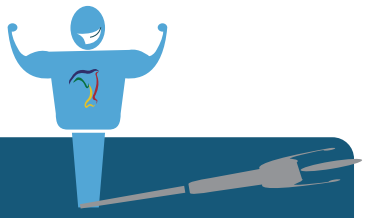
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

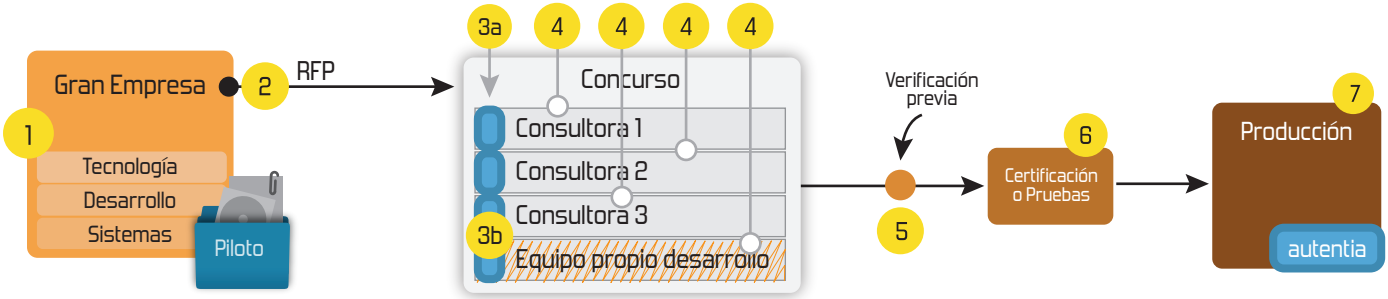
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
Gestor de contenidos (Alfresco)
Aplicaciones híbridas

Control de autenticación y acceso (Spring Security)
UDDI
Web Services
Rest Services
Social SSO
SSO (Cas)

JPA-Hibernate, MyBatis
Motor de búsqueda empresarial (Solr)
ETL (Talend)

Dirección de Proyectos Informáticos.
Metodologías ágiles
Patrones de diseño
TDD

Tareas programadas (Quartz)
Gestor documental (Alfresco)
Inversión de control (Spring)

BPM (jBPM o Bonita)
Generación de informes (JasperReport)
ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Miguel Arlandy Rodríguez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/JEE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2012-12-12 Tutorial visitado 1 veces Descargar en PDF

Sonar y Total Quality: midiendo la calidad total de nuestros proyectos.

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Instalando el plugin.
  - 3.1 Instalación con Sonar.
  - 3.2 Instalación manual.
- 4. Configurando el plugin.
- 5. ¿Cómo analiza Total Quality la calidad de nuestros proyectos?.
- 6. Calculando la calidad de la arquitectura.
  - 6.1 Package Tangle Index (TI).
- 7. Calculando la calidad del diseño.
  - 7.1 Complejidad ciclomática (NOM).
  - 7.2 Lack of Cohesion In Methods (LCOM).
  - 7.3 Response for Class (RFC).
  - 7.4 Coupling Between Object Clases (CBO).
  - 7.5 Depth of Inheritance Tree (DIT).
- 8. Calculando la calidad del código.
  - 8.1 Documented API Density (DOC).
  - 8.2 Rules Compilance Index (RULES).
  - 8.3 Duplicated Lines Density (DRYNESS).
- 9. Calculando la calidad de los tests.
  - 9.1 Cobertura (COV).
  - 9.2 Unit test success (SUC).
- 10. Referencias.
- 11. Conclusiones.

1. Introducción

Como sabemos, Sonar es una excelente herramienta open source que tiene como objetivo analizar y medir la calidad de nuestros proyectos. Para conseguir este propósito hace uso de una gran cantidad de métricas de calidad del software además de un amplio juego de reglas (PMD, Checkstyle, Findbugs...).

Existe un plugin de Sonar llamado Total Quality que agrupa una serie de métricas de Sonar para ofrecernos una **evaluación general de la calidad** de nuestro proyecto. Este plugin permite que configuremos aquellos parámetros que más se ajusten a nuestras necesitades o requerimientos.

En este tutorial veremos cómo el plugin Total Quality de Sonar calcula la calidad total de nuestros proyectos y comentaremos las métricas en las que se apoya (complejidad ciclomática, LCOM, RFC, código duplicado, etc...).

