FACULTAD DE INGENIERÍA

UNIVERSIDAD CÉSAR VALLEJO

Escuela Profesional de:
INGENIERÍA DE SISTEMAS

TALLER DE COMPETENCIAS ESPECÍFICAS

CALIDAD DEL SOFTWARE

EXPERIENCIA CURRICULAR

PRÁCTICA PREPROFESIONAL

Mgtr. Even Pérez Rojas

eperezr@ucv.edu.pe

Introducción

“Nosotros solo confiamos en Dios, para todo lo demás traigan datos”

W. Edward Deming

INTRODUCCIÓN

- **Medición del software**: necesidad de obtener datos objetivos que ayuden a mejorar la calidad.
- **Creación de modelos de calidad**: útiles para discutir, planificar y obtener índices de calidad.
- **Aplicación de estándares de calidad**: directrices para el aseguramiento externo e interno de la calidad.

CALIDAD TOTAL (TQ)

La **Calidad Total** implica poner énfasis anticipadamente en visualizar y comprender el destino, y luego saber como acceder a los caminos que conduzcan a ese destino.

La clave central del sistema de **Calidad Total es el consumidor.**

Actitud permanente que involucra a todo el personal en su conjunto.

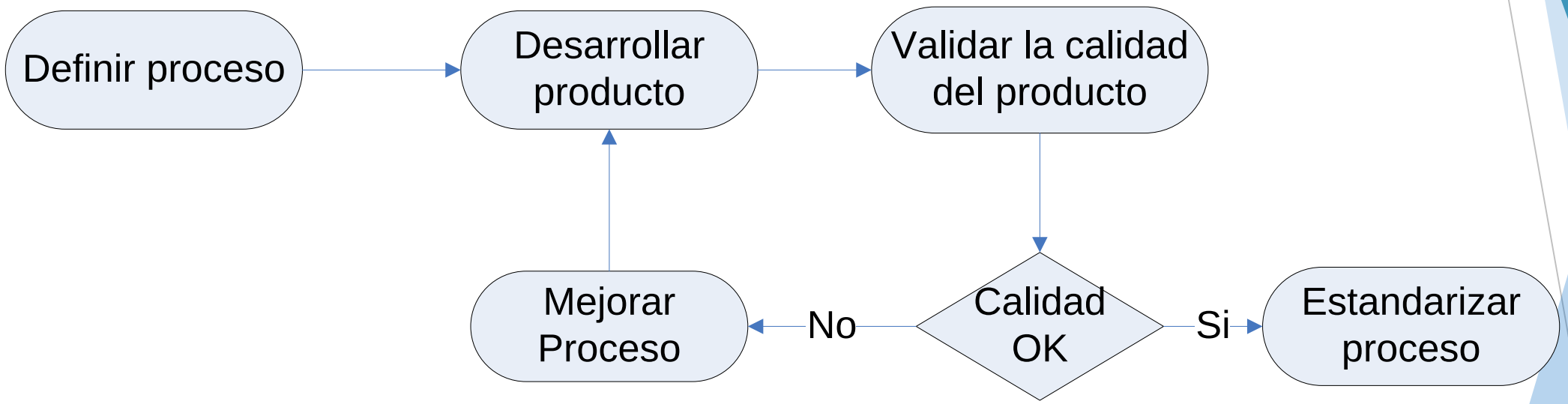
Rubén Roberto Rico

Administración de la calidad del Software

- Se refiere a lograr un nivel de calidad requerido en el producto de software.
- Involucra a la definición de estándares de calidad apropiados y procedimientos que permitan asegurar que estos se cumplan.
- Debe llevar a desarrollar una cultura de calidad en donde la calidad es responsabilidad de todos.

