TEMA 2
Medición del software

María N. Moreno García
Departamento de Informática y Automática
Universidad de Salamanca

Contenidos

1. Conceptos básicos
2. Medidas y modelos
3. Alcance de las métricas del software
4. Clasificación de las métricas de software
5. Recogida de datos métricos
6. Medición de atributos internos del producto
7. Medición de atributos externos del producto
8. Medición de recursos
9. Métricas para sistemas orientados a objetos

Conceptos básicos (I)

- **Medida**

  Una medida proporciona una indicación cuantitativa de extensión, cantidad, dimensiones, capacidad y tamaño de algunos atributos de un proceso o producto.

- **Métrica**

  Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado (IEEE, 1993).

- **Indicador**

  Métrica o combinación de métricas que proporcionan una visión profunda, del proceso de software, del proyecto de software o del producto en sí (Ragland, 1995).

Conceptos básicos (II)

- **Tipos de indicadores**
  - Los ***indicadores de proceso*** permiten tener una visión más profunda de la eficacia de un proceso ya existente. Se recopilan de todos los proyectos de la organización durante un largo periodo de tiempo con objeto de obtener mejoras de los procesos de software a largo plazo.
  - Los ***indicadores de proyecto*** permiten:
    - Evaluar el estado del proyecto en curso.
    - Seguir la pista de los riesgos potenciales.
    - Detectar áreas de problemas antes de que sean críticas.
    - Ajustar el flujo y las tareas del trabajo.
    - Evaluar la habilidad del equipo del proyecto en controlar la calidad de los productos.
  - Los ***indicadores de producto*** permiten evaluar su calidad.

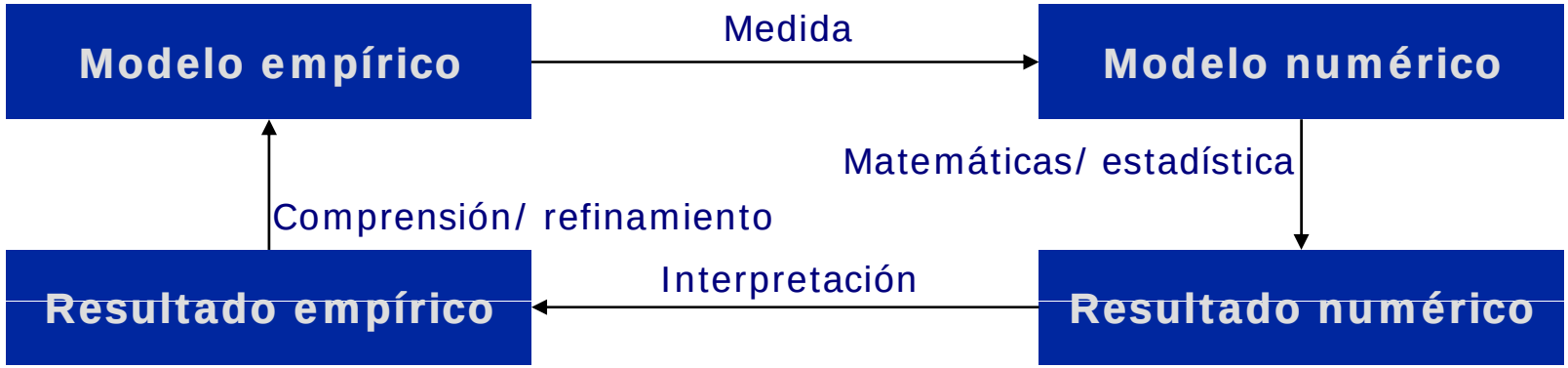
Medidas y modelos (I)

- *Un **modelo** es una abstracción de la realidad que permite observar los detalles de una entidad o concepto desde una perspectiva particular*.

  Los modelos son necesarios para asociar las entidades o atributos del mundo real con los elementos de un sistema numérico. La interpretación de las medidas requiere el uso de modelos:

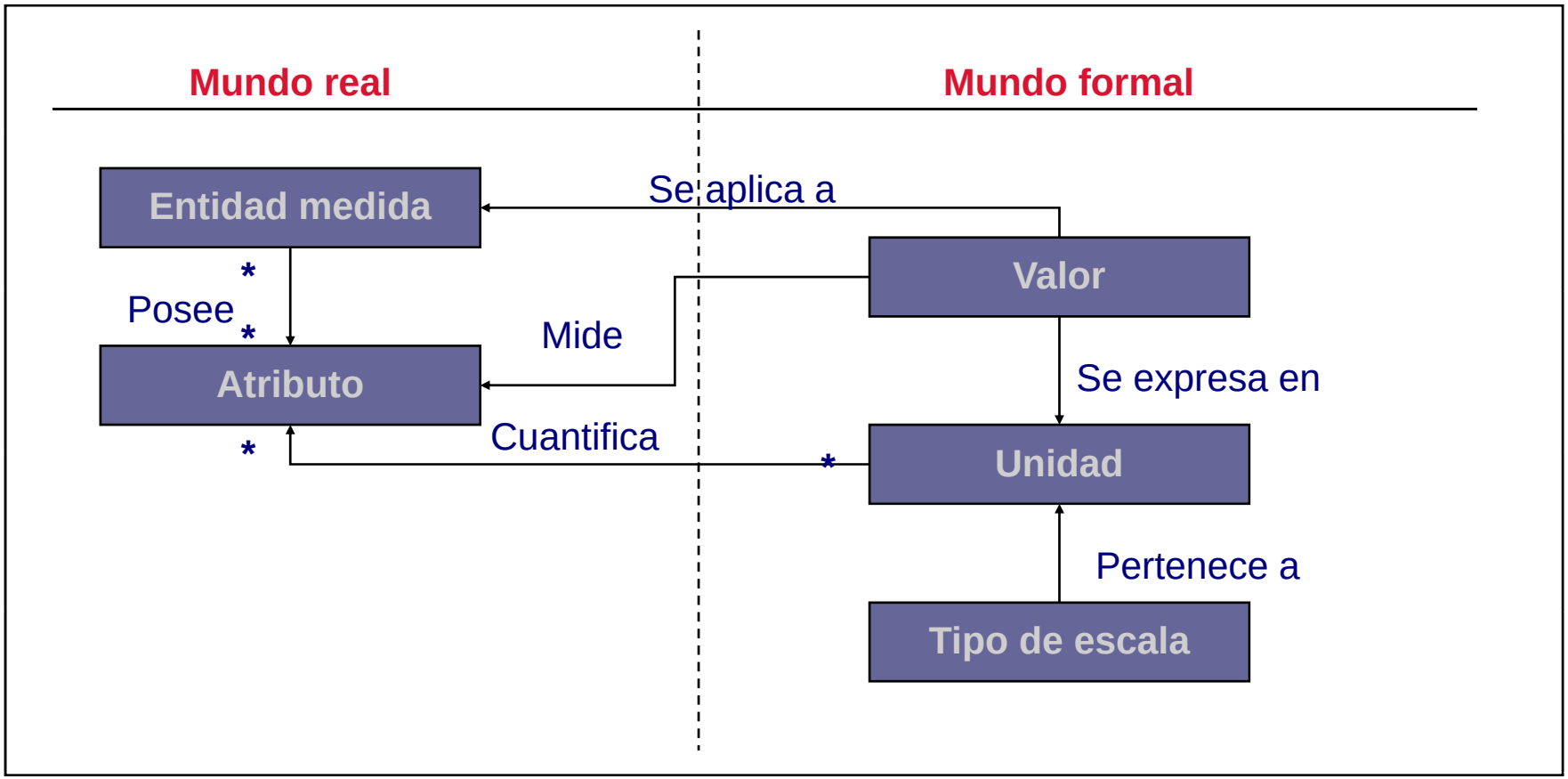
  - **Modelos empíricos.** *Contexto empírico del mundo real*
  - **Modelos numéricos.** *Formalización de las medidas del contexto empírico*

- **Medición**: proceso por el cual se asignan números o símbolos a atributos de entidades del mundo real de tal forma que los describa de acuerdo con reglas claramente definidas [Fenton y Pfleeger, 1997]



Medidas y modelos (II)

- **Elementos implicados en la medición**



Modelo estructural [Kitchenham et al., 1995]

Medidas y modelos (III)

- **Atributo medible**
  - Propiedad o característica distinguible de una entidad software
- **Escala**
  - Conjunto ordenado de valores, continuos o discretos, o conjunto de categorías
  - Define un intervalo de valores posibles que se pueden producir al aplicar un método de medida
- **Tipos de escala**
  - **Nominal**: clasifica las entidades medidas en categorías que no implican un orden
  - **Ordinal**: clasifica las entidades medidas en categorías que implican un orden
  - **De intervalo**: la diferencia entre dos valores consecutivos de la escala es igual
  - **Ratio**:
    - comienza con el valor 0 correspondiente a la ausencia del atributo medido
    - Crece a intervalos iguales (unidades)
    - Distancias iguales corresponden a cantidades iguales del atributo
  - **Absoluta**: la medición se realiza mediante el recuento del número de elementos en una entidad

Medidas y modelos (IV)

- **Medidas directas e indirectas**:
  - Una **medida directa** de una entidad o atributo no involucra ninguna otra entidad o atributo (longitud del código fuente, duración del proceso de prueba, número de defectos...)
  - Una **medida indirecta** se obtiene a partir de medidas directas (productividad, estabilidad de requisitos, densidad de defectos en un módulo...)
- **Objetivos de las medidas**:
  - **Evaluación**: comprobación del cumplimiento de ciertas características por una entidad que ya existe (calidad del diseño, fiabilidad del software...)
  - **Predicción**: estimación de los atributos que tendrá una entidad que no existe aún (coste de un proyecto, esfuerzo necesario). Las medidas para hacer predicciones siempre requieren algún modelo matemático que relacione los atributos que se van a predecir con los que se pueden medir ahora

Medidas y modelos (V)

- **Validación de las medidas**: confirmación de la validez de los atributos, de las unidades, del instrumento y del protocolo de medida
  - **Validación teórica**: confirma si la medida viola alguna de las propiedades que deben cumplir los elementos involucrados en la medición
  - **Validación empírica**: consiste en determinar, mediante experimentos, si los valores obtenidos con una medida encajan con la idea existente sobre el atributo que se mide

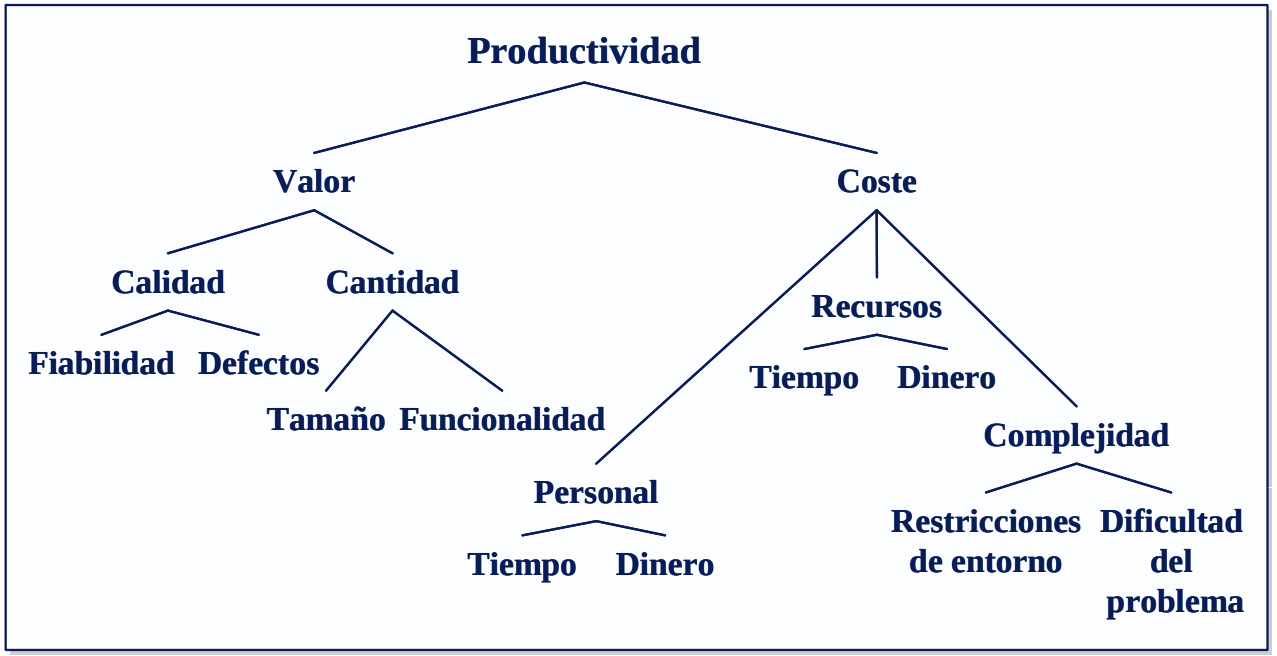
Alcance de las métricas del software (I)

- Las métricas del software abarcan muchas actividades:
  - Estimación de coste y esfuerzo
  - Modelos y medidas de productividad
  - Modelos y medidas de calidad
  - Modelos de fiabilidad
  - Evaluación del rendimiento
  - Métricas estructurales y de complejidad
  - Valoración de capacidad de madurez
  - Gestión mediante métricas
  - Evaluación de métodos y herramientas

Alcance de las métricas del software (II)

- **Estimación de coste y esfuerzo**
  - Las medidas son necesarias para predecir los costes del proyecto al comienzo del ciclo de vida
  - La mayoría de los modelos de estimación (COCOMO, SLIM, Albrecht ...) expresan el coste y el esfuerzo en función de variables que se obtienen por medición del software (tamaño del producto, nivel de reutilización ...)
- **Modelos y medidas de productividad**
  - Permiten valorar la productividad en diferentes procesos de software y en diferentes entornos

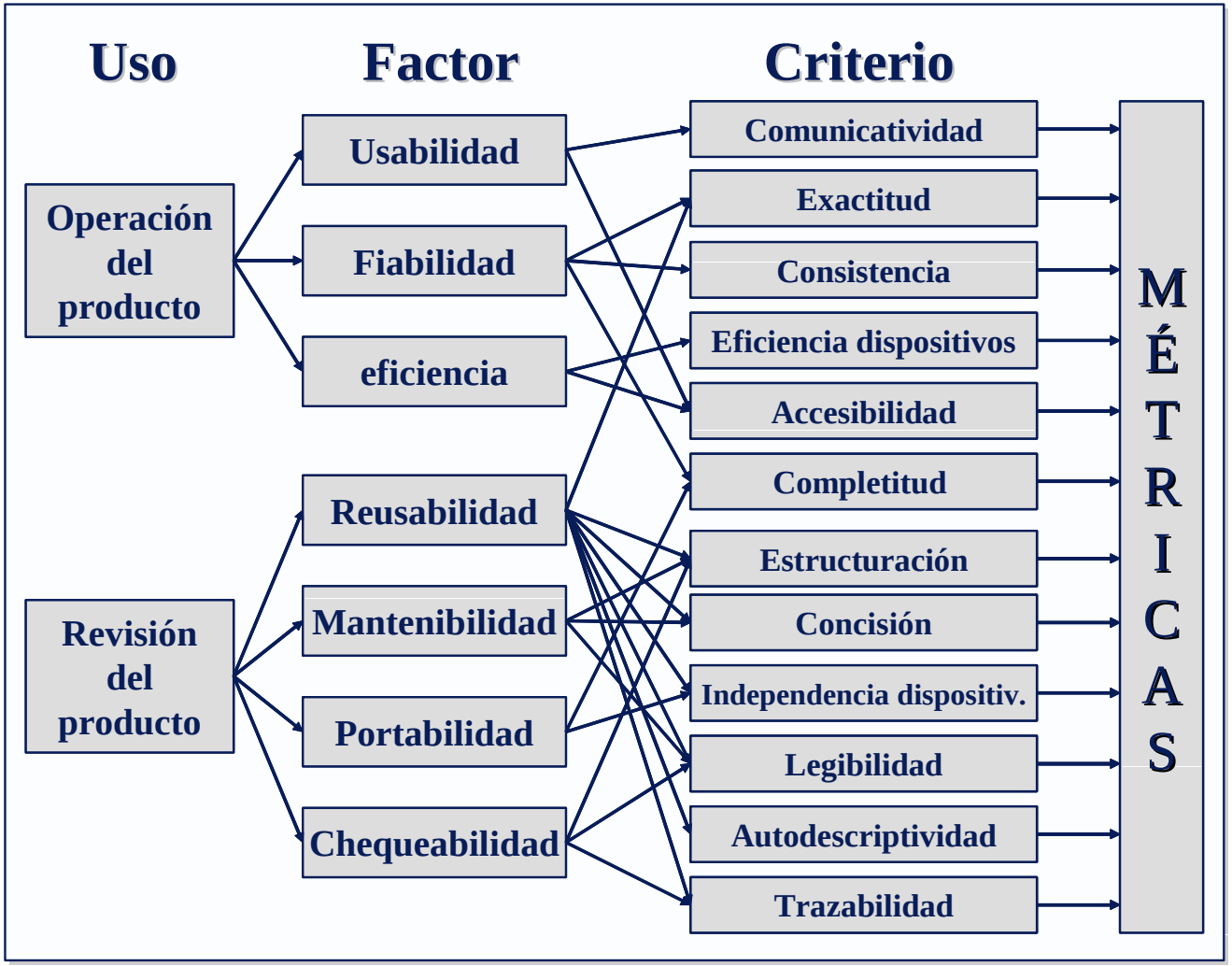
Alcance de las métricas del software (III)



