INGENIERÍA DEL SOFTWARE

UNIDAD 2: MEDIDAS, MÉTRICAS E INDICADORES DEL PROCESO

CICLO LECTIVO 2013

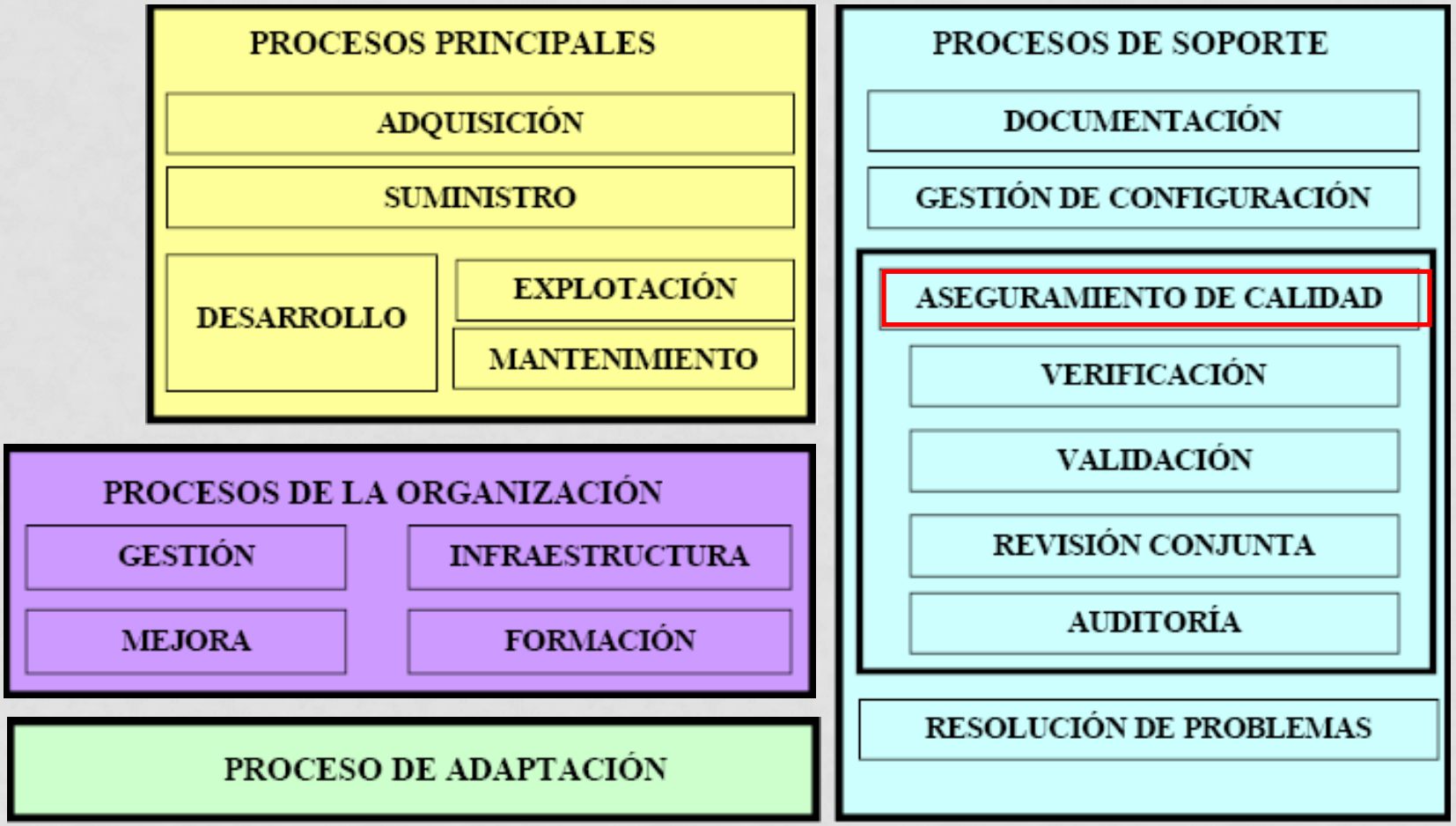
OBJETIVOS DE LA CLASE

- **Métricas del proyecto.**
- **Métricas del producto**
- **Métricas según cada objetivo.**
- **Paradigma GQM**
- **Métricas: privacidad por niveles.**

CALIDAD DEL SOFTWARE

- El **Aseguramiento de Calidad (AC)** del software es el conjunto de actividades planificadas y sistemáticas necesarias para controlar que el **producto** (proceso,...) satisface los requisitos de **Calidad.**
- El AC debe estar presente en:
  - Métodos y herramientas de Análisis, Diseño, Programación y Testeo
  - Control de la Documentación
  - Registro de Auditorías e Informes
  - Métodos para Medición y Evaluación

ESTÁNDAR ISO/IEC 12207



CALIDAD DEL SOFTWARE

- **Medición de software:** 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 (prescripciones) para el aseguramiento externo e interno de la calidad.
  - ISO 9126, etc.

MEDICIÓN DE SOFTWARE (PRESSMAN)

- La medición se utiliza a lo largo de un proyecto de software como apoyo en:
  - La estimación,
  - El control de calidad,
  - La valoración de la productividad,
  - El control del proyecto.
- La medición es aplicada por los ingenieros de software para auxiliar la evaluación de la calidad de los productos de trabajo y para apoyar la toma de decisiones

MEDICIONES DE SOFTWARE

Mejora objetiva: midiendo atributos.

- Identificar atributos.
- Desarrollar métricas para esos atributos.
- Utilizar las métricas para obtener indicadores.
- Planificar e implementar el proceso de mejora.

