o que el usuario quiere

Vision de la manufactura → calidad en conformidad con la especificación de manufactura.

Vision de producto → vinculado a las practicas inherentes del producto. hace lo que hace en la forma especificada en que debe hacerlo. (ejemplo el ram de una compu, la ssd cual tiene 'mejor' independientemente de lo que erigiría o quiere el usuario)

Vision basada en el valor → calidad depende de la cantidad de plata que el usuario este dispuesto a pagar por el producto.

manufactura es la mas objetiva. después producto, dsp usuario, dsp valor y desp trascendental.

Como evaluar la calidad - Modelos de Calidad de Software



iso25010 → centrado en la vision de producto.

vision basada en las características intrínsecas dentro del producto y cada item dentro de la ISO hablan del producto

▼ Adecuación Funcional

la capacidad del producto para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando e producto se usa en las condiciones especificadas

cuidado con las funciones implícitas. debe hacer lo declarado y la implícito

- **Completitud Funcional:** grado en el cua e conjunto de funcionalidades cubre las tareas y objetivos especificados de usuadio
- **Correccion funcional:** Capacidad del producto/sist para proveer resultados correctos con el nivel de precision requerido
- **Pertinencia Funcional:** Capacidad de prod de software para proporcionar un conjunto apropiado de funciones para tareas y objetivos de usuario especificados.

▼ Eficiencia de Desempeño

El desempeño relativo a la cantidad de recursos usados bajo determinadas condiciones

Comportamiento temporal: Los tiempos de respuesta y procesamiento y ratios de throughput de un sistema cuando lleva a cabo sus funciones bajo condiciones determinadas en relación con un banco de pruebas establecido

Utilización de recursos: cantidades y tipos de recursos usados cuando el software cumple sus función bajo condiciones determinadas

Capacidad: Grado en que los limites maximos de un parametro de un producto/sistema de software cumplen con los requisitos

▼ **Compatibilidad**

Capacidad de dos o mas sistemas o componentes para intercambiar informacion y llevar a cabo sus funciones requerida cuando comparten el mismo entorno de hardware/software

Coexistencia: capacidad de un producto para coexistir con otro software independiente, en un entorno comun, compartiendo recursos comunes sin detrimento.

Interoperabilidad: Capacidad de dos o mas sistemas o componentes para intercambiar informacion y usar informacion intercambiada

▼ **Usabilidad**

Capacidad de un prod de software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando es usado bajo determinadas condiciones

Capacidad de reconocer su adecuación: lo que le permite a usuario entender si el software es adecuado para sus necesidades

Capacidad de aprendizaje: Permite que el usuario aprenda la aplicación?

Capacidad para ser usado: permite al usuario operarlo y controlarlo con facilidad?

Protección contra errores de usuario: protege al usuario de sus errores?

Estética de la interfaz del usuario: le agrada y satisface la interacción con el usuario?

Accesibilidad: puede ser utilizado por usuarios con determinados impedimentos y discapacidades?

▼ **Fiabilidad**

Capacidad para desempeñar as funciones especificadas, cuando se usa bajo condiciones y periodo de tiempo determinados

Madurez: Cap de sistema para satisfacer las necesidades de fiabilidad en condiciones normales

Disponibilidad: Cap de sistema de estar operativo y accesible para su uso cuando se requiere

Tolerancia a fallos: cap de sistema para operar segun lo previsto en presencia de fallos de hardware/software

Capacidad de recuperacion: recuperar los datos directamente afectados y reestablecer el estado deseado de sistema en caso de interrupcion o fallo

▼ Seguridad

Capacidad de proteccion de la informacion y los datos de manera que personas/sistemas no autorizados no puedan acceder o modificarlos

Confidencialidad: Cap de proteccion contra el acceso a los datos e informacion no autorizados, ya sea accidenta o deliberadamente.

Integridad: cap del sistema o componente para prevenir accesos o modificaciones no autorizadas a datos o programas de ordenador

No Repudio: capacidad de demostrar las acciones o eventos que han tenido lugar, de manera que dichas acciones o eventos no puedan ser repudiados posteriormente

Responsabilidad: Cap de rastrear de forma inequivoca las acciones de una entidad

Autenticidad: cap de demostrar la identidad de un sujeto o recurso

▼ Mantenibilidad

mantenibilidad → vision de manufactura

cap del producto para ser modificado efectiva y eficientemente, debido a necesidades evolutivas, correctivas o perfectivas.

Modularidad: Cap del sistema que permite que un cambio en un componente tenga un impacto minimo en los demas

Reusabilidad: Cap de un activo que permite que sea utilizado en mas de un sistema de software o en la construccion de otros activos

Analizabilidad: facilidad con la que se puede evaluar el impacto de un determinado cambio en el resto del software, diagnosticar las deficiencias o causas de fallos en el software, o identificar partes a modificar

Capacidad para ser modificado: Cap de un producto que permite que sea modificado de forma efectiva y eficiente sin introducir defectos o degradar el desempeño

Capacidad de ser probado: facilidad con la que se pueden establecer criterios de prueba para un sistema o componente y con la que se pueden llevar a cabo dichas pruebas.

▼ Portabilidad

Cap de un producto o componente de ser transferido de forma efectiva y eficiente de un entorno de hardware, software, operacional o de utilización a otro

Adaptabilidad: capacidad del producto que le permite ser adaptado de forma efectiva y eficiente a diferentes entornos determinados de hard, soft, etc

Capacidad para ser instalado: Facilidad con la que el producto se puede instalar o desinstalar de forma exitosa en un determinado entorno

Capacidad para ser reemplazado: Cap del producto para ser utilizado en el lugar de otro producto de software determinado con el mismo propósito y en el mismo entorno

No se puede todo

A veces hay que sacrificar un aspecto de la calidad para priorizar otro

Ejemplo: utilización de recursos vs tolerancia a fallas

tolerancia a fallas requiere redundancia, y si tengo información redundante no estoy maximizando los recursos.