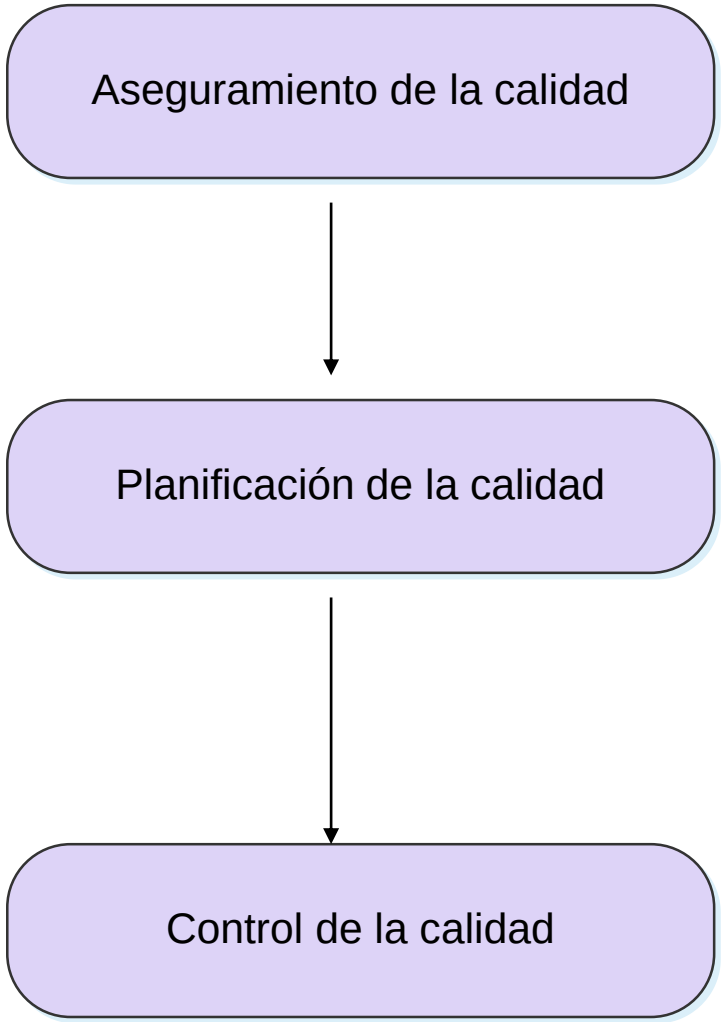
Administración de Calidad

- Calidad del proceso y del producto



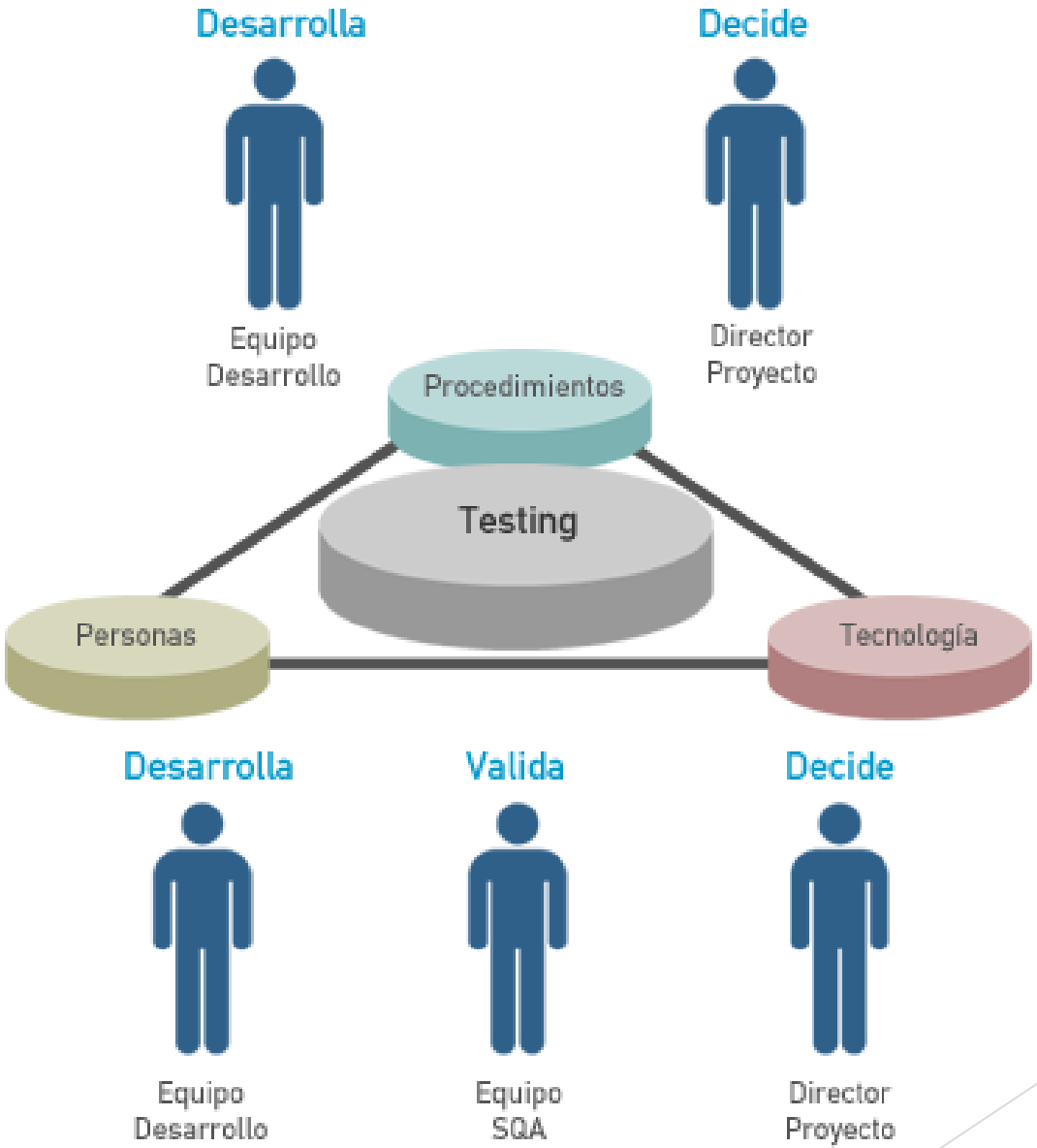
Calidad basada en procesos

Administración de la calidad del Software



- tres actividades principales
  - **Aseguramiento de la calidad**
    - Establecimiento de un marco de trabajo de procedimientos y estándares corporativos que conduzcan a la obtención de software de alta calidad.
  - **Planificación de la calidad**
    - Selección de procedimientos y estándares adecuados a partir de ese marco de trabajo y adaptación de éstos para un proyecto de software específico.
  - **Control de la calidad**
    - Definición y aplicación de los procesos que aseguren que los procedimientos y estándares son seguidos por el equipo de desarrollo.

¿Qué es la calidad de software?



Definición de calidad de software: Pressman

- Conformidad con los requisitos funcionales y de rendimiento, estándares explícitos de desarrollo y las características implícitas que se esperan de todo el software desarrollado profesionalmente.
- Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente que desea el usuario.

Definición de calidad de software: IEEE

1. El nivel en el que un sistema, componente o proceso cumple con los requisitos especificados.
2. El nivel en el que un sistema, componente o proceso cumple con las necesidades o expectativas del cliente o usuario.

Definición de calidad de software: ISO

La totalidad de características de un producto, proceso o servicio que cuenta con la habilidad de satisfacer necesidades explícitas o implícitas.

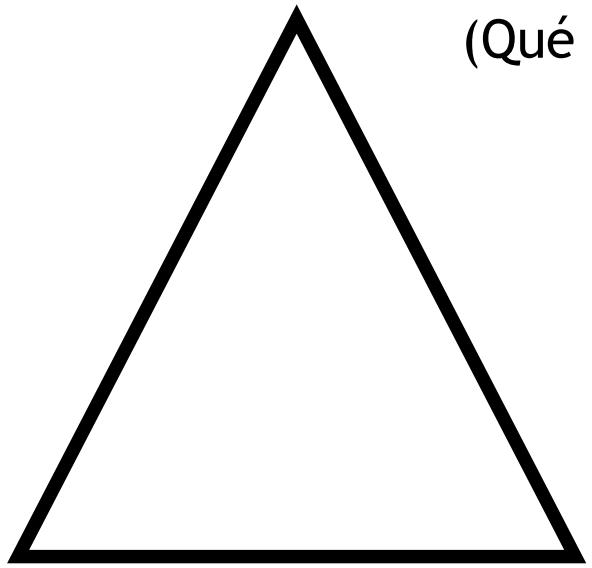
Definición de aseguramiento de calidad de software: IEEE

1. Un patrón planeado y sistemático de todas las acciones necesarias para generar la confianza adecuada que un elemento o producto cumple con los requisitos técnicos establecidos.
2. Conjunto de actividades diseñadas para evaluar el proceso con el que los productos se desarrollan y producen.

PERSPECTIVA DE LA CALIDAD DEL SOFTWARE

PRODUCTO

(Qué hacemos)



CLIENTE
(Para quien lo hacemos)

PROCESO
(Cómo lo hacemos)

¿Cómo controlar la calidad del software?

- Software para explotar durante un largo período (10 años o más) necesita ser confiable, mantenible y flexible.
- Es necesario, ante todo, definir los parámetros, indicadores o criterios de medición.
- Tom De Marco: “No se puede controlar lo que no se puede medir".

MODELOS DE CALIDAD

- **Modelo de Boehm [Boehm et al., 1978]**
- **Modelo FCM** (F*actors/Criteria/Metrics*) [McCall et al., 1977]
- **Marco ISO 9126** [ISO/IEC, 1991]:
- **Paradigma GQM** (*Goal-Question-Metric*) [Basili y Rombach, 1988]:
- **Modelo de Gilb** [Gilb, 1988]:
- **Modelo CMM** (*Capability Maturity Model*) [Paulk, 1993]:
- **Modelo SPICE** (*Software Process Improvement and Capability determination*) [Rout, 1995], [SPICE, 1999]:

MÉTRICAS

- Por término general, para la evaluación de la calidad, es más habitual centrarse en medidas del producto que en medidas del proceso.
- Una métrica es una asignación de un valor a un atributo (tiempo, complejidad, etc.) de una entidad software, ya sea un producto (código) o un proceso (pruebas).

MODELO DE CALIDAD DE MCCALL

Modelo factores/criterios/métricas (McCall) (ii)

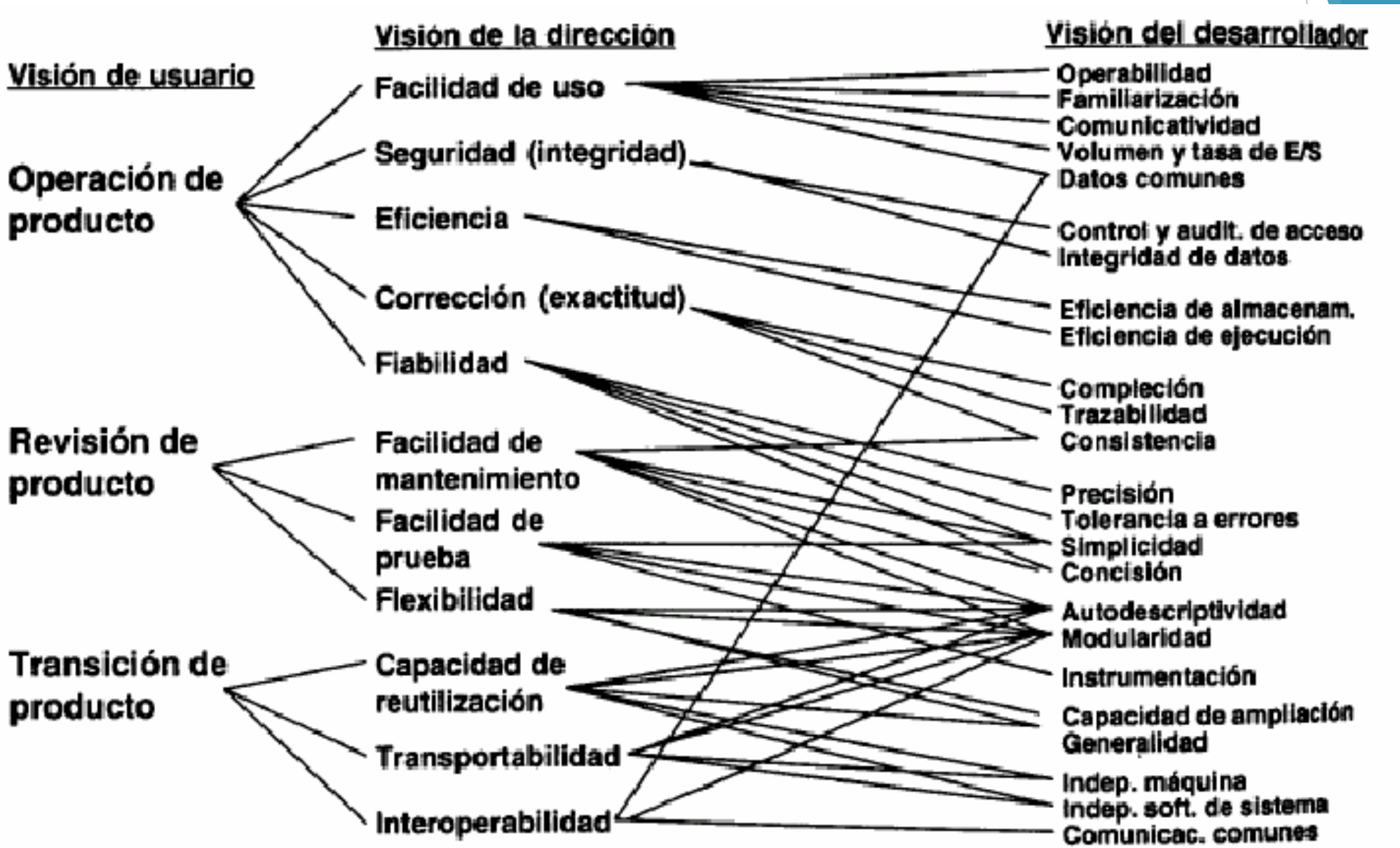
- Cada factor determinante de la calidad se descompone en una serie de criterios o propiedades que determinan su calidad.
- Los criterios pueden ser evaluados mediante un conjunto de métricas. Para cada criterio deben fijarse unos valores máximo y mínimo aceptables para cada criterio.