MÉTRICAS DEL PROYECTO

- Primera aplicación de las métricas del proyecto ➔ ¿Cuándo?
- A medida que avanza el proyecto, las medidas del esfuerzo y el tiempo se comparan con las de la planificación.
- El líder de proyecto utiliza estos datos para supervisar y controlar el avance.
- Ejemplos:
  - tiempo y esfuerzo medios de corrección de errores,
  - errores detectados antes de la entrega del software,
  - defectos detectados e informados por los usuarios finales, productos de trabajo entregados, etc.
  - …

MÉTRICAS DEL PROYECTO

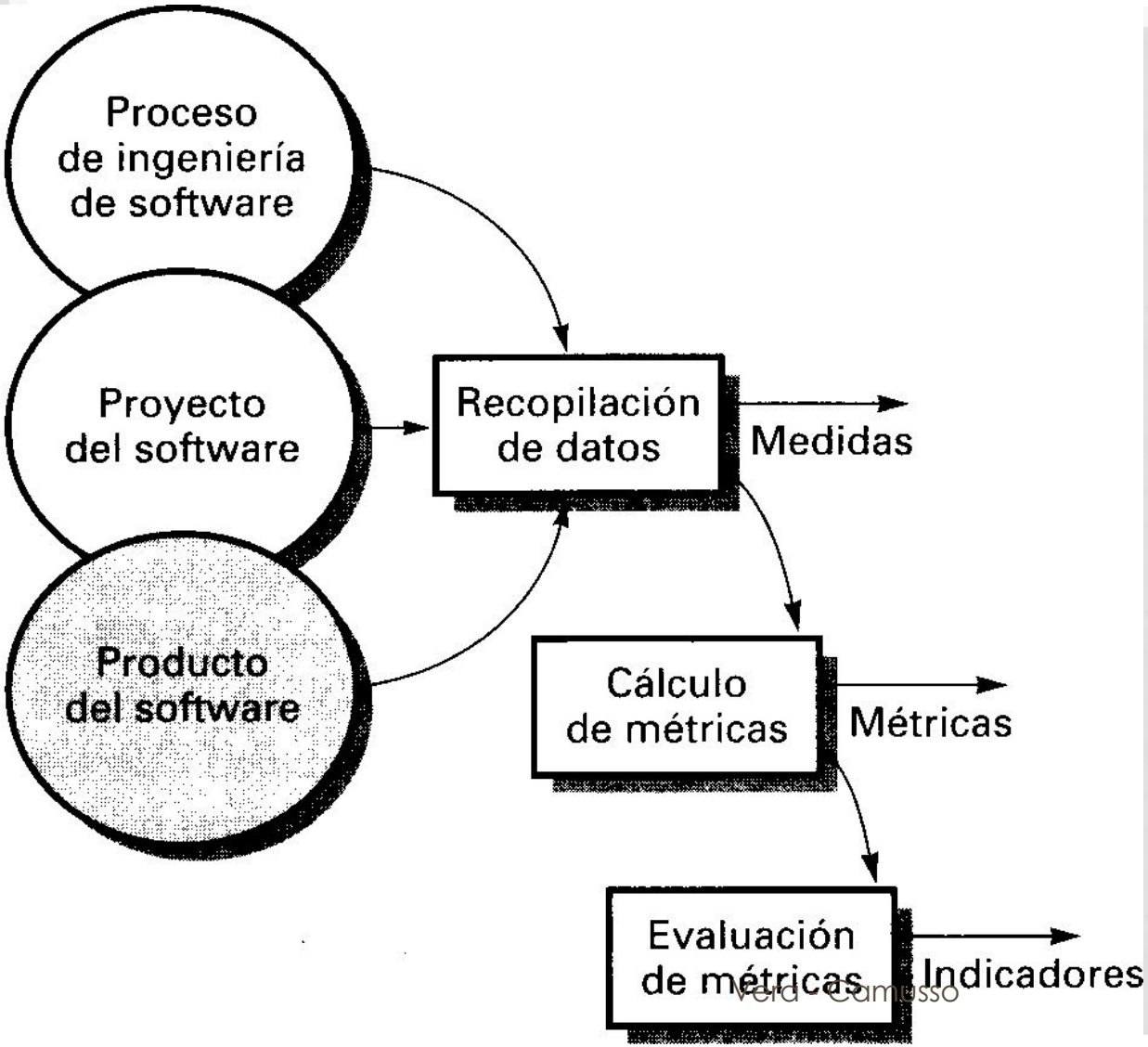
➔ La utilización fundamental de las métricas del proyecto son dos:

- *Optimizar la planificación*, guiando los ajustes necesarios que eviten retrasos y mitiguen problemas y riesgos potenciales.
- *Evaluar la calidad de los productos en el momento actual*, modificando el enfoque técnico para mejorar la calidad, si es necesario.

MÉTRICAS: DIMENSIONES [SEI, 1997]

- Métricas del Producto.
  - Asegurar la aceptación del cliente.
  - Cuestiones de mayor peso: principalmente, las relativas a los atributos físicos y dinámicos del producto (arquitectura, productividad, confiabilidad, usabilidad, estabilidad, performance, etc.)

PROCESO DE RECOPILACIÓN DE MÉTRICAS



PROGRAMA DE MÉTRICAS DE SOFTWARE DIRIGIDO POR METAS [SEI, 1997]

- 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** y los **indicadores** relacionados que se emplearán como apoyo para lograr los objetivos.
- 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.

