Universidad Católica de Córdoba
Universidad Jesuita

Ingeniería de software III

Unidad 6 - Métricas

Medición y métricas del software

✧La medición del software
- ✧se ocupa de derivar un valor numérico para un atributo de software como su complejidad o confiabilidad

✧Comparando los valores medidos entre ellos y con los estándares de la organización se pueden extraer conclusiones acerca de
- ✧la calidad del software
- ✧evaluar la efectividad del proceso, herramientas o métodos

✧En el mundo ideal la calidad podría basarse en mediciones de atributos que afectan la calidad del software

✧El objetivo final de la medición del software es usar la medición para hacer juicios sobre la calidad del software

✧Idealmente, un sistema podría ser evaluado usando un rango de métricas para medir sus atributos
- ✧de las medidas obtenidas inferir un valor de su calidad y compararlo contra un threshold para saber si la calidad requerida ha sido alcanzada

✧Sin embargo la industria está aún lejos de la situación ideal

Medición y métricas del software

✧Un métrica del software es una característica de un sistema de software, de su documentación, o del proceso de desarrollo que puede ser medido de manera efectiva

✧Las métricas pueden ser
- ✧de control
- ✧de predicción

✧Las métricas de control dan soporte a la gestión del proceso

✧Las métricas de predicción ayudan a predecir características del software

Medición y métricas del software

- Ejemplo de métricas de proceso
  - El tiempo que lleva completar un proceso particular
    - calendar time, etc
  - Los recursos requeridos para un proceso particular
    - cuantas personas/día, etc
  - El número de ocurrencias de un evento particular
    - Número de defectos promedio encontrados en una inspección
    - número de pedidos de cambio de requerimientos
    - número de líneas de código modificadas por pedidos en los requerimientos
    - etc

Medición y métricas del software

- Ejemplo de métricas de predicción o de producto
  - están asociadas con el software en sí mismo
  - ejemplos
    - complejidad ciclomática
    - longitud promedio de los identificadores
    - número de atributos y operaciones que tiene un clase
    - etc

Medición y métricas del software

- ✧Ambos tipos de métricas pueden influenciar en las decisiones del management
  - ✧Los managers usan
    - ✧mediciones de proceso para decidir si se debería cambiar el proceso
    - ✧métricas de predicción para decidir si el software tiene que ser cambiado o si está listo para ser liberado

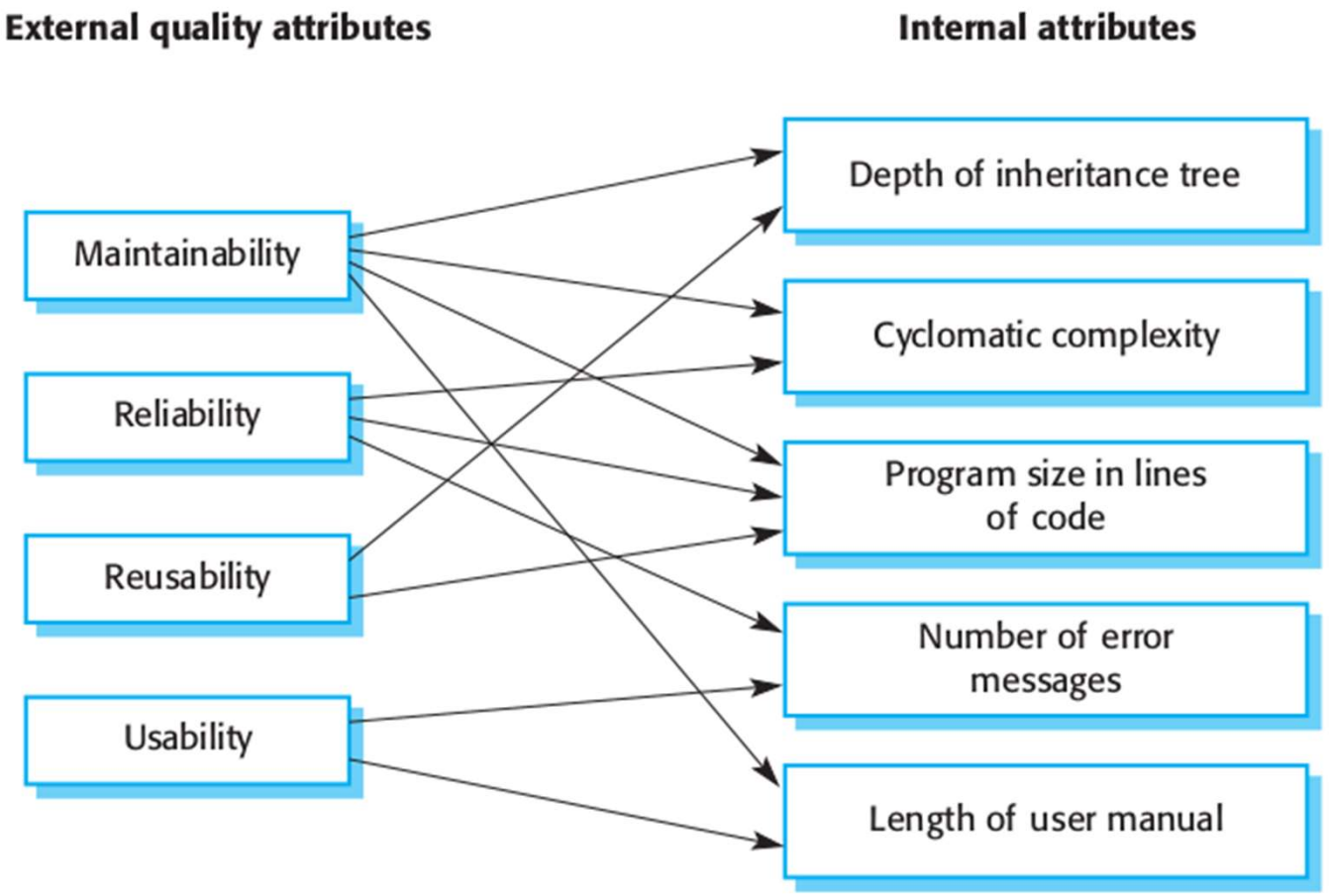
Métricas del software

- Las métricas del software deberían usarse de 2 maneras
  - Midiendo características del sistema y agregandolas para evaluar atributos de calidad del sistema
  - Para identificar los componentes del sistema que están por debajo de los estandares

- Es difícil hacer una medición directa de los atributos de calidad del software (mantenibilidad, usabilidad, etc)
  - Estos son atributos externos relacionados a como los desarrolladores y lo usuarios experimentan el sistema
  - No pueden medirse objetivamente
  - Pero se pueden medir atributos internos del software como el tamaño o la complejidad y asumir una relación entre ellos

Métricas del software

✧Relación intuitiva entre atributos internos y atributos externos de calidad



Métricas del software

- ✧Condiciones para que una medida de un atributo interno pueda utilizarse como predictor de un atributo de calidad externo
  - ✧El atributo interno debe poder medirse de manera precisa
    - ✧Muchas veces se utilizan tools específicas
  - ✧Debe existir una relación entre el atributo interno que se mide y el atributo de calidad externo que se intenta predecir
  - ✧La relación entre el atributo interno y el atributo de calidad externo debe ser entendida, validada y expresada en términos de una fórmula o modelo
    - ✧Identificar la forma funcional del modelo (lineal, exponencial, etc)
    - ✧Identificar los parámetros que se incluirán en el modelo
    - ✧Calibrar estos parámetros usando datos existentes

- ✧O se puede utilizar data-mining para descubrir relaciones y hacer predicciones sobre los atributos externos

Métricas del software

- ✧Durante los 90´s grandes compañías invirtieron en programas de métricas centrados en
  - ✧ Defectos de programa, o encontrados durante los procesos de validación y verificación
- ✧Hoy en día, se toman métricas de proceso
  - ✧ nro de peticiones de cambio
  - ✧ nro de defectos encontrados en pruebas
- ✧ Y internas de product
- ✧Pero no hay mucho al respecto
- ✧¿Por qué no es tan común el que se adopte un programa de métricas en una organización?
  - ✧Es difícil cuantificar el ROI de invertir en su introducción en la organización
  - ✧Se ha mejorado mucho en la calidad del software en los últimos años sin el uso de métricas
  - ✧No existen estándares de métricas del software o procesos estandarizados de mediciones y analysis
  - ✧Puede requerir la introducción de herramientas especializadas
  - ✧La mayor parte de los estudios e investigaciones de métricas se han hecho en torno a un proceso dirijido por plan.
    - ✧Más y más se utilizan los métodos ágiles
  - ✧Agrega overhead al proceso, lo cuál va en contra de los principios ágiles

Métricas de producto

- Las métricas de producto son predictores usados para cuantificar atributos internos del software
  - Ejemplos: loc, nro de métodos de una clase, etc
- Desafortunadamente no existe una relación universal entre los atributos internos del software que pueden ser medidos y los atributos de calidad externos
  - Varía dependiendo del proceso de desarrollo, la tecnología, el tipo de software, etc
- Se clasifican en
  - Métricas dinámicas
    - Colectadas midiendo un programa en ejecución
      - Por ejemplo, nro de bugs reportados, tiempo de respuesta, etc
  - Métricas estáticas
    - Colectadas midiendo representaciones de un sistema, como el diseño, el programa o la documentación
    - Análisis estático del Código
      - Existen numerosas herramientas
        - Sonar
        - Codacy
        - Klockwork
        - Etc

Métricas de producto

- Las métricas dinámicas y estáticas están relacionadas a diferentes tipos de atributos de calidad
  - Dinámicas
    - Sirven para evaluar eficiencia y confiabilidad
  - Estáticas
    - Sirven para evaluar complejidad, mantenibilidad, understandability

- Las métricas dinámicas pueden relacionarse fácilmente con los atributos de calidad
  - Tiempo de respuesta, tiempo de inicialización, etc se pueden relacionar directamente con la eficiencia
  - Nro de fallas, etc pueden relacionarse directamente con la confiabilidad del software

- Las métricas estáticas tienen relaciones más complejas con los atributos de calidad
  - Muchas intentos se han hecho, ninguno concluyente
  - Tamaño del programa o complejidad del control parecen ser bastante apropiados para predecir mantenabilidad, understandability, complejidad del sistema.

Métricas estáticas o dinámicas?

Software metric	Description
Fan-in/Fan-out	Fan-in is a measure of the number of functions or methods that call another function or method (say X). Fan-out is the number of functions that are called by function X. A high value for fan-in means that X is tightly coupled to the rest of the design and changes to X will have extensive knock-on effects. A high value for fan-out suggests that the overall complexity of X may be high because of the complexity of the control logic needed to coordinate the called components.
Length of code	This is a measure of the size of a program. Generally, the larger the size of the code of a component, the more complex and error-prone that component is likely to be. Length of code has been shown to be one of the most reliable metrics for predicting error-proneness in components.
Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability. I discuss cyclomatic complexity in Chapter 8.
Length of identifiers	This is a measure of the average length of identifiers (names for variables, classes, methods, etc.) in a program. The longer the identifiers, the more likely they are to be meaningful and hence the more understandable the program.
Depth of conditional nesting	This is a measure of the depth of nesting of if-statements in a program. Deeply nested if-statements are hard to understand and potentially error-prone.
Fog index	This is a measure of the average length of words and sentences in documents. The higher the value of a document's Fog index, the more difficult the document is to understand.

Métricas de producto

✧Métricas específicas para OO

✧http://www.virtualmachinery.com/sidebar3.htm

Object-oriented metric	Description
Weighted methods per class (WMC)	This is the number of methods in each class, weighted by the complexity of each method. Therefore, a simple method may have a complexity of 1, and a large and complex method a much higher value. The larger the value for this metric, the more complex the object class. Complex objects are more likely to be difficult to understand. They may not be logically cohesive, so they cannot be reused effectively as superclasses in an inheritance tree.
Depth of inheritance tree (DIT)	This represents the number of discrete levels in the inheritance tree where subclasses inherit attributes and operations (methods) from superclasses. The deeper the inheritance tree, the more complex the design. Many object classes may have to be understood to understand the object classes at the leaves of the tree.
Number of children (NOC)	This is a measure of the number of immediate subclasses in a class. It measures the breadth of a class hierarchy, whereas DIT measures its depth. A high value for NOC may indicate greater reuse. It may mean that more effort should be made in validating base classes because of the number of subclasses that depend on them.
Coupling between object classes (CBO)	Classes are coupled when methods in one class use methods or instance variables defined in a different class. CBO is a measure of how much coupling exists. A high value for CBO means that classes are highly dependent. Therefore, it is more likely that changing one class will affect other classes in the program.
Response for a class (RFC)	RFC is a measure of the number of methods that could potentially be executed in response to a message received by an object of that class. Again, RFC is related to complexity. The higher the value for RFC, the more complex a class, and hence the more likely it is that it will include errors.
Lack of cohesion in methods (LCOM)	LCOM is calculated by considering pairs of methods in a class. LCOM is the difference between the number of method pairs without shared attributes and the number of method pairs with shared attributes. The value of this metric has been widely debated, and it exists in several variations. It is not clear if it really adds any additional, useful information over and above that provided by other metrics.

Métricas

✧Response Time / Tiempo de Respuesta

- Tiempo que un sistema requiere para procesar un pedido visto desde fuera

✧Responsiveness / Tiempo de reacción

- Cuan rápido un sistema acepta (hace acknowledge) un pedido
- ¿Cuándo el tiempo de reacción es igual al tiempo de respuesta?

✧Latency / Latencia

- El tiempo mínimo requerido para obtener cualquier tipo de respuesta
- Sistemas locales vs sistemas remotos

✧Throughput / Rendimiento

- Cuanto trabajo puede hacer el sistema en una cantidad dada de tiempo
- Bytes/s – Transactiones/s

✧Load / Carga / Stress

- Es una medida de cuan estresado/cargado están un sistema
- Usando generalmente como contexto de otra medida de performance
- Tiempo de respuesta 0.5 s con 10 usuarios y 2 s con 20 usuarios

Performance

✧Load Sensitivity / Sensibilidad a la Carga

- Una expresión que indica cómo varía el tiempo de respuesta con la carga
- Sistema A: 0.5 s con 10 a 20 usuarios
- Sistema B: 0.2 s con 10 usuarios y 2 s con 20 usuarios

✧Efficiency / Eficiencia

- Es la performance dividido los recursos empleados
- Sistema A: 30 tps con 2 CPUs
- Sistema B: 40 tps con 4 CPUs (iguales a la anterior)

✧Scalability / Escalabilidad

- Es una medida que indica como se ve afectada la performance cuando se agregan nuevos recursos
- Scale Up: Agregar más o mejores recursos a un servidor
- Scale Out: Agregar más servidores

✧Capacidad

- Máxima carga o máximo rendimiento efectivos.
- Máximo absoluto o un punto dónde la performance se vuelve inacceptable

✧Ejemplo

Recursos	Sistema A	Sistema B
1 servidor	20 tps	40 tps
2 servidores	35 tps	50 tps
3 servidores	50 tps	60 tps
4 servidores	65 tps	70 tps
5 servidores	80 tps	80 tps

Preguntas