Comportamiento temporal vs protección frente a errores del usuario

Calidad de proceso

Vision de la manufactura

CMMI - Capability Maturity Model Integrated.

Determina la madurez de un proceso y organiza el esfuerzo para mejorarlo describiendo un camino incremental de mejora. Ayuda a saber donde estamos y tener un mapa que indique a donde ir.

Para la mejora y evaluación de procesos de desarrollo, mantenimiento y operación de SW.

Cuanto mejor es el proceso de fabricación, mas calidad tendrá el producto.

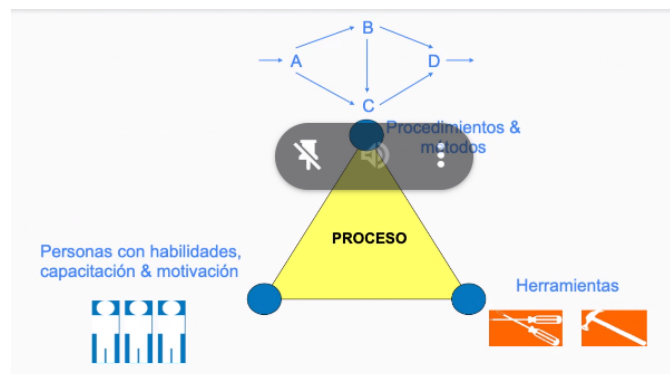
5 niveles de madurez

Definiciones

Proceso: Conjunto de actividades que la gente usa para desarrollar/mantener el software y sus productos asociados

Madurez: Capacidad organizacional, no individual, de cumplir sistemáticamente con los objetivos. **El grado en el que esta un proceso (definido y documentado / administrado y controlado / medido y es efectivo)**

Capacidad de un proceso: Habilidad inherente de un proceso de SW para **producir los resultados planificados**. CMMI se enfoca en que la organización produzca productos de alta calidad en forma consistente y predecible.



Proceso Maduro

- Soportado por la gerencia → Toda la gerencia lo hace seguir
- Definido, documentado, conocido y practicado
- Existe infraestructura adecuada para soportarlo
- Adecuadamente medido y controlado
- Presupuestos y plazos realistas
- Riesgo conocido y controlado

Proceso Inmaduro

- La gerencia dice soportarlo
- Improvisado sobre la marcha
- Aunque este definido no se sigue rigurosamente
- No hay entrenamiento formal ni herramientas para sustentarlo
- Presupuestos y plazos son generalmente excedidos por estimaciones no realistas

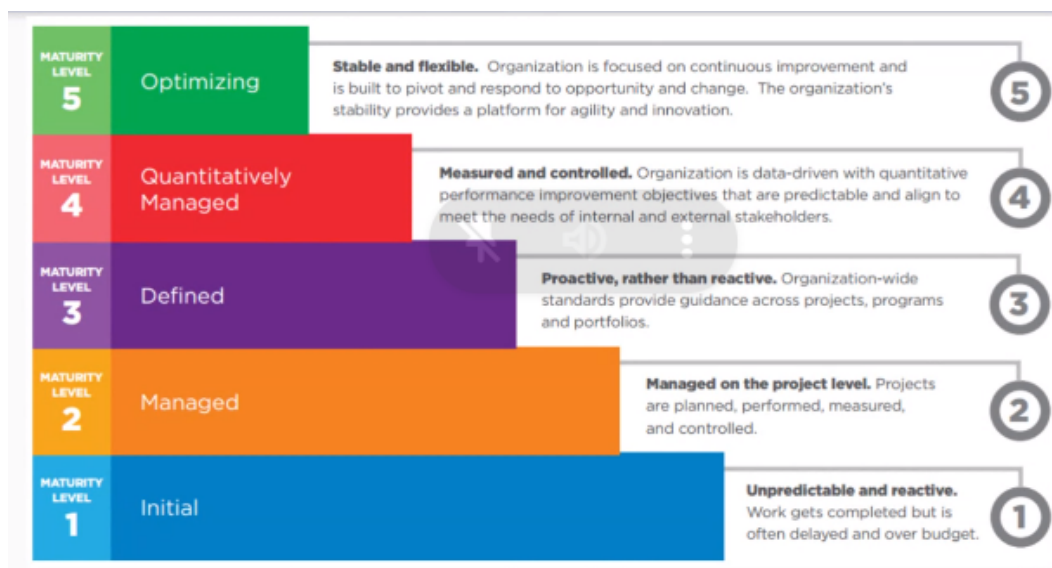
- Proactivo
- Es una organización reactiva
- **Institucionalizado**



Cuando las cosas se complican, los procesos no suelen seguirse. Cuando un proceso se sigue a pesar de las complicaciones es porque es un proceso maduro

Representaciones CMMI

Nivel	Representación continua Niveles de capacidad	Representación por etapas Niveles de madurez
Nivel 0	Incompleto	
Nivel 1	Realizado	Inicial
Nivel 2	Gestionado	Gestionado
Nivel 3	Definido	Definido
Nivel 4		Gestionado cuantitativamente
Nivel 5		En optimización



El nivel inicial no tiene practicas

nivel 2 → gerenciamiento básico (planeación, seguimiento, administración de calidad y mediciones básicas). no esta uniformizado en toda la organización

nivel 3 → Toda la organización sigue el proceso (ej scrum)

Nivel 4 → nivel 3 mas evolucionado. las decisiones se toman basadas en métricas.

Scampi - método de evaluación de una empresa

El SCAMPI A es el único que te genera rating.

