TEMA 7. Métricas de Producto

Calidad del Software

Dr. José Luis Abellán Miguel

Grado en Ingeniería Informática

Dr. José Luis Abellán Miguel - Tlf: (+34) 968 278 657
Universidad Católica San Antonio de Murcia www.ucam.edu

Índice

Bibliografía

- Pressman, R. ***Ingeniería del Software: Un enfoque práctico***. 7ª edición. Madrid: McGraw Hill, 2010. ISBN: 9701054733 (disponible en la biblioteca UCAM) → **Capítulo 23**

Introducción (1/3)

- La ingeniería es una disciplina cuantitativa
  - **Medición**: elemento clave de cualquier proceso de ingeniería.
    - ¡Cuidado!: veremos que Medición != Métrica != Indicador
- Las mediciones y métricas del software con frecuencia son indirectas, están abiertas a debate
  - *Algunos miembros de la comunidad del software continúan argumentando que el software es "inmensurable" o que los intentos por medir deben posponerse hasta comprender mejor el software y los atributos que deben usarse para describirlo.*

Introducción (2/3)

- Las métricas de producto **ayudan** a los ingenieros de software a obtener comprensión acerca del diseño y la construcción del software que elaboran
  - Enfocarse en atributos mensurables específicos de los productos de trabajo de la ingeniería del software
- Las métricas de producto proporcionan una base desde donde el **análisis, el diseño, la codificación y las pruebas** pueden realizarse de manera más objetiva y valorarse de modo más cuantitativo
- **¿Quién la hace?** Los ingenieros de software usan métricas de proyecto para auxiliarse en la construcción de software de mayor calidad

Introducción (3/3)

- **¿Pasos para las métricas de producto?**
  1. Derivar las mediciones y métricas del software que sean adecuadas para la presentación del software que se está construyendo
  2. Recolectan los datos requeridos para derivar las métricas formuladas
  3. Las métricas adecuadas se analizan con base en políticas y reglas preestablecidas y en datos anteriores
  4. Los resultados del análisis se interpretan para obtener comprensión acerca de la calidad del software
  5. Los resultados de la interpretación conducen a modificación de requerimientos y modelos de diseño, código fuente o casos de prueba

Marco Conceptual para las Métricas

- **Medida!=Métrica!=Indicador**
  - **Medida**: indicio cuantitativo de la extensión, cantidad, dimensión, capacidad o tamaño de algún atributo de un producto o proceso
    - E.g.: Número de errores dentro de un diagrama UML
    - **Medición**: acto de determinar una medida
  - **Métrica**: "una medida cuantitativa del grado en el que un sistema, componente o proceso posee un atributo determinado"
    - Relaciona en alguna forma las medidas individuales:
    - E.g.: Número promedio de errores que se encuentran por revisión o tasa de fallos, tiempo medio entre fallas, etc.
  - **Indicador:** métrica o combinación de métricas que proporcionan comprensión acerca del proceso de software, el proyecto de software o el producto en sí.
    - Si la tasa de fallos de las pruebas > 0.2 se revisará el producto SW

Marco Conceptual para las Métricas

❑ Principios de medición

1. **Formulación**. La derivación de medidas y métricas de software apropiadas para la representación del software que se esta construyendo.
2. **Recolección**. Mecanismo que se usa para acumular datos requeridos para derivar las métricas formuladas.
3. **Análisis**. El cálculo de métricas y la aplicación de herramientas matemáticas.
4. **Interpretación**. Evaluación de las métricas resultantes para comprender la calidad de la representación.
5. **Retroalimentación**. Recomendaciones derivadas de la interpretación de las métricas del producto, transmitidas al equipo de software.