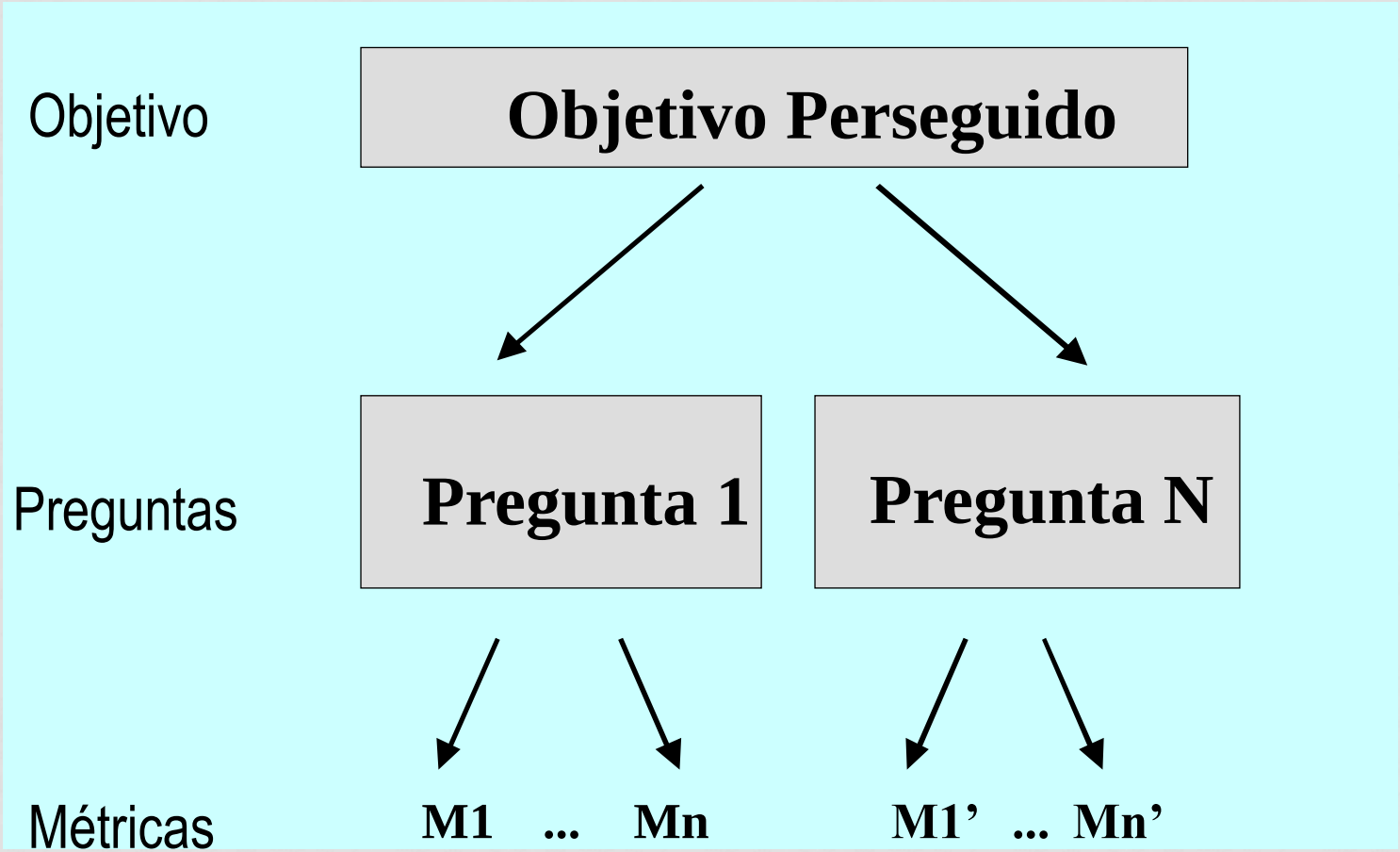
PARADIGMA GQM (GOAL-QUESTION-METRIC) [BASILI Y WEISS, 1984]

- Fundamento:

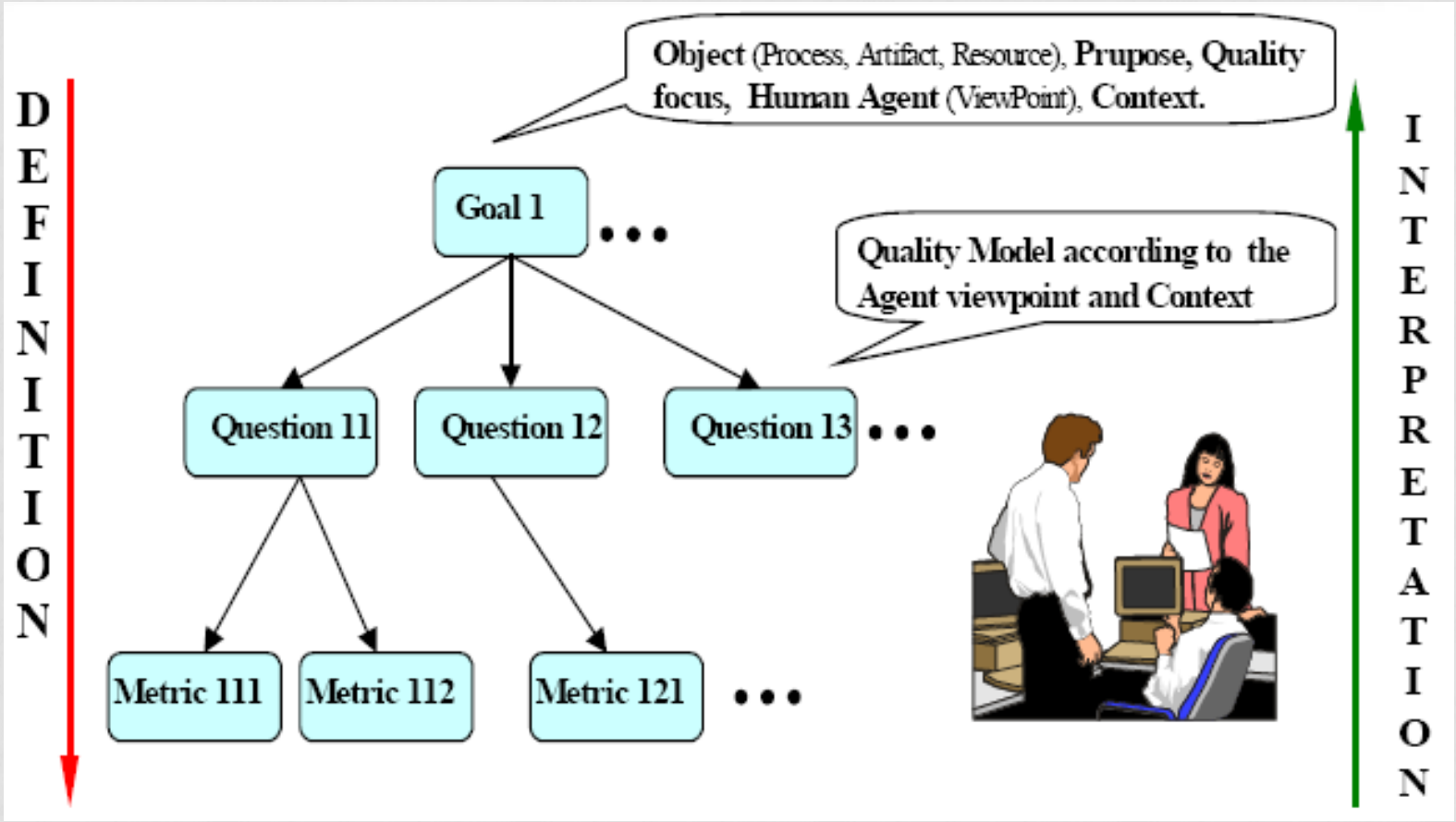
  "La medición debe basarse en modelos y objetivos".