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

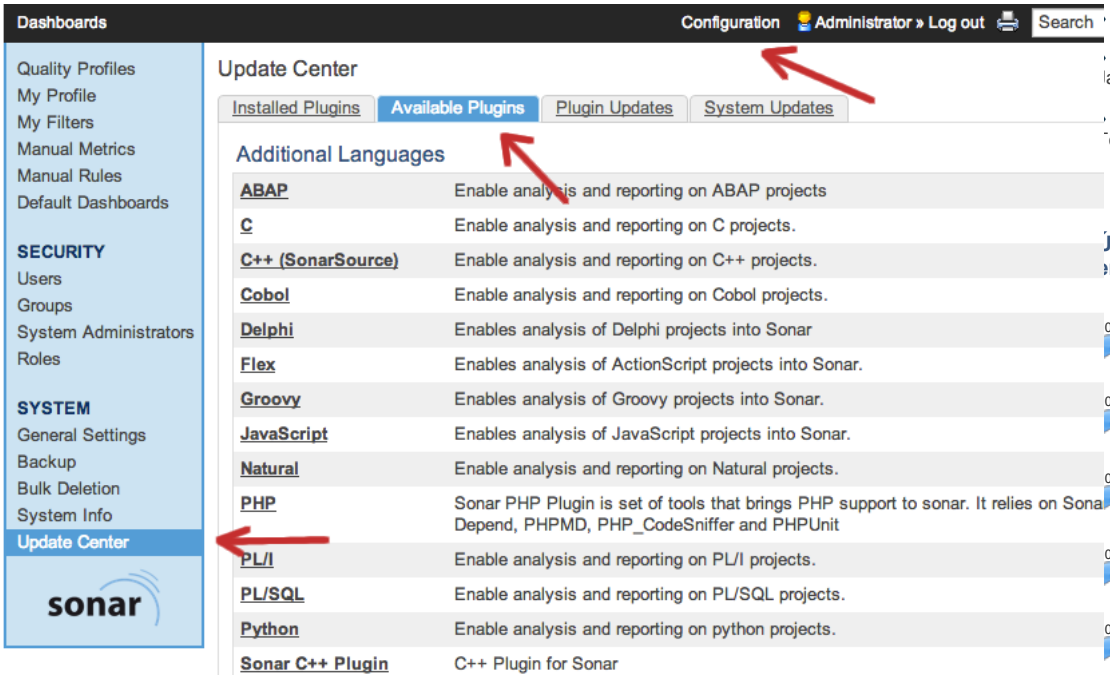
- Hardware: Portátil MacBook Pro 15' (2.2 Ghz Intel Core I7, 8GB DDR3).
- Sistema Operativo: Mac OS Mountain Lion 10.8
- Entorno de desarrollo: Intellij Idea 11.1 Ultimate.
- Sonar 3.3.2
- Total Quality 1.0.2

3. Instalando el plugin.

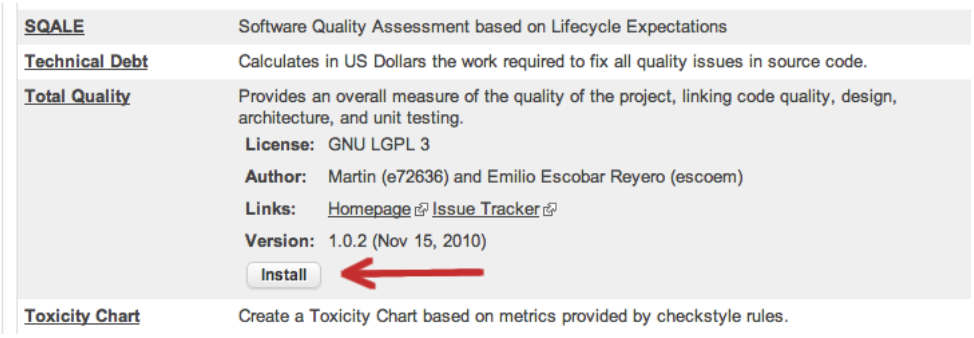
Instalar el plugin es muy sencillo. Podemos hacerlo de dos formas: a través de Sonar o de forma manual.

3.1 Instalación con Sonar.

Lo primero que debemos hacer es acceder a la aplicación y logarnos con un usuario administrador. Luego pulsamos sobre **Configuration (parte superior derecha) > Update Center (menú izquierda) > Available Plugins (pestaña)**



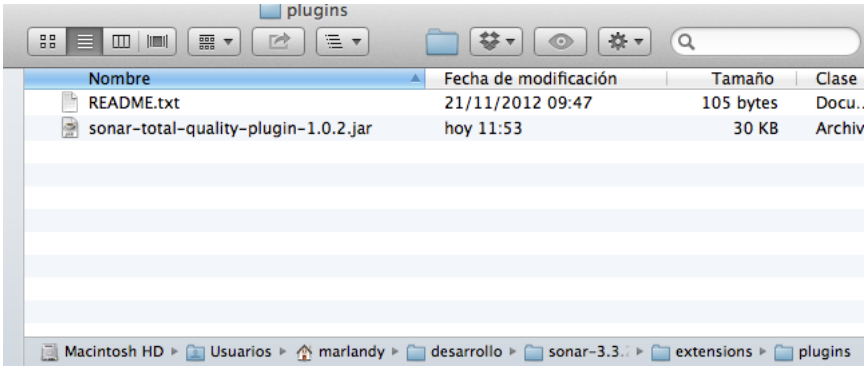
A continuación, en el listado de plugins disponibles, buscamos el plugin Total Quality, pinchamos sobre él y pulsamos sobre el botón "Install".