Marco Conceptual para las Métricas

- **Principios de medición (**Lethbridge'03**)**
  - **Una métrica debe tener propiedades matemáticas deseables**: el valor de la métrica debe estar en un rango significativo
    - E.g.: de 0 a 1, donde 0 realmente significa ausencia, 1 indica el valor máximo y 0.5 representa el "punto medio".
  - **Cuando una métrica representa una característica de software que aumenta cuando ocurren rasgos positivos o que disminuye cuando se encuentran rasgos indeseables, el valor de la métrica debe aumentar o disminuir en la misma forma**
  - Cada métrica debe **validarse de manera empírica** en una gran **variedad de contextos** antes de publicarse o utilizarse para tomar decisiones.
    - Una métrica debe medir el factor de interés, independientemente de otros factores.
    - Debe "escalar" a sistemas mas grandes y funcionar en varios lenguajes de programación y dominios de sistema.

Atributos deseables de las Métricas (1/2)

- **Simple y calculable**. Debe ser relativamente fácil aprender como derivar la métrica y su cálculo no debe demandar esfuerzo o tiempo excesivo.
- **Empírica e intuitivamente convincente**. Debe satisfacer las nociones intuitivas del ingeniero acerca del atributo de producto que se elabora (por ejemplo, una métrica que mide la cohesión del modulo debe aumentar en valor conforme aumenta el nivel de cohesión).
- **Congruente y objetiva**. Siempre debe producir resultados que no tengan ambigüedades.
  - Una tercera parte independiente debe poder derivar el mismo valor de métrica usando la misma información acerca del software.

Atributos deseables de las Métricas (2/2)

- **Constante en su uso de unidades y dimensiones**. El cálculo matemático de la métrica debe usar medidas que no conduzcan a combinaciones extrañas de unidades.
  - E.g.: multiplicar personas en los equipos de proyecto por variables de lenguaje de programación en el programa da como resultado una mezcla sospechosa de unidades que no son intuitivamente convincentes.
- **Independiente del lenguaje de programación**. Debe basarse en el modelo de requerimientos, el modelo de diseño o la estructura del programa en sí. No debe depender de los caprichos de la sintaxis o de la semántica del lenguaje de programación.
- **Un mecanismo efectivo para retroalimentación de alta calidad**. Debe proporcionar información que pueda conducir a un producto final de mayor calidad.

Métricas para el Modelo de Requisitos

Métrica de punto de función (PF) (1/6)

- Medio para medir la funcionalidad que entra a un sistema utilizando datos históricos (mediciones de productos SW ya desarrollados)
- Sirve para:
  - Estimar el costo o esfuerzo requerido para diseñar, codificar y probar el software
  - Predecir el numero de errores que se encontrarán durante las pruebas
  - Prever el número de componentes y/o de líneas fuente proyectadas en el sistema implementado

Métricas para el Modelo de Requisitos

Métrica de punto de función (PF) (2/6)

- Los puntos de función se derivan usando una relación empírica basada en medidas contables del **dominio de información del software** y en valoraciones cualitativas de la complejidad del software
- Valores de dominio de información (1/2):
  - **Número de entradas externas (EE).** Cada *entrada externa* se origina por un usuario o se transmite desde otra aplicación, y proporciona distintos datos orientados a aplicación o información de control.
  - **Número de salidas externas (SE).** Cada *salida externa* es un dato o conjunto de datos derivados dentro de la aplicación que ofrecen información al usuario. En este contexto, salida externa se refiere a reportes, pantallas, mensajes de error, etc.

Métricas para el Modelo de Requisitos

Métrica de punto de función (PF) (3/6)

- Valores de dominio de información (2/2):
  - **Número de consultas externas (CE).** Una consulta externa se define como una entrada que da como resultado la generación de alguna respuesta de software inmediata
  - **Número de archivos lógicos internos (ALI)**. Cada archivo lógico interno es un **agrupamiento lógico de datos** que reside **dentro** de la frontera de la aplicación y se mantiene mediante entradas externas
  - **Número de archivos de interfaz externos (AIE).** Cada archivo de interfaz externo es un **agrupamiento lógico de datos** que reside **fuera** de la aplicación, pero que proporciona información que puede usar la aplicación

Métricas para el Modelo de Requisitos

Métrica de punto de función (PF) (4/6)

- Cálculo de puntos de función (1/2)

Valor de complejidad suele ser un valor subjetivo y depende de la experiencia

Valor de dominio de información	Conteo		Factor ponderado: Simple	Promedio	Complejo	
Entradas externas (EE)		×	3	4	6	=
Salidas externas (SE)		×	4	5	7	=
Consultas externas (CE)		×	3	4	6	=
Archivos lógicos internos (ALI)		×	7	10	15	=
Archivos de interfaz externos (AIE)		×	5	7	10	=
Conteo total →						

$$PF = \text{conteo total} \times [0.65 + 0.01 \times \Sigma (F_i)]$$

Métricas para el Modelo de Requisitos

Métrica de punto de función (PF) (4/6)

- Cálculo de puntos de función (2/2)

$$PF = \text{conteo total} \times [0.65 + 0.01 \times \Sigma (F_i)]$$

  - Los Fi (i = 1 a 14) son **factores de ajuste de valor** (FAV) con base en respuestas a las siguientes preguntas:

- Preguntas para calcular el **FAV** (1/2): **Responder de 0 a 5**

1. ¿El sistema requiere respaldo y recuperación confiables?
2. ¿Se requieren comunicaciones de datos especializadas para transferir información hacia o desde la aplicación?
3. ¿Existen funciones de procesamiento distribuidas?
4. ¿El desempeño es crucial?
5. ¿El sistema correrá en un entorno operativo existente enormemente utilizado?

Métricas para el Modelo de Requisitos

Métrica de punto de función (PF) (4/6)

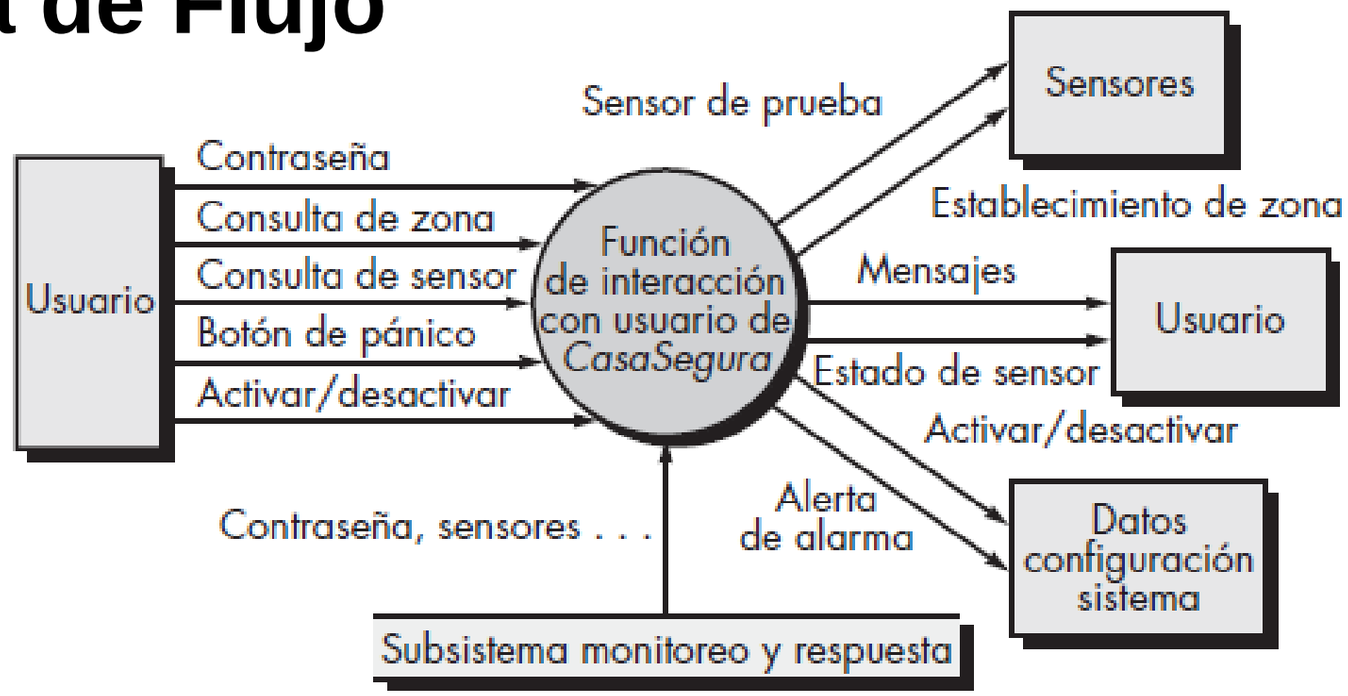
- Preguntas para calcular el **FAV** (2/2): **Responder de 0 a 5**

6. ¿El sistema requiere entrada de datos en línea?
7. ¿La entrada de datos en línea requiere que la transacción de entrada se construya sobre múltiples pantallas u operaciones?
8. ¿Los ALI se actualizan en línea?
9. ¿Las entradas, salidas, archivos o consultas son complejos?
10. ¿El procesamiento interno es complejo?
11. ¿El código se diseña para ser reutilizable?
12. ¿La conversión y la instalación se incluyen en el diseño?
13. ¿El sistema se diseña para instalaciones múltiples en diferentes organizaciones?
14. ¿La aplicación se diseña para facilitar el cambio y su uso por parte del usuario?

Métricas para el Modelo de Requisitos

Métrica de punto de función (PF) (5/6)

❑ Ejemplo: Diagrama de Flujo

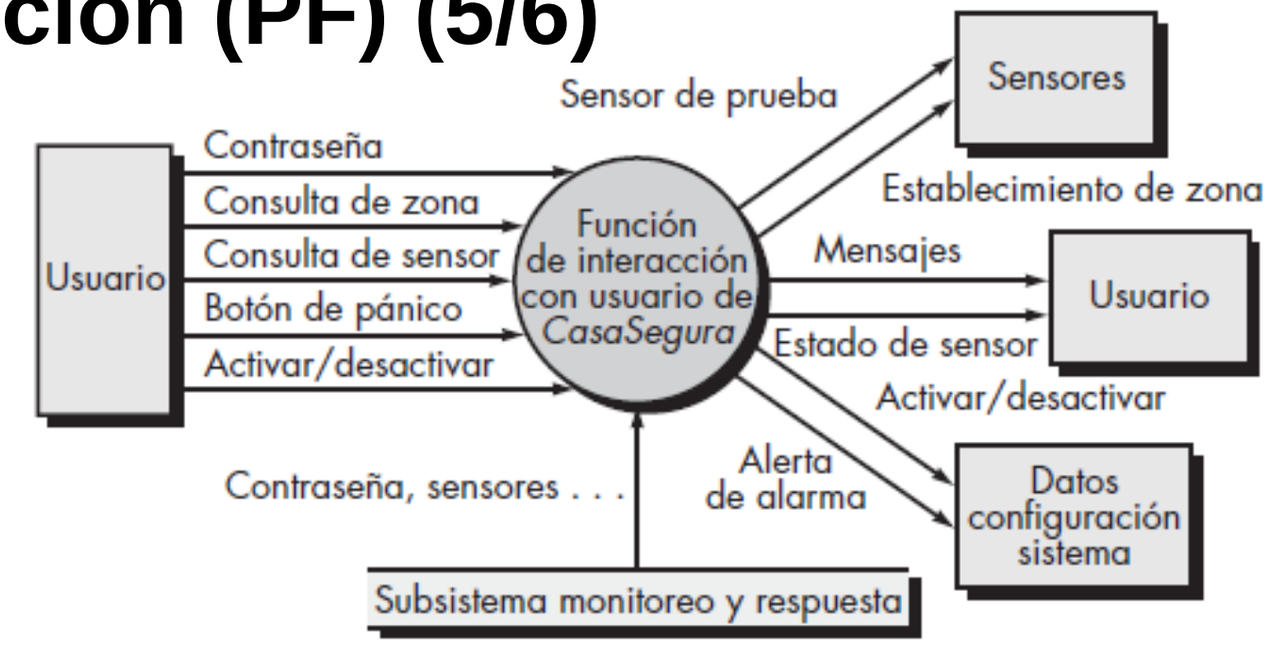


- 3 entradas externas (**contraseña**, **botón de pánico** y **activar/desactivar**)
- 2 salidas externas (**mensajes** y **estado de sensor**)
- 2 consultas externas (**consulta de zona** y **consulta de sensor**).
- 1 ALI (**subsistema monitoreo y respuesta**)
- 4 AIE (**sensor de prueba**, **establecimiento de zona**, **activar/desactivar** y **alerta de alarma**).

Métricas para el Modelo de Requisitos

Métrica de punto de función (PF) (5/6)

- Ejemplo:



Valor dominio de información	Conteo		Factor ponderado: Simple	Promedio	Complejo		
Entradas externas (EE)	3	×	3	4	6	=	9
Salidas externas (SE)	2	×	4	5	7	=	8
Consultas externas (CE)	2	×	3	4	6	=	6
Archivos lógicos internos (ALI)	1	×	7	10	15	=	7
Archivos de interfaz externos (AIE)	4	×	5	7	10	=	20
Conteo total →							50

$$PF = 50 \times [0.65 + (0.01 \times 46)] = 56$$

Métricas para el Modelo de Requisitos

Métrica de punto de función (PF) (6/6)

- Conclusiones
  - Del ejercicio anterior hemos estimado: PF=56
  - La empresa de SW podrá utilizar su histórico de proyectos realizados para conocer a qué equivale cada punto de función (**pf**):
    - **Líneas de código por cada pf** = 20
    - **Horas por cada pf** = 2,5h
    - **pfs al mes** = 20
  - **A continuación pueden realizar estimaciones:**
