Universidad Nacional de Tucumán

Universidad Nacional de Tucumán
Facultad de Ciencias Exactas y Tecnología

Ingeniería de Software II

Mediciones del Software

Mg. Héctor A. Valdecantos

Introducción

“Cuando puedes medir lo que estás hablando y expresarlo en números, sabes algo al respecto; pero cuando no puedes medir, cuando no lo puedes expresar en números, tu conocimiento es escaso e insatisfactorio”

[Lord Kelvin, Scottish Physicist – 1884]

Introducción

“Si no lo puedes medir, probablemente no lo puedas entender, y seguramente no lo puedas controlar”

[(cita adaptada) Lord Kelvin, Scottish Physicist – 1884]

¿Para qué medimos?

Conceptos fundamentales: términos

- Medida
  - Es un dato crudo
  - Una forma de averiguar o tasar un valor comparándolo con una norma [1]
  - Proporcionan una base para generar métricas, no se evalúan ni se comparan con ningún estándar o punto de referencia.
- Métrica
  - Función que tiene “medidas” como argumentos, conlleva una interpretación de la/las medida/medidas
  - Una cuantificación del grado en que un sistema, componente, o proceso posee un cierto atributo [2]
- Indicador
  - Una métrica o combinación de métricas que provee visión o percepción de un proceso, producto, proyecto, etc.

[1] IEEE-STD-1061-1998 Standard for Software Quality Metrics Methodology
[2] IEEE-STD-610.12-1990 Glossary of Software Engineering Terminology

[Pressman, Ingenieria de software, un enfoque practico. 7th edicion, p. 527]

Conceptos fundamentales: tipo de escala

- Nominal
  - Diferentes categorías. Ejemplo: tipos de defectos
- Ordinal
  - Categorías con orden. Ej: nivel madurez en CMMI, severidad de defectos, escala Likert.
- Intervalo
  - Numérico con escala relativa. No tiene un “cero genuino”. Ej: temperatura 0°C = 32°F
- Proporción
  - Útil para hacer comparaciones.
  - Exposición al riesgo vs proporción cruda
  - $$\text{densidad de defectos} = \frac{Num.\,de\,defectos}{Oportunidad\,de\,error}$$

Conceptos fundamentales: observación

Concepto

Definición

Definición operacional

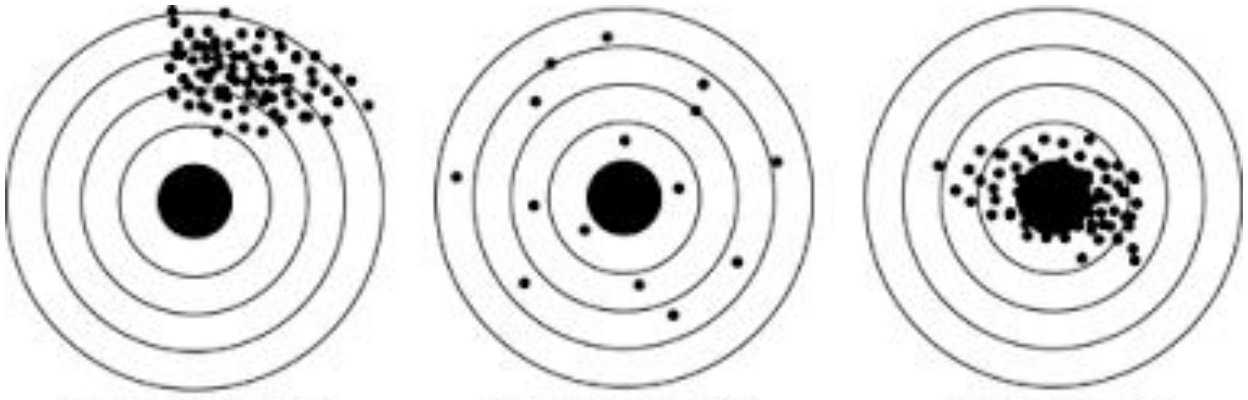
Mediciones

- Concepto que se quiere medir. Por ejemplo “cycle time” (o duración de un ciclo)
- Se necesita una definición: “tiempo transcurrido para realizar una tarea”
- Detalles procedimentales sobre cómo se obtiene la medición: ej: “tiempo entre la aprobación de requerimientos y la liberación del producto”

[Van Solingen, R., Basili, V., Caldiera, G., & Rombach, H. D. (2002). Goal question metric (GQM) approach. Encyclopedia of software engineering.]