Modelo de productividad

- **Modelos y medidas de calidad**
  - Las medidas de productividad deben ir acompañadas de medidas que permitan valorar la calidad del producto
  - Los modelos de estimación, generalmente se construyen descomponiendo las características de calidad

Alcance de las métricas del software (IV)



Modelo de calidad del software

Alcance de las métricas del software (V)

- **Modelos de fiabilidad**
  - Están incluidos en la mayoría de los modelos de calidad
  - La especialización de los modelos de fiabilidad permite aumentar el entendimiento y control de los productos
- **Evaluación del rendimiento**
  - Aunque es otro aspecto de la calidad, la valoración del rendimiento incluye características observables como tiempos de respuesta y características internas como eficiencia de los algoritmos
- **Métricas estructurales y de complejidad**
  - Para realizar predicciones sobre atributos de calidad (fiabilidad, facilidad de mantenimiento ...) se pueden medir atributos estructurales sobre representaciones del software que están disponibles antes que el código

Alcance de las métricas del software (VI)

- **Gestión mediante métricas**
  - La realización de gráficos basados en diferentes medidas a lo largo del proyecto permite conocer el estado del mismo
- **Evaluación de métodos y herramientas**
  - Las investigaciones cuidadosas, con análisis y mediciones controladas sobre una herramienta o método permiten hacerlos más productivos para situaciones particulares
- **Evaluación de capacidad de madurez**
  - Se pueden medir diferentes atributos del desarrollo para evaluar la capacidad de una organización de desarrollar software de calidad

Clasificación de las métricas del software (I)

- El primer paso de la medición es identificar los atributos o entidades a medir. Estos pueden ser de tres tipos:
  - ***Procesos***: atributos de actividades relacionadas con el software.
  - ***Productos***: componentes, entregas o documentos resultantes de una actividad de proceso.
  - ***Recursos***: entidades requeridas por una actividad de proceso
- Dentro de cada clase anterior se puede distinguir:
  - ***Atributos internos***: Son aquellos que pueden ser medidos examinando el proceso, producto o recurso mismo.
  - ***Atributos externos***: se miden con respecto a como el proceso, producto o recurso se relaciona con su entorno.

Clasificación de las métricas del software (II)

ENTIDADES	ATRIBUTOS	
	Internos	**Externos**
Productos Especificaciones, diseño, código...	Tamaño, reusabilidad, modularidad, funcionalidad, acoplamiento, complejidad...	Comprensión, mantenibilidad, calidad, fiabilidad ...
Procesos Realización de la especificación, del diseño, del código...	Tiempo, esfuerzo, cambios en requisitos, fallos en la especificación	Calidad, coste, estabilidad
Recursos Personal, equipos, hardware, software...	Edad, precio, tamaño del equipo, velocidad, tamaño de memoria	Productividad, experiencia, calidad, usabilidad, fiabilidad

Clasificación de las métricas del software (III) Procesos

- Los aspectos relacionados con el proceso de desarrollo de software que pueden medirse son:
  - ***Atributos internos***:
    - Duración de un proceso o de una de sus actividades
    - Esfuerzo asociado con el proceso o con una de sus actividades
    - Número de incidentes de un tipo determinado que ocurren durante el proceso o una de sus actividades
    - ...
  - ***Atributos externos***:
    - Controlabilidad
    - Observabilidad
    - Estabilidad
    - ...

Clasificación de las métricas del software (V)
Productos

- Las características medibles del producto son:
  - ***Atributos externos***:
    - Usabilidad
    - Integridad
    - Eficiencia
    - testeabilidad
    - Reusabilidad
    - Portabilidad
    - ...
  - ***Atributos internos***:
    - Tamaño del producto
    - Longitud de las especificaciones
    - Modularidad del diseño
    - Acoplamiento y cohesión
    - Complejidad del código
    - ...

El uso principal de los atributos internos es la predicción de los atributos externos

Midiendo y controlando algunos atributos internos del producto se puede controlar su calidad (fiabilidad, mantenibilidad, usabilidad...)

Clasificación de las métricas del software (I)
Recursos

- Los recursos incluyen cualquier entrada en la producción de software
  - Personal
  - Materiales
  - Herramientas
  - Métodos …
- Las medidas de recursos ayudan a controlar el proceso indicando cómo el proceso está usando y transformando las entradas en salidas
- ***Atributos internos***
  - Tamaño del equipo
  - Tiempo de experiencia
  - Madurez de las herramientas ...
- ***Atributos externos***
  - Coste
  - Productividad …

productividad = cantidad de salida / esfuerzo de entrada

La productividad combina una medida de proceso (entrada) con una medida de producto (salida)

Recogida de datos métricos (I)

- El ***proceso de medición*** incluye las actividades siguientes [McGarry et al., 2002]:
  - **Planificación de las mediciones**: proporciona un método consistente para identificar las necesidades de información del proyecto, seleccionar y especificar las medidas e integrarlas en los procesos técnicos y de gestión del proyecto
  - **Realización de las mediciones**: implica la recolección de datos de medida, el análisis de los datos y la presentación de resultados
  - **Evaluación de las mediciones**: el proceso de medición y las medidas específicas se evalúan periódicamente y se mejoran si es necesario
  - **Establecimiento y mantenimiento de un compromiso**: consiste en establecer los recursos, formación y herramientas para implementar un programa de medida efectivo, así como asegurar que existe el compromiso de usar la información producida

Recogida de datos métricos (II)

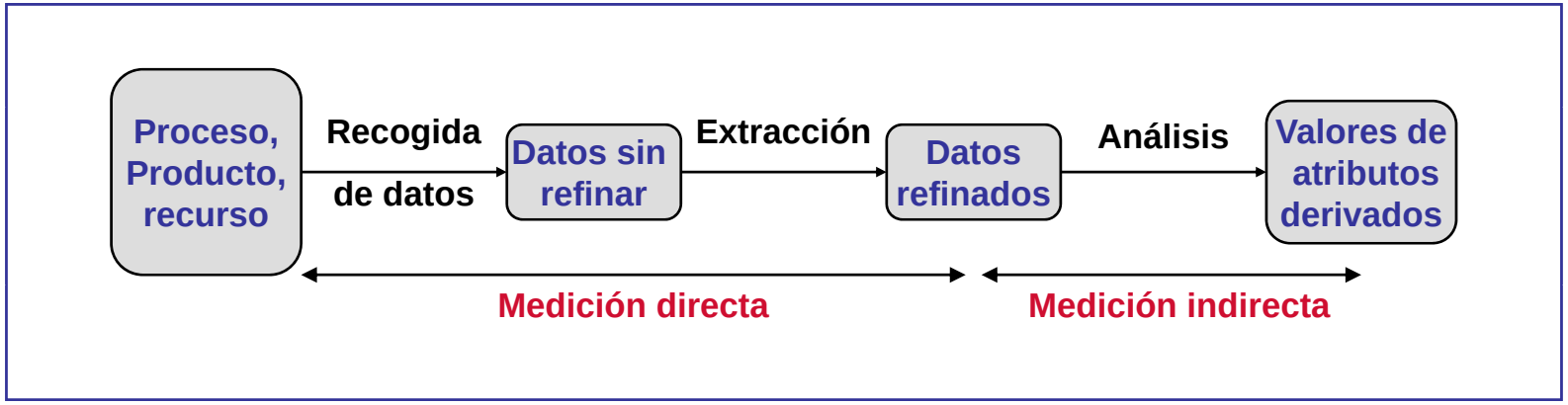
- **Actividades de recogida y procesamiento de datos**:
  - Capturar y visualizar los datos de forma apropiada
  - Asegurar la calidad de los datos
  - Almacenar y gestionar los datos para su análisis
- **Propiedades de los datos**:
  - **Correctos**: la recogida debe hacerse de acuerdo a las reglas exactas de la definición o de la métrica
  - **Exactos**: la diferencia entre el valor resultante de la medida y el valor real del dato debe ser lo mínima posible
  - **Precisos**: el número de cifras utilizadas para expresarlos debe ser la apropiada
  - **Consistentes**: evaluaciones diferentes sobre los mismos datos deben dar los mismos resultados.
  - **Replicables**: deben servir para comparar datos obtenidos en circunstancias diferentes.
  - **Asociados con una actividad o periodo de tiempo** particular.

Recogida de datos métricos (III)

- **Definición de los datos**

  Hay dos tipos de datos:

  - ***Datos sin refinar*** resultantes de la medición inicial
  - ***Datos refinados*** obtenidos extrayendo los elementos de datos relevantes de los datos sin refinar

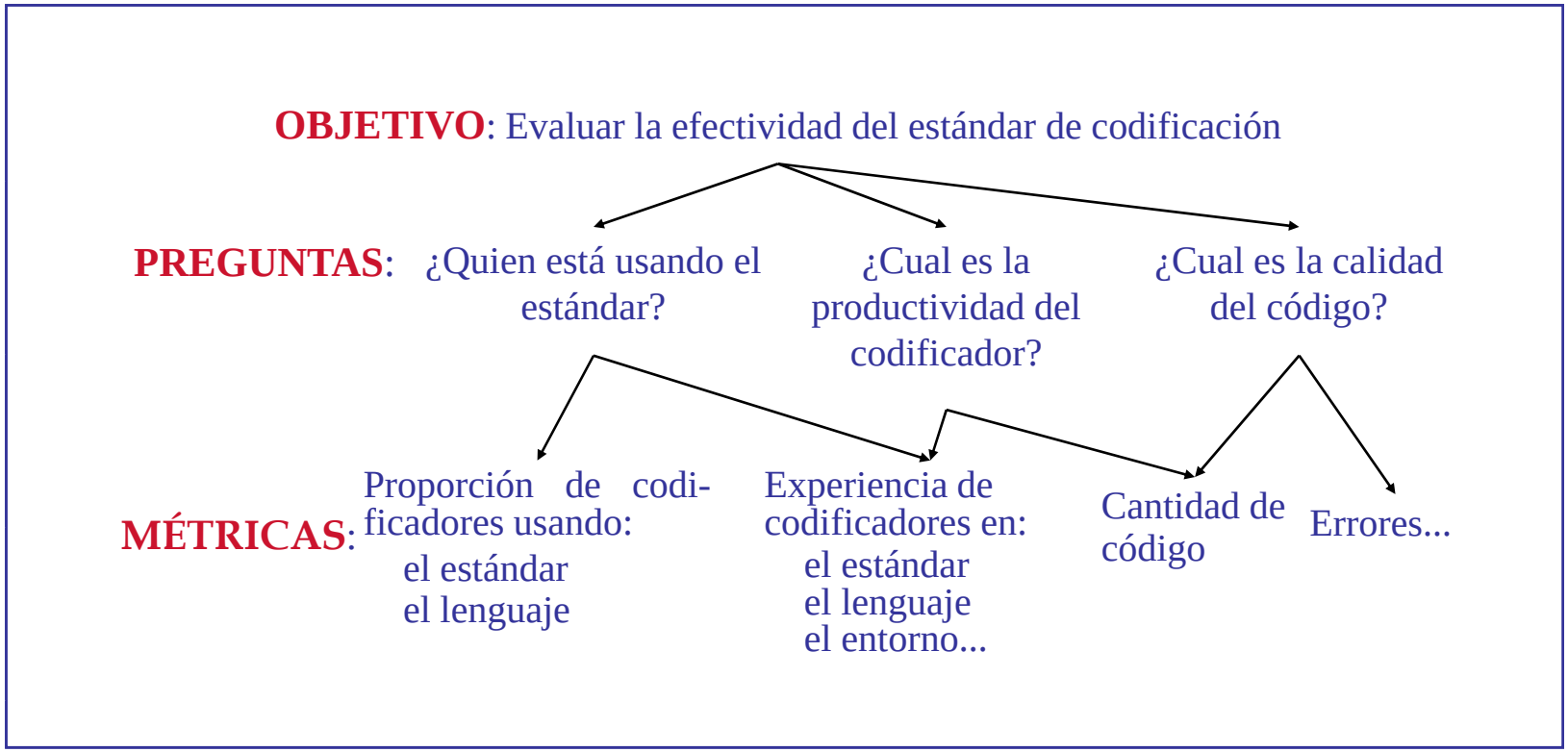


Recolección de datos para la medición

Recogida de datos métricos (IV)

- **Modelo de medición**
  - El primer paso en la aplicación de las métricas es decidir qué medir. Los objetivos para procesos, proyectos y productos deben estar bien definidos.
  - El **enfoque GQM** (*Goal-Question-Metric*) (Basili y Weiss, 1984) (Basili y Rombach, 1988) puede utilizarse para seleccionar e implementar métricas de una manera efectiva.
  - El enfoque GQM se aplica en tres pasos:
    - Lista de los objetivos principales del desarrollo y mantenimiento del proyecto.
    - Para cada objetivo obtener las preguntas que deben contestarse para saber si se están cumpliendo los objetivos.
    - Decidir qué medir para poder contestar las preguntas de forma adecuada.

Recogida de datos métricos (V)



Ejemplo de métricas derivadas de objetivos y preguntas (GQM)

Medición de atributos internos del producto (I)

Los atributos internos describen los productos de software de forma que dependen únicamente del producto mismo

- El producto puede ser descrito en función de:
  - Su ***tamaño***
  - Sus ***propiedades estructurales***
- Atributos usados para medir el **tamaño** del software:
  - ***Longitud***: tamaño físico del producto.
  - ***Funcionalidad***: funciones que proporciona el producto al usuario.
  - ***Complejidad***:
    - Del problema
    - Algorítmica
    - Estructural
    - Cognitiva

Medición de atributos internos del producto (II)

- Las ***propiedades estructurales*** del software son atributos internos relacionados con la calidad del producto. Los tipos de medidas estructurales son:
  - ***Flujo de control***: secuencia en que se ejecutan las instrucciones.
  - ***Flujo de datos***: seguimiento de cómo los datos se crean y se manejan por un programa.
  - ***Estructura de los datos***: organización de los datos independiente del programa.

Medición de atributos internos del producto (III)
Longitud (I)

Los principales productos que es útil medir son la especificación, el diseño y el código.