Modelo de Calidad de McCall

Visión del Usuario

Ejes (o punto de vista)	Factores
***Operación* del Producto**	●Facilidad de Uso ●Integridad ●Corrección ●Fiabilidad ●Eficiencia
***Revisión* del Producto**	●Facilidad de Mantenimiento ●Facilidad de Prueba ●Flexibilidad
***Transición* del Producto**	●Facilidad de Reutilización ●Interoperabilidad ●Portabilidad

Modelo factores/criterios/métricas (McCall) (iii)

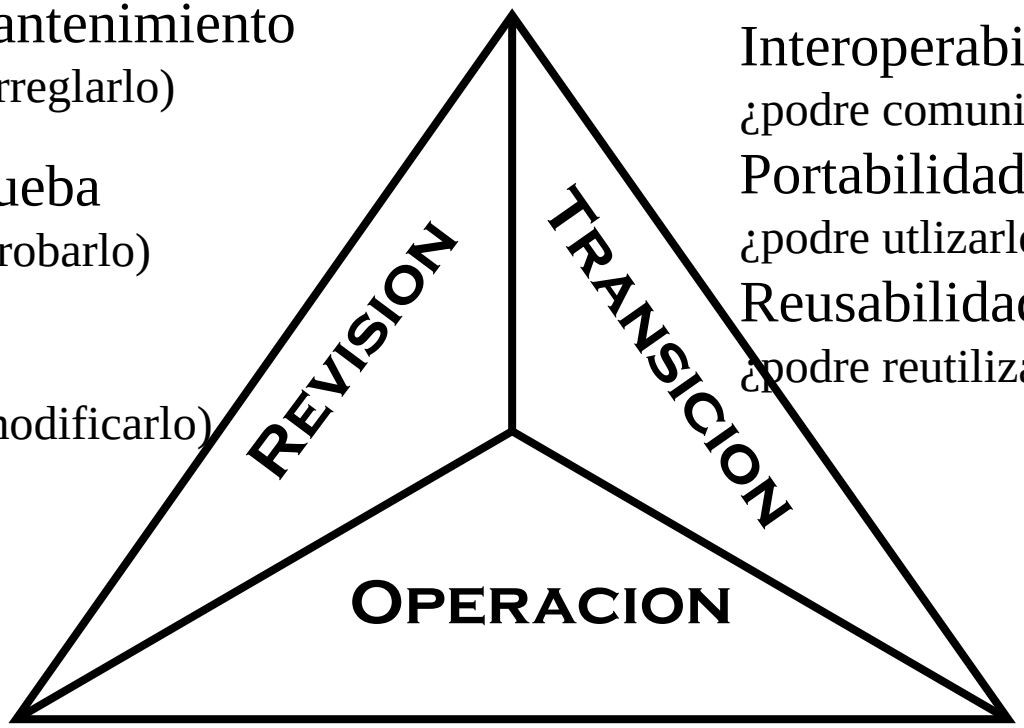


Modelo de Calidad de McCall

Facilidad de Mantenimiento
(puedo arreglarlo)

Facilidad de Prueba
(puedo probarlo)

Flexibilidad
(puedo modificarlo)



Interoperabilidad
¿podre comunicarlo con otro sistema?

Portabilidad
¿podre utlizarlo en otra maquina?

Reusabilidad
¿podre reutilizar parte del software?