- Debemos entonces establecer objetivos medibles y dirigidos por el modelo apropiado.
- GQM:
  - Se utiliza a nivel de proceso y proyecto para decidir qué mediciones hacer y cómo utilizarlas.
  - Provee un marco de trabajo para interpretar los datos y entender el enfoque sobre los objetivos.

PARADIGMA GQM



PARADIGMA GQM



PARADIGMA GQM

- **Metas** (goals): aquello que la organización o el proyecto intenta alcanzar. Se define un objetivo para una entidad. Por ejemplo:
  1. mejorar la productividad de los programadores,
  2. reducir tiempos de desarrollo,
  3. incrementar la confiabilidad del producto,
  4. ...

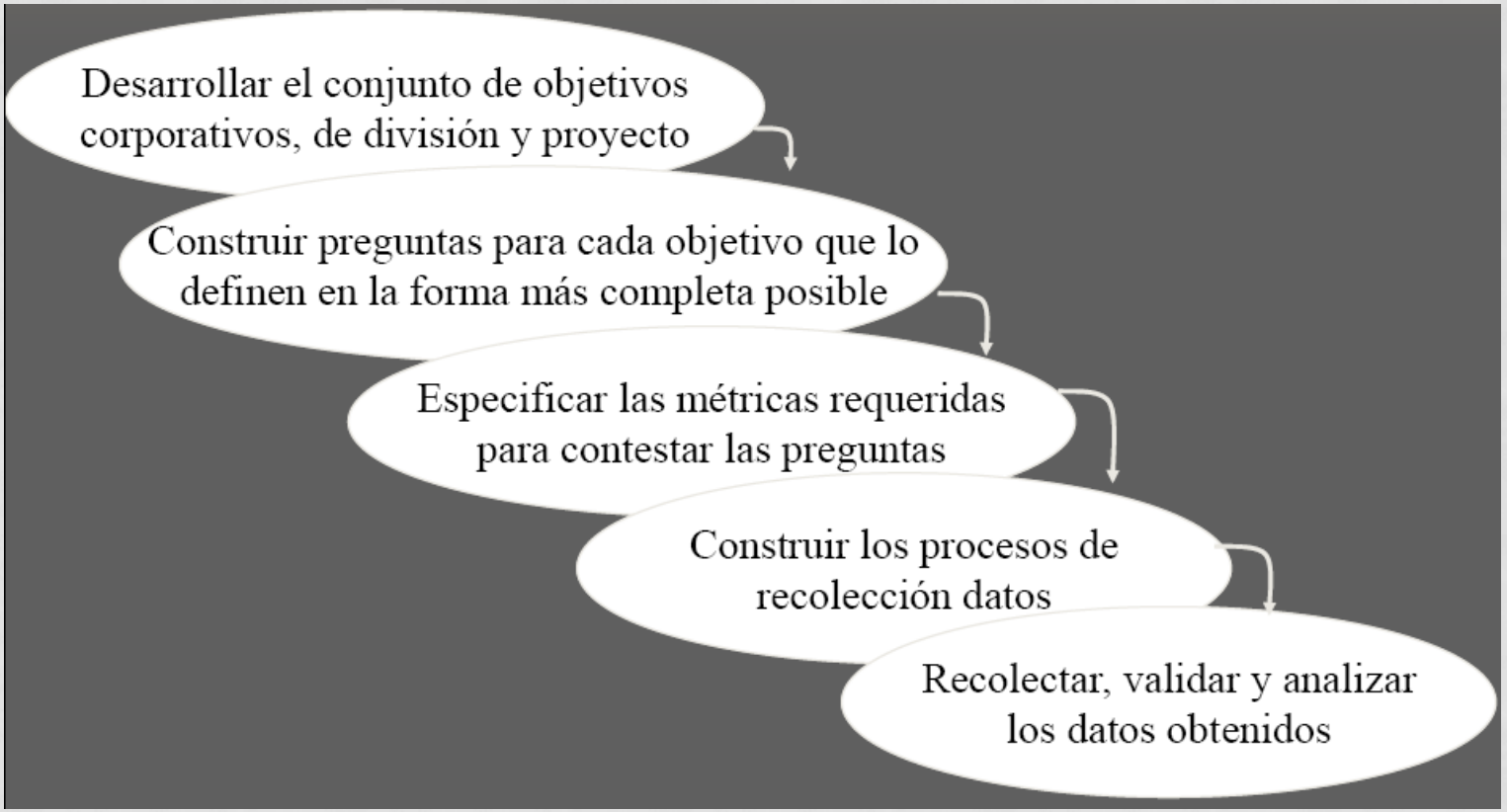
PARADIGMA GQM

- **Preguntas** (questions): refinamientos de las metas en las que se identifican áreas específicas de incertidumbre. Ejemplos:
  1. (productividad) ¿cómo se puede incrementar el número de LDC depuradas?
  2. (tpo. desarrollo) ¿cómo se puede reducir el tiempo requerido para finalizar la implementación de los requerimientos?
  3. (confiabilidad) ¿cómo se pueden llevar a cabo evaluaciones de confiabilidad más efectivas?

PARADIGMA GQM

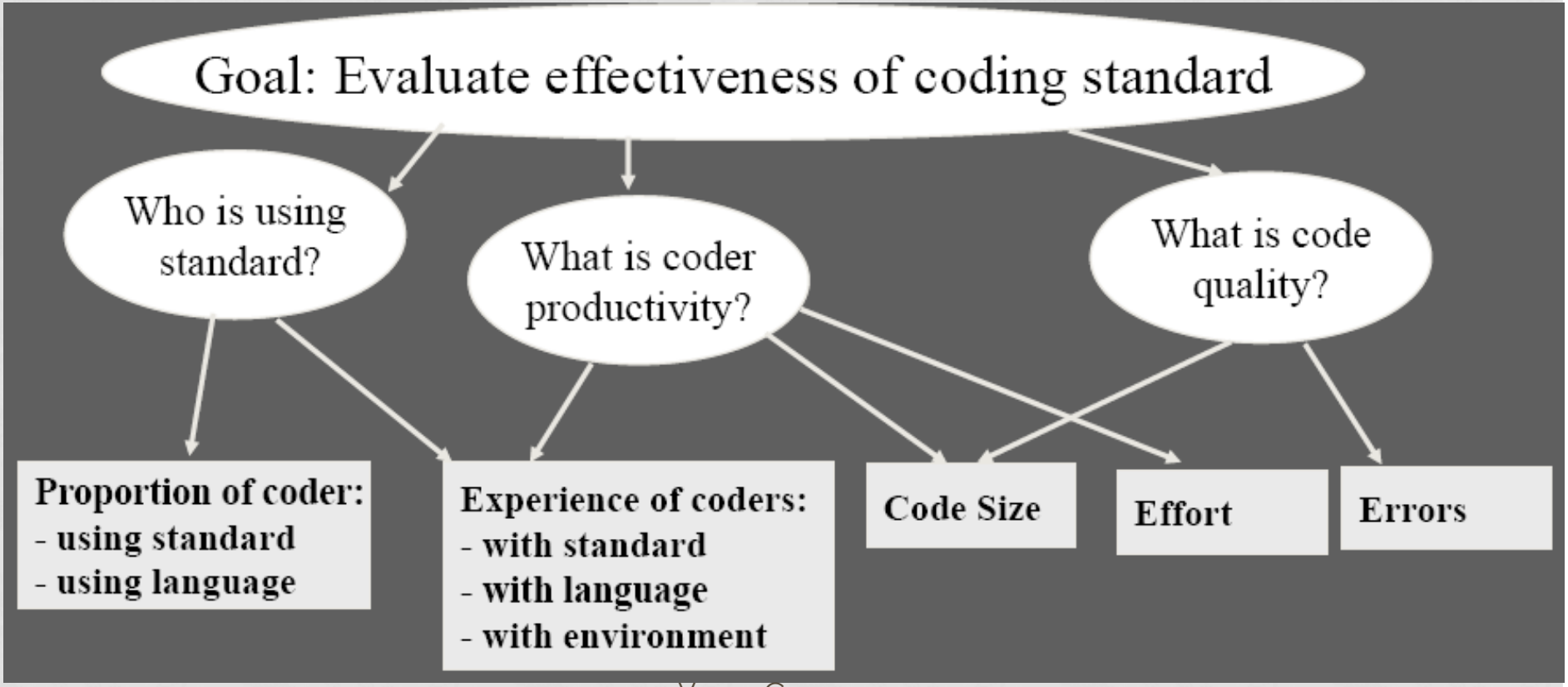
- **Métricas**: las mediciones necesarias para ayudar a responder a las preguntas y confirmar si las mejoras del proceso o proyecto cumplieron su objetivo. Ejemplos:
  1. (productividad) medir la productividad de los programadores en LDC y nivel de experiencia.
  2. (tpo. desarrollo) medir número de comunicaciones formales entre cliente y analista para cada cambio de requerimientos.
  3. (confiabilidad) medir el número de pruebas requeridas para provocar una caída en el producto.

GQM: PASOS BÁSICOS



GQM: EJEMPLO

Objetivo: "Efectividad de usar estándares para la codificación"



GQM: EJEMPLO 2

- **Objetivo:** "Mejorar la Planificación del Tiempo"
- **Pregunta:** ¿Cuál fue la exactitud de la estimación del cronograma del proyecto?
- **Métrica:** Exactitud en Estimación del Cronograma (EEC)

EEC = Duración real del proyecto / duración estimada del proyecto

GQM: EJEMPLO 3

- **Objetivo:** "Rediseñar la página web para facilitar su uso"
- **Preguntas:**
  - ¿Qué enlaces se utilizan más?
  - ¿Cuán fácil es acceder a la información?
  - ¿…?
- **Métricas:**
  - Cantidad de accesos a los enlaces.
  - Cantidad de saltos desde la página de inicio.
  - ¿…?

GQM: propuesta de pasos

1. Identificar diversas entidades que necesitan ser evaluadas para lograr la meta planteada.
   - productos, recursos, artefactos, actividades
   - código fuente, casos de prueba, solicitudes de cambio, tareas de ingeniería de software, planificación, etc.
2. Identificar preguntas asociadas a cada una de las entidades definidas del proyecto relacionadas con el objetivo planteado.

GQM: propuesta de pasos

3. Considerando las preguntas definidas:
   - Agrupar preguntas relacionadas (agrupando cuestiones relativas a algún área particular, al producto de software, gestión de proyecto, gestión de cambios, interacciones con el cliente).
   - Identificar para cada grupo una submeta o subobjetivo relacionado con el original.
4. Identificar subentidades y atributos necesarios a evaluar para las submetas definidas en el punto anterior.
5. Definir de manera completa métricas e indicadores para los atributos identificados en el punto anterior.

GQM: PARA PENSAR…

Objetivo: "Aumentar la Satisfacción del Cliente"

- Preguntas?
- Métricas?

GQM: PARA PENSAR…

Objetivo: "Aumentar la Satisfacción del Cliente"

1. Entidades:
   - Requerimientos de software.
   - Manual de Usuario.
   - Interfaces.
   - Casos de prueba.
   - Plazos.
   - Contrato.
   - ….

GQM: PARA PENSAR…

2. Preguntas para las entidades:
   - Requerimientos de software:
     - ¿Se elicitaron completamente? ¿Se cumplieron en su totalidad?
   - Documentación:
     - ¿Es completa? ¿Es entendible? ¿Es útil? …
   - Interfaces:
     - ¿Son atractivas? ¿Muestran la información deseada? ¿El diseño es intuitivo? …
   - Casos de prueba.
     - ¿Son suficientes los casos de prueba? ¿Cuántos defectos encontró el cliente? …
   - Plazos.
   - Contrato.
   - ….

GQM: PARA PENSAR…

3. a) Áreas:
   - Relación con el cliente.
   - Calidad del producto de software.
   - Calidad de la documentación.

   b) Submetas relacionadas con la original
   - Elicitar y cumplir la totalidad de requerimientos.
   - Mejorar la interacción con el cliente.
   - Minimizar los defectos encontrados por el cliente.
   - Mejorar la calidad de la documentación del producto.

GQM: PARA PENSAR…

4. Subentidades y atributos:
   - **Requerimientos**: Requerimientos encontrados, requerimientos cumplidos, requerimientos no cumplidos, requerimientos modificados durante el proceso de desarrollo, etc.
   - **Defectos y Errores**: Defectos encontrados antes de la entrega, errores encontrados por el cliente, etc.
   - **Manual de Usuario**: tamaño, legibilidad, consistencia, errores, modificabilidad, etc.

GQM: PARA PENSAR…

5. **Métricas:**
   - **Requerimientos**:
     - #Reqs Cumplidos / #Reqs Totales
     - #Reqs Validados / #Reqs Totales
   - **Defectos y Errores**:
     - #Defectos / (#Defectos + #Errores)
   - **Manual de Usuario**:
     - Índice Fog: Medida de la longitud promedio de las palabras y declaraciones en los documentos ➔ a mayor índice de Fog, más difícil será comprender el documento.

¿qué está faltando? → INDICADORES

GQM: OTRO EJEMPLO PARA PENSAR…

Objetivo: "Disminuir el retrabajo en desarrollo"

- Preguntas?
- Métricas?

MÉTRICAS PRIVADAS VS. MÉTRICAS PÚBLICAS

- Métricas Públicas: Engloban información originalmente privada para particulares y equipos.
  - Ejemplos: esfuerzo total, 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.

MÉTRICAS: PRIVACIDAD SEGÚN NIVELES [WIEGERS, 2007]

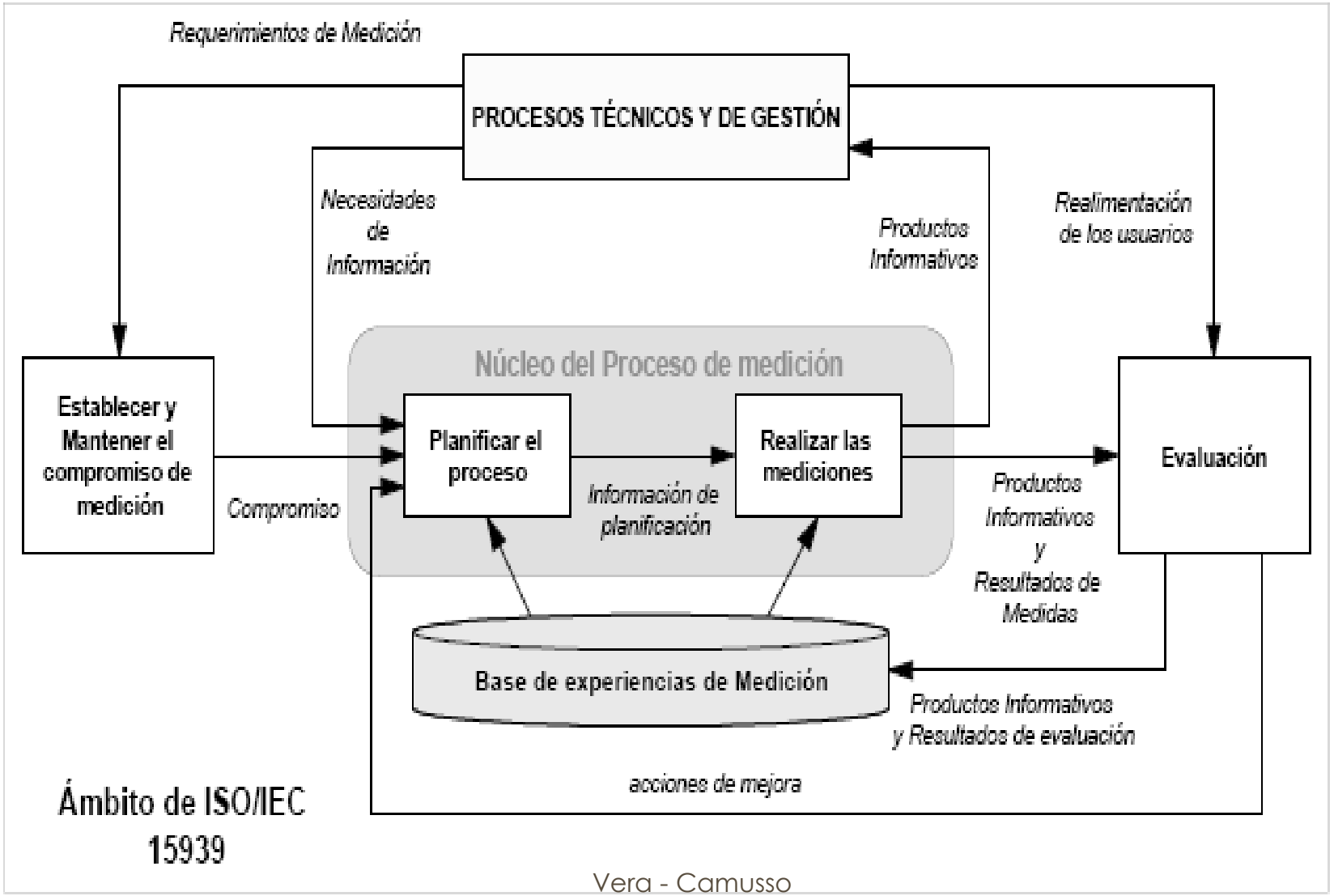
- Propone clasificar las métricas directas o básicas en algún nivel de privacidad:
  - **Individual:** sólo los individuos que recolectan los datos sobre su propio trabajo conocen su información. Por ej.: horas semanales individuales dedicadas a actividades de desarrollo.
  - **Equipo de proyecto:** Los datos son privados para el equipo de proyecto. Por ej.: horas semanales totales dedicadas a actividades de desarrollo.
  - **Organización:** los datos son compartidos a lo largo de la organización. Por 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 software e ingeniería de sistemas.