    - Duración_Estimada: 56/20 = 2,8 meses de trabajo
    - Líneas_de_Código: 20 x 56 = 1120
    - Tiempo_Total(horas) = 2,5 x 56 = 140 horas

Métricas para el Modelo de Diseño OO

- Conforme un modelo de diseño OO crece en tamaño y complejidad, una visión mas objetiva de las características del diseño puede beneficiar tanto al diseñador experimentado (quien adquiere comprensión adicional) como al principiante (quien obtiene un indicio de la calidad que de otro modo no tendría disponible)

Métricas para el Modelo de Diseño OO

- Características medibles en un diseño OO (Whitmire'97)
  - **Tamaño**:
    - **Población**: se mide al realizar un conteo estático de entidades OO, tales como clases u operaciones.
    - **Volumen** son idénticas a las medidas de población pero se recolectan de manera dinámica: en un instante de tiempo determinado.
    - **Longitud** es una medida de una cadena de elementos de diseño interconectados (e.g., profundidad de un árbol de herencia es una medida de longitud).
    - **Funcionalidad** : valor entregado al cliente por la aplicación OO

Métricas para el Modelo de Diseño OO

- Características medibles en un diseño OO (Whitmire'97) (1/2)
  - **Complejidad**: características estructurales al examinar cómo se relacionan mutuamente las clases de un diseño OO
  - **Acoplamiento.** Las conexiones físicas entre elementos del diseño OO (por ejemplo, el numero de colaboraciones entre clases o el de mensajes que pasan entre los objetos) representan el acoplamiento dentro de un sistema OO.
  - **Suficiencia**. Un componente de diseño (e.g.: clase) es suficiente si refleja por completo todas las propiedades del objeto de dominio de aplicación que se modela
  - **Completitud**. Más general que la suficiencia que sólo se centra en la aplicación actual que se desarrolla.
  - **Cohesión**. Un componente OO debe diseñarse de manera que tenga todas las operaciones funcionando en conjunto para lograr un solo propósito bien definido