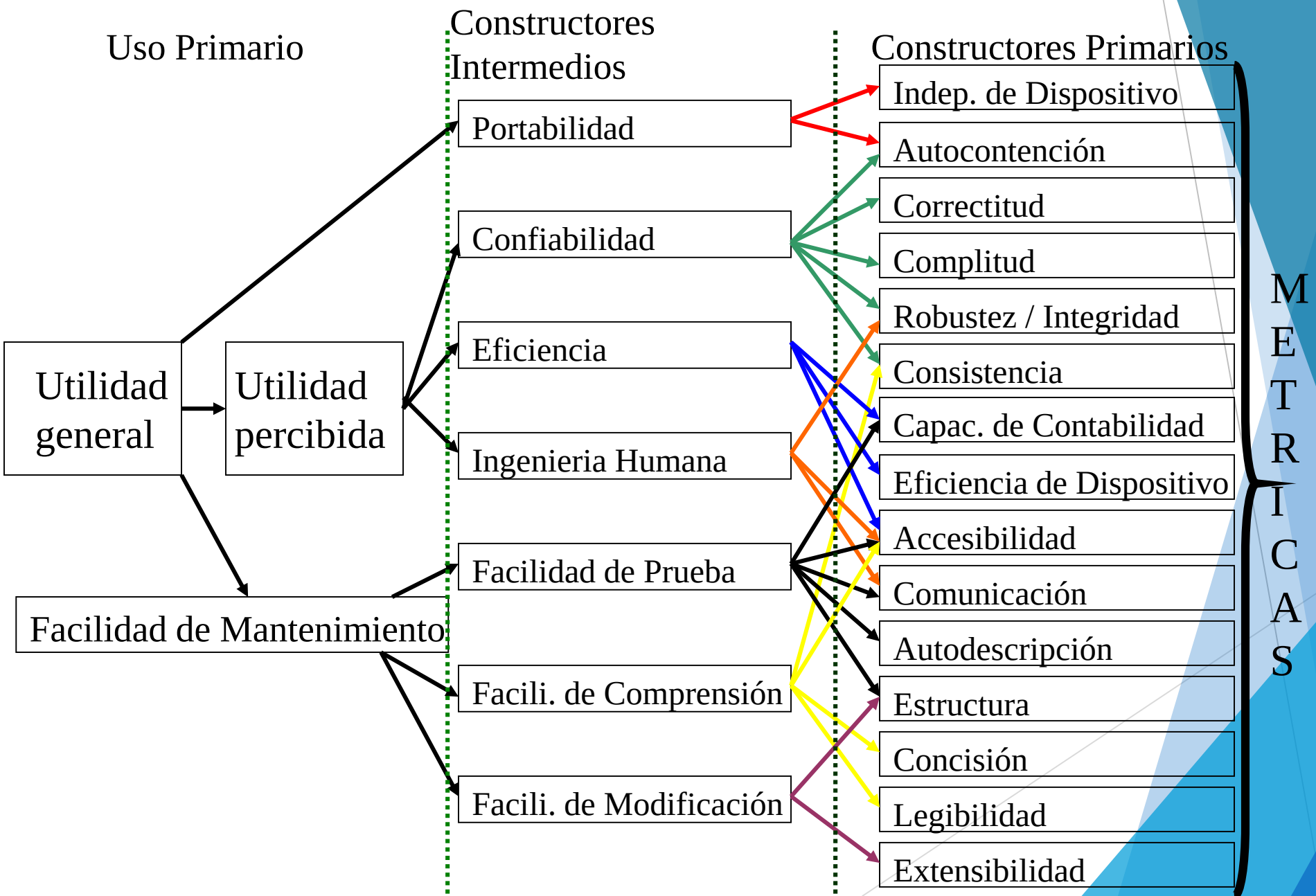
Corrección ¿hace el software lo que yo quiero?
Fiabilidad ¿lo hace de forma exacta todo el tiempo?
Eficiencia ¿se ejecutará sobre mi hardware lo mejor posible?
Integridad ¿es seguro?
Facilidad de uso ¿puedo ejecutarlo?

MODELO DE BOEHM

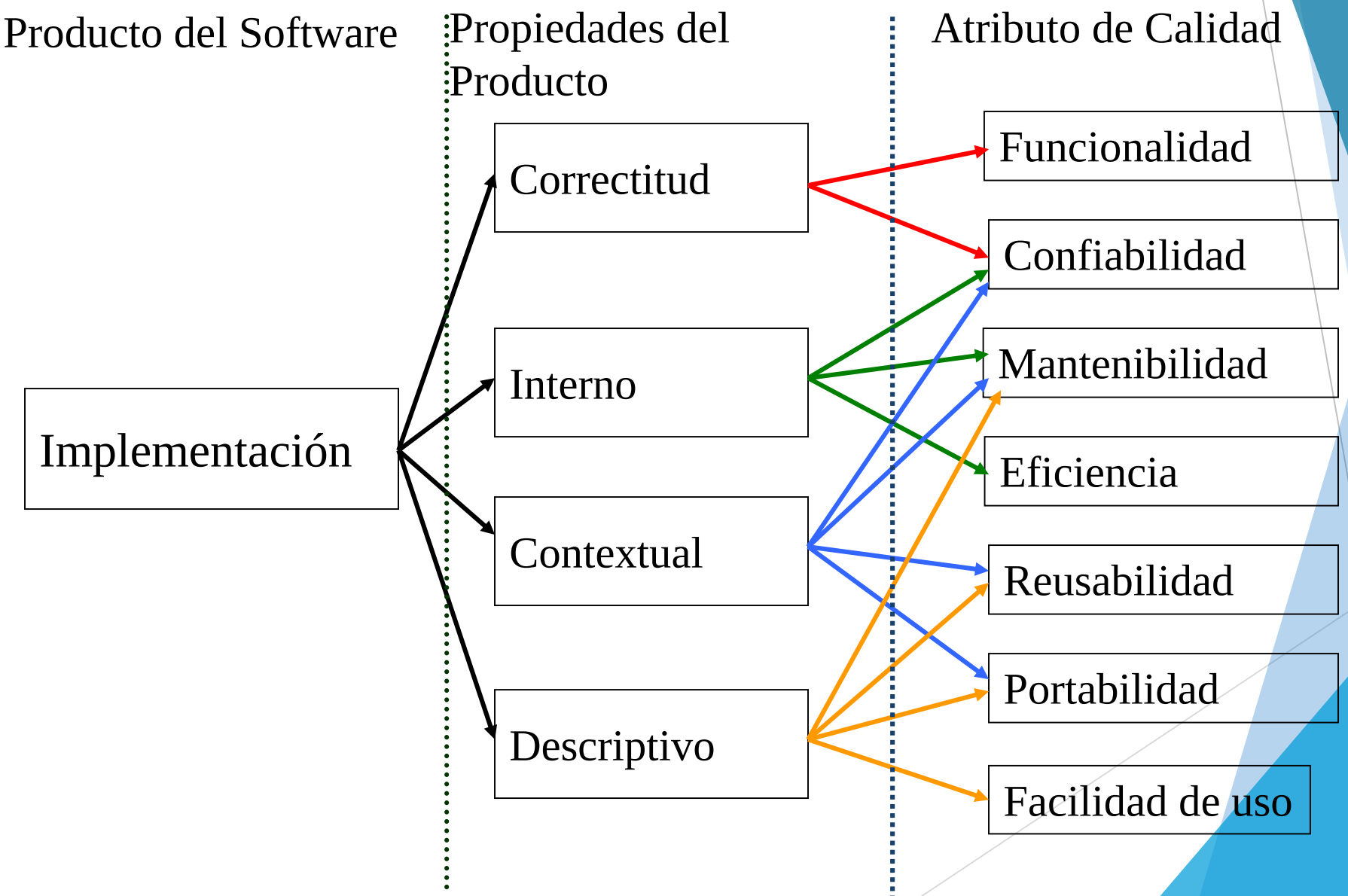
Este aseguramiento de la calidad se realiza a través de modelos. Los más conocidos son los siguientes:

- modelo de descomposición de características de calidad del software en tres niveles antes de aplicar las métricas: usos principales, componentes intermedios y componentes primitivos.
- Los componentes o constructores del modelo se centran en el producto final. Se identifican características de calidad desde el punto de vista del usuario.

MODELO DE CALIDAD DE BOEHM

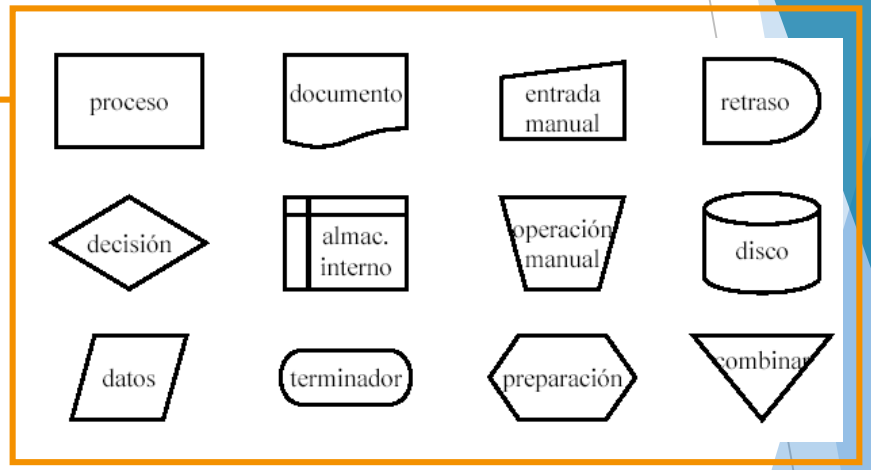


Modelo de Calidad de Dromey



Herramientas de Calidad: Básicas

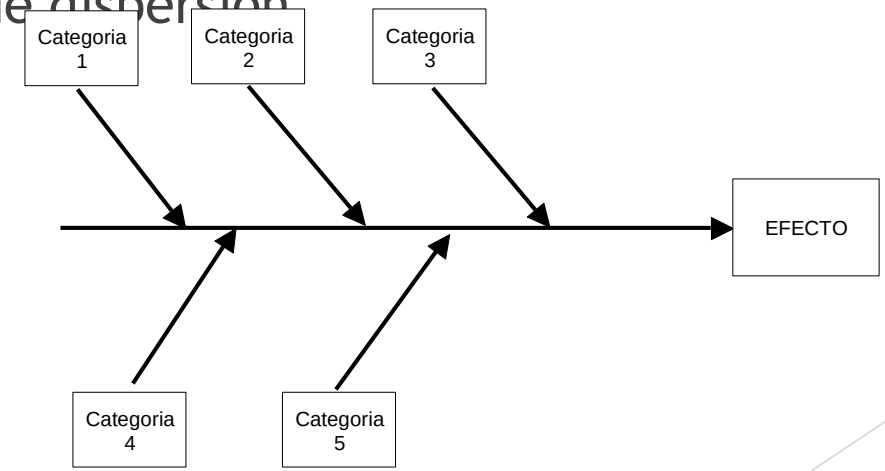
- Diagrama de flujo
- Diagrama causa-efecto
- Diagrama de Pareto
- Hoja de chequeo
- Grafo de control
- Histograma
- Diagrama de dispersión



Herramientas de Calidad: Básicas

- Diagrama de flujo
- Diagrama causa-efecto
- Diagrama de Pareto
- Hoja de chequeo
- Grafo de control
- Histograma
- Diagrama de dispersión

1. **Definir el efecto que se quiere analizar**
2. **Determinar causas/subcausas (5 M: Método, Material, Maquinaria, Mano de obra, Medio ambiente)**
3. **Revisar causas y su interacción**
4. **Seleccionar las causas según su grado de contribución al efecto**