Características	Clase A	Clase B	Clase C
Cantidad de evidencia objetiva	High	Medium	Low
Ratings generados	Yes	No	No
Necesidad de recursos	High	Medium	Low
Tamaño del equipo	Large	Medium	Small

Engineering Approaches

Proyecto - Definición



Esfuerzo temporal emprendido para crear un producto o servicio único para lograr un objetivo

Posee recursos

Consta de sucesiones de actividades o fases en las que se coordinan los recursos.

Project Management: La admin de proyectos es una disciplina que consiste en planificar, organizar

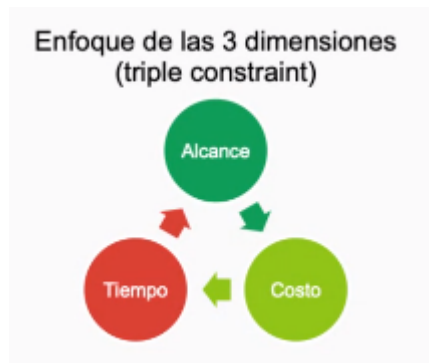
Dimensiones de un proyecto de software

No se puede cumplir con todas a la vez.

Cada dimension puede ser

- **Driver:** objetivo vital a lograr, tiene poca/nada flexibilidad
- **Restricción:** no esta bajo nuestro control directo y tiene poca/nada flexibilidad

- **Grado de libertad:** Libertad para filtrar objetivos. Existe flexibilidad para manejar la variable



Son **dependientes** una de la otra, en cualquiera de los dos enfoques.

Plan de Proyectos

1. **Presupuesto**
2. **Objetivos** → Pueden ser de negocio, de proyecto y de producto. Un objetivo debe ser SMART (Specific Measurable Attainable Relevant Time-Related). El éxito del Proyecto deberá medirse en el grado de cumplimiento de dichos objetivos.
3. **Alcance** → Define los límites del sistema (lo que incluye, de ser necesario se aclara lo que "no" incluye).
4. **Riesgos**

Riesgos

Un riesgo es la posibilidad de que suceda un evento negativo que impacte en los objetivos del proyecto.

Se caracterizan por tener los siguientes atributos:

- Probabilidad de ocurrencia (1-100%)
- Impacto (\$; 1-10; Bajo/Medio/Alto)
- Exposición = Probabilidad de ocurrencia * Impacto

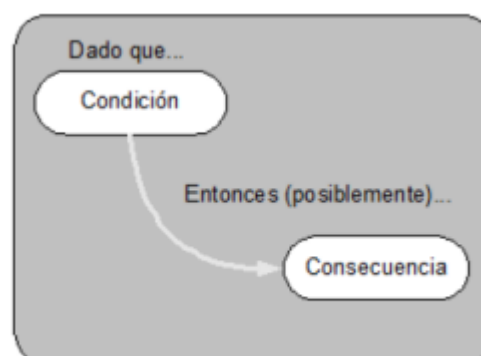
Dado que los riesgos son problemas que aún no llegaron, es necesario gestionar los mismos (Risk Management).

La gestión del riesgo trata de identificar, resolver y comunicar los riesgos. Se basa en tomar decisiones bajo niveles de incertidumbre que tienen incidencia en el futuro. No es una actividad aislada, debe acompañar a todo el ciclo de vida de desarrollo de SW

Paradigma de Gestión del Riesgo (según SEI)



1. **Identificación** → Consiste en localizar los riesgos antes de que se conviertan en problemas y afecten al programa. Se utilizan varias técnicas como brainstorming, taxonomías, repaso de experiencia, etc. Deben documentarse. Para enunciarlos se utiliza la representación de Glutch:



2. **Análisis** → Consiste en transformar la información anteriormente capturada en información que permita tomar decisiones. Un buen análisis debe:
 - a. Estimar probabilidad e impacto
 - b. Estudiar causas y acciones correctivas
 - c. Identificar causas comunes

- d. Identificar tiempos de ocurrencia
3. **Planificación** → Transformar la información acerca de los riesgos en decisiones y acciones. La prioridad se establece en función del grado de exposición y de la urgencia que demande la acción correctiva. Un plan de acción puede tener cualquiera de las siguientes formas:
- a. Evitar el riesgo.
 - b. Reducir la probabilidad de ocurrencia con planes de mitigación.
 - c. Atacar el impacto con planes de contingencia.
 - d. Aceptar el riesgo sin tomar acciones, aceptando las consecuencias derivadas de su posible ocurrencia

Plan de mitigación	Plan de contingencia
Se toman acciones tempranas sobre el impacto y la probabilidad de ocurrencia de los riesgos, independiente de la ocurrencia de los mismos	Se planifican ciertas acciones, pero se monitorea señales de alerta (triggers). Se toman las acciones solo cuando se disparan dichas señales
Se gasta tiempo y dinero por adelantado debido a una condición de riesgo identificada. Proactivo	No se gasta tiempo y dinero por adelantado, pero se puede hacer una reserva para gastarla cuando sea necesaria

4. **Seguimiento** → Consiste en monitorear el "status" del riesgo, y que las acciones que fueron definidas en el plan se ejecuten, informando tanto las posibles desviaciones respecto de los objetivos, como el impacto en el presupuesto, calendario y consideraciones técnicas. Para esto se definen métricas, y eventos que se disparan para asegurarse que las acciones estipuladas en la planificación funcionan correctamente
5. **Control** → Realizar las correcciones en los desvíos ocurridos en la planificación.
6. **Comunicación** → Es el centro del paradigma porque sin comunicación efectiva, ningún approach para gestionar el riesgo es viable. La comunicación es crítica porque es la que permite la interacción entre los elementos del paradigma. Otorga feedback acerca de las actividades que podrían causar algún riesgo, de los riesgos actuales, de los emergentes.

Roles principales de un proyecto

Stakeholder: todos los involucrados por el proyecto. Tienen poder de decisión con capacidad de influir en la marcha de este. todos los de abajo son ejemplos de stakeholder

Sponsor: owner del proyecto. Tiene la autoridad para llevarlo. Lo paga. La persona mas interesada en el proyecto

Usuario Campeón/Product owner: Experto en el dominio del problema del proyecto. Asegurar su capacidad para la función y su disponibilidad. Son muy demandados. Deben asegurarse de transmitir su conocimiento

Usuarios directos: Interactúan directamente con el sistema. lo opera

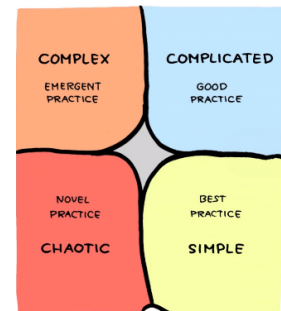
Usuario indirecto: Hace uso del sistema pero no lo opera directamente

CYNEFIN

4 dominios de complejidad: simple, complicado, complejo, caótico.

ordenados: los de la derecha. Desordenados: los de la izq

Ayuda a hacer un análisis entendiendo el domino en el cual estamos. Podemos determinar en que tipo de framework nos conviene usar



En lo desordenado no existe el causa y efecto.

Mundo ordenado → Gestion basada en **hechos**

Mundo desordenado → Gestion basada en **patrones**

Contexto Simple

Se pueden aplicar las best practices. si pasa A, siempre va pasar B

Relación causa efecto es clara. **Problemas conocidos**

Conozco lo que sucede y como categorizarlo según cada caso. No hay incertidumbre

Sense → categorize → respond

Puede haber problemas por categorizar mal una situación, o por sobre simplificarla. Entrained Thinking → lideres cegados a nuevas alternativas y soluciones por perspectivas ganadas de experiencias pasadas exitosas. **en el modelo, el contexto simple esta al lado del caótico, y es porque por**

complacencia, es muy fácil no ver que el sistema simple esta fallando y reaccionar tarde, terminando en el caos. HAY QUE ESTAR ATENTOS SIEMPRE A LOS POSIBLES CAMBIOS DE CONTEXTO.

Contexto Complicado

Cuando pasa algo, hay multiples respuestas posibles. Hay buenas practicas que puedo aplicar para resolver el problema. La relación causa efecto necesita análisis o experiencia

Sense → analyze → respond

known unknowns.

Los expertos en vez de los lideres los que pueden caer en **entrained thinking**, no se aceptan ideas innovadoras para solucionar problemas complicados.

Analysis Paralysis: los expertos llegan a un punto donde no pueden acordar una solución por ego o por entrained thinking.

Contexto Complejo

Mundo desordenado

No sabemos como se resuelve el causa y efecto. Análisis de retrospectiva, tengo que ver que paso. No hay respuesta correcta. Se pueden desarrollar diseños útiles y informativos (**practica emergente**).

Se exploran soluciones, analizan y se responde al problema.

Probe → sense → response. Prueba y error/prueba y análisis

No es predecible

| Las metodologías ágiles viven en este contexto.

Características de un sistema complejo

- Incorpora muchos elementos que interactúan entre si
- Interacciones no lineares. Pequeñas modificaciones pueden producir consecuencias desmedidas.
- Sistema es **dinámico**, el **total es mayor que la suma de sus partes** y no se pueden imponer soluciones, sino que se van apareciendo de las situaciones que ocurre. **emergence**

- El sistema tiene historia y el pasado esta integrado al presente. Evolución irreversible
- **Hindsight does not lead to foresight**, external conditions and systems constantly change
- No hay constraints. Los agentes y el sistema se restringen mutuamente a lo largo del tiempo, lo que lleva a incertidumbre sobre el futuro

Los sistemas complejos humanos no son como los sistemas complejos en la naturaleza y no se pueden modelar por la inteligencia y la impredecibilidad humana.

Conditions of scarcity often produce more creative results than conditions of abundance

Contexto Caótico

Todo es incierto. Incertidumbre. Tener experiencia no ayuda.

Cualquier acción que tomes es correcta. Se busca establecer el orden como se pueda

Act → Sense → Respond

Un ransomware x ejemplo.

No vale la pena buscar right answers.

es muy difícil aplicarle un marco metodológico, pero puede meterse un Kanban adaptado, onda un trello

Kanban

Administración visual del flujo de trabajo que ayuda a realizar mas con menos estrés

Viene de Lean, Tablero Visual. Ágil

Como ordenar tareas que sufren cambios de prioridad

Se usa Kanban cuando una de las siguientes preguntas tiene si como respuesta:

- Parece que hay miles de tareas para realizar constantemente?
- siempre hay que cambiar de una tarea a otra perdiendo el foco y sin tener suficiente seguimiento?

- Parece que se trabaja siempre sin parar pero aun así sin ser productivo?
- Hay problemas de comunicación provocando esfuerzos duplicados, defectos, retrabajo y mas?

Conceptos básicos

- Visualizar el flujo de trabajo
- Limitar el trabajo en progreso
- Medir y administrar el workflow
- Políticas explícitas y mejora continua colaborativa y acumulativa



Touch time: tiempo trabajado neto. si se lo pone en pausa, no toma en cuenta el tiempo en pausa

Lead time: tiempo desde que se pidió hasta que se entrego. cuenta TODO. Tiempo sin empezar, tiempo pausado, todo

Cycle Time: Cuenta desde el momento que se empieza a trabajar en la tarea, y tiene en cuenta las pausas.