Reiniciamos Sonar y listo.

3.2 Instalación manual.

Si preferimos la opción manual podemos hacerlo de la siguiente forma. Nos descargamos el plugin (es un .jar) de la página de Total Quality y lo copiamos en el directorio **$SONAR_HOME/extensions/plugins/**, donde $SONAR_HOME es el directorio de instalación de Sonar.

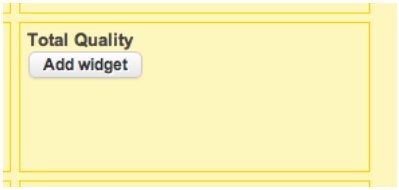


Reiniciamos Sonar y listo.

4. Configurando el plugin.

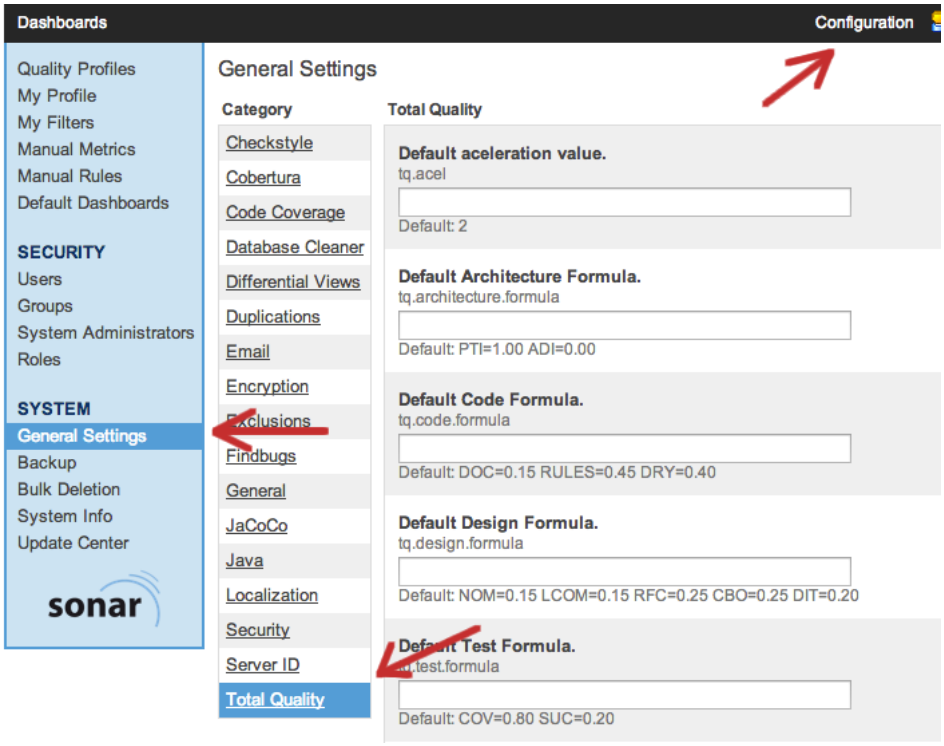
Una vez tenemos el plugin instalado ya podemos proceder a su configuración.

Lo primero que haremos será añadir el Widget que nos proporciona el plugin a nuestro Dashboard. Para ello, pulsamos sobre la opción **"Configure widgets"** (en el Dashboard), buscamos el Widget de Total Quality y pulsamos sobre "Add widget".



Como veremos en los siguientes apartados, Total Quality calcula la calidad de nuestro proyecto en base a **fórmulas donde intervienen diferentes métricas de Sonar**. Dichas fórmulas tienen unos parámetros por defecto de forma que puden dar más importancia a alguna métrica que a otra a la hora de calcular la calidad total. Sin embargo, estos **parámetros son totalmente configurables**, por lo que podemos ajustarlos de la forma que más se ajuste a nuestras necesidades.

Para configurar los parámetros lo haremos pulsando sobre **Configuration (parte superior derecha) > General Settings (menú izquierda) > Total Quality**



A continuación explicaremos cómo Total Quality analiza la calidad total de nuestros proyectos y cada uno de los factores que se tienen en cuenta :-).

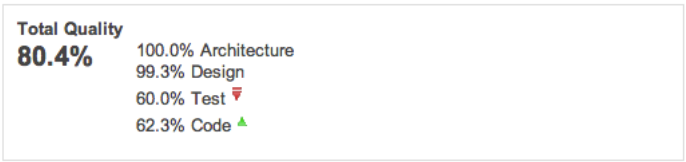
5. ