Asociación de Técnicos de Informática

# II Jornadas de Calidad de Software

Sesión de mañana del 2 de julio de 1998

# Métricas de calidad del software en orientación objetos

***Juan Carlos Granja Álvarez***

***Grupo Lenguajes y Sistemas Informáticos e Ingeniería del Software***

***Universidad de Granada***

***e-mail: jcgranja@goliat.ugr.es***





## Lord Kelvin dijo:

*•Cuando puedes medir lo que estás haciendo y expresarlo en números, sabes algo sobre ello; pero cuando no puedes medirlo, cuando no puedes expresarlo con números, tu conocimiento es escaso e insatisfactorio; puede ser el comienzo del conocimiento, pero en tus pensamientos, apenas estás avanzando hacia el escenario de la ciencia.*

# PROBLEMÁTICA EN O.O.

•Hay una serie de métricas en O.O. que miden algunas características del software que me permiten conocerlo y estudiarlo pero no son lo suficientemente buenas ni completas.

•No permiten detectar algunos tipos de errores en el software, algunas no son independientes del estilo de desarrollo del software, no son aplicables a nuevas herramientas de desarrollo de software, etc...





# Criterios convencionales que determinan la calidad del software

•corrección: el grado en que el software satisface las especificaciones.

•fiabilidad: el grado en que se espera que un sistema lleve a cabos sus funciones encomendadas de forma correcta.

•eficiencia: recursos requeridos por el software para llevar a cabos sus funciones.

# Criterios convencionales que determinan la calidad del software

•integridad: grado en que puede controlarse el acceso al software o a los datos por personal no autorizado.

•facilidad de uso: esfuerzo para aprender un programa, trabajar con él, ...

•facilidad de mantenimiento: esfuerzo requerido para localizar y arreglar un error en un programa





# Criterios convencionales que determinan la calidad del software

•flexibilidad: esfuerzo requerido para modificar un programa operativo.

•facilidad de prueba: esfuerzo requerido para probar un programa de forma que se asegure que realiza su función requerida.

•portabilidad: esfuerzo requerido para transferir el programa desde un hardware y/o un entorno de sistemas de software a otro.

# Criterios convencionales que determinan la calidad del software

•reusabilidad: grado en que un programa (o partes de un programa) se puede reusar en otras aplicaciones.

•facilidad de interoperación: esfuerzo requerido para acoplar un sistema a otro.





# SITUACIÓN ACTUAL DE LAS MÉTRICAS EN ORIENTACIÓN A OBJETOS

# Número de instrucciones en un método/Líneas de código por método/Media del tamaño de los métodos

•Esta métrica nos da una indicación de la calidad del diseño con respecto a la perspectiva de los objetos. Si la media del tamaño de los métodos es elevada indica una alta probabilidad de que el código esté escrito orientado a función. Si ésta es baja puede indicar que el código esté escrito orientado a objetos.





Para solucionar ésto habría que dividir el código del método en varios métodos más pequeños o pasar la funcionalidad de éste a otras subclases.

## Número de métodos en una clase (métodos públicos, protegidos y privados); Media del número de métodos por clase

•El número de variables de instancia de una clase es una medida de su tamaño. El que una clase tenga un número alto de variables de instancia puede indicar que tiene más relaciones de las deseables con otros objetos del sistema o que la clase está haciendo más cosas de las que ella debería hacer.





•Hay una serie de métricas en O.O. que miden algunas características del software que me permiten conocerlo y estudiarlo pero no son lo suficientemente buenas ni completas.

•No permiten detectar algunos tipos de errores en el software, algunas no son independientes del estilo de desarrollo del software, no son aplicables a nuevas herramientas de desarrollo de software, etc...

- Para determinar si nuestra clase tiene demasiadas variables de instancia, deberíamos hallar la media del número de variables de instancia por clase. Si una clase tiene más variables de instancia que la media de las clases, deberíamos revisarla ya que puede haber algo anormal.





## Los factores que se pueden ver afectados son

•reusabilidad: será más difícil reutilizar las clases ya que se puede dar el caso de que la clase tenga muchos métodos o métodos muy grandes

# Nivel de anidación en la jerarquía de clases

•Una gran profundidad de anidación indica un problema de diseño, donde los desarrolladores pueden ser entusiastas en encontrar y crear objetos. Esto da generalmente una subclase que no es especialización de una superclase, una subclase debe extender la funcionalidad de una superclase. Puede darse el caso de que existan clases que no tengan prácticamente responsabilidades.