El limite del WIP seria la cantidad de items que pueden trabajarse en paralelo por cada etapa del proceso (lo que marca la diferencia entre Kanban y una lista to do visual)

Scrum

es bueno para el escenario complejo

Marco metodológico de trabajo para la administración de proyectos que enfatiza en el trabajo en equipo, en el proceso iterativo hacia un objetivo bien definido. **Construir productos de forma incremental.**

Te dice lo que hay que hacer, no como.

Empieza con una premisa simple: **Comenzar con lo que puede ser visto o conocido y después seguir el avance y modificar lo necesario.**

Sprint → medida de tiempo fijo en el que se obtiene un incremento de producto. es de entre 1 y 4 semanas.

Cada incremento es una versión mejorada del producto que alcanza criterios de aceptación y el nivel de calidad requerido.

Valores de Scrum

- Foco
- Coraje
- Apertura
- Compromiso
- Respeto

Dividir el alcance (entregables chicos, priorizados por valor, estimables)

Dividir la organización (equipos chicos, auto organizados, interdisciplinarios que incluyen al cliente)

Dividir el tiempo (Seguimiento diario, Iteraciones cortas, se planifica al comienzo y evalúa al final, generan productos de valor)

Optimizar regularmente (el proceso con retrospectivas, e producto con revisiones, el plan como resultado del aprendizaje iterativo, se busca un ritmo sostenible)

Roles

Scrum master

- Dueño del proceso
- colabora con el equipo

- Esta a cargo de resolver cualquier impedimento, el responsable
- Protege y cuida al equipo
- Cuida el proceso, asegurándose que scrum se aplique correctamente

Equipo

- Dueño de los procesos de producción/ingeniería
- De 5 a 9 personas
- Interdisciplinario
- Auto-organizado
- Responsables de construir el producto
- Define colaborativamente como transformar el product backlog en un incremento de funcionalidad al final de la siguiente iteración

Product Owner

- Dueño de la definición de éxito/terminado
- Representa al cliente-usuario
- Dueño del backlog
- Define las prioridades
- Establece plan preliminar de entregas
- Administra el ROI (retorno de inversión) priorizando los requerimientos
- Cuida el valor del producto
- Se asegura que el equipo trabaje de forma adecuada desde la perspectiva del negocio

Ceremonias

Sprint planning

Participan el scrum master, dev team y product owner

Debe definirse que se entregara al final del sprint y que trabajo es necesario para hacer el incremento del producto y como se hara.

Al finalizar la reunion:

- Se compromete el Sprint backlog
- Se asignan user stories comprometidas
- Se generan tareas.

Frecuencia: 1 vez por sprint, al inicio de este

Duración: entre 1 y dos horas

el user story es el requerimiento del usuario. contiene tareas

Daily Meeting

Scrum master, dev team y product owner

Dura 10 a 15 minutos, todos los días.

Cada miembro del equipo de scrum explica que hizo para el proyecto desde la ultima reunion, que piensa hacer hasta la próxima reunion y que impedimentos tiene.

Sirve para sincronizar tareas y fomentar la participación/comunicación. **no se deberían refinar historias o tareas, o resolver problemas.**

Cuando finaliza:

- El equipo intenta remover los impedimentos

Sprint Review

Se expone el objetivo del sprint, se muestran las funcionalidades desarrolladas y se obtiene el feedback del negocio.

Frecuencia: 1 vez por sprint, al final de este

Duración: puede llegar a ser 4 hrs para un sprint de 1 mes.

Retrospective:

Se obtienen lecciones aprendidas y acciones para mejorar en el siguiente sprint. Se inspecciona lo que funciono bien, lo que no resulto como se esperaba, propuestas de mejoras, lecciones aprendidas y próximas acciones.

Frecuencia: 1 vez por sprint, al final

Duración: 1 o 2 hrs, no se recomienda mas que eso

Lean

Una filosofía y un enfoque que hace hincapié en la eliminación de residuos o de no valor añadido al trabajo, a través de un enfoque en la mejora continua

Se centra en ofrecer una mayor calidad y reducir tiempos de ciclos y costos.

Busca guiar y reforzar los principios lean de manera efectiva en toda la organización a través del uso de distintas herramientas y sistemas con sus rutinas de trabajo generando los comportamientos necesarios que conduzcan a los resultados deseados, creando así una cultura de mejora continua de creación de **valor** al **cliente**, y desarrollo de las **personas**

4 ejes principales

- Aprender de lo que ya hice
- Apoyarme en las personas
- Darle valor a los procesos
- Es una filosofía

Se piensa a largo plazo



14 ejes de gestión



- **Gestionar basado en una filosofía de largo plazo**
- **Crear flujos de procesos continuos para evidenciar problemas**
- **Usar sistemas de procesos "Pull" para evitar a sobreproducción**

- **Nivelar la carga de trabajo: HEIJUNKA**
- **Generar una practica y cultura de parar para resolver problemas y asegurar la calidad**
- **Estandarizar, base fundacional de la mejora continua y e empoderamiento de las personas**
- **Utilizar controles visuales para evidenciar os problemas**
- **Recurrir únicamente a tecnología confiable y correctamente testada para servir a las personas y a los procesos**
- **Desarrollar líderes que entiendan el trabajo, vivan la filosofía y se a enseñen a otros**
- **Desarrollar personas y equipos excepcionales que sigan la filosofía de su organización**
- **Respetar a tu red de contratistas y proveedores desafiándolos y ayudándolos a mejorar**
- **Ver y observar por uno mismo para comprender profundamente la situación (genchi genbutsu)**
- **Tomar decisiones por consenso, teniendo en cuenta todas las opciones (nemawashi) e implementar ágilmente**
- **Propiciar el proceso de convertirse en una organización que aprende a través de la reflexión (Hansei) y la mejora continua (kaizen)**

En general nos preguntamos cuanto tiempo vamos a tardar en terminar?
cuantas hrs se necesitan? cuanto va a costar?