Herramientas de Calidad: Básicas

Nivel de Madurez	Descripción	Herramientas
BAJO	No existe sistema de calidad formal o no se usa. Reclamaciones y costes de fallos son altos. No hay mejora continua normal. Departamento de calidad es responsable	▪Auditorías ▪Coste de calidad ▪Control est. Proceso
MEDIO	Coste de calidad internos altos, los externos bajos. Cada departamento acepta su papel en sistema de gestión de calidad. Proyectos de mejora con empleados	▪H. Creatividad ▪Encuestas clientes ▪FMEA / Dis. Exp. ▪Benchmarking
ALTO	Los sistemas de gestión de calidad, seguridad, finanzas, etc.integrados y dirigidos por la estrategia org. Dptos. y procesos monitorizan desempeño y mejoran diaria.	▪H. de gestión ▪Encuestas a empleados ▪QFD

Tipos de error

- Errores de código
- Errores de procedimiento
- Errores de documentación
- Errores de datos

Causas de errores de software

- Errores en la definición de requisitos
- Fallas de comunicación entre cliente-desarrollador
- Desviaciones deliberadas de los requisitos de software
- Errores de diseño
- Errores de programación
- Incumplimiento de las políticas de documentación y programación
- Dificultades del proceso de pruebas
- Errores de procedimiento
- Errores de documentación
- Etc.

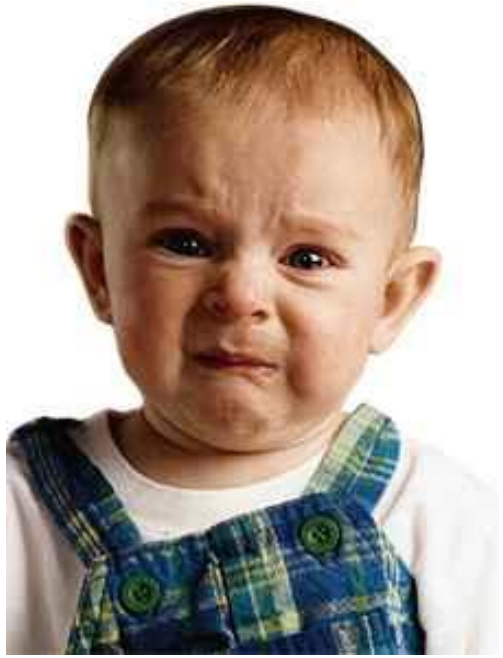
Casos de error famosos

- Therac-25
- Sistema de misiles Patriot
- NASA's Mars Polar Lander
- Sistema de lanzamiento Ariane 5
- Apagón 2003

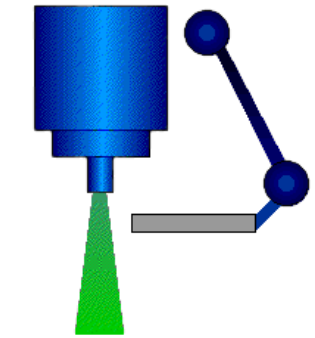
Qué nos obliga a asegurar la Calidad en el SW (y X): mantenimiento de producto

- El software no se termina cuando se entrega al cliente, sino que ¡¡¡NACE!!!

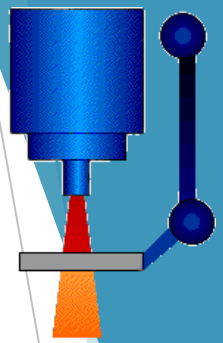
... un bebé un poco problemático



Therac-25



Electron Mode



X-Ray Mode

- Era una máquina empleada en terapia de radiación, producida por Atomic Energy of Canada Limited.
- Causó al menos seis accidentes entre 1985 y 1987, y le costó la vida al menos a cinco personas.
- El problema estaba en la interfaz gráfica que permitía proporcionar dosis de radiaciones mortales a los pacientes.
- El software no detectaba la rotación generada en los electrodos y no notaba que el paciente estaba recibiendo una dosis de raciación letal.

Therac-25 - Razones

- El diseño no incluía ningún bloqueo del hardware para prevenir que se llegara a ese alto nivel de energía sin que estuviera todo en posición.
- Se reutilizó el software de otros modelos que sí tenían el bloqueo anterior y no eran vulnerables a este problema.
- El hardware no proveía al software un modo para verificar que todo estuviera funcionando correctamente.
- La tarea de control del equipo no se sincronizaba con la del operador. Las condiciones del problema se dieron cuando el operador cambiaba la configuración muy rápido. Esto no se daba durante las pruebas porque apenas estaban aprendiendo.

Patriot Missile System

- En Febrero 25, 1991, el misil patriot estuvo en operación por 100 horas. En este momento el reloj interno se había movido un tercio de segundo.
- Para un objetivo que se movía tan rápido el error en posición era de 600 mt.
- El sistema de radar detectó el misil tipo Scud y predijo dónde iba a estar. Por el error de reloj, el sistema buscó el misil y no lo encontró en el momento por lo que asumió que había sido un error de detección.
- Dado que no se interceptó el misil este impacto y murieron 28 militares.

Mars Polar Lander

- La Mars Polar Lander (MPL) era una sonda espacial estadounidense lanzada por un cohete Delta II 7425 el 3 de enero de 1999, que llegó a Marte el 3 de diciembre de 1999 tras un viaje de nueve meses.
- La sonda llegó en buen estado a Marte.
- El 3 de diciembre de 1999, diez minutos antes de aterrizar, se perdió el contacto.
- La causa de la pérdida de comunicación se desconoce.
- Una posible causa es que durante el descenso la apertura de las patas de aterrizaje era tan brusca que podría haber activado los sensores que indicaban que se había tocado suelo, responsables de parar el motor. Así, la sonda habría parado su motor en pleno vuelo, estrellándose fatalmente contra el suelo.