ACTIVIDAD	TAREAS
Establecer y Mantener el Compromiso de Medición	Aceptar los requisitos de la medición
	Asignar recursos
Planificar el Proceso de Medición	Obtener las características de la organización
	Identificar las necesidades de información
	Seleccionar las medidas
	Definir los procedimientos de recolección de datos, análisis e informes
	Definir los criterios de evaluación de los productos de información y el proceso de medición
	Revisar, aprobar y proporcionar recursos para las tareas de medición
	Adquirir y utilizar tecnologías de apoyo
Realizar el Proceso de Medición	Integrar los procedimientos
	Recoger los datos
	Analizar los datos y desarrollar productos de información
	Comunicar los resultados
Evaluar la Medición	Evaluar los productos de información y el proceso de medición
	Identificar las mejoras potenciales

ISO/IEC 15939 - 2007



… PARA CONSIDERAR …

"No se puede controlar lo que no se puede medir"

Tom De Marco

"No se puede predecir lo que no se puede medir"

Norman Fenton

REFERENCIAS

- **Bibliografía Obligatoria**
  - [Pressman, 2005] Pressman, Roger S. "Ingeniería del Software: un Enfoque Práctico". **Capítulos 4**. 6ta. Edición. Mc Graw-Hill. 2005.
  - [Sommerville, 2005] Sommerville, Ian. "Ingeniería de Software". **Capítulo 27**. Séptima Edición. Pearson Education, 2005.
- **Bibliografía Complementaria**
  - [Basili, Caldiera y Rombach, 1994]. Basili, V., Caldiera, G., Rombach, H.D. The Goal Question Metric Approach. 1994.
  - [SEI, 1997]. Florac, W.A.; Park, R.E.; Carleton, A.D. Practical Software Measurement: Measuring for Process Management and Improvement. April, 1997.
  - [ISO, 2007] Std. ISO/IEC 15939-2007, Software Engineering - Software Measurement Process.
  - [Franzoy y Vrancken, 2010]. Franzoy, M. y Vrancken, L. Sistema de Medición y Análisis. Proyecto Final de Carrera, Ingeniería en Sistemas de Información. FRSF – UTN. Octubre, 2010.
  - [Wiegers, 2007]. Wiegers, K.E. Practical Project Initiation: A Handbook with Tools. Microsoft Press, 2007.

INGENIERÍA DEL SOFTWARE

UNIDAD 3: ESTIMACIÓN DE PROYECTOS DE SOFTWARE
CICLO LECTIVO 2013

OBJETIVOS DE LA CLASE

- Gestión de proyectos de software.
- Complejidad del proyecto.
- Tamaño del proyecto.
- Grado de incertidumbre.
- Estimaciones.
- Factores críticos del éxito.

GESTIÓN DE PROYECTOS DE SOFTWARE

- La gestión de un proyecto de software comienza con un conjunto de actividades que globalmente se denominan ***planificación del proyecto***. *Antes de que el proyecto comience,* el gestor y el equipo de software deben realizar una estimación del trabajo a realizar, de recursos necesarios y del tiempo que transcurrirá desde el comienzo hasta el final de su realización.
- Siempre que estimamos, estamos mirando hacia el futuro y aceptamos cierto grado de incertidumbre

GESTIÓN DE PROYECTOS DE SOFTWARE

- Existen técnicas útiles para la estimación del esfuerzo y del tiempo.
- Las métricas del proyecto y del proceso proporcionan una perspectiva histórica y una potente introducción para generar estimaciones cuantitativas. La experiencia anterior (de todas las personas involucradas) puede ayudar en gran medida al desarrollo y revisión de las estimaciones.

GESTIÓN DE PROYECTOS DE SOFTWARE

- La estimación de recursos, costos y planificación temporal de un esfuerzo en el desarrollo de software requiere experiencia, acceder a una buena información histórica y el coraje de confiar en predicciones (medidas) 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.

COMPLEJIDAD DEL PROYECTO

- La complejidad del proyecto tiene un gran efecto en la incertidumbre, que es inherente en la planificación. Sin embargo, la complejidad es una medida relativa que se ve afectada por la familiaridad con esfuerzos anteriores.
- Se podría considerar una aplicación sofisticada de comercio electrónico como ***excesivamente compleja*** para un desarrollador que haya realizado su primera aplicación. Sin embargo para un equipo de software que desarrolle su enésimo sitio web de comercio electrónico podría considerarse ***sumamente fácil*** (una de tantas).

COMPLEJIDAD DEL PROYECTO

- Se han propuesto una serie de medidas cuantitativas de la complejidad del software. Tales medidas se aplican en el nivel de diseño y de codificación, y por consiguiente son difíciles de utilizar durante la planificación del software (antes de que exista un diseño o un código).
- Sin embargo, 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

- El tamaño del proyecto es otro factor importante que 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 software. El problema de la descomposición, un enfoque importante hacia la estimación, se hace más difícil porque los elementos descompuestos pueden ser todavía excesivamente grandes.
- Parafraseando la ley de Murphy: "lo que puede ir mal irá mal", y si hay más cosas que puedan fallar, más cosas fallarán.

GRADO DE INCERTIDUMBRE ESTRUCTURAL

- El grado de incertidumbre estructural tiene también efecto en el riesgo de la estimación.
- En este contexto, 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.

GRADO DE INCERTIDUMBRE ESTRUCTURAL

- 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 del proyecto o los requisitos del proyecto están sujetos a cambios, la incertidumbre y el riesgo son peligrosamente altos. El planificador del software debería solicitar definiciones completas de rendimiento y de interfaz (dentro de una especificación del sistema).
- El planificador y, lo que es más importante, el cliente, deben tener presente que cualquier cambio en los requisitos del software significa inestabilidad en el coste y en la planificación temporal.

FIABILIDAD DE LA ESTIMACIÓN

- la complejidad del proyecto,
- el tamaño del proyecto y
- el grado de incertidumbre estructural

afectan a la fiabilidad de la estimación.

FACTORES CRÍTICOS DE ÉXITO

- Soporte Ejecutivo.
- Involucramiento del Cliente.
- Gerente de Proyecto experimentado.
- Objetivos de negocio claros.
- Alcances claros y acotados.
- Infraestructura tecnológica probada.
- Requerimientos consolidados.
- Proceso formal y maduro de desarrollo.
- Estimaciones realistas.

ESTIMACIONES

- No son simples, pero se necesitan a menudo.
- Una organización solo mejora en sus estimaciones cuando las usa:

Falacia: Como estimamos mal, no vale la pena hacerlo.

- Creadas, Usadas y Refinadas durante
  - Planeamiento Estratégico.
  - Factibilidad/SOW / Propuesta
  - Evaluación de Sub-Contratistas
  - Planeamiento básico de proyecto (iterativo)

ESTIMACIONES

- Una "estimación exacta" es una contradicción.