Código (I)

- El **numero de líneas de código** (LOC) es la medida más usada para medir la longitud del código fuente. Se han realizado muchas propuestas para su contabilización:
  - **Líneas de código efectivas** (ELOC: *effective lines of code*) o **no comentadas** (NCLOC)
    - Propuesta de HP
    - Es La más extendida
    - No contabiliza las líneas comentadas ni en blanco

Medición de atributos internos del producto (IV)
Longitud (II)

Código (II)

- Es útil medir por separado las **líneas comentadas** (CLOC) de las no comentadas para calcular esfuerzo, productividad, etc. La longitud total será:

$$LOC = NCLOC + CLOC$$

- También puede se útil calcular la **densidad de comentarios**:

$$CLOC/LOC$$

- Para propósitos tales como la prueba es importante conocer cuanto código ejecutable se produce, para ello se mide el **número de sentencias ejecutables** (ES), ignorando los comentarios, declaraciones de datos y cabeceras
- Otra propuesta consiste en contabilizar únicamente el **código entregado al cliente**. Se cuenta el número de DSI (*delivered source instruction*) que incluye las declaraciones de datos, las cabeceras y las instrucciones fuente

Medición de atributos internos del producto (V)
Longitud (III)

Código (III)

- **Métricas de Halstead** [Halstead, 1977] **(I)**
  - Considera un programa P como un conjunto de ***tokens*** (*token*: unidad sintáctica más elemental distinguible por un compilador) que se pueden clasificar como operandos y operadores
  - Las métricas básicas para los *tokens* son:
    - n_1: número de operadores únicos
    - n_2: número de operandos únicos
    - N_1: número total de ocurrencias de operadores
    - N_2: número total de ocurrencias de operandos

Medición de atributos internos del producto (VI)
Longitud (IV)

Código (IV)

- **Métricas de Halstead (II)**
  - Métricas compuestas para un programa:
    - **Longitud estimada**:

      $L = n_1 \log_2 n_1 + n_2 \log_2 n_2$

    - **Volumen**:

      $V = N \log_2 n$

      donde $N = N_1 + N_2$ y $n = n_1 + n_2$

    - **Nivel**:

      $L = V^*/V$

      donde V^* = volumen del tamaño mínimo

    - **Esfuerzo de implementación**:

      $E = (n_1 N_2 N \log_2 n) / (2 n_2)$

Medición de atributos internos del producto (VII)
Longitud (V)

Especificaciones y diseño

- Los documentos de especificación de requisitos y de diseño tienen representaciones de muchos tipos (texto, gráficos, símbolos...)
- La medición del atributo longitud exige la identificación de elementos atómicos que puedan contarse. Ejemplo:
  - Diagramas de flujo de datos: procesos, entidades externas, almacenes de datos y flujo de datos
  - Especificaciones algebraicas: salidas, funciones, operaciones y axiomas
  - Esquemas Z: líneas de especificación (declaraciones y predicados)
- Se pueden definir páginas de documentación como objetos atómicos

Vista	Diagrama	Objetos atómicos
Funcional	Diagrama de flujo de datos Diccionario de datos	Burbujas Elementos de datos
Datos	Diagrama entidad relación	Objetos, relaciones
Estado	Diagrama de transición de estados	Estados, transiciones

Componentes atómicos del análisis estructurado

Medición de atributos internos del producto (VIII) Funcionalidad (I)

- Para realizar estimaciones de esfuerzo y duración es mejor estimar el tamaño del producto en base a su funcionalidad
- La mayoría de los enfoques miden la funcionalidad de los documentos de especificación

Puntos de función (I)

- El análisis de los puntos de función (FPA) es una medida de la funcionalidad propuesta en el método de estimación del esfuerzo de Albrecht [Albrecht, 1979]
- El paso previo al cálculo de los puntos de función, **FP**, es el cálculo de **UFC** (*unadjusted function point count*):
  - Se determinan los siguientes elementos de alguna representación del software:
    - ***Entradas externas***: entradas de usuario que proporcionan datos a la aplicación
    - ***Salidas externas***: Salidas que proporcionan información al usuario
    - ***Consultas externas***: peticiones interactivas que requieren una respuesta
    - ***Ficheros externos***: interfaces con otros sistemas legibles por la máquina
    - ***Ficheros internos***: ficheros maestros lógicos del sistema

Medición de atributos internos del producto (IX) Funcionalidad (II)

Puntos de función (II)

- A cada item se le asigna un índice de complejidad (simple, medio o complejo) y un factor de peso en función del índice, cuyos valores se recogen en la tabla siguiente

	Factor de peso		
Item	**Simple**	**Medio**	**Complejo**
Entradas externas	3	4	6
Salidas externas	4	5	7
Consultas externas	3	4	6
Ficheros externos	7	10	15
Ficheros internos	5	7	10

Medición de atributos internos del producto (X) Funcionalidad (III)

Puntos de función (III)

- UFC se calcula mediante la siguiente fórmula:

$$UFC = \sum_{i=1}^{15} ((\text{número de items de la clase i}) * \text{peso}_i)$$

- Para completar el cálculo de los PF es necesario conocer el factor de complejidad técnica (**TCF**) que engloba los 14 factores que aparecen en la tabla siguiente

Componentes del factor de complejidad técnica		
F_1 Copias de seguridad y recuperación fiables	F_6 Entrada interactiva de datos	F_{11} Reusabilidad
F_2 Comunicación de datos	F_7 Facilidad operativa	F_{12} Facilidad de instalación
F_3 Funciones distribuidas	F_8 Actualización interactiva	F_{13} Múltiples sitios
F_4 Rendimiento	F_9 Interfaces complejas	F_{14} Facilidad de cambios
F_5 Configuración muy usada	F_{10} Procesamiento complejo	

Medición de atributos internos del producto (XI)
Funcionalidad (IV)
Puntos de función (IV)

- Cada componente de la tabla anterior se sitúa en una escala entre 0 y 5 según su influencia:
  - Ninguna influencia 0
  - Incidental 1
  - Moderado 2
  - Medio 3
  - Significativo 4
  - Esencial 5
- La siguiente fórmula combina los 14 factores:

$$TCF = 0.65 + 0.01 \sum_{i=1}^{14} F_i$$

- Los valores constantes de la ecuación y los factores de ponderación se obtienen empíricamente
- Cálculo final de los puntos de función:

$$FP = UFC * TCF$$

- La técnica de puntos de función presenta muchos problemas debido fundamentalmente a la subjetividad en la aplicación de los factores y a la inexactitud de las medidas

Medición de atributos internos del producto (XII) Funcionalidad (V)

- **Modificaciones y alternativas a FPA**

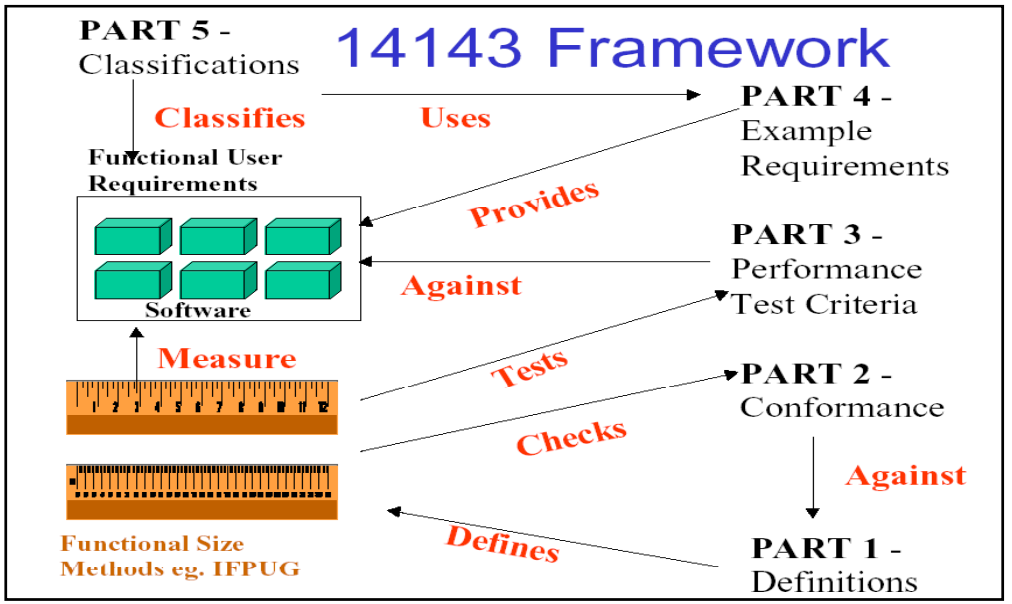
  Críticas al método de Albrecht: semántica confusa, método de contabilización complejo, asignación arbitraria de pesos, sólo considera funciones del usuario final, sin tener en cuenta el esfuerzo de implementación de funciones internas...

  - Puntos de función **Mark II** (MkII FPA) [Symons, 1991]
  - **Medición del tamaño funcional** [ISO/IEC 14143, 2003]
  - **IFPUG** (*The International Function Point Users' Group*) [ISO/IEC 20926:2003]: estándar de uso de FPA
  - **NESMA** [ISO/IEC 24570, 2003]
  - **COSMIC-FFP** [ISO/IEC 19761:2003]
  - **UCP** (***Use Case Points***) [Karner, 1993]

Medición de atributos internos del producto (XIII)
Funcionalidad (VI)

Medición del tamaño funcional

- Conjunto de estándares [ISO/IEC 14143 parts 1 – 5] creados como respuesta a la gran proliferación de propuestas de FP
- **Definiciones** (parte 1):
  - **Tamaño funcional**: tamaño del software derivado de la cuantificación de los requisitos funcionales de usuario



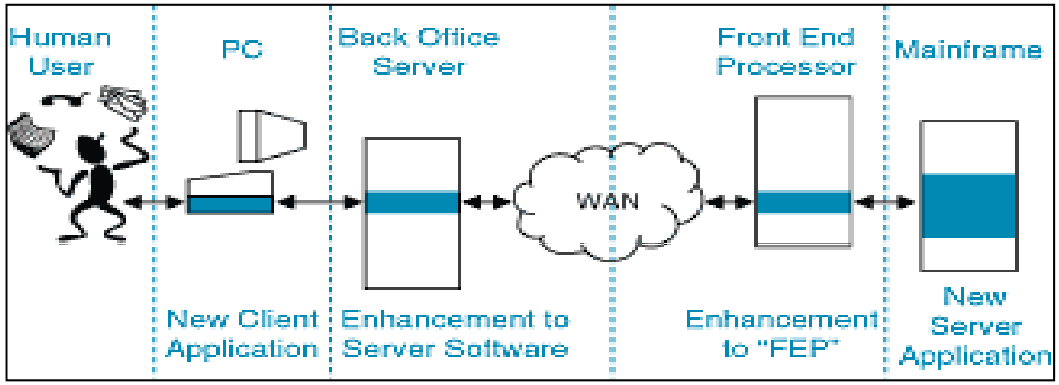
  - **Medición del tamaño funcional**: proceso de medir el tamaño funcional
  - **Índices de productividad**: tamaño del producto software entregado / esfuerzo
  - **Efectividad del coste**: tamaño del producto software entregado / coste
  - **Calidad del producto**:

    defectos entregados / tamaño del producto software entregado

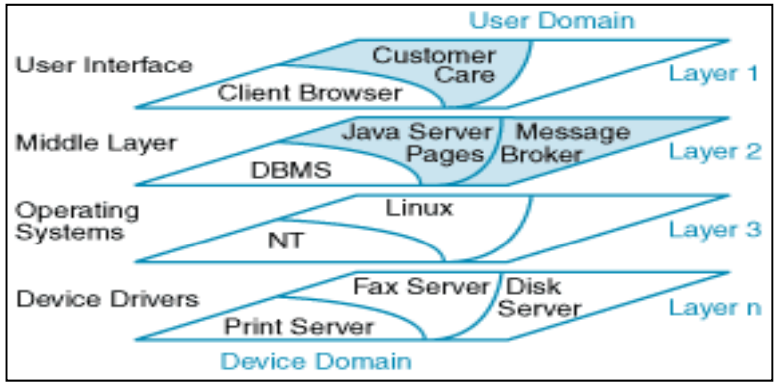
Medición de atributos internos del producto (XIV)
Funcionalidad (VII)

COSMIC-FFP [ISO/IEC 19761:2003]

- Propuesta del **CO**mmon **S**oftware **M**easurement **I**nternational **C**onsortium
- Para sistemas cliente servidor y aplicaciones multicapa
- Compatible con técnicas actuales de especificación de requisistos (casos de uso y prototipado)
- Aplicable en proyectos que usan COTS



Ejemplo de un sistema cliente-servidor actual



Capas de la arquitectura software

Medición de atributos internos del producto (XV) Funcionalidad (VIII)

UCP (*Use Case Points*) (I)

- Métrica similar al análisis de puntos de función pero basada en la funcionalidad representada en forma de casos de uso
- Los UCP se utilizan para estimar el esfuerzo de desarrollo
- Considera actores, escenarios y factores técnicos y de entorno
- Requiere el cálculo de tres variables:
  - **UUCP** (*Unadjusted Use Case Points*)
    - **UUCW** (*Unadjusted Use Case Weight):* considera el número y complejidad de los casos de uso
    - **UAW** (*Unadjusted Actor Weight):* considera el número y complejidad de los actores
  - **TCF** (*Technical Complexity Factor*)
  - **ECF** (*Environment Complexity Factor*)
- Los UCP se calculan mediante la expresión:

$$\mathbf{UCP = UUCP * TCF * ECF}$$

Medición de atributos internos del producto (XVI) Funcionalidad (IX)

UCP (*Use Case Points*) (II)

- Se definen trece **factores de complejidad técnica** (TCF) y ocho **factores de complejidad del entorno** (ECF)
- A cada factor se le asigna un **peso** (W) acorde a su impacto y **una complejidad percibida** (F) correspondiente a la percepción de complejidad que tiene el equipo de desarrollo
- Cálculo de TCF:

$$TCF = C_1 + C_2 \sum_{i=1}^{13} W_i F_i$$

- Cálculo de ECF:

$$ECF = C_1 + C_2 \sum_{i=1}^{8} W_i F_i$$

Medición de atributos internos del producto (XVII) Funcionalidad (X)

UCP (*Use Case Points*) (III)

Factores de complejidad técnica		
T_1 Sistemas distribuidos	T_6 Facilidad de instalación	T_{11} Características especiales de seguridad
T_2 Rendimiento	T_7 Facilidad de uso	
T_3 Eficiencia del usuario final	T_8 Portabilidad	T_{12} Acceso directo a terceras partes
T_4 Procesamiento interno complejo	T_9 Facilidad de cambio	T_{13} Se requiere entrenamiento especial del usuario
T_5 Reusabilidad	T_{10} Concurrencia	

Factores de complejidad del entorno	
E_1 Familiaridad con UML	E_5 Experiencia en orientación a objetos
E_2 Trabajadores a tiempo parcial	E_6 Motivación
E_3 Capacidad de los analistas	E_7 Dificultad del lenguaje de programación
E_4 Experiencia en la aplicación	E_8 Estabilidad de los requisitos

Medición de atributos internos del producto (XVIII) Funcionalidad (XI)

Enfoque COCOMO

- COCOMO es un modelo para predecir el esfuerzo en función del tamaño del sistema
- Utiliza una medida del tamaño que puede ser aplicada al comienzo del desarrollo: los ***puntos objeto***
- El cálculo de los puntos objeto se realiza contabilizando el número de pantallas, informes y componentes de 3GL de la aplicación. A cada objeto se le asigna un factor de peso según su grado de dificultad

Tipo de objeto	Simple	Medio	Difícil
Pantalla	1	2	3
Informe	2	5	8
Componente 3GL	-	-	10

- Los pesos reflejan el esfuerzo relativo requerido para implementar un objeto de un determinado nivel
- Si existe reutilización los objetos ponderados se suman y se multiplican por un factor de reutilización para dar un nuevo número de puntos objeto:

$$\text{Puntos objeto nuevos} = (\text{puntos objeto}) * (100 - r) / 100$$

Medición de atributos internos del producto (XIX) Funcionalidad (XII)

Métricas *bang* (I)

- Medida de la funcionalidad propuesta por DeMarco [DeMarco, 1978, 1982]
- Para calcular la métrica se deben evaluar un conjunto de primitivas:
  - Primitivas funcionales (PFu)
  - Elementos de datos (ED)
  - Objetos (OB)
  - Relaciones (RE)
  - Estados (ES)
  - Transiciones (TR)
- Se determinan cuentas adicionales para:
  - Primitivas modificadas de la función manual (PMFu)
  - Elementos de datos de entrada (EDE)
  - Elementos de datos de salida (EDS)
  - Elementos de datos retenidos (EDR)
  - Muestras (*tokens*)de datos (TC_i)
  - Conexiones de relación (RE_i)

Medición de atributos internos del producto (XX) Funcionalidad (XIII)

Métricas *bang* (II)

- Se calculan para dos dominios:
  - ***Dominio de la función***: medida del número de primitivas funcionales de los diagramas de flujo
  - ***Dominio de los datos***: medida basada en el número de entidades y relaciones del diagrama entidad-relación

RE/PFu < 0,7	aplicación de dominio de función
0,8 < RE/PFu < 1,4	aplicación híbrida
RE/PFu > 1,5	aplicación de dominio de datos

- Las métricas se calculan a partir de incrementos PFu y OB corregidos (IPFuC e IOBC):

TC_i	IPFuC
2	1,0
5	5,8
10	16,6
15	29,3
20	43,2

RE_i	IOBC
1	1,0
3	4,0
6	9,8

- Las métricas *bang* pueden definirse formalmente y calcularse automáticamente mediante herramientas CASE

Medición de atributos internos del producto (XXI) Complejidad (I)

- ***Complejidad de un problema***: cantidad de recursos que se requieren para una óptima solución del problema
- ***Complejidad de la solución***: recursos necesarios para implementar una solución particular:
  - ***Complejidad de tiempo***: el recurso es tiempo de ordenador
  - ***Complejidad de espacio***: el recurso es memoria de ordenador
- La complejidad de la solución se puede medir en términos de ***eficiencia*** de esa solución, como puede ser la eficiencia algorítmica (función del número de operaciones primitivas que se requieren para una entrada dada):
  - Las entradas se pueden caracterizar mediante un parámetro de tamaño, ***n***
  - Si ***F*** es el conjunto de todas las funciones de ***n***, se puede establecer una relación “>” que se corresponda con la relación empírica “más eficiente”
  - La notación “**big-O**” permite definir una relación de orden sobre funciones ***f(n)*** encontrando el término dominante e ignorando las constantes que multiplican: ***O(g)*** es el conjunto de funciones que dominan asintóticamente la función g
  - La eficiencia de un algoritmo A es O(f(n)) si para una entrada de tamaño n se requieren O(f(n)) operaciones en el peor caso

Medición de atributos internos del producto (XXII) Complejidad (II)

Big-O de f(n)	Tipo de complejidad
O(1)	Complejidad constante
O(log n)	Complejidad logarítmica
O(n)	Complejidad lineal
$O(n^2)$	Complejidad cuadrática
$O(n^i)$	Complejidad polinomial
$O(c^n)$, n>1	Complejidad exponencial

- La relación empírica “más eficiente” se representa mediante la relación de inclusión:

$$O(n) \subset O(n^2)$$

- La complejidad del problema se puede medir en función de la complejidad de sus soluciones utilizando la notación “big-O”:
  - Un problema que tiene una solución de complejidad polinomial se dice que es ***factible*** o ***viable***. La clase de todos los problemas factibles se llama **P**. Cualquier solución factible tiende a P.
  - Hay problemas que no tienen una viabilidad conocida, pero existen métodos (algoritmos con límite polinomial) que se pueden aplicar para comprobar una solución propuesta. Esta clase de problemas se llaman **NP**.
  - Los problemas NP tienen un subconjunto de problemas más complejos de categoría **NP-total**.

Medición de atributos internos del producto (XXIII)
Medidas estructurales (I)

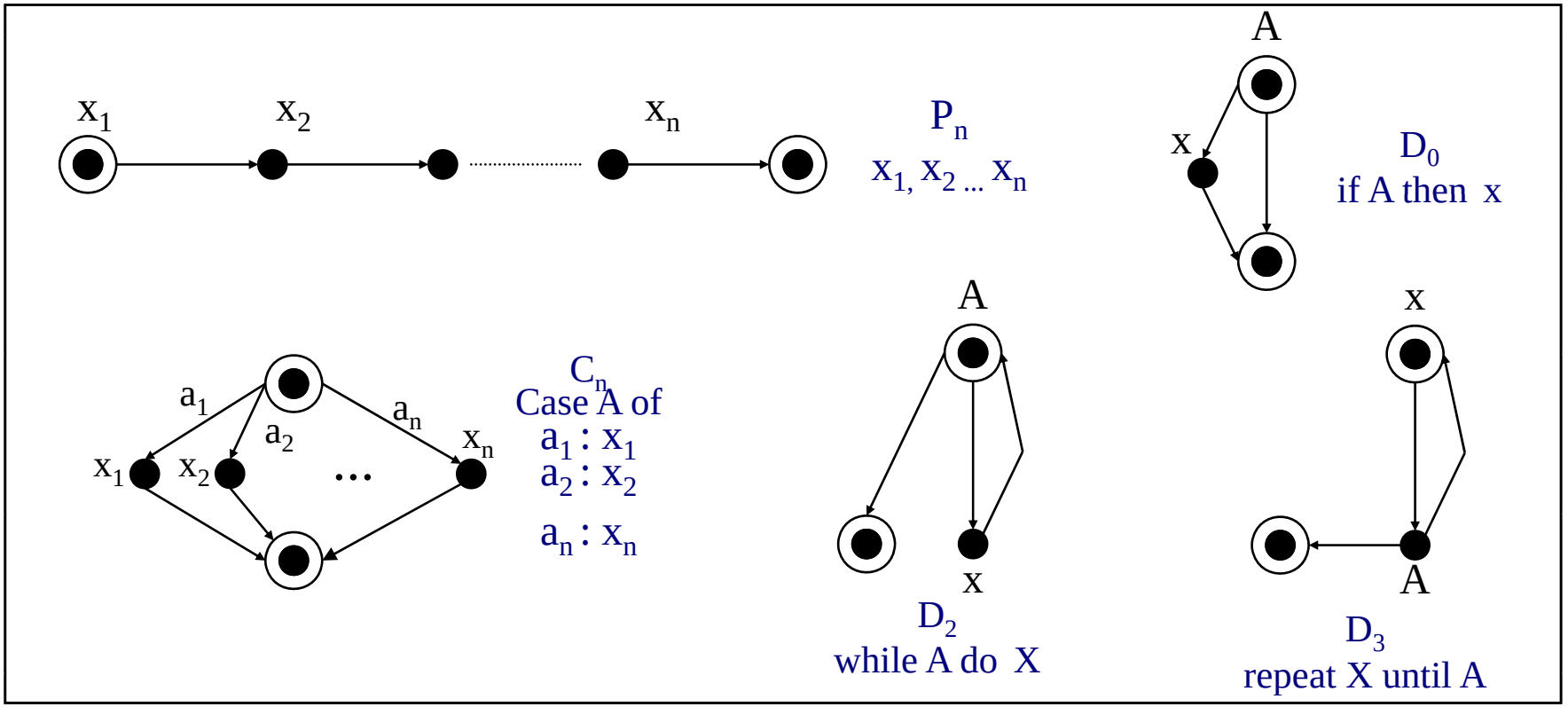
Flujo de control. Grafos de flujo de control (I)

- Las medidas de flujo de control generalmente se modelan mediante grafos dirigidos denominados ***grafos de flujo*** o ***grafos de flujo de control***
- Un grafo dirigido está formado por un conjunto de puntos (nodos) y líneas (arcos). Cada arco tiene asignada una dirección representada por una flecha.
- Un arco se puede describir como un conjunto ordenado de pares, <*x*,*y*>, donde *x* e *y* son los nodos conectados por el arco.
- En un grafo de flujo se pueden definir los siguientes conceptos:
  - ***Grado de entrada a un nodo***: número de arcos que llegan al nodo.
  - ***Grado de salida***: número de arcos que salen del nodo.
  - ***Camino***: secuencia consecutiva de arcos dirigidos, algunos de los cuales pueden atravesarse más de una vez.
  - ***Camino simple***: camino que no tiene arcos repetidos.
  - ***Nodos procedimiento***: nodos cuyo grado de salida es igual a uno.
  - ***Nodos predicado***: nodos cuyo grado de salida esa mayor que uno.

Medición de atributos internos del producto (XXIV)
Medidas estructurales (II)

Flujo de control. Grafos de flujo de control (II)

- Los nodos principio y fin del grafo se representan rodeados por una circunferencia.



Grafos de flujo para diferentes estructuras de programa

Medición de atributos internos del producto (XXV)
Medidas estructurales (III)

Flujo de control. Grafos de flujo de control (III)

- Existen dos operaciones que se pueden usar para construir grafos de flujo a partir de otros: ***concatenación*** y ***anidamiento***
  - Si A y A' son dos bloques de código, la **concatenación** de los grafos de flujo F(A) y F(A') se representa como:

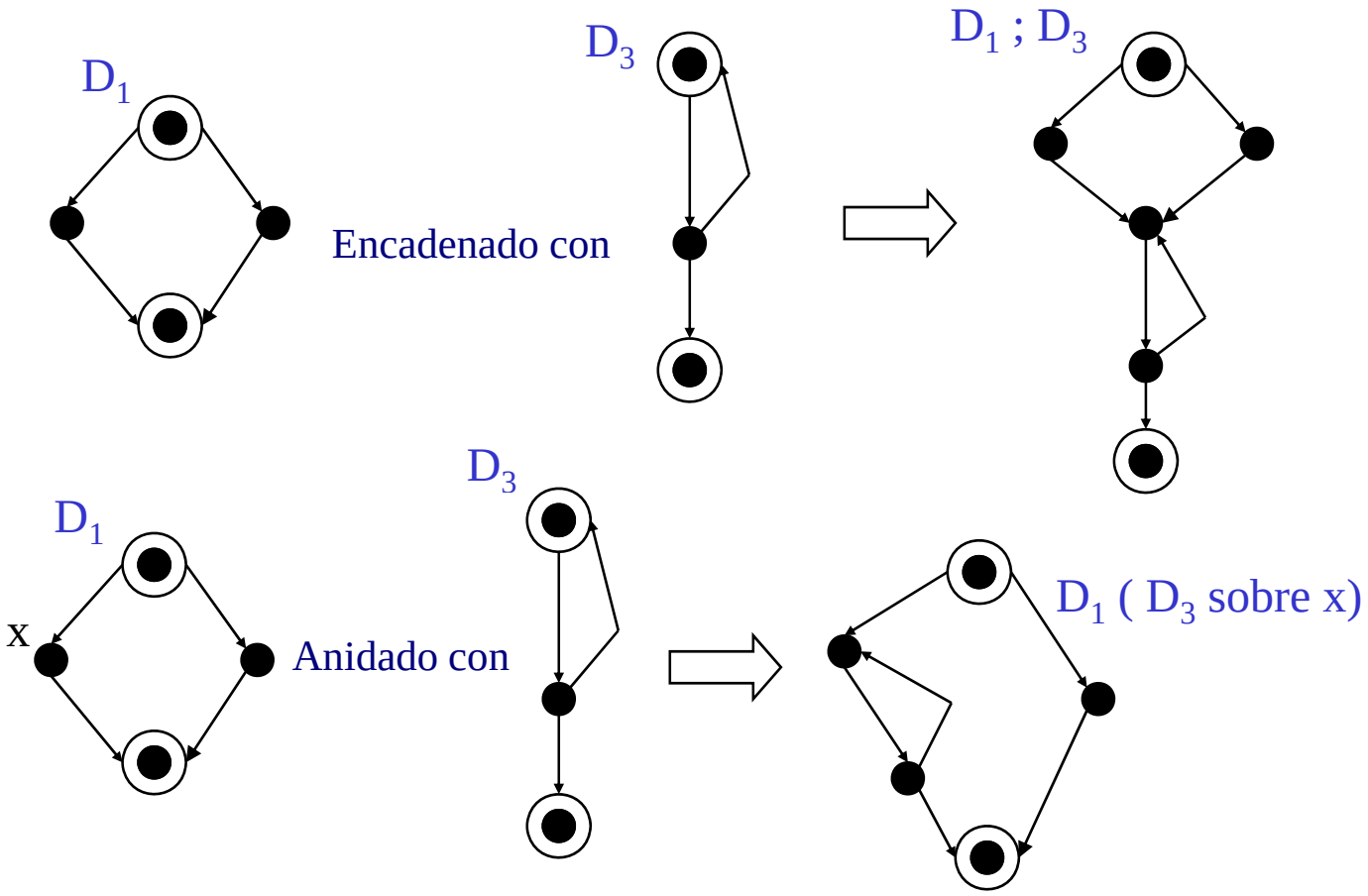
$$F(A ; A') = F(A) ; F (A')$$

  - Sean F_1 y F_2 dos grafos de flujo tal que F_1 tiene un nodo procedimiento x. El **anidamiento** de F_2 sobre x de F_1 se representa:

$$F_1(F_2 \text{ sobre } x)$$

Medición de atributos internos del producto (XXVI)
Medidas estructurales (IV)

Flujo de control. Grafos de flujo de control (IV)



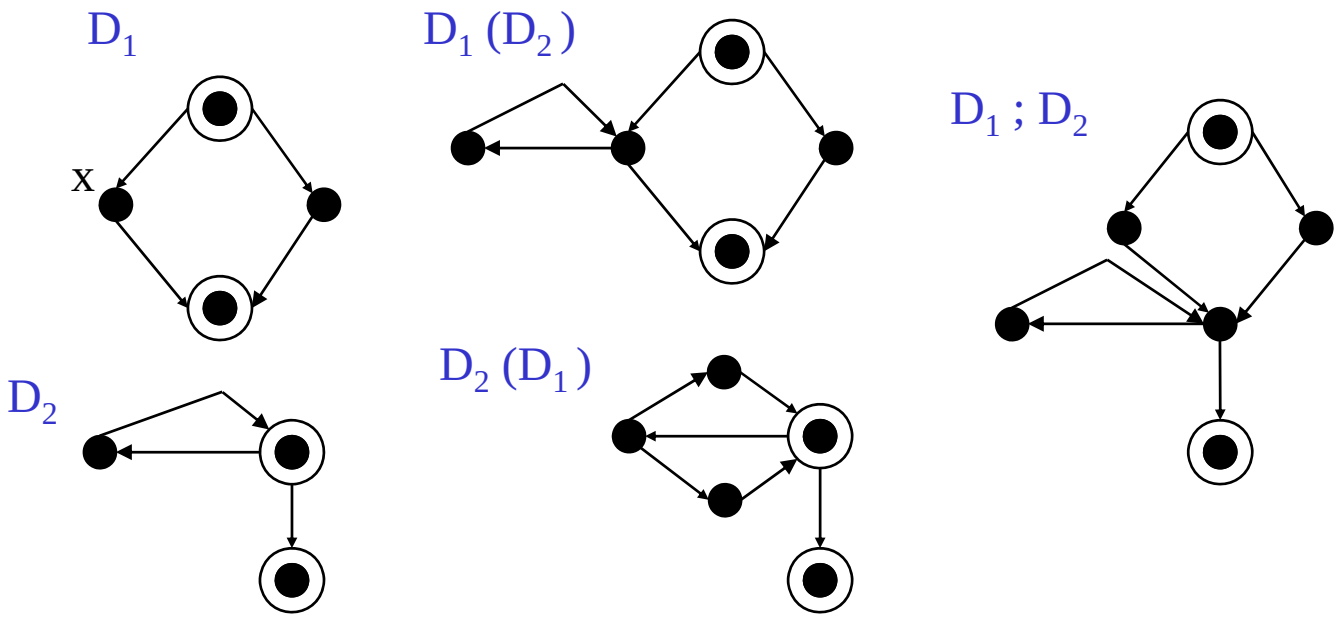
Medición de atributos internos del producto (XXVII) Medidas estructurales (V)

Flujo de control. Grafos de flujo de control (V)

- Examinando los grafos de un programa se puede evaluar si está estructurado o no. Una familia de grafos de flujo es miembro de los grafos S (estructurados) si satisface las siguientes reglas:
  - Cada miembro de S es S-estructurado
  - Si F y F' son grafos S-estructurados entonces lo son:
    - F ; F'
    - F(F')
  - Ningún grafo de flujo es S-estructurado a menos que se pueda mostrar que es generado por un número finito de aplicaciones de los pasos anteriores
- Se pueden definir un conjunto de estructuras de control como S-grafos básicos (primitivas). Cualquier conjunto derivado de ellos será S-estructurado

Medición de atributos internos del producto (XXVIII)
Medidas estructurales (VI)

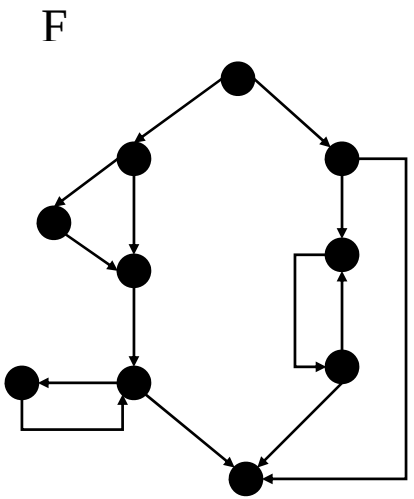
Flujo de control. Grafos de flujo de control (VI)

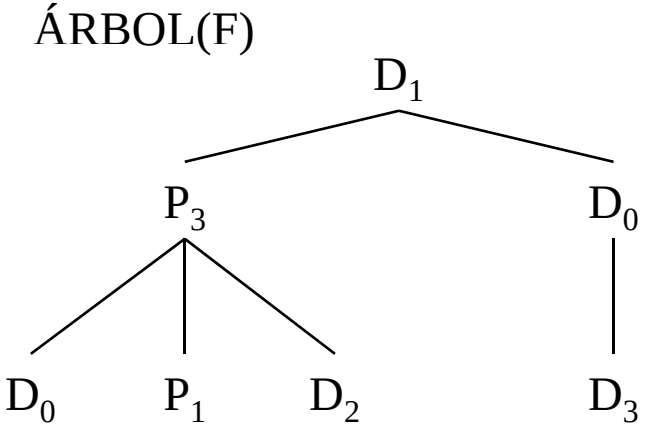


Medición de atributos internos del producto (XXIX)
Medidas estructurales (VII)

Flujo de control. Grafos de flujo de control (VII)

- Cualquier grafo de flujo puede asociarse con un ***árbol de descomposición*** que describe como se construye el grafo por concatenación y anidamiento de grafos básicos





$$F = D_1\ ((D_0\ ;\ P_1\ ;\ D_2),\ D_0\ (D_3))$$

Medición de atributos internos del producto (XXX)
Medidas estructurales (VIII)

Flujo de control. Grafos de flujo de control (VIII)

- ***Teorema de la descomposición básica:*** *Cada grafo de flujo tiene una única descomposición en una jerarquía de primitivas (grafos básicos)*
- El teorema de descomposición proporciona una manera de determinar si un grafo cualquiera es S-estructurado o no, para alguna familia de primitivas S
  - Si los nodos del árbol son miembros de S o son P_n entonces el grafo de flujo es un S-grafo

Medición de atributos internos del producto (XXXI)
Medidas estructurales (IX)

Flujo de control. Medidas jerárquicas (I)

El árbol de descomposición de los grafos de flujo es la base para realizar muchos tipos de medidas.