Métricas para el Modelo de Diseño OO

- Características medibles en un diseño OO (Whitmire'97) (2/2)
  - **Primitivismo**: grado en el que una operación es atómica
    - La operación no puede construirse a partir de una secuencia de otras operaciones contenidas dentro de una clase (clase con alto grado de primitivismo encapsula sólo operaciones primitivas
  - **Similitud.** El grado en el que dos o mas clases son similares en términos de su estructura, función, comportamiento o propósito se indica mediante esta medida.
  - **Volatilidad.** De un componente de diseño OO mide la probabilidad de que ocurrirá un cambio.

Métricas para Código Fuente

- Halstead'77 asignó leyes cuantitativas al desarrollo de software usando un conjunto de medidas primitivas que pueden derivarse después de generar el código o de que el diseño este completo

n_1 = número de operadores distintos que aparecen en un programa
n_2 = número de operandos distintos que aparecen en un programa
N_1 = número total de ocurrencias de operador
N_2 = número total de ocurrencias de operando

- Longitud de programa: $N = n_1 \log_2 n_1 + n_2 \log_2 n_2$
- Volumen de programa (bits): $V = N \log_2 (n_1 + n_2)$

Métricas para Pruebas OO (1/2)

- *Las métricas pueden ayudar a dirigir los recursos de prueba en hebras, escenarios y paquetes de clases que son "sospechosas" con base en las características medidas.*

1. **Falta de cohesión en métodos (FCOM).** Mientras más alto sea el valor de la FCOM más estados deben ponerse a prueba para garantizar que los métodos no generan efectos colaterales
2. **Porcentaje público y protegido (PPP).** Indica el porcentaje de los atributos de clase que son públicos o protegidos.
   - Valores altos de PPP aumentan la probabilidad de efectos colaterales entre las clases porque los atributos públicos y protegidos conducen a alto potencial para acoplamiento.
   - Las pruebas deben diseñarse para garantizar el descubrimiento de tales efectos colaterales.

Métricas para Pruebas OO (2/2)

3. **Acceso público a miembros de datos (APD).** Número de clases (o métodos) que pueden acceder a otros atributos de clase (violación de la encapsulación). Valores altos de APD conducen al potencial de efectos colaterales entre clases.
   - Las pruebas deben diseñarse para garantizar el descubrimiento de tales efectos colaterales.
4. **Número de clases raíz (NCR).** Conteo de las distintas jerarquías de clase que se describen en el modelo de diseño. Deben desarrollarse las suites de prueba para cada clase raíz y la correspondiente jerarquía de clase.
   - Conforme el NCR aumenta, también aumenta el esfuerzo de prueba.
5. ***Fan-in* (FIN).** Cuando se usa en el contexto OO, el *fan-in* (abanico de entrada) en la jerarquía de herencia es un indicio de herencia múltiple.
   - FIN > 1 indica que una clase hereda sus atributos y operaciones de mas de una clase raíz.
   - FIN > 1 debe evitarse cuando sea posible.
6. **Número de hijos (NDH) y profundidad del árbol de herencia (PAH).** Los métodos de superclase tendrán que volverse a probar para cada subclase

Métricas para Mantenimiento

- IEEE Std. 982.1-1988 [IEE93] sugiere un *índice de madurez de software* (IMS) que proporcione un indicio de la estabilidad de un producto de software (con base en cambios que ocurran para cada liberación del producto)

M_T = número de módulos en la liberación actual
F_c = número de módulos en la liberación actual que cambiaron
F_a = número de módulos en la liberación actual que se agregaron
F_d = número de módulos de la liberación anterior que se borraron en la liberación actual

$$\text{IMS} = \frac{M_T - (F_a + F_c + F_d)}{M_T}$$

- Conforme el IMS tiende a 1.0, el producto comienza a estabilizarse