Integrar:

- Los factores discernibles que afectan la estimación;
  - Tamaño, Complejidad, Capacidades…..
  - Experiencias anteriores.
- Ningún proyecto soporta ser estimado en el "peor caso".
- Tampoco en el "mejor caso".
- Ideal: Media + 5-10%.
- En nuestra industria las estimaciones suelen estar corridas en 25-100%

ESTIMACIONES

- 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.
- La estimación es muy precisa al final, pero ahí no se necesita…
  - Pero es importante capturarla para el siguiente proyecto!!!
- Las mejores estimaciones se basan en la historia.
- Sobre-estimacion.
  - Ley de Parkinson, Costo inviable.
- Infra-Estimacion.
  - Escasez de Recursos.
- Cambios
  - Tecnología, nuevas funciones.

OBJETIVO DE LA PLANIFICACIÓN

Cuanto más sepa, mejor realizará la estimación.
Por consiguiente, actualice sus estimaciones a medida que progresa el proyecto.

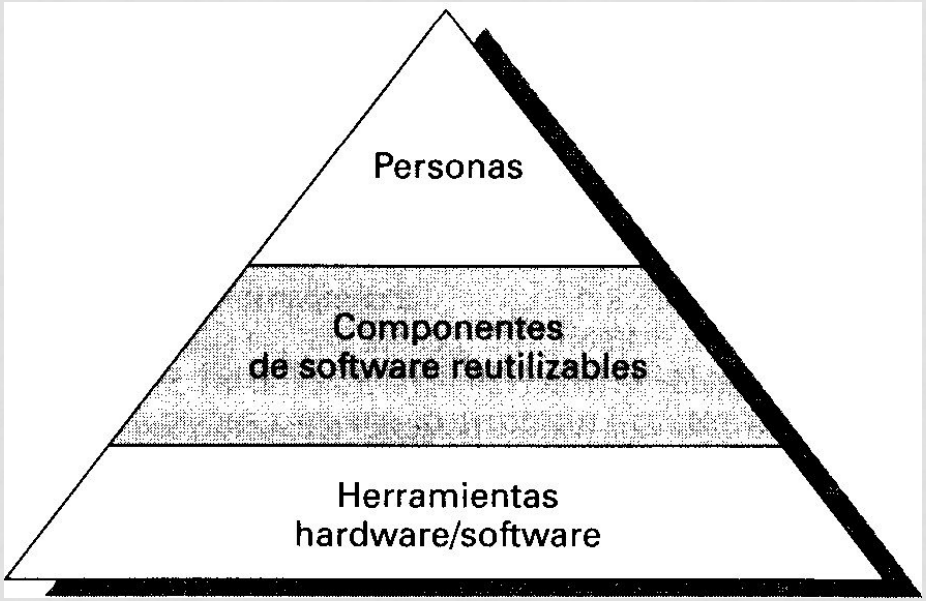
- Proporcionar un marco de trabajo que permita al gestor hacer estimaciones razonables de recursos, costo ***y planificación temporal***.
- Estas estimaciones se hacen dentro de un marco de tiempo limitado al comienzo de un proyecto de software, y deberían actualizarse regularmente a medida que progresa el proyecto.
- Las estimaciones deberían definir los escenarios del ***mejor caso*** y ***peor caso*** de forma que los resultados del proyecto puedan limitarse

ÁMBITO DEL SOFTWARE

- La primera actividad de la planificación del proyecto de software es determinar el ámbito del software. Se debe delimitar la declaración del ámbito del software.
- El ámbito del software describe el control y los datos a procesar, la función, el rendimiento, las restricciones, las interfaces y la fiabilidad.
- Para esto se trabaja con las técnicas ya vistas de recolección de información con todos los stakeholders.

RECURSOS

- La segunda tarea de la planificación del desarrollo de software es la estimación de los recursos requeridos para acometer el esfuerzo de desarrollo de software.



RECURSOS

- Cada recurso queda especificado mediante cuatro características: descripción del recurso, informe de disponibilidad, fecha cronológica en la que se requiere el recurso, tiempo durante el que será aplicado el recurso.
- Las dos últimas características pueden verse como una ventana temporal. La disponibilidad del recurso para una ventana específica tiene que establecerse lo más pronto posible.

RECURSOS HUMANOS

- El número de personas requerido para un proyecto de software sólo puede ser determinado después de hacer una estimación del esfuerzo de desarrollo (por ejemplo, personas-mes).
- Hay que especificar tanto la posición dentro de la organización (por ejemplo: gestor, ingeniero de software experimentado, etc.) como la especialidad (por ejemplo: telecomunicaciones, bases de datos, cliente/servidor).

RECURSOS DE SOFTWARE REUTILIZABLES

- La ingeniería del software basada en componentes destaca la reutilización, esto es, la creación y la reutilización de bloques de construcción de software.
- **Componentes ya desarrollados.** El software existente se puede adquirir de una tercera parte o provenir de uno desarrollado internamente para un proyecto anterior.
- **Componentes ya experimentados. Especificaciones,** diseños, código o datos de prueba existentes desarrollados para proyectos anteriores que son similares al software que se va a construir para el proyecto actual.

RECURSOS DE SOFTWARE REUTILIZABLES

- **Componentes con experiencia parcial. Especificaciones**, diseños, código o datos de prueba existentes desarrollados para proyectos anteriores que se relacionan con el software que se va a construir para el proyecto actual, pero que requerirán una modificación sustancial.
- **Componentes nuevos. Los componentes de software** que el equipo de software debe construir específicamente para las necesidades del proyecto actual.

RECURSOS DE ENTORNO

- El entorno es donde se apoya el proyecto de software, llamado a menudo entorno de ingeniería del software (EIS), incorpora hardware y software.
- Cuando se va a desarrollar un sistema basado en computadora (que incorpora hardware y software especializado), el equipo de software puede requerir acceso a los elementos en desarrollo por otros equipos de ingeniería.

ESTIMACIONES SEGURAS

- Dejar la estimación para más adelante (obviamente, podemos realizar una estimación al cien por cien fiable tras haber terminado el proyecto).
- Basar las estimaciones en proyectos similares ya terminados.
- Utilizar técnicas de descomposición relativamente sencillas para generar las estimaciones de coste y de esfuerzo del proyecto.
- Utilizar uno o más modelos empíricos para la estimación del coste y esfuerzo del software.

Desgraciadamente, la primera opción, aunque atractiva no es práctica.

¿Dudas, consultas?