- ***Profundidad del anidamiento*** (α) de un grafo de flujo F:
  - **Primitivas**:

    $\alpha(P_1) = 0$ y $\alpha(F) = 1$ si F es una primitiva $\neq P_1$.
  - **Secuencia**:

    $\alpha(F_1; .....; F_n) = \max(\alpha(F_1), ..., \alpha(F_n))$
  - **Anidamiento**:

    $\alpha(F(F_1, ..., F_n)) = 1 + \max(\alpha(F_1), ..., \alpha(F_n))$

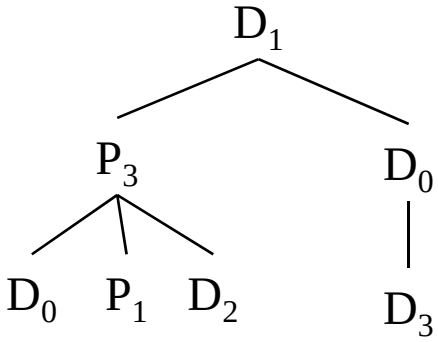
Ejemplo:

$\alpha(F) = \alpha(D_1((D_0 ; P_1 ; D_2), D_0(D_3)))$

$= 1 + \max(\alpha(D_0 ; P_1 ; D_2), \alpha(D_0(D_3)))$

$= 1 + \max(\max(\alpha(D_0); \alpha(P_1); \alpha(D_2)), 1 + \alpha(D_3))$

$= 1 + \max(\max(1, 0, 1), 2) = 1 + \max(1, 2) = 3$



Medición de atributos internos del producto (XXXII)
Medidas estructurales (X)

Flujo de control. Medidas jerárquicas (II)

- Si S es un conjunto arbitrario de primitivas, se puede decir que ***m*** es una ***medida jerárquica*** de ellas si se puede definir sobre un conjunto de S-grafos especificando:
  - m(F) para cada F $\in$ S (Regla M_1)
  - la secuencia de función(es) (regla M_2)
  - el anidamiento de función(es) (regla M_3)
- La unicidad de la descomposición básica implica que un S-grafo puede construirse de una manera única. Por tanto podemos definir reglas propias para crear nuevas medidas jerárquicas:

  Medida de la longitud (*v*) a partir del número de nodos (*n*)

  - M_1: $v(P_1) = 1$ y para todo $F \neq P_1$, $v(F) = n + 1$ (n=nº de nodos de proceso)
  - M_2: $v(F_1; .....; F_n) = \sum v(F_i)$
  - M_3: $v(F(F_1, ..., F_n)) = 1 + \sum v(F_i)$ para todo $F \neq P_1$
- Una vez que se haya caracterizado una medida jerárquica en términos de las reglas M_1, M_2 y M_3, se pueden calcular las medidas para todos los S-grafos

Medición de atributos internos del producto (XXXIII)
Medidas estructurales (XI)

Flujo de control. Medidas jerárquicas (III)

- ***Complejidad ciclomática de McCabe***
  - La complejidad de un programa se mide mediante el ***número ciclomático*** (***v***) de su grafo de flujo (F):

    $$v(F) = e - n + 2$$

    siendo *e* el número de arcos y *n* el número de nodos de F
  - El número ciclomático mide el número de caminos linealmente independientes
  - Se puede demostrar que para cualquier grafo F,

    $$v(F) = 1 + d$$

    donde *d* es el número de nodos predicado
  - Se puede definir *v* como una medida jerárquica:
    - M_1: $v(F) = 1 + d$ para cualquier primitiva F
    - M_2: $v(F_1; .....; F_n) = \sum v(F_i) - n + 1$
    - M_3: $v(F(F_1, ..., F_n)) = v(F) + \sum v(F_i) - n$ para cualquier primitiva F
  - El número ciclomático es un indicador útil de la dificultad de prueba y mantenimiento de un programa o módulo
  - Cuando se obtiene un número ciclomático superior a 10 para un módulo, éste puede ser problemático

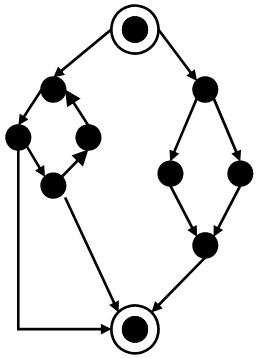
Medición de atributos internos del producto (XXXIV)
Medidas estructurales (XII)

Flujo de control. Medidas jerárquicas (IV)

- ***Complejidad esencial de McCabe***
  - Proporciona una medida del grado de estructuración de un programa. Se define como:

    $$ev(F) = v(F) - m$$

    siendo *m* el número de subgrafos de F que son grafos básicos D-estructurados (D_0, D_1, D_2 o D_3)
  - La complejidad esencial indica el grado en que un grafo puede descomponerse en subgrafos que sean primitivas D-estructuradas
  - La complejidad esencial de un programa D-estructurado es 1
  - Ejemplo: complejidad esencial del siguiente grafo



$v(F) = 5$

$m = 1$

$ev(F) = v(F) - m$

Medición de atributos internos del producto (XXXV)
Medidas estructurales (XIII)

Modularidad y flujo de información (I)

- Modelos de modularidad y flujo de información:
  - ***Diagramas de diseño***: representan la información referente a las relaciones entre módulos necesaria en el diseño
  - ***Grafos de llamada a módulos***: representación utilizada para realizar medidas intermodulares a un nivel más abstracto que los diagramas de diseño
  - ***Grafos de dependencia de datos***: Modelo de flujo de información utilizado para realizar medidas intra-modulares

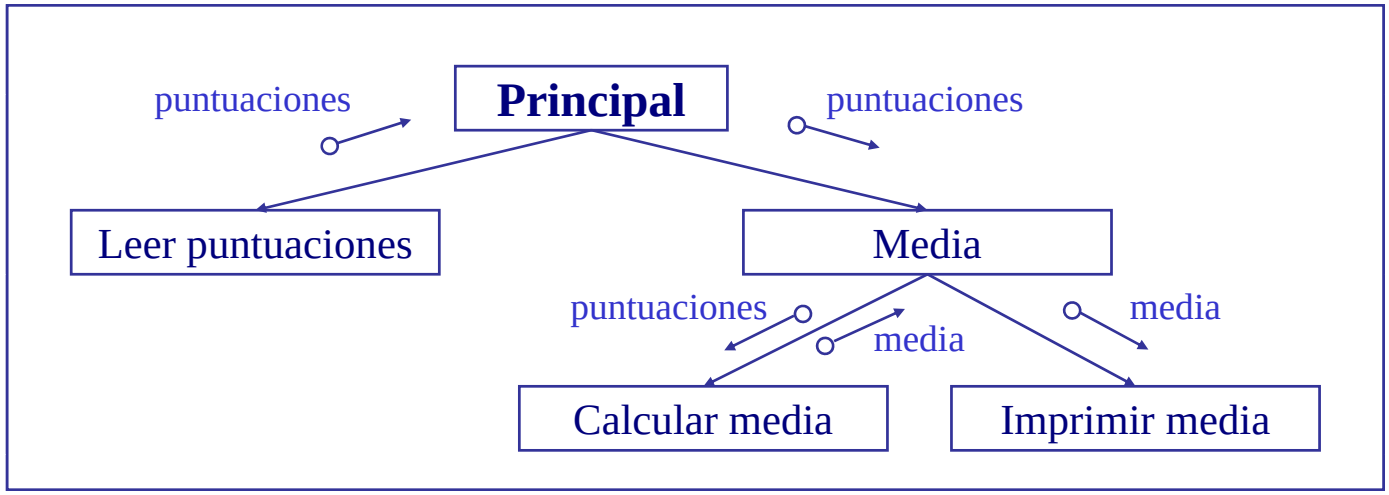
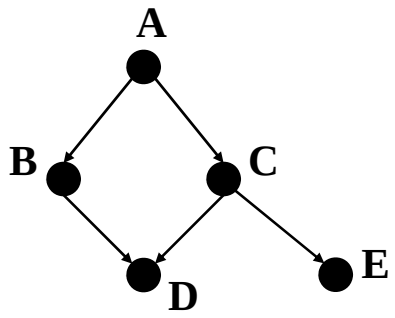


Diagrama de diseño

Medición de atributos internos del producto (XXXVI)
Medidas estructurales (XIV)

Modularidad y flujo de información (II)

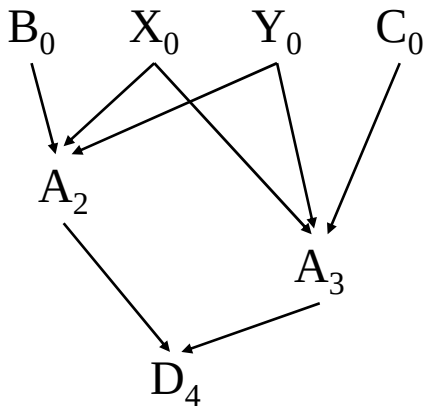


Módulo A llama a B, C

Módulo B llama a D

Módulo C llama a D, E

Grafo de llamada a módulos



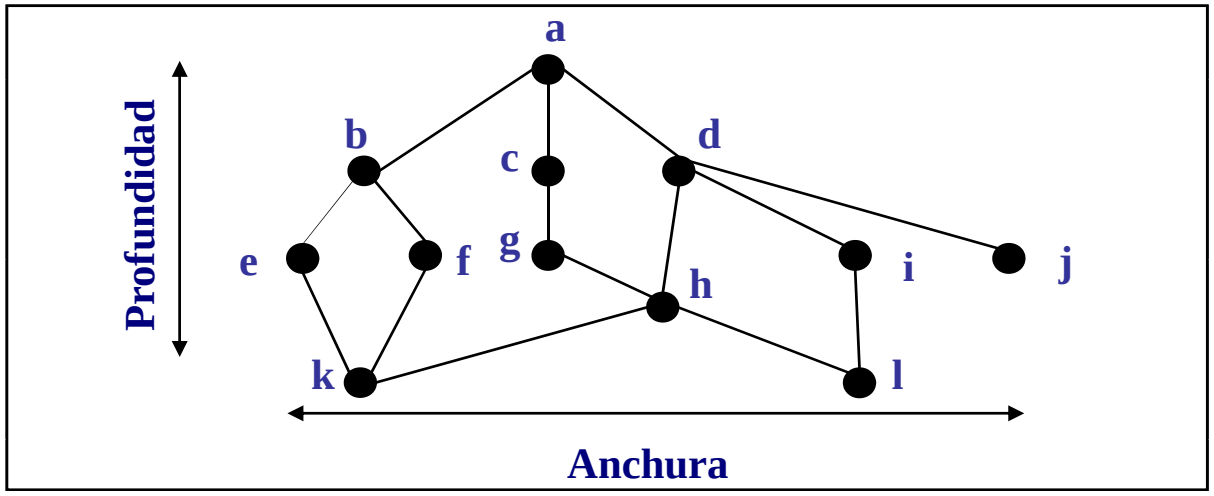
0 Inicializar
1 if X < Y
2 then A = B
3 else A = C
4 D = A

Grafo de dependencia de datos

Medición de atributos internos del producto (XXXVII)
Medidas estructurales (XV)

Modularidad y flujo de información (III)

- **Morfología:** *Forma de la estructura total del sistema.*
  - Se representa mediante ***grafos de dependencia.***
  - Características morfológicas medibles directamente:
    - ***Tamaño***: medida del número de nodos, número de arcos o combinación de ambos.
    - ***Profundidad***: medida del número de niveles del camino más largo entre el nodo raíz y un nodo hoja.
    - ***Anchura***: Medida del número máximo de nodos de un nivel.
    - ***Relación entre nodos y arcos***: medida de la densidad de conectividad.



Grafo de dependencia

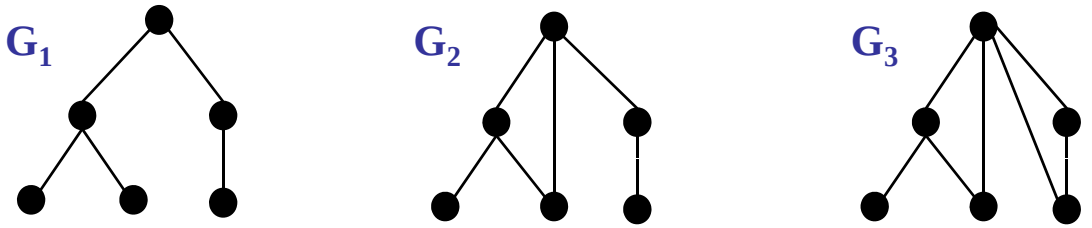
Medición de atributos internos del producto (XXXVIII)
Medidas estructurales (XVI)

Modularidad y flujo de información (IV)

- Conceptos relativos a grafos y árboles:
  - ***Grafo conectado***: si para cada par de nodos hay un camino entre ellos
  - ***Grafo completo***: si cada par de nodos del grafo está conectado directamente

    Un grafo completo con n nodos tiene n(n-1)/2 arcos
  - ***Árbol***: grafo conectado que no tiene ciclos (caminos que empiezan y terminan en el mismo nodo)
  - ***Impureza de árbol***: medida de la desviación de un grafo respecto a la estructura de árbol

    Un árbol con n nodos siempre tiene n-1 arcos
  - Para todo grafo conectado G, se puede construir al menos un ***árbol incluido*** sobre los mismos nodos que G
  - Un ***subgrafo incluido*** G' de un grafo G se construye sobre los mismos nodos que G pero con un subconjunto de arcos, de manera que cualquier par de nodos de G' estén conectados por un camino



Grafos con diferentes grados de impureza

Medición de atributos internos del producto (XXXIX)
Medidas estructurales (XVII)

Modularidad y flujo de información (V)

- ### Impureza de árboles
  - Los grafos de dependencia presentan ciertas propiedades que pueden servir para evaluar la calidad del diseño. Cualquier medida *m* de impureza de árboles debe satisfacer al menos cuatro propiedades:
    - **Propiedad 1**: m(G) = 0 si y sólo si G es un árbol
    - **Propiedad 2**: m(G) > m(G') si G se diferencia de G' sólo en la inserción de un arco extra
    - **Propiedad 3**: si A y A' son el número de arcos de G y G' respectivamente, y N y N' el número de nodos

      si N>N' y A - N + 1 = A' - N' + 1 entonces $m(G) < m(G')$
    - **Propiedad 4**: para todos los grafos G, $m(G) \leq m(K_N)$, donde N es el número de nodos de G y K_N es el grafo completo de N nodos
  - Se puede definir una medida de impureza de árboles ***m*(G)** que satisfaga esas propiedades y que se define como:

    Relación entre el nº de arcos de más respecto a la estructura de árbol y el máximo nº de arcos de más respecto a la estructura de árbol

    $$m(G) = 2\,(a - n + 1) / (n-1)(n-2)$$

    donde *a* es el número de arcos y *n* el número de nodos de G
    - Para asegurar la calidad del diseño los valores de m deben ser cercanos a cero

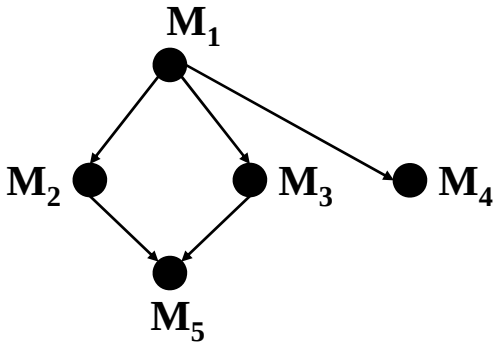
Medición de atributos internos del producto (XL)
Medidas estructurales (XVIII)

Modularidad y flujo de información (VI)

- **Reutilización interna:** *Grado con que los módulos se reutilizan dentro del mismo producto*
  - Una medida de reutilización es la ***medida de diseño del sistema***:

$$r(G) = a - n + 1$$

donde *a* es el número de arcos y *n* el número de nodos

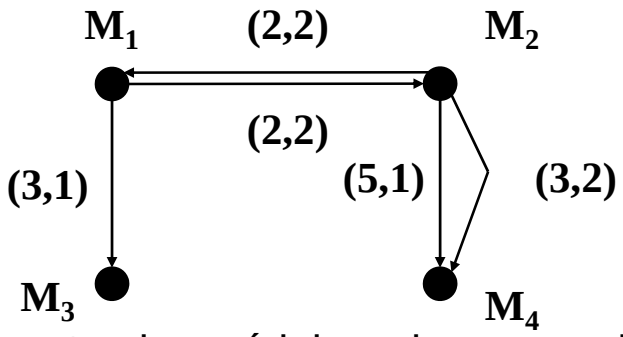


- **Acoplamiento:** *Grado de interdependencia entre módulos*
  - Tipos de acoplamiento entre módulos
    - R_0: relación sin acoplamiento
    - R_1: relación de acoplamiento de datos
    - R_2: relación de acoplamiento por estampado
    - R_3: relación de acoplamiento de control
    - R_4: relación de acoplamiento común
    - R_5: relación de acoplamiento por contenido

Medición de atributos internos del producto XLI
Medidas estructurales (XIX)

Modularidad y flujo de información (VII)

- Para representar el acoplamiento entre módulos se emplean unos grafos especiales llamados ***grafos de modelo de acoplamiento***:



  - Los nodos representan los módulos y los arcos el acoplamiento entre parejas de módulos
  - Cada arco está etiquetado con el par (i,j) donde i representa la relación R_i y j el número de veces que tiene lugar el acoplamiento de ese tipo

- Se puede medir el acoplamiento global de un sistema S que contiene los módulos D_1, ..., D_n :

$$c(D_i, D_j) = k + (n / (n + 1))$$

Donde k es el número de la peor relación de acoplamiento R_k entre D_i y D_j y n el número de interconexiones entre dichos módulos

$C(S)$ = valor medio del conjunto $\{ c(D_i, D_j): 1 \leq i \leq j \leq n\}$

Medición de atributos internos del producto (XLII)
Medidas estructurales (XX)

Modularidad y flujo de información (VIII)

- **Cohesión:** *Grado en que los componentes locales a un módulo colaboran para realizar una tarea concreta.*
  - Niveles de cohesión:
    - **Funcional**
    - **Secuencial**
    - **Por comunicaciones**
    - **Procedimental**
    - **Temporal**
    - **Lógica**
    - **Por coincidencia**
  - No existen procedimientos de medida para determinar el nivel de cohesión de un módulo, pero se puede obtener una idea del grado escribiendo una frase que describa el propósito del módulo .
  - La cohesión es también un atributo intermodular. Ésta puede calcularse de la forma:

$$\text{índice de cohesión} = \frac{\text{nº de módulos con cohesión funcional}}{\text{nº total de módulos}}$$

Medición de atributos internos del producto (XLIII)
Medidas estructurales (XXI)

Modularidad y flujo de información (IX)

- Conceptos sobre el **flujo de información** entre módulos:
  - Podemos decir que existe un ***flujo local directo*** si
    - un módulo invoca a un segundo módulo y le pasa información, o
    - el módulo invocado devuelve un resultado al que lo llama
  - Un ***flujo local indirecto*** existe si
    - el módulo invocado devuelve información que se pasa a un segundo módulo invocado
  - Un ***flujo global*** existe si la información va de un módulo a otro mediante estructuras globales de datos
- Atributos del fujo de información:
  - "***fan-in***" de un módulo M es el número de flujos locales que terminan en M más el número de estructuras de datos recuperadas por M
  - "***fan-out***" de un módulo M es el número de flujos locales que parten de M más el número de estructuras de datos actualizadas por M
  - ***Complejidad del flujo de información*** (CFI) propuesta por Henry y Kafura:

$$CFI(M) = \text{longitud}(M) * (\text{fan-in}(M) * \text{fan-out}(M))^2$$

Medición de atributos internos del producto XLIV
Medidas estructurales (XXII)

Modularidad y flujo de información (X)

- Las medidas de la complejidad del flujo de información de Henry y Kafura presentan problemas derivados de las definiciones informales del modelo, de la noción de flujos indirectos y de las medidas realizadas sobre módulos que se reutilizan
- Refinamientos propuestos por Shepperd :
  - Las llamadas recursivas a módulos se tratan como llamadas normales
  - Cualquier variable compartida por dos o más módulos se tratará como una estructura de datos global
  - Los módulos de biblioteca se ignorarán
  - Los flujos locales indirectos se contabilizarán sólo a través de un nivel de la jerarquía
  - Se ignorarán los flujos duplicados
  - La longitud del módulo se considerará un atributo independiente de la complejidad
- Medida de la complejidad de Shepperd (CS):

$$CS(M) = (\text{fan-in}(M) * \text{fan-out}(M))^2$$

Medición de atributos internos del producto XLIV
Medidas estructurales (XXIII)

Estructura de los datos (I)

- **Medidas locales**: Medida de las propiedades de las estructuras de datos individuales
  - Las medidas de estructuras de datos complejas se definen jerárquicamente en términos de los valores de los datos primitivos que las forman y las operaciones necesarias para construirlas
- **Medidas globales**: Globalmente se puede medir la cantidad de datos de un sistema dado
  - Medida del número de operandos distintos:

    μ_2 = nº de variables + nº de constantes + nº de etiquetas

  - Medida del tamaño de las bases de datos (modelo COCOMO):

$$D/P = \frac{\text{Tamaño de la base de datos en bytes o caracteres}}{\text{Tamaño del programa en DSI*}}$$

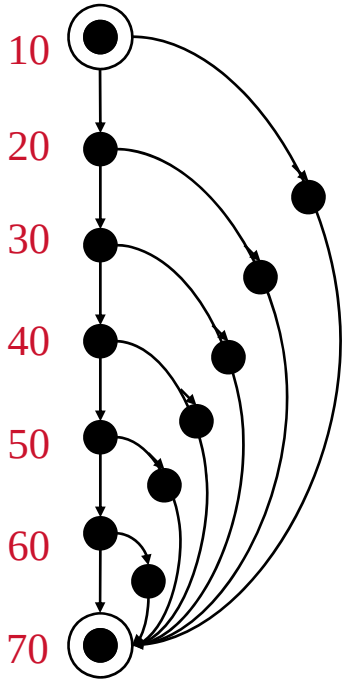
*DSI: *Delivered Source Instructions*

Medición de atributos internos del producto (XLIV)
Medidas estructurales (XXIV)

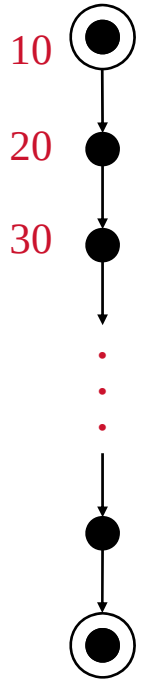
Estructura de los datos (II)

- Relación entre la estructura de los datos y la estructura de control de un programa: Se puede transferir la complejidad estructural del flujo de control de un programa a la estructura de los datos

```
10  Si sc >= 90 entonces
            mostrar “grado A”
            ir a 70
. . .
60  Si sc >= 40 entonces
            mostrar “grado E”
            ir a 70
70  FIN
```



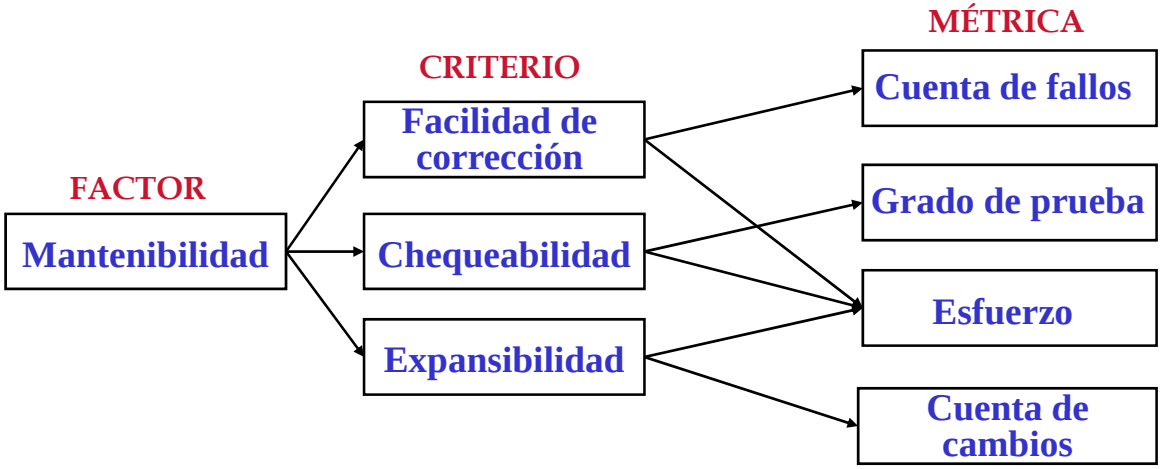

```
10  A[9] = “grado A”
. . .
60  A[4] = “grado E”
70  I = parte entera de (sc/10)
80  mostrar A[I]
90 FIN
```



Medición de atributos externos del producto (I)

Los atributos externos de un producto son aquellos que pueden medirse únicamente con respecto a cómo el producto se relaciona con su entorno.

- Los atributos externos sólo son medibles cuando el producto esta completo.
- La mayoría de los atributos externos están relacionados con algún aspecto de la calidad.
  - Los modelos de calidad recogen atributos clave denominados ***factores de calidad***, que normalmente son atributos externos de alto nivel.
  - Los factores de calidad pueden descomponerse en atributos de bajo nivel denominados ***criterios de calidad***.
  - Los criterios de calidad pueden asociarse con un conjunto de atributos de bajo nivel medibles directamente obteniendo las ***métricas de calidad***.

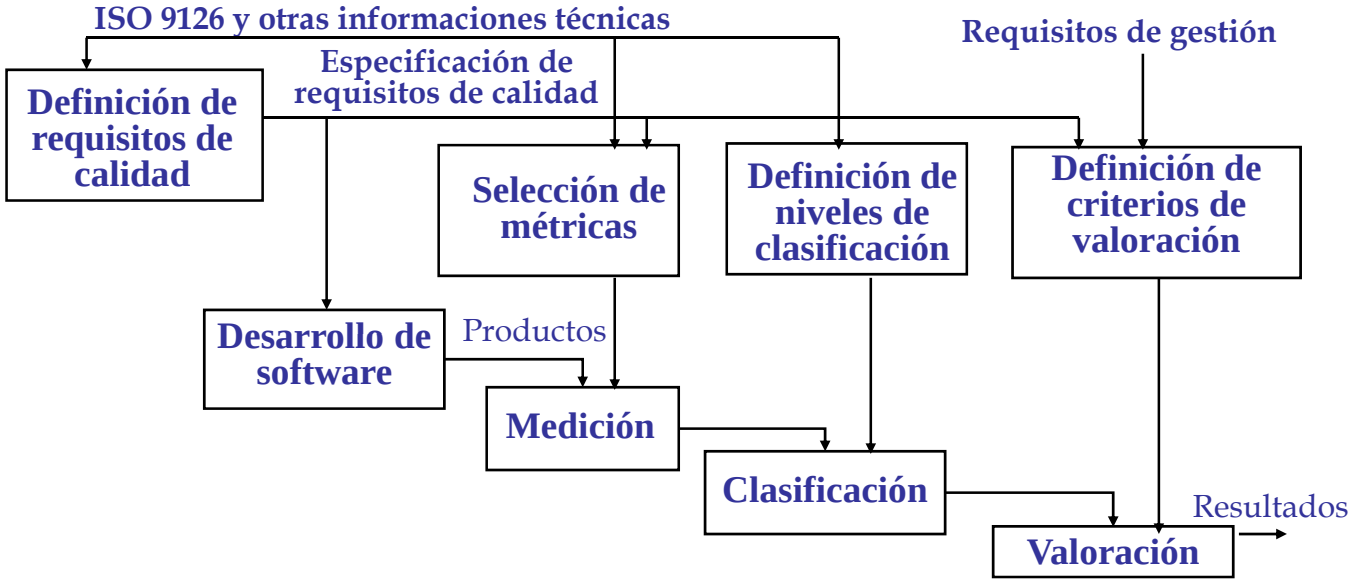


Descomposición del factor “mantenibilidad”

Medición de atributos externos del producto (II)

El modelo de calidad estándar ISO 9126

- En el estándar denominado ***Evaluación de Productos Software: Características de calidad y guías para su uso***, la calidad se descompone en seis factores:
  - Funcionalidad
  - Fiabilidad
  - Eficiencia
  - Usabilidad
  - Mantenibilidad
  - Portabilidad
- El estándar define un proceso para evaluar la calidad del software.



Modelo del proceso de evaluación ISO 9126

Medición de atributos externos del producto (III)

Medición de la portabilidad

- Se entiende por portabilidad la facilidad de mover una aplicación de un entorno “host” a otro
- La portabilidad se puede expresar como:

$$\text{portabilidad} = 1 - (ET/ER)$$

ET: medida de los recursos necesarios para mover el sistema a otro entorno (*Target Environment*).

ER: medida de los recursos necesarios para crear el sistema en el entorno residente (*Resident Environment*)

Medición de atributos externos del producto (IV)

Medición de la densidad de defectos

- Para cualquier producto software se pueden considerar dos tipos de defectos:
  - Defectos conocidos
  - Defectos latentes
- La densidad de defectos se puede definir en función de los primeros:

$$\text{Densidad de defectos} = \frac{\text{número de defectos conocidos}}{\text{tamaño del producto}}$$

Medida de la calidad basada en defectos

- Se puede medir la calidad en función de la relación:

$$\frac{\text{Tiempo empleado en la corrección de defectos “post-release”}}{\text{Tiempo total de desarrollo del sistema}}$$

Medición de atributos externos del producto (V)

Medidas de usabilidad (I)

- Boehm define usabilidad como el grado en que un producto se puede usar de forma apropiada y práctica.
- **ISO 9241-1:** Define la usabilidad como el *"instante en el cual un producto puede ser utilizado por usuarios específicos para alcanzar metas específicas con efectividad, eficiencia, y satisfacción"*.
- **ISO 9126-4:** Describe a la calidad en uso como compuesta por las características productividad, efectividad, seguridad y satisfacción
- La buena usabilidad incluye:
  - Manuales bien estructurados
  - Buen uso de menús y gráficos
  - Mensajes de error informativos
  - Funciones de ayuda
  - Interfaces consistentes
- La usabilidad se puede descomponer en atributos medibles de los siguientes tipos:
  - Nivel de entrada
  - Nivel de aprendizaje
  - Facilidad de manejo

Medición de atributos externos del producto (VI)

Medidas de usabilidad (II)

- Las medidas de usabilidad se suelen expresar en función del rendimiento del usuario. Se han propuesto varias medidas de ese tipo:
  - ***Efectividad de las tareas***:

    % de efectividad = (cantidad * calidad)/100
  - ***Eficiencia temporal***:

    eficiencia = efectividad/tiempo de tarea
  - ***Periodo de tiempo productivo***:

$$\text{periodo productivo} = \frac{\text{tiempo de tarea - tiempo no productivo}}{\text{tiempo de tarea}} \times 100$$

  - ***Eficiencia relativa del usuario***:

$$\text{eficiencia relativa} = \frac{\text{eficiencia del usuario}}{\text{eficiencia del experto}} \times 100$$

Medición de atributos externos del producto (VII)

Medidas de usabilidad para aplicaciones Web (I)

- La usabilidad es uno de los principales factores de calidad de una aplicación Web
- Puede medirse en función de atributos que caracterizan la usabilidad de las interfaces de usuario [Nielsen, 1996]
  - Facilidad de aprendizaje
  - Eficiencia
  - Facilidad de recordar
  - Errores
  - Satisfacción

Medición de atributos externos del producto (VIII)

Medidas de usabilidad para aplicaciones Web (II)

- **Modelo de medición de usabilidad**
  - Requisitos de una aplicación usable:
    - Debe ser amigable
    - Debe ser fácil de entender y de aprender
    - Debe ayudar a los usuarios finales a alcanzar sus objetivos
    - Debe mejorar la productividad de los usuarios finales
  - Compuesto de cuatro subcaracterísticas:
    - Facilidad de uso
    - Entendibilidad
    - Efectividad
    - Eficiencia

Medición de atributos externos del producto (IX)

Medidas de usabilidad para aplicaciones Web (III)

- **Facilidad de uso**
  - Indicaciones de diseño
    - Ayuda navegacional
    - Ayuda y documentos en línea
    - Teclas de función preasignadas
    - Utilizaciones de zonas destacadas, subrayados, coloreados, y otros indicadores que ayuden al usuario a localizar elementos
    - El menor número de pantallas posible para implementar la funcionalidad
    - Soporte bilingüe
    - Soporte multilingüe

Rango	Número de indicaciones incluidas
0	Ninguna
1	De 1 a 3
2	De 4 a 5
3	6 o más pero no hay requisitos de usuario relacionados con la eficiencia
4	6 o más junto con requisitos de usuario para eficiencia suficientemente fuertes para requerir tareas de diseño
5	6 o más junto con requisitos de usuario para eficiencia suficientemente fuertes para requerir el uso de herramientas especiales y procesos para demostrar que los objetivos son alcanzados

Medición de atributos externos del producto (X)

Medidas de usabilidad para aplicaciones Web (IV)

- **Entendibilidad**: refleja la capacidad de la aplicación para interactuar de manera sencilla con los usuarios finales.
  - Indicadores
    - Número de veces que el usuario final accede, durante unas sesión, a las funciones de ayuda en línea
    - Número de páginas web accedidas y rápidamente abandonadas
    - Números de mensajes de error visualizados durante una sesión
    - Tiempo que necesita el usuario para pasar de una página web a otra (valores de 0 a 3 de acuerdo a un rango de posibles valores)
  - La medición final de la entendibilidad se hace calculando la media aritmética entre los valores obtenidos de la medición de los tres primeros indicadores y el rango asignado al cuarto
- **Efectividad**:
  - Indicadores
    - Número de transacciones diarias abandonadas (no completadas) por usuarios finales
    - El número de sesiones de trabajo diarias abandonadas sin ninguna transacción completada
  - La medición final de la efectividad se hace calculando la media aritmética entre los valores obtenidos de la medición de los indicadores

Medición de atributos externos del producto (XI)

Medidas de usabilidad para aplicaciones Web (V)

- **Eficiencia**: capacidad de ayudar a los usuarios finales a alcanzar sus objetivos en menos tiempo que el requerido para alcanzar los mismos objetivos utilizando otros instrumentos
  - Indicadores
    - El tiempo necesario para completar exitosamente una transacción
    - Número de transacciones completadas con éxito en un periodo de tiempo dado
  - La medición final de la eficiencia se hace calculando la media aritmética entre los rangos asignados a cada indicador en las siguientes tablas

Rango	Tiempo
0	tiempo <= 3 minutos
1	3 minutos < tiempo <= 5 minutos
2	5 minutos < tiempo <= 10 minutos
3	tiempo > 10 minutos

Rango	Nº de transacciones completadas (TC)
0	TC> 20
1	12< TC <= 20
2	6 < TC <= 12
3	TC <= 6

Medición de atributos externos del producto (XII)

Medidas de mantenibilidad (I)

- La mantenibilidad de un producto refleja su facilidad de comprensión, mejora y corrección
- Se puede aplicar tanto al código como a los documentos de diseño y especificación de requisitos
- Una medida de mantenibilidad es la denominada **MTTR** (***mean time to repair***). Para calcularla se requiere conocer:
  - Tiempo de reconocimiento del problema
  - Tiempo de demora administrativa
  - Tiempo de recolección de herramientas
  - Tiempo de análisis del problema
  - Tiempo de especificación del cambio
  - Tiempo de cambio
- Otras medidas dependientes del entorno son:
  - Número de problemas no resueltos
  - Tiempo utilizado en problemas no resueltos
  - Porcentaje de cambios que introducen nuevos fallos
  - Número de módulos modificados en la implementación de un cambio

Medición de atributos externos del producto (XIII)

Medidas de mantenibilidad (II)

- La mantenibilidad puede predecirse también a partir de atributos internos como la complejidad o la calidad de la documentación
  - Se han realizado estudios sobre la relación entre el número ciclomático y el esfuerzo de mantenimiento
  - En [Porter y Selby, 1990] se utilizan árboles de decisión para determinar los atributos más influyentes en la aparición de errores de interfaz entre módulos durante el mantenimiento del sistema
  - Belady y Lehman [Belady y Lehman, 1976] sugieren un modelo para predecir el esfuerzo de mantenimiento en función de atributos externos e internos

$$M = p + K^{(c-d)}$$

M: esfuerzo total de mantenimiento
p: esfuerzo productivo
K: constante empírica
c: medida de complejidad
d: medida del grado de familiaridad con el software

Medición de recursos (I)

Los recursos son las entidades que se requieren en las actividades de proceso,

- Los recursos incluyen cualquier entrada en la producción de software:
  - Personal
  - Materiales
  - Herramientas
  - Métodos …
- Atributos de recursos:
  - El ***coste*** generalmente se mide, respecto a todos los recursos, pudiéndose ver como el coste de las entradas afecta al coste de las salidas.
  - La ***productividad*** es otro atributo externo importante que depende del proceso de desarrollo. Normalmente se mide de la forma:

cantidad de salida/ esfuerzo de entrada

Medición de recursos (II)

Productividad (I)

- La productividad de un recurso de software se mide en función de la proporción entre lo que entra y sale de un proceso de producción de software
- Ecuación de productividad:

tamaño de software/ esfuerzo

- Unidades más utilizadas:
  - Tamaño: líneas de código
  - Esfuerzo: persona-meses

Recurso	Proceso	Producto	Ejemplo de recursos usados
Programador	Codificación	Programa	Compilador
Programador	Codificación	Grupo de programas	Compilador y analizador estático
Equipo de programación	Codificación	Programa	Compilador
Equipo de programación	Codificación	Grupo de programas	Compilador y LAN
Diseñador	Desarrollo	Diseño detallado de módulos	Herramienta CASE
Programador	Mantenimiento	Programas y documentación	Programa
Programador	Mantenimiento	Programas y documentación	Herramientas de reingeniería
Compilador	Compilación	Código	Microprocesador

Medición de recursos (III)

Productividad (II)

- La medición de la productividad en función del número de líneas de código puede presentar problemas:
  - Dependencia de la técnica de contar
  - Dependencia del lenguaje de programación
  - Penalización del buen estilo de programación ...
- Para solventar esos problemas algunos autores han propuesto medir la productividad a partir de la funcionalidad en lugar del tamaño:

número de puntos de función/ persona-meses

- Ventajas de los puntos de función:
  - Reflejan mejor el valor de la salida
  - Pueden usarse en cualquier momento del ciclo de vida
  - Se pueden utilizar para medir el progreso comparando los puntos de función completados frente a los incompletos
- El principal problema que presentan los puntos de función es que son más difíciles de medir que las líneas de código

Medición de recursos (IV)

Equipos (I)

- La estructura del equipo del proyecto es un factor clave en la productividad
- El factor particular que afecta a la productividad es la ***complejidad de las comunicaciones***: *complejidad causada por el número de individuos implicados y el método de comunicación requerido entre los miembros de un proyecto* [Grady y Caswell, 1987]
- Si se representa la estructura del equipo mediante un grafo donde los nodos son los miembros del equipo y los arcos los caminos de comunicación directa entre ellos, se pueden establecer las siguientes medidas:
  - ***Tamaño***: número de nodos del grafo
  - ***Densidad de comunicación***: relación entre arcos y nodos
  - ***Nivel de comunicación***: medida de la impureza del árbol
  - ***Nivel de comunicación individual***: grado del nodo correspondiente (suma de *fan-in* y *fan-out* del nodo)
  - ***Promedio del nivel de comunicación individual***

Medición de recursos (V)

Equipos (II)

- Otro factor clave que afecta a la productividad es la experiencia del equipo
- Fenton establece la siguiente escala ordinal para medir la experiencia de cada miembro del equipo:
  - 0 Ninguna experiencia previa
  - 1 Familiaridad pero sin experiencia práctica
  - 2 Experiencia práctica en un proyecto (más de 20 h.)
  - 3 Experiencia en varios proyectos (entre 21 y 100 h.)
  - 4 Amplia experiencia.
- La experiencia del equipo se calcula hallando la media de las medidas de experiencia individual.
- Algunos métodos como COCOMO utilizan escalas de medida ordinales (muy baja, baja, nominal, alta y muy alta) para cada uno de los atributos de personal: experiencia en la aplicación, en el lenguaje, en el uso de herramientas, etc.
- La productividad del equipo no sólo depende de las productividades individuales de sus miembros, sino también de la comunicación entre ellos

Medición de recursos (VI)

Métodos y herramientas (I)

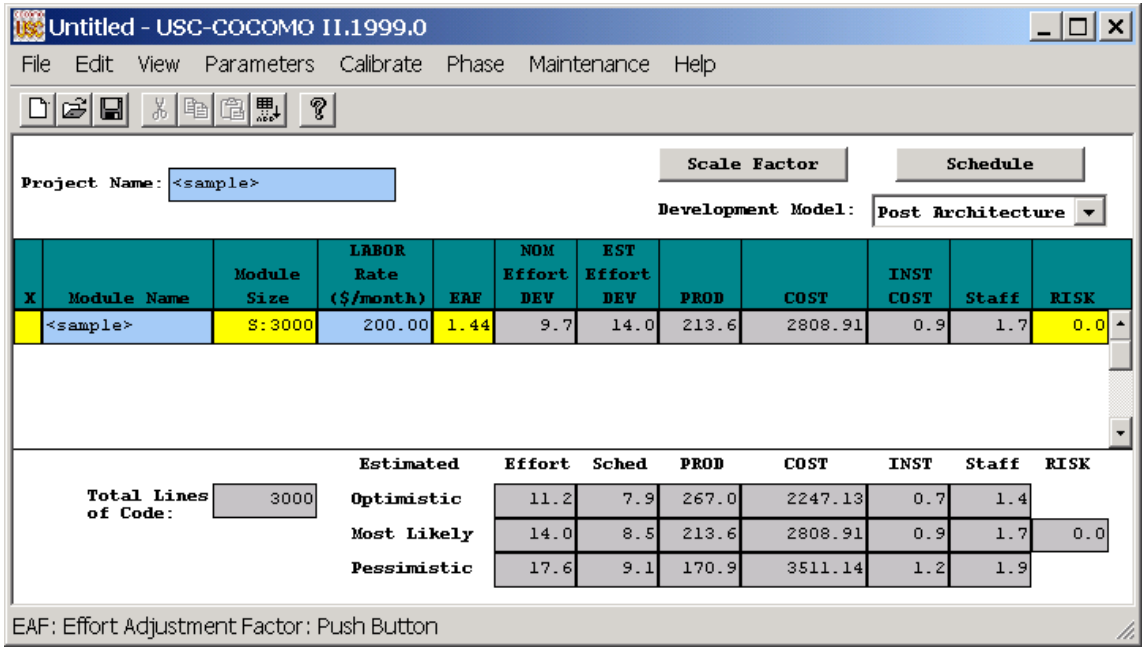
- Existen pocos estudios sobre el grado en que las herramientas aumentan la productividad
- Los modelos de estimación del esfuerzo requieren una medida del nivel de uso de métodos y herramientas
- El modelo COCOMO incluye dos guías de coste: *uso de herramientas* y *uso de técnicas modernas de programación* con las escalas de medida que muestra la tabla siguiente:

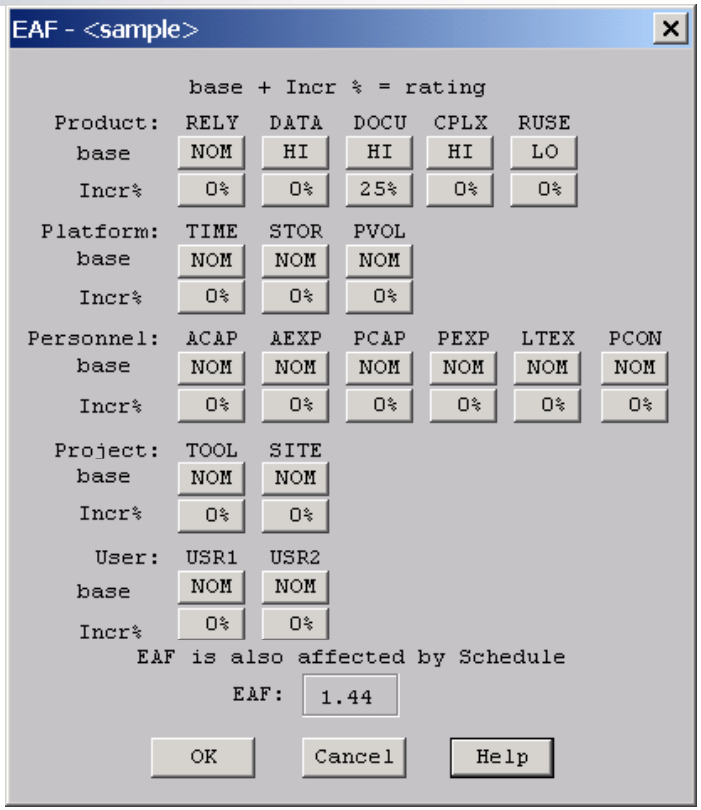
Uso de herramientas	
Categoría	**Significado**
Muy baja	Editores, depuradores
Baja	CASE front-end, back-end, baja integración
Nominal	Herramientas de ciclo de vida básicas, integración moderada
Alta	Herramientas de ciclo de vida potentes y maduras, integración moderada
Muy alta	Herramientas de ciclo de vida potentes y maduras, bien integradas con procesos, métodos y reutilización

Medición de recursos (VII)

Métodos y herramientas (II)

- Fenton propone la siguiente escala para evaluar el uso de herramientas por cada diseñador:
  0 No se usan herramientas.
  1 Las herramientas sirven de ayuda en el 20% de la documentación.
  2 Las herramientas se usan para documentar al menos el 50% del diseño de alto nivel.
  3 Las herramientas se usan para documentar al menos el 50% del diseño de alto nivel y diseño detallado.
  4 Las herramientas se usan para el diseño y la generación automática de código de al menos el 50% del sistema.
  5 Las herramientas se usan para el diseño y la generación automática de código de al menos el 90% del sistema.
- Escalas similares se pueden utilizar para cuantificar otros recursos:
  - Homogeneidad y compatibilidad del equipo del proyecto.
  - Uso de correo electrónico y otras facilidades de comunicación.
  - Nivel de soporte administrativo.
  - Nivel de recursos de información.
  - . . .