Pocas veces nos preguntamos cual es el tamaño de lo que tenemos que construir? ESTA DEBERIA SER LA PRIMERA PREGUNTA QUE NOS HACEMOS

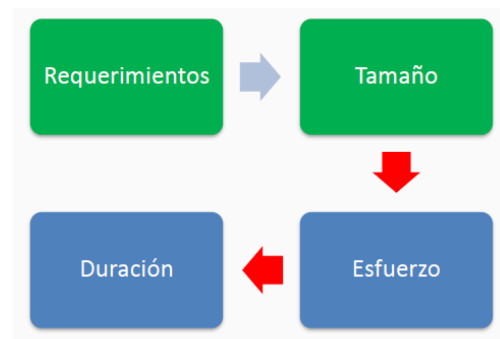
El proceso de estimar

Requerimientos → lo que nos lleva.

Tamaño → noción de esfuerzo / volumen a construir

Esfuerzo → bajarlo a trabajo actual, horas humanas ininterrumpidas para resolver los requerimientos

Duración → aplico reglas (ej, sab y dom no se trabaja, 1 hora de almuerzo, puedo trabajar 9 hrs x día, etc)



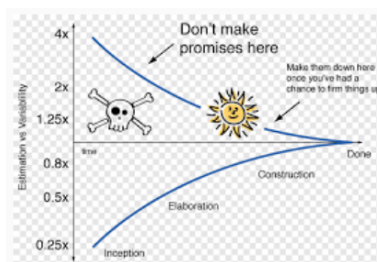
una tarea de 16hrs de esfuerzo podrían ser 2 días de duración

Gannt calcula la duración

Porque fallan las estimaciones

- Optimismo
- Estimaciones informales
- No hay historia
- Mala definición del alcance falta de experiencia
- Complejidad del problema a resolver
- Mala administración de los requerimientos
- No hay seguimiento y control
- Se confunde el progreso con esfuerzo

El nivel de incertidumbre



Cono que determina que tan certero puedo ser, se va achicando con el tiempo

La incertidumbre va bajando si yo voy ganando conocimiento del producto.

Se recomienda dar estimaciones en rangos, especialmente en las primeras etapas

Tips para estimar

Diferenciar entre estimaciones, objetivos y compromisos

Asociar a las estimaciones un % de confiabilidad

Es recomendable presentar las estimaciones como rangos en lugar de un único valor

Siempre presentar junto con la estimación los supuestos que se tuvieron en cuenta para llegar a ella.

Ley de Parkinson → toda tarea se expande para ocupar su tiempo asignado

Considerar todas las actividades relacionadas al desarrollo de sw, no solamente la codificación y testing

No asumir que solo por el paso del tiempo y de las fases de un proyecto se avanza con menor incertidumbre en las estimaciones (el cono)

Recolectar datos históricos para tener como referencia

Estimaciones creíbles

Las estimaciones las hacen las personas, no herramientas ni modelos

Las estimaciones se basan en comparaciones.

Para que a gente pueda estimar necesitamos **historia** de proyectos pasados para poder comparar

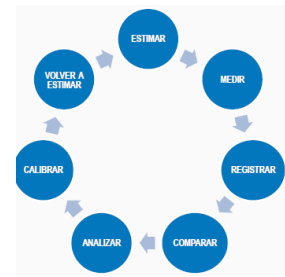
Un método creíble debe ser de caja blanca

Ciclo de estimación

Para crear y perfeccionar métodos de estimación

No siempre hay que calibrar

- **Estimar:** comenzar x estimar el tamaño para derivar el esfuerzo y el costo
- **Medir:** mientras evoluciona el proyecto, medir el tamaño, el esfuerzo y costo incurrido
- **Registrar:** dejar claras las mediciones tomadas
- **Analizar:** Razones de los desvíos, supuestos que quizás variaron, temas no contemplados, etc.
- **Calibrar:** Ajustar c/u de las variables y parámetros que intervienen en el proceso de estimación. No siempre es necesario
- **Volver a estimar** → el mismo proyecto pero ahora con mas información que antes, o nuevos proyectos con el proceso ajustado por la calibración



Métodos de estimación

No Paramétricos → Juicio Experto, PERT, Wideband Delphi, Panning poker

Paramétricos → Function Points, Use Case Points, Object Points

Juicio Experto

Depende de la persona que haga la estimación

Se basa en la experiencia de personal, se recurre a analogías

Desventaja: la opinion de 1 único experto. Que pasa si le pasa algo al experto?
no hay documentación que releve el proceso de estimación. no es sostenible en el tiempo

Ventaja: es fácil de implementar y es rápido

Método PERT - Clark

Program Evaluation and Review Technique

Se basa en el juicio experto, incluye los factores optimistas y pesimistas en su estimación

Estimación = (Optimista + 4 * Medio + Pesimista) / 6

Se puede utilizar para estimar **tamaño y esfuerzo**

Ventajas: incluye una formula, mas documentado.

Desventaja: Sigue basándose en un experto.

Wideband Delphi

Juicio d experto en grupos, da lugar a la participación de todos los involucrados.

Puede usarse en etapas tempranas del proyecto

Se recomienda usarlo en proyectos poco conocidos en donde no hay historia

Ventajas: fácil de implementar, da sentido de propiedad sobre la estimación, es como "colegiado"

Desventaja: Sigue sin documentarse.

▼ Planning Poker

Variación del método Wideband Delphi, pero se adapta a proyectos actuales

El que mas se usa hoy en día

Se estiman tamaños de tareas relativas entre si, tomando alguna como modelo. La estimación puede variar entre proyectos, pero no puede variar el tiempo

buena para proyectos ongoing/en desarrollo incremental a largo plazo. No es buena para primeras estimaciones o desarrollos de 0 a 100% en corto tiempo.