Conceptos fundamentales: confiabilidad y validez

- Confiabilidad y validez



- Confiabilidad: mediciones consistentes usando el mismo método en el mismo objeto.
- Validez: si la medida o métrica revela algo del concepto o atributo de calidad que se quiere entender.

[Kan, S. H. (2002). Metrics and models in software quality engineering. Addison-Wesley Longman Publishing Co., Inc., página 77]

Medición del Software

- Medidas directas
  - Eje: la longitud de un tornillo
  - Del proceso: costo, esfuerzo, duración,
  - Del producto: SLOC, velocidad de ejecución, tamaño de memoria, defectos observados en un período
  - En general son fáciles de obtener
- Medidas indirectas
  - Eje: la calidad de un tornillo (% de tornillos rechazados)
  - Del producto: funcionalidad, calidad, complejidad, eficiencia, mantenibilidad, confiabilidad, etc
  - En general son más difíciles de obtener

[Pressman, Ingenieria de software, un enfoque practico. 7th edicion, p. 575]

Métricas del Software

- Medidas en el mundo físico:
  - Medidas directas: longitud de un tornillo.
  - Medidas indirectas: calidad de los tornillos, medida por el porcentaje de rechazados. (SW -ilidades)
- Ej. de medidas directas sobre proyectos de software:

Proyecto	esfuerzo	costo	pag. doc.	defectos	fallas	pers
alpha	24	168000	365	134	29	3
beta	62	440000	1224	321	86	5
gamma	43	314000	1050	256	64	6

[Pressman, R. S. (2009). Software engineering: a practitioner's approach. Palgrave Macmillan., p. 672]

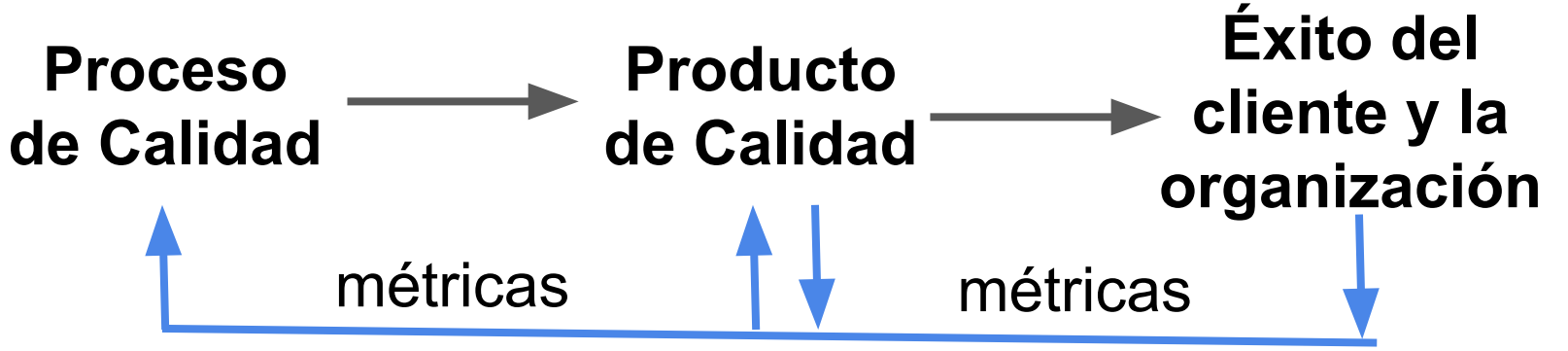
Métricas del Software

- Métricas del producto
  - Controlar y entender características del producto
  - Ej: tamaño, complejidad, performance, calidad, etc
- Métricas del proceso (propósitos estratégicos)
  - Usadas para mejor el proceso de desarrollo
  - Ej: eficiencia en la remoción de defectos (DRE), tiempo de respuesta para corregir un proceso, etc
- Métricas del proyecto (propósitos tácticos)
  - Parámetros del proyecto (o recursos)
  - Ej: tamaño equipo, costo, schedule, cycletime, etc.

[pressman, 7ed, p. ~~671~~ 584]

¿Por qué medimos el Software?

- Indicar la calidad del producto, proceso, y entender el uso de los recursos del proyecto
- Evaluar la productividad de la gente que lo desarrolla
- Evaluar los beneficios al usar métodos y herramientas
- Establecer líneas bases para la estimación y comparación
- Determinar el estado actual respecto al plan
- Justificar el uso de nuevas herramientas o de capacitación



"the quality of a product is highly influenced by the quality of the processes used to produce it" [Humphrey 1989].

Tamaño del Software

- Longitud: tamaño físico del producto
- Funcionalidad: funciones que provee al usuario
- Complejidad computacional o del problema
  - Algorítmica: mide la eficiencia del software
  - Estructural: mide la estructura del software
  - Cognitiva: mide el esfuerzo requerido para entender el software (comprensibilidad)
- Reuso: uso de partes existentes.

[apuntes cátedra]

Medidas del tamaño del Software

- Mediciones candidatas:
  - AbcMetric [Jerry Fitzpatrick, 1997]
  - Líneas de código fuente (SLOC)
  - Puntos de función (FP) [Albrecht, 1970]
  - Tablas en la base de datos
  - Cantidad total de usuarios
  - Tamaño del binario ejecutable
  - Cantidad de interfaces gráfica o “pantallas”
  - Tamaño de la suite de test
  - Casos de usos (Use Cases) | Total story points
  - Número de archivos
  - Total páginas de documentación

[adaptada de http://wiki.c2.com/?SystemSizeMetrics]

Medidas del tamaño del Software

- Mediciones candidatas:
  - AbcMetric [Jerry Fitzpatrick, 1997]
  - **Líneas de código fuente (SLOC)**
  - **Puntos de función (FP)** [Albrecht, 1970]
  - Tablas en la base de datos
  - Cantidad total de usuarios
  - Tamaño del binario ejecutable
  - Cantidad de interfaces gráfica o “pantallas”
  - Tamaño de la suite de test
  - **Casos de usos (Use Cases) | Total story points**
  - Número de archivos
  - Total páginas de documentación

[adaptada de http://wiki.c2.com/?SystemSizeMetrics]

Source Lines Of Code

- Cantidad de líneas físicas de código fuente (SLOC)
  - Método simple y objetivo
  - Contar líneas de código puede resultar ambiguo:
    - Comentarios, líneas en blanco, sentencias largas en dos líneas, etc. (definición operacional)
  - Con convenciones de código es más confiable
  - Depende del lenguaje y estilo de programación
  - Es el método más viejo de medir el tamaño:
    - Resulta confiable si usamos assembly
    - Traducir y luego contar (lenguajes interpretados?)
  - El conteo no puede empezar hasta la implementación
  - La paradoja SLOC con respecto a la productividad

Source Lines Of Code

- Cantidad de líneas de código fuente (SLOC)

```
int maximo(int vector[], int n) {
  int max = vector[0];
  for(int i=1; i<n; i++){
    if(vector[i] > max)
      max = vector[i];
  }
  return max;
}
```

8 SLOC

```
int maximo(int vector[], int n)
{
  int max = vector[0];
  int i = 1;
  while (i < n)
  {
    if(vector[i] > max)
    {
      max = vector[i];
    }
    i = i + 1;
  }
  return max;
}
```

14 SLOC

Source Lines Of Code: normalización

- Métricas orientadas al tamaño usando LOC

Proyecto	LOC	esfuerzo	$ costo	pag. doc.	defectos	fallas	personas
alpha	12100	24	168000	365	134	29	3
beta	27200	62	440000	1224	321	86	5
gamma	20200	43	314000	1050	256	64	6