Product attributes

- Required software reliability (RELY)
- Database size (DATA)
- Documentation match to life-cycle needs (DOCU)
- Product complexity (CPLX)
- Required Reusability (RUSE)

Platform attributes

- Execution time constraint (TIME)
- Main storage constraint (STOR)
- Platform volatility (PVOL)

Personnel attributes

- Analyst capabilities (ACAP)
- Applications experience (AEXP)
- Programmer capabilities (PCAP)
- Platform experience (PEXP)
- Programming language experience (LEXP)
- Personnel Continuity (PCON)

Project attributes

- Use of software tools (TOOL)
- Multisite Development (SITE)

Métricas para sistemas orientados a objetos (I)

- La mayoría de las métricas orientadas a objetos se basan en las características distintivas del software orientado a objetos respecto al software convencional:
  - ***Localización***: forma en que se concentra la información dentro de un programa
  - ***Encapsulamiento***: empaquetamiento de una colección de elementos.
  - **Ocultamiento de la información**: supresión de los detalles operativos de un componente.
  - ***Herencia***: mecanismo que permite la propagación de responsabilidades de un objeto a otro.
  - ***Abstracción***: mecanismo que permite concentrarse en los detalles esenciales de un componente sin considerar los de nivel inferior.
- Clasificación de las métricas:
  - Métricas a nivel del sistema
  - Métricas orientadas a clases
  - Métricas orientadas a operaciones
  - Métricas para pruebas orientadas a objetos
  - Métricas para proyectos orientados a objetos

Métricas para sistemas orientados a objetos (II)

Métricas a nivel del sistema

- ***Métricas* MOOD (*Metrics for Object Oriented Design*)** [Brito e Abreu y Melo 1996]
  - ***Proporción de métodos ocultos*** (PMO). Es la proporción entre la suma de los grados de invisibilidad de los métodos en todas las clases (métodos protegidos y privados) y el número total de métodos definidos en el sistema
  - ***Proporción de atributos ocultos*** (PAO). Es la proporción entre la suma de los grados de invisibilidad de los atributos en todas las clases (atributos protegidos y privados) y el número total de atributos definidos en el sistema
  - ***Proporción de métodos/atributos heredados*** (PMH/PAH). Proporción entre el número de métodos/atributos heredados y el número total de métodos/atributos
  - ***Proporción de polimorfismo*** (PDP): es la proporción entre el número real de posibles situaciones polimorfas para una clase C_i y el máximo número posible de situaciones polimorfas en C_i. Es decir, es el número de métodos heredados redefinidos dividido entre el máximo número de situaciones polimorfas distintas posibles

$$POF = \frac{\sum_{i=1}^{TC} M_o(C_i)}{\sum_{i=1}^{TC} [M_d(C_i) \times DC(C_i)]}$$

Donde

$M_d(C_i) = M_n(C_i) + M_o(C_i)$

$DC(C_i)$ es el número de descendientes de C_i.

$M_n(C_i)$ es el número de métodos nuevos

$M_o(C_i)$ es el número de métodos redefinidos

TC es el número total de clases

Métricas para sistemas orientados a objetos (III)

Métricas orientadas a clases (I)

- ***Conjunto de métricas CK*** [Chidamber y Kemerer 1994]
  - ***Métodos ponderados por clase*** (MPC): recoge la noción de complejidad.

    Para una clase C con M_1, M_2, ...,M_n métodos con un peso de complejidad c_1, c_2, ..., c_n respectivamente,

    $$MPC = \sum c_i$$

    Si todos los métodos tienen igual complejidad $c_i = 1$ y MPC = n
  - ***Profundidad del árbol de herencia*** (PAH): longitud del camino máximo entre un nodo y la raíz del árbol
  - ***Número de hijos*** (NH): es el número de descendientes inmediatos de una clase (nodo)
  - ***Acoplamiento entre clases*** (AC): número de clases que se acoplan con una clase dada. Hay dependencia entre dos clases cuando una de ellas usa métodos o variables de la otra clase
  - ***Respuesta para una clase*** (RPC): es el número de métodos locales a una clase más el número de métodos llamados por los métodos locales
  - ***Métrica de falta de cohesión*** (MFC): número de grupos de métodos locales que no acceden a atributos comunes. MFC es el número de conjuntos disjuntos de métodos locales

Métricas para sistemas orientados a objetos (IV)

Métricas orientadas a clases (II)

- ***Métricas de Lorenz y Kidd*** [Lorenz y Kidd 1994]
  - ***Tamaño de clase*** (TC). El tamaño general de una clase se determina utilizando las siguientes medidas:
    - Número total de operaciones
    - Número de atributos
  - ***Número de operaciones invalidadas por una subclase*** (NOI). La invalidación consiste en la sustitución en la subclase de una operación heredada por una versión especializada (reescritura de métodos).
  - ***Número de operaciones añadidas por una subclase*** (NOA): operaciones propias de la subclase que no aparecen en las superclases.
  - ***Índice de especialización*** (IE):

$$IE = (NOI \times nivel)/ M_{total}$$

nivel: nivel de la clase en la jerarquía de clases

M_{total}: Número total de métodos de la clase

Métricas para sistemas orientados a objetos (V)

Métricas orientadas a operaciones

- ***Métricas de Churcher y Shepperd*** [Churcher y Shepperd 1995]
  - ***Tamaño medio de operación*** (TO_{med}): Número de mensajes enviados por la operación
  - ***Complejidad de operación*** (CO): se puede aplicar cualquier métrica de complejidad del software convencional
  - ***Número medio de parámetros por operación*** (NP_{med})

Métricas para pruebas orientadas a objetos (I) [Binder 1994]

- ***Encapsulamiento***
  - ***Carencia de cohesión en métodos*** (CCM): se aplica la métrica CK.
  - ***Porcentaje público y protegido*** (PPP): porcentaje de atributos públicos de la clase
  - ***Acceso público a datos miembro***(APD): número de clases que pueden acceder a atributos de otras clases

Métricas para sistemas orientados a objetos (VI)

Métricas para pruebas orientadas a objetos (II)

- ***Herencia***
  - ***Número de clases raíz*** (NCR): número de jerarquías de clases distintas.
  - ***Admisión*** (ADM): indicación de la herencia múltiple.
  - ***Número de descendientes*** (NDD) y ***profundidad del árbol de la herencia*** (APM): se aplican las correspondientes métricas CK

Métricas para proyectos orientados a objetos

- ***Métricas de Lorenz y Kidd*** [Lorenz y Kidd 1994]
  - ***Número de guiones de escenario*** (NGE): un guión de escenario es una secuencia detallada de pasos que describen la interacción entre el usuario y la aplicación.
  - ***Número de clases clave*** (NCC): clases que se centran en el dominio de negocio del problema en cuestión.
  - ***Número de subsistemas*** (NSUB): proporcionan una idea de la asignación de recursos, de la planificación y del esfuerzo global de integración.

Métricas para sistemas orientados a objetos (VII)

Métricas para diagramas UML (I)

Métricas de Marchesi (I) [Marchesi, 1998]

- Definidas para diagramas de clases
- Tres categorías:
  - Relativas a clases simples
  - Relativas a paquetes
  - Relativas al sistema completo

Variables y constantes empleadas en las métricas de Marchesi

Variables y Constantes	Descripción
C	Array con las clases del sistema.
NC =dim(C)	Número total de clases.
P	Array con los paquetes del sistema.
NP =dim(P)	Número total de paquetes.
G	Array con las clases raíz de las jerarquías de herencia
NG = dim(G)	Número total de clases raíz
$B^{(i)}$	Array con las superclases de la clase c_i (en todos los niveles)
$b^{(i)}$	Array con los índices de todas las superclases de la clase c_i. Por tanto: $B^{(i)} = \{c_k \mid k\ b^{(i)}\}$
$R^{(k)}$	Array con las responsabilidades de la clase c_k
$NR_k = dim(R^{(k)})$	Número de responsabilidades de la clase c_k
NA_k	Número de responsabilidades abstractas de la clase c_k
$S^{(k)}$	Array con las subclases inmediatas de la clase c_k
$NS_k = dim(S^{(k)})$	Número de subclases inmediatas de la clase c_k
$D^{(k)}$	Array con las dependencias desde la clase c_k
$ND_k = dim(D^{(k)})$	Número de dependencias desde la clase c_k
$[P]_{NCxNP}$	Matriz clase-paquete: el elemento p_{ik} es 1 si la clase C_i pertenence al paquete P_k . Cada fila de [P] tiene un, y sólo un, elemento igual a 1; todos los demás son 0.
$[D]_{NCxNC}$	Matriz dependencia: el elemento d_{ik} es el número de dependencias entre la clase C_i (cliente) y la clase C_k (servidor)

Métricas para sistemas orientados a objetos (VIII)

Métricas para diagramas UML (II)

- ***Métricas de Marchesi*** (II)
  - **Métricas de clase**:
    - CL1: número ponderado de responsabilidades de una clase, heredadas o no.
    - CL2: número ponderado de dependencias de una clase, las específicas y las heredades tendrán diferentes pesos.
  - **Métricas de paquete**:
    - PK1: relativa al número de dependencias entre las clases del paquete, P_k, y las clases de los otros paquetes.
    - PK2: dependencias cuyas clases servidoras pertenecen al paquete P_k.
    - PK3: media de los valores de la métrica PK1.
  - **Métricas de sistema**:
    - OA1: número total de clases, N_C.
    - OA2: número total de jerarquías de herencia, N_G.
    - OA3: media ponderada del número de clases.
    - OA4: desviación estándar del número ponderado de clases
    - OA5: media del número de dependencias directas de las clases.
    - OA6: desviación estándar del número de dependencias directas de las clases.
    - OA7: porcentaje de responsabilidades heredadas respecto al número total de ellas.

Métricas para sistemas orientados a objetos (IX)

Métricas para diagramas UML (III)

- ***Métricas de Genero et al.* (I)** [Genero et al. 2000]
  - Definidas para relaciones en diagramas de clases
  - Se dividen en tres categorías:
    - Métricas de asociación
    - Métricas de agregación
    - Métricas de dependencia
  - **Métricas de asociación**:
    - ***Número de Asociaciones de una Clase (NAC):*** se define como el número total de asociaciones que tiene la clase en un diagrama de clases
    - ***Número de Asociaciones en un Paquete (NAP):*** se define como el número total de asociaciones dentro del paquete
    - ***Número de Asociaciones vs. Clases en un Paquete (NAVCP):*** es definido como el ratio entre el número de asociaciones de un paquete (*NAP*) dividido por el número de clases en el paquete
  - **Métricas de dependencia**:
    - ***Número de Dependencias In (NdepIn):*** se define como el número de clases que son dependientes de la clase dada
    - ***Número de Dependencias Out (NDepOut):*** se define como el número de clases de las que depende la clase dada

Métricas para sistemas orientados a objetos (X)

Métricas para diagramas UML (IV)

- ***Métricas de Genero et al.* (II)**
  - **Métricas de agregación**:
    - ***Altura de Agregación.(HAgg):*** la altura de una clase en una jerarquía de agregación se define como la ruta máxima desde la clase raíz hasta las hojas.
    - ***Número de Partes Directas (NODP):*** se define como el número total de clases - “partes directas” las cuales componen una clase compuesta.
    - ***Número de Partes (NP):*** se define como el número de clases – “parte” (directas e indirectas) de una clase completa. Si se compara con la jerarquía de generalización se podría expresar como el número de descendientes de la clase.
    - ***Número de agregados (NW):*** se define como el número de clases “agregados” (directas e indirectas) de una clase “parte”. Si se compara con la jerarquía de generalización, sería el número de predecesores de la clase medida .
    - ***Agregación Múltiple (MAgg):*** se define como el número de clases “agregados directos” que tiene una clase en una jerarquía de agregación.
    - ***Número de Relaciones de Agregación (NAggR):*** se define como el número de relaciones de agregación existentes en el paquete

Bibliografía

Albrecht, A.J., *Measuring application development*, Proc. of IBM applications Development Joint SHARE/GUIDE Symposium, Monterrey, CA, 83-92, 1979.

Basili, V.R. y Weiss, D., *A Methodology for collecting valid software engineering data*, IEEE Transaction on Software Engineering,10(6), 728-38 1984.

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.

Belady, L.A., and Lehman, M.M., *A Model of Large Program Development*, IBM Systems Journal, 15 (3), pp. 225-252, 1976.

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

Brito e Abreu, F. y Melo, W., *Evaluating the Impact of Object-Oriented Design on Software Quality*, In Proceedings of the 3 rd International Software Metrics Symposium, 90-99, 1996.

Chidamber, S.R. y Kemerer, C.F., *A metrics suite for object-oriented design* ,IEEE Trans. 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.

DeMarco, T., *Structured Analysis and Sistem Specification*, Yourdon Press, 1978.

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.

Fenton, N.E. y Pfleeger, S.L., *Software metrics. A rigorous & practical approach* , 1997.

Genero, M., Piattini, M. y Calero, C*. Early Measures For UML class diagrams.* L´Objet*.* 6(4), Hermes Science Publications, 489-515, 2000.

Halstead, M.H., *Elements of Software Science*, Elsevier North-Holland, 1977.

IEEE *Software Engineering Standards*,. Standard 610.12-1990, 1993.

IEEE *Software Engineering Standards*,. Standard 610.12-1990, 1993.

ISO/IEC 9126-1:2001, "*Software engineering -- Product quality -- Part 1: Quality model*, 2001.

ISO/IEC TR 9126-2:2003, "*Software engineering -- Product quality -- Part 2: External metrics*", 2003 .

ISO/IEC TR 9126-3:2003, "*Software engineering -- Product quality -- Part 3: Internal metrics*", 2003.

ISO/IEC TR 9126-4:2004, "*Software engineering -- Product quality -- Part 4: Quality in use metrics*", 2004.

Bibliografía

ISO/IEC 14143-1 *Information Technology - Software Measurement - Functional Size Measurement - Part 1: Definition of Concepts*, 2003.

ISO/IEC 19761:2003 *Software engineering -- COSMIC-FFP -- A functional size measurement method*, 2003.

ISO/IEC 20926:2003 *Software engineering -- IFPUG 4.1 Unadjusted functional size measurement method -- Counting practices manual*, 2003.

ISO/IEC IS 24570 *Software Engineering - NESMA functional size measurement method version 2.1 - Definitions and counting guidelines for the application of Function Point Analysis*, 2003.

Karner, G., *Resource Estimation for Objectory Projects*. Objective Systems SF AB, 1993.

Kitchenham, B., Pfleeger, S.L., and Fenton, N.E., Towards a Framework for Software Measurement Validation, IEEE Transactions on Software Engineering, 21 (12), 929-944, 1995.

Lorenz, M. and Kidd, J., *Object_oriented Software Metrics,* Prentice Hall 1994.

Marchesi M., *OOA Metrics for the Unified Modeling Language*, Proceedings of the 2nd Euromicro Conference on Software maintenance an reengineering, p. 67-73, 1998.

McConnell, S., *Desarrollo y gestión de proyectos informáticos,* Mc Graw Hill 1997.

McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean,J. And Hall, F. Practical Software Measurement, Addison-Wesley, 2002.

Nielsen, J. Usability Metrics: Tracking Interface Improvements, IEEE Software, 13 (6), 12-13, 1996.

Paulk, M. et al., *Capability Maturity Model for Software*, Software Engineering Institute, Carnie Mellon University, Pittsburgh, P.A., 1993.

Pressman, R.S., *Ingeniería del Software, un enfoque práctico*, Mc Graw Hill, 1998.

Porter, A.A. and Selby, R.W.*Empirically Guided Software Development Using Metric-Based Classification Trees*.*IEEE Software* ,7(2),pp.46-54,March 1990.

Ragland, B., *Measure, Metric or Indicator: What´s the Difference?*, Crosstalk, 8 (3), 29-30, 1995.

Symons, C.R., *Software sizing and estimating: Mk II FPA (function point analysis)*, John Wiley & Sons, 1991

Tuya, J., Ramos, I. y Dolado, J. (eds.) "Técnicas Cuantitativas para la Gestión en Ingeniería del Software", Netbiblo, S.L., 2007.

Vázquez, P.J., Moreno, M.N. y García, F.J., *Métricas Orientadas a Objetos*, Informe técnico DPTOIA-IT-2001-002, noviembre, 2001.