Reglas

Participantes: todo el equipo debe participar

Timebox: Fija la cantidad de tiempo que va a estimar y se cumple

Priorizar: Todos los user stories tienen que estar priorizadas y leídas previamente por el equipo

Escala: se miden las user story en story points que siguen la escala de Fibonacci.

Story points y no hour value porque e story point incluye la complejidad y e tiempo requeridos.

Tarea Pivote: Definir una user story pivote y darle tamaño en story points. 5 u 8 SP son buenos candidatos. Puede ser una User story actual o una anterior que sea significativa y todos la conocen.

Velocity: definir la velocidad del equipo (basado en sprints anteriores) para saber el numero mínimo de SP que necesito estimar para completar un sprint

Definición de Terminado: Cuando esta terminado el User Story? incluye el testing?. **Cada tarea debe tener su definición**

Otros tamaños

- Fibonacci (0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89)
- Modified Fibonacci (0, 1/2, 1, 2, 3, 5, 8, 13, 20, 40, 100, ∞, café)
- T-shirt Sizes (xxs, xs, sm, med, lrg, xl, xxl)
- The Power of 2 (0, 2, 4, 8, 16, 32, 64...)

Cafecito significa pausa. Infinito es que no hay elementos para estimar.

Final

Al finalizar a estimación se visualiza todas las tareas y las estimaciones juntas y comparar entre ellas para ajustarlas

Teniendo en cuenta la velocity del equipo y las prioridades, se planifica el próximo sprint

Ventajas:

- Se estima en tamaño. si cambia el equipo, las tareas no se modifican sino que se modifica la velocity del sprint.
- El sprint es corto. Se corrigen los desvíos cada 2 semanas aprox
- Poco tiempo perdido en planning





Planning poker mide tamaño? VERDADERO

▼ Object Points

Objetos → pantallas, reportes, módulos (implementaciones nuevas tambien consideran los stored procedures, cases, scripts SQLS, etc como objetos)

Se asigna a cada componente un peso de acuerdo a su clasificación por complejidad (simple, medio, difícil)

Considera como factor de ajuste el porcentaje de re-uso de codigo

	Number and sources of data tables			
	Number of Views Contained	Total < 4	Total < 8	Total 8+
	<3	simple	simple	medium
	3-7	simple	medium	difficult
	Number and source of data tables			
	Number of Sections Contained	Total < 4	Total < 8	Total 8+
	0-1	simple	simple	medium
	2-3	simple	medium	difficult
	Number and source of data tables			
	4+	medium	difficult	difficult

Se le asigna un peso a cada nivel de complejidad

OPs → e resultado de sumar la cantidad de objetos de cada complejidad

Con el reuso, obtenido NewOP (NOP) = $OP * [(100 - \%reuso) / 100]$

▼ Function Points

Miden el tamaño del SW en base a la funcionalidad capturada de los requerimientos.

Usa el modelo lógico o conceptual para trabajar.

Es aceptado en a industria a pesar de su complejidad y antigüedad.

Muchas heurísticas están basadas en Function Points.

los FP son independientes de la tecnología y requieren un análisis de requerimientos avanzado.

Elementos del sistema

Archivos Logicos internos - **Internal Logical Files** (ALI-ILF)

Datos logicos de información o de control, identificables por el usuario, mantenidos a través de procesos elementales de la aplicación

Archivo de Interface Externa - **External Interface Files** (AIE)

Datos lógicos de info o control identificables por el usuario que son referenciados por la aplicación pero mantenidos por procesos elementales de una aplicación externa.

NO SON MANTENIDOS POR MI APLICACION

Entradas Externas (EE) o **External Inputs**

Proceso elemental de la aplicación que procesa datos o información de control que entran desde el exterior del sistema.

Los datos procesados mantienen uno o mas ILFs

La info de control puede o no ser mantenida en un ILF

Salidas Externas - **External Outputs SE**

Proceso de la app que envía datos o info de control fuera de los limites del sistema

Puede hacer update de un ILF

El procesamiento debe contener al menos una formula matemática o calculo, o crear datos derivados

Una Eo tbn puede mantener uno o mas ILFs o alterar el comportamiento del sistema

External Inquiry EQ

Proceso que envía datos o info de control fuera de los limites del sistema

NO posee formulas matemáticas ni cálculos ni crea datos derivados.

No mantiene ILFs durante su procesamiento, ni altera el comportamiento del sistema

Unadjusted Function Points (UFP) se calculan así:

Function Type	Low	Average	High
External Input	x3	x4	x6
External Input	x4	x5	x7
Logical Internal File	x7	x10	x15
External Interface File	x5	x7	x10
External Inquiry	x3	x4	x6

	1-5 Data element types	6-19 Data element types	20+ Data element types
0-1 File types referenced	Low	Low	Average
2-3 File types referenced	Low	Average	High
4+ File types referenced	Average	High	High

TCF - Technical Complexity Factor = $0.65 + (\text{sum de factores}) / 100$

TCF y CGS son lo mismo (características Generales de Sistema)

14 factores de complejidad técnica.