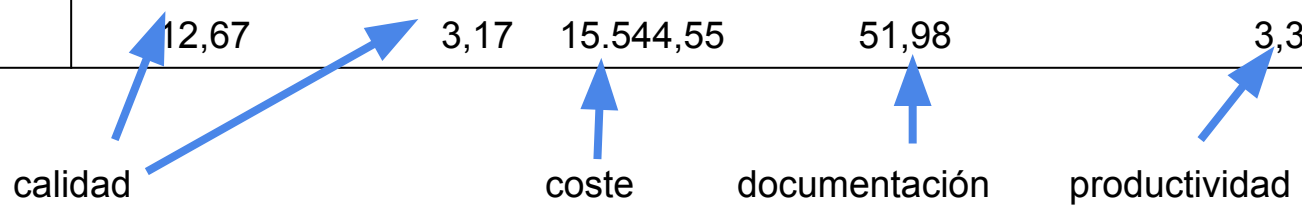
Proyecto	def/KLOC	fallas/KLOC	$/KLOC	Page/KLOC	KLOC/persona-mes
alpha	11,07	2,40	13.884,30	30,17	4,03
beta	11,80	3,16	16.176,47	45,00	5,44
gamma	12,67	3,17	15.544,55	51,98	3,37

Source Lines Of Code: normalización

- Métricas orientadas al tamaño usando LOC

Proyecto	LOC	esfuerzo	$ costo	pag. doc.	defectos	fallas	personas
alpha	12100	24	168000	365	134	29	3
beta	27200	62	440000	1224	321	86	5
gamma	20200	43	314000	1050	256	64	6

Proyecto	def/KLOC	fallas/KLOC	$/KLOC	Page/KLOC	KLOC/persona-mes
alpha	11,07	2,40	13.884,30	30,17	4,033
beta	11,80	3,16	16.176,47	45,00	5,440
gamma	12,67	3,17	15.544,55	51,98	3,367



Método del punto de función (FP)

Métrica propuesta por Alan Albrecht en 1979. La métrica de Punto de Función puede ser usada para:

- Estimar el costo o esfuerzo para el diseño, código, test
- Predecir el número de errores durante el testeo
- Pronosticar el número de componentes y/o las SLOC

Miden la cantidad de funcionalidad de un sistema descripto en una especificación

Método del punto de función

Para calcular FP se tiene en cuenta la siguiente información

- **Número de entradas externas:** entradas externas originadas por los usuarios o transmitidas desde otra aplicación. **EE**
- **Número de salidas externas**: salidas externas que la aplicación provee al usuario. Reportes, pantallas, mensajes de error, etc. **SE**
- **Número de Peticiones externas**: se define como una entrada online que resulta de una respuesta inmediata del SW en forma de una salida online (usualmente desde un archivo lógico interno) **PE**
- **Números de Archivos lógicos internos**: agrupación lógica de datos que es mantenida vía entradas externas. (puede ser una DB) **ALI**
- **Número de archivos de Interfaces externas**: interfaces que la aplicación entiende y usa para transmitir información a otros sistemas. A**IE**

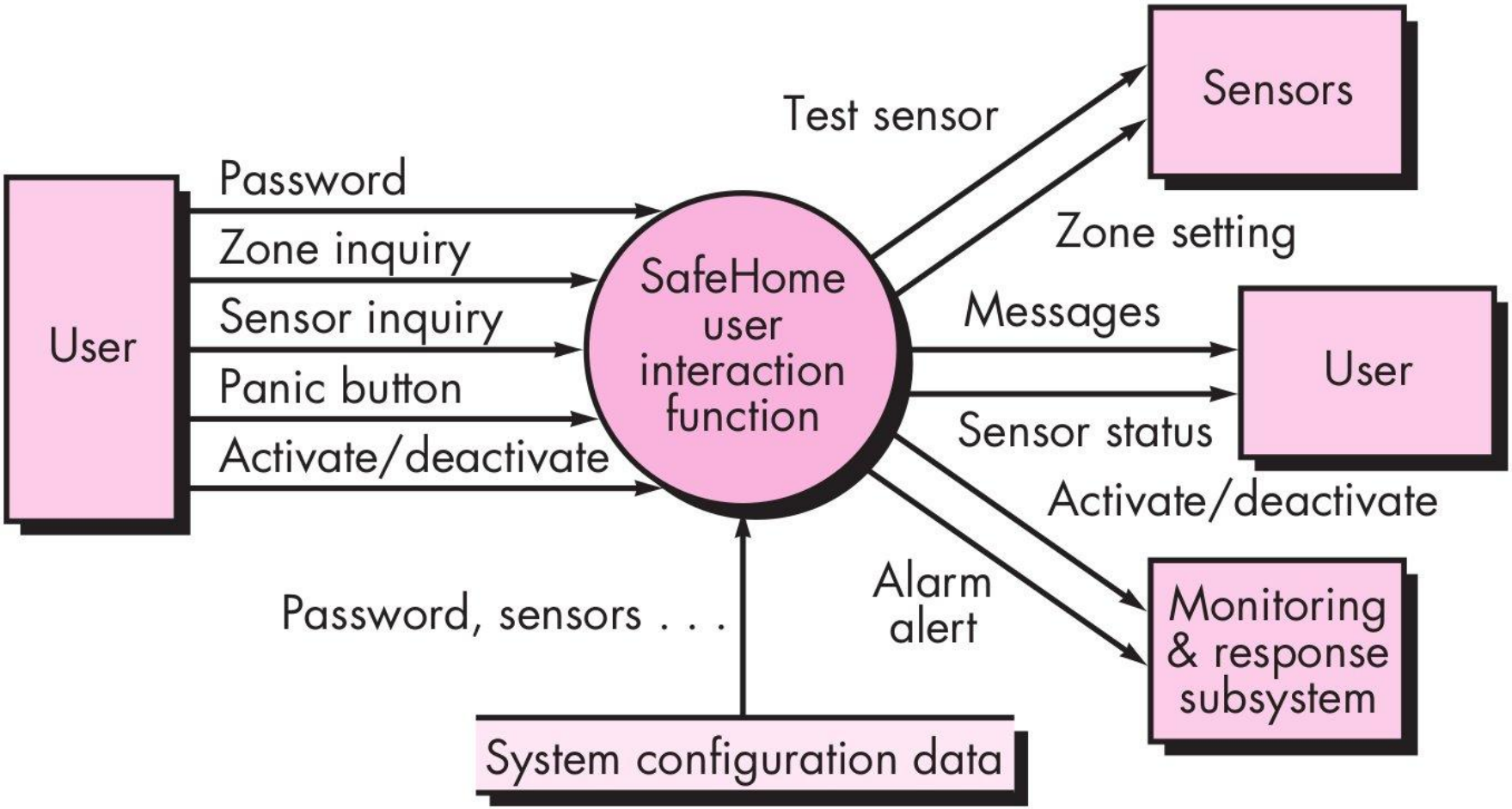
Método del punto de función

Una vez levantados los datos se llena la tabla siguiente:

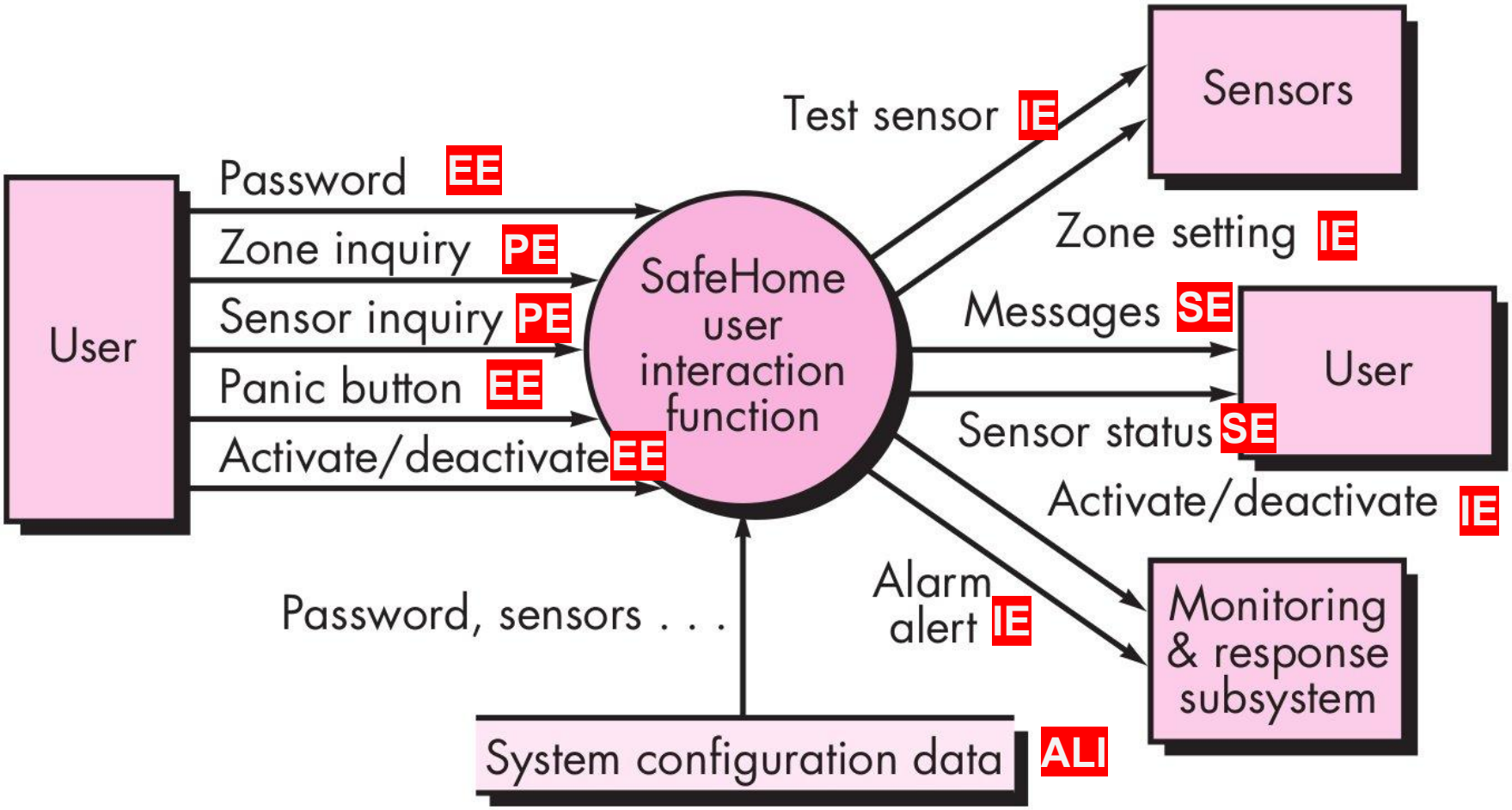
		factor de peso			
Parámetro de medida	Cuenta	Simple	Medio	Complejo	
Entradas	…..	3	4	6	…..
Salidas	…..	4	5	7	…..
Peticiones	…..	3	4	6	…..
Archivos	…..	7	10	15	…..
Interfaces	…..	4	7	10	…..
				total =	…..

[Pressman, Ingenieria de software, un enfoque practico. 7th edicion, p. 532]

Método del punto de función



Método del punto de función



Método del punto de función

Una vez levantados los datos se llena la tabla siguiente:

		factor de peso			
Parámetro de medida	Cuenta	Simple	Medio	Complejo	
Entradas (EE)	3	**3**	~~4~~	~~6~~	9
Salidas (SE)	2	**4**	~~5~~	~~7~~	8
Peticiones (PE)	2	**3**	~~4~~	~~6~~	6
Archivos (ALI)	1	**7**	~~10~~	~~15~~	7
Interfaces (IE)	4	**4**	~~7~~	~~10~~	16
				total =	50

Método del punto de función

$$PF = total * \left[0.65 + 0.01 * \sum_{1}^{14}(F_i)\right]$$

$$PF = 50 * \left[0.65 + 0.01 * \sum_{1}^{14}(F_i)\right]$$

Método del punto de función

Factores de ajuste de valor basados en respuestas de escala Likert

1. El sistema requiere de backup y recuperación confiable?
2. Se necesita de comunicación de datos especializada?
3. El sistema tiene funciones distribuidas?
4. Es la performance crítica?
5. El sistema correrá en un ambiente operacional intensivo?
6. Se requiere de entrada de datos en línea?
7. La entrada de datos online requiere de transacciones entre múltiples pantallas u operaciones?
8. Se necesitan actualizaciones on-line?
9. Son las entradas, salidas, archivos, o peticiones complejas?
10. El procesamiento interno es complejo?
11. Es el código diseñado para ser reutilizado?
12. Es el sistema de fácil instalación?
13. Es un sistema con múltiples instalaciones? Múltiples sites?
14. Es una aplicación de fácil uso y fácil de cambiar?

Método del punto de función

$$PF = total * \left[0.65 + 0.01 * \sum_{1}^{14}(F_i)\right]$$

$$PF = 50 * \left[0.65 + 0.01 * \sum_{1}^{14}(F_i)\right]$$

$$PF = 50 * [0.65 + 0.01 * 46]$$

$$PF = 56$$

Método del punto de función

- Medida indirecta basada en información del dominio y complejidad del software.
- Independiente del lenguaje de programación
- La medición comienza antes de la implementación
- Más subjetiva, no tiene un significado físico directo
- Aborda el problema de SLOC respecto a las líneas contadas y productividad
- Más difícil, más costosa, de implementar que SLOC
- Tan controversial como SLOC

Function Points: normalización

- Métricas orientadas a la función

Proyecto	FP	esfuerzo	costo	pag. doc.	defectos	fallas	gente
alpha	52	24	168000	365	134	29	3
beta	122	62	440000	1224	321	86	5
gamma	96	43	314000	1050	256	64	6

Proyecto	Def/FP	Fallas/FP	$/FP	Page/FP	FP/persona-mes
alpha	2,58	0,56	3.230,77	7,02	17,33
beta	2,63	0,70	3.606,56	10,03	24,40
gamma	2,67	0,67	3.270,83	10,94	16,00

Medidas de tamaño del Software

- Conversión FP a LOC

Una LOC en C++ provee ~2.4 más funcionalidad que una LOC en C.

Smalltalk ~6.2

- Problema para medir productividad con LOC

	LOC por FP			
Prog. Lang	**Prom**	**complejo**	**medio**	**simple**
Ada	154	—	104	205
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
COBOL	77	77	14	400
FORTRAN	100	—	—	—
FoxPro	32	35	25	35
Java	63	53	77	—
JavaScript	58	63	42	75
Perl	60	—	—	—
Smalltalk	26	19	10	55
SQL	40	37	7	110
Visual Basic	47	42	16	158

[pressman, 2009, p.676]

Métricas de calidad del Software

- Relacionados a los atributos de calidad del SW
  - Funcionalidad, usabilidad, performance, confiabilidad, portabilidad, mantenibilidad, disponibilidad, etc
- Mediciones de diferentes aspectos de la calidad
  - Más complejas, más sencillas, durante el desarrollo, después de la entrega
- Satisfacción al cliente: “adecuación al uso”
- Las más ampliamente usadas incluyen:
  - Correctitud, Mantenibilidad, Integridad, y Usabilidad [Pressman 2009, p. 681]

Métricas de calidad del Software

- Correctitud:

$$\text{densidad de defectos} = \frac{Num.\,de\,defectos}{KLOC}$$

- Mantenibilidad:

$$TMEC = \frac{\sum_1^n (TEC_i))}{n}$$

- Integridad:

$$Integridad = \sum_1^n [1 - amenaza_i * (1 - seguridad_i)]$$

- Usabilidad
  - Software Usability Measurement Inventory (SUMI)

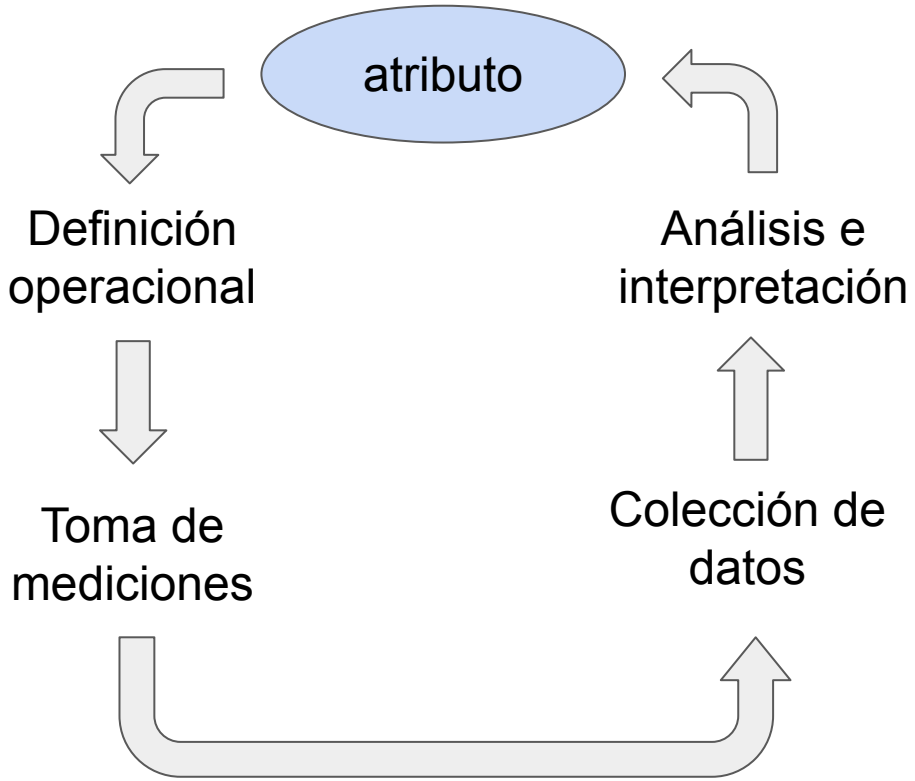
Métricas del Software: "top 10" list

1. Número de defectos después de la entrega
2. Peticiones de cambios
3. Satisfacción del usuario o cliente
4. Número de defectos durante el desarrollo
5. Completitud de la documentación
6. Tiempo para identificar y corregir defectos
7. Distribución de defectos por tipo/clase
8. Errores por funcionalidad/feature
9. Cobertura de pruebas de especificaciones
10. Cobertura de pruebas de código

[Glass, R. L. (2002). Facts and fallacies of software engineering, pag. 126]

Integración de métricas al proceso de Ing. soft.

- Mantener métricas es un proceso en sí mismo:
  - Definición, obtención datos, evaluación, mejora



Integración de métricas al proceso de Ing. soft.

- Las métricas son usadas muy poco en la práctica. Tienen que representar un costo adecuado: [Glass, 2002]
  - Obtención de datos, no más de 3%
  - Procesamiento y análisis, entre 4% y 6%
- Cuando sea posible, automatizar la obtención de datos
- Uso la datos históricos
  - sistemas de versionado: e.g. github jekyll
  - issue tracking: e.g bugzilla chromium
- Identificar y definir métricas operacionalmente (GQM).

Baselines

- Datos de proyectos anteriores que sean precisos
  - Tantos proyectos como sean posibles
- Métricas consistentes
  - Mismas definiciones operacionales
  - El tamaño, LOC, duración, severidad de defectos, etc
- Comparar entre similares
  - Productos: dominio, complejidad
  - Procesos: ágil, clásico
  - Tipo de proyectos: gubernamentales, personales, OSS
- Pueden ser simples como la tabla presentada o complejas con miles de proyectos y métricas avanzadas
- Fundamentales para mejorar el proceso y estimaciones

Conclusión

- Las métricas están atada a un atributo de calidad del software con las que buscamos estimar y/o mejorar
- El contexto determina las definiciones operacionales de las métricas: modelos de desarrollo, dominio, negocio, etc
- SLOC y FP representan indicadores de la oportunidad para cometer errores en las métricas de tasa de defecto.
- Las métricas son indispensables cuando pensamos en calidad y estimación de riesgo.
- No dejarse atraer demasiado por lo que las métricas conciben. El otro lado de las métricas!
    - No medir lo que es fácil, si lo que es esencial.

Bibliografía

Pressman, Roger S. Software engineering: a practitioner's approach. 7th edition. Palgrave Macmillan, (2010).

Ian Sommerville. Software Engineering (9th ed.). Addison-Wesley Publishing Company, USA, (2010).

Pfleeger, S. & Atlee, J. (2010). Software engineering : theory and practice. Upper Saddle River N.J: Prentice Hall.

Kan, S. H. (2002). Metrics and models in software quality engineering. Addison-Wesley Longman Publishing Co., Inc..

Van Solingen, Rini, Vic Basili, Gianluigi Caldiera, and H. Dieter Rombach. "Goal question metric (gqm) approach." Encyclopedia of software engineering (2002).

Glass, R. L. (2002). Facts and fallacies of software engineering. Addison-Wesley Professional.

IEEE-STD-1061-1998 Standard for Software Quality Metrics Methodology

IEEE-STD-610.12-1990 Glossary of Software Engineering Terminology