# Los factores a los que puede afectar

•reusabilidad: va a perjudicar a que se reutilice el código (por ej. cuando se de el caso de que una subclase no es una especialización de una superclase).

# Número de métodos con el mismo nombre que un método de la superclase

•Un gran número de métodos de este tipo indica un problema de diseño. Dado que una subclase debería ser una especialización de su superclase, la subclase debería extender los servicios de una superclase. Esto daría lugar a nuevos métodos.





•Tener varios métodos de este tipo indica una subclasificación por la conveniencia de reusar código y/o variables de instancia cuando la nueva subclase no es puramente una especialización de su superclase. Una subclase extiende los servicios de una superclase pero no los sobrescribe.

## Los factores a los que afecta

•reusabilidad: puede dar problemas si quiero reusar desde una subclase el comportamiento de una clase superior pero entre ambas hay un método con el mismo nombre que otro método de la superclase. Puede inducir a errores.





## Uso de herencia múltiple

•C++ permite el uso de la herencia múltiple. Hay complicaciones que resultan del uso de herencia múltiple:

- se producen colisiones: si dos superclases tienen ambas un método con el mismo nombre, ¿qué método se ejecutaría?
- falta de comprensión por parte de los desarrolladores.

•En la industria generalmente no se usa la herencia múltiple dado que hay un número de formas apropiadas para modelar el sistema y la herencia múltiple solo puede introducir problemas. En general se aconseja no utilizarla.





## Puede afectar a los siguientes factores

•reusabilidad: va a provocar que la reusabilidad sea menor. Al tener un diseño complejo y en cierto modo erróneo y menos comprensible hace que el sistema sea menos reusable.

# Número de métodos añadidos por una subclase

•Las subclases normalmente heredan el comportamiento de una superclase en forma de métodos y el estado de los datos en forma de variables de instancia. El número de métodos heredados de una superclase indica la fortaleza de la subclasificación por especialización.





# Número de métodos añadidos por una subclase

•Una subclase debería definir nuevos métodos, extendiendo el comportamiento de la superclase. Una clase que no añade nuevos métodos es cuestionable. El número de nuevos métodos debería decrementar conforme se baja en la jerarquía de clases.

•Si esto no sucede habría que reconsiderar el diseño de las clases ya que esto podría acarrear problemas en el futuro.

# PROBLEMÁTICA ACTUAL

•Estas métricas miden algunas características del software que nos permiten saber si éste es un software de calidad, si el mantenimiento de éste será costoso, conocer ciertas características del software, etc...

•Pero plantean el problema de que no son capaces de detectar errores sino de detectar posibles fuentes de error. Luego es labor del programador el comprobar si hay un error o no. No ahorra al programador comprobar si hay un error en esa posible fuente de error.





•Otro problema que plantea el uso de estas métricas en el ámbito industrial es que no son aplicables a la programación visual, tipo de programación que surge ahora con fuerza. Estas métricas serían solo aplicables en la programación orientada a objetos, pero tradicional, sin utilizar herramientas visuales. jerarquías de herencia?

•Además estas métricas carecen de formalización. Hace falta métricas que sean estrictas para no definir con vaguedades y suficientemente amplio para abarcar todo el campo del sujeto. Por ejemplo el nivel de anidación en la jerarquía de clases podemos pensar que es una métrica formal, pero nada más lejos de la realidad.





•Cuando tenemos herencia múltiple, el concepto de raíz no está nada claro. Además, ¿cuál es el significado real de esta medida?, ¿es otra medida de complejidad, en este caso de las jerarquías de herencia?

# CONCLUSIONES SOBRE LAS MÉTRICAS ORIENTADAS A OBJETOS

- Como conclusión podemos exponer que las actuales métricas orientadas a objetos proporcionan una escasa información sobre el software que no nos permite conocerlo con profundidad, detectar errores sobre éste.
- Además no son útiles para la programación basada en herramientas visuales, estilo de programación que en estos momentos tiene mucho auge.