Se evalúan de 0 al 5 de acuerdo a su grado de influencia

Calculando los function points

PFs neto = PFN = UFP * TCF

- Fig. 2. Factores de complejidad técnica
- Data communications
 - Performance
 - Heavily used configuration
 - Transaction rate
 - Online data entry
 - End user efficiency
 - Online update
 - Complex processing
 - Reusability
 - Installation ease
 - Operations ease
 - Multiple sites
 - Facilitate change
 - Distributed functions

Conversion de FP a LOC:

	Mínimo	Media	Máximo
Java	40 LOC	55 LOC	80 LOC
C++	40 LOC	55 LOC	80 LOC
Cobol	65 LOC	107 LOC	150 LOC
SQL	7 LOC	13 LOC	15 LOC

Use Case Points

La cantidad de use case points depende de

- Cantidad y complejidad de los casos de uso
- Cantidad y complejidad de los actores intervinientes en el sistema
- Factores técnicos y ambientales

El método requiere que sea posible contar el numero de transacciones de cada caso de uso

Transacción: evento que ocurre entre el actor y el sistema

Basado en Function Points

Elementos del sistema:

▼ Casos de uso

Simple: transacciones ≤ 3. PESO = 5

Medio: 4 a 7 transacciones. PESO = 10

Complejo: transacciones > 7. PESO = 15

$UUCW = \text{Unadjusted Use Case Weight} = \sum(\text{for each cat (num por cat * peso de la cat)})$

- Transacciones de casos de uso

▼ Actores

Actores simples: sistemas externos, predecibles. Todo sistema con interfaz de aplicación bien definida. PESO = 1

Actores Promedio: Dispositivos de hardware. Requieren esfuerzo para controlarlos y mas propensos a errores. PESO = 2

Actores Complejos: humanos, impredecibles, dificiles de controlar. Personas que interactuan a traves de una GUI. PESO = 3

$UAW (\text{Unadjusted Actor Weight}) = \sum(\text{for each cat (num d actores por cat * peso por cat)})$

UUPC = Unadjusted use case points = UUCW + UAW

a su vez, a cada factor tecnico o del entorno que puede influir en el costo total se le asigna un valor del 0 al 5. UNA TABLA PARA LOS DE ENTORNO, OTRA PARA LOS TECNICOS

TCF = Technical Complexity Factor = $0.6 + [0.01 * \sum(\text{cada factor tecnico * su peso})]$

EF = Environmental factor = $1.4 + [-0.03 * \sum(\text{cada factor de entorno * su peso})]$

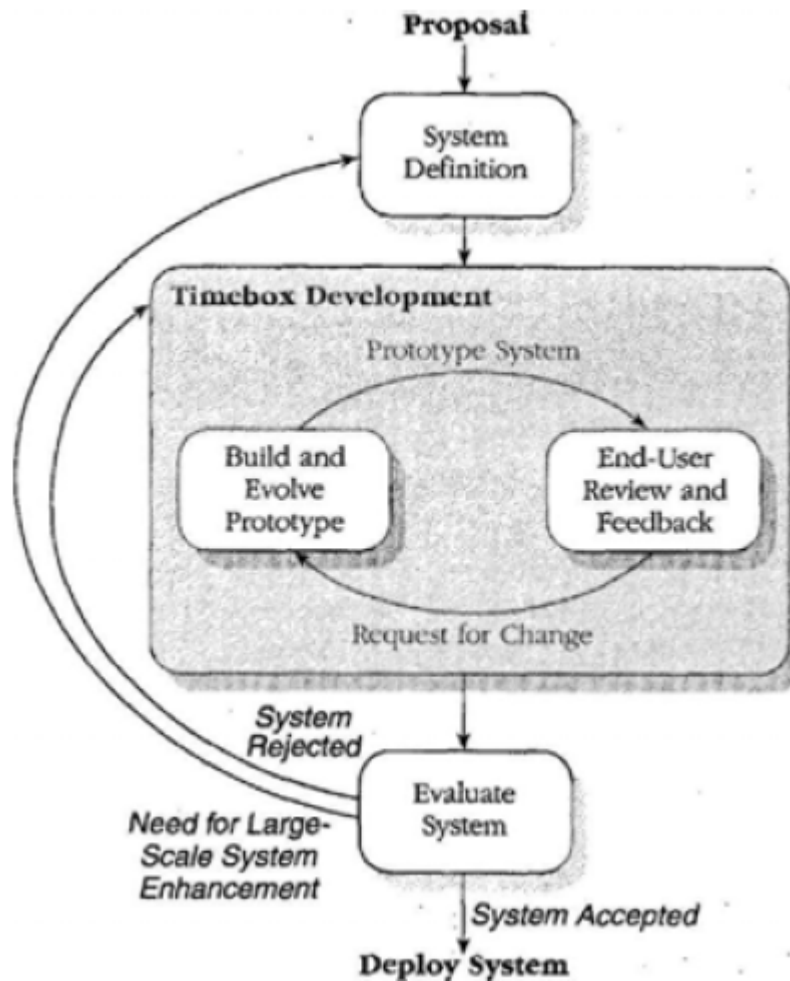
UCP = Adjusted use case points = UUCP * TCF * EF

Si fueran 20 hrs por hombre por ucp $\Rightarrow \text{COSTO PROYECTO EN HH} = \text{UCP} * 20 \text{ HH}$

Timebox

Hace que el schedule sea FIXED (entre 60 y 120 días), poniendo énfasis en lo que es prioridad hacer.

Controla el feature creep, ya que al ponerle un limite al tiempo de desarrollo limitas el tiempo en el que se pueden pedir nuevos features. Genera una sensación de apuro que motiva a los end users y a los desarrolladores.



Criterio para usar timebox

- Lista de features con **prioridad** → **se usa la prioridad para elegir que feature saco cuando veo que no llego con los tiempos**
- Schedule realístico creado por el team → solo va a tener éxito el método si realmente podemos terminar la cosas en el tiempo que propusimos, por lo que debe ser factible
- Right kind of project → in house business software. aplicaciones muy customizadas, que deben codearse no son buenas para este método.
- End user involvement suficiente → basado en prototipado, por lo que necesitamos buen feedback.

Riesgos de aplicarlo

- Aplicarlo en sistemas donde no debería aplicarse (como upstream activities and beginning of the food chain stuff)

- **Sacrificar calidad en vez de features** → si sufre la calidad, sufre el schedule, ya que no va a pasar el test de aprobación y hay que volver al timebox.