Ariane 5 Rocket

- El Ariane 5 es un cohete de un sólo uso diseñado para colocar satélites en órbita geoestacionaria y para enviar cargas a órbitas bajas.
- En Junio 4 de 1996 fue el vuelo de prueba para el sistema de lanzamiento del Ariane 5.
- El cohete se destruyó 37 segundos después del lanzamiento convirtiendo la falla en uno de los errores de software más costosos de la historia.
- El software reutilizó especificaciones del Ariane 4. Pero la ruta era muy distinta y fuera del rango para el que se diseñó el anterior software.
- Específicamente, el Ariane 5 tenía 5 veces más aceleración y esto causó que los computadores fallaran.
- Las pruebas no se realizaron sobre el las condiciones de vuelo del Ariane 5.
- Por la diferencia de ruta se ocasionó un error de conversión lo que llevó a una cascada de problemas cumlminando en la destrucción del vuelo.

2003 Apagón

Agosto 14 2003

- Se encontró que FirstEnergy no tomó ninguna acción o alertó a otros centros de control hasta que era demasiado tarde.
- Un error de software del sistema de administración de energía de General Electric prevenía que las alarmas se mostraran en el sistema de control.
- Todas las alarmas y eventos se represaron y el servidor principal falló en 30 minutos. El servidor secundario también falló por la misma razón y todas las aplicaciones dejaron de funcionar.

Conclusiones

Construir modelos para:

- ayudar a entender qué estamos haciendo
- proporcionar una base para definir objetivos
- proporcionar una base para la medición

Construir modelos de:

- gente, procesos, productos
- y estudiar sus interrelaciones

Conclusiones

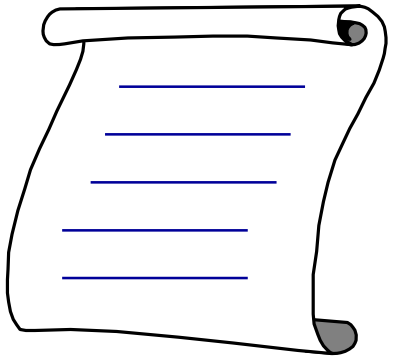
Usar modelos para

- clasificar el proyecto en curso
- distinguir los entornos pertinentes del proyecto
- encontrar tipos de proyectos con características y objetivos similares

Los modelos proporcionan un contexto para:

- Definición de objetivos
- Objetos/experiencias reutilizables
- Selección de procesos
- Evaluación/comparación
- Prediccion

Referencias bibliográficas



Referencias

Albrecht, A.J., "*Measuring application development*", Proc. of IBM Applications DevelopmentJoint SHARE/GUIDE Symposium, Monterey, CA, pp 83-92, 1979.

Arthur, J.D. y Stevens, K.T. , *"Assessing the adequacy of documentation through document quality indicators*", Proceedings of the IEEE Conference of Software Maintenance, pp. 40-49, 1989.

Basili, V.R. y Rombach, H.D., "*The TAME Project: Towards Improvement-Oriented Software Environments"*, IEEE Transaction on Software Engineering,14(6), 758-73 1988.

Basili, V.R. y Weiss, D., "*A Methodology for Collecting Valid Software Engineering data*", IEEE Transaction on Software Engineering, 10 (6), 728-38 1984.

Binder, R., "*Testing Object-Oriented Systems*", American Programmer, 7(4), 22-29, 1994.

Boehm, B.W., Kaspar, J.R. y otros "*Characteristics of Software Quality*", TRW Series of Software Technology, 1978.

Boehm, B.W., Clark, B., Horowitz, E. et al., "*Cost Models for future life cycle processes: COCOMO 2.0*", Annals of Software Engineering 1(1), pp 1-24, 1995.

Brykczynskki, B., "A survey of software inspection checklist", ACM Software Engineering Notes, 24(1), pp 82-89, 1999.

Chidamber, S.R. y Kemerer, C.F., "*A Metrics Suite for Object-Oriented Design"* ,IEEE Transactions of Software Engineering, 20(6), 476-493, 1994.

Churcher, N.I. and Shepperd, M.J., "*Towards Conceptual Framework for Object-Oriented Metrics"*, ACM Software Engineering Notes, 20 (2), 67-76, 1995.

Referencias

DeMarco, T., "*Controlling Software Projects*", Yourdon Press, 1982.

Dolado, J.J. y Fernández, L. (coordinadores). "*Medición para la Gestión en la Ingeniería del Software*". Ra-ma, 2000.

Farbey, B., "*Software Quality metrics: considerations about requirements and requirements specification*", Information and Software Technology, 32 (1), pp 60-64, 1990.

Fenton, N.E. y Pfleeger, S.L., "*Software Metrics. A Rigorous & Practical Approach*" , PWS, 1997.

French, J.C., Knight, J.C. y Powell, A.L., "*Applying hipertext structures to software documentation", Information Processing and Management"*, 33 (2), pp 219-231, 1997.

Genero, M., Manso, M.E., Piattini, M. y García F.J. *"Assessing the quality and the Complexity of OMT Models*" 2nd European Software Measurements Conference-FESMA 99, Amsterdam, Netherlands, pp 99-109, 1999.

Genero, M., Piattini, M. y Calero, C. "*Una propuesta para medir la calidad de los diagramas de clases en UML*", IDEAS´2000, Cancun, México, pp 373-384, 2000.

Gilb, T. "*Principles of Software Engineering Management*", Addison-Wesley, 1988.

ISO/IEC "*Information Technology - Information Resources Dictionary System (IRDS) - Framework*", ISO/IEC intl. Standard edition, 1990.

ISO/IEC 9126, "*Software Product Evaluation – Quality Characteristics and Guidelines for their Use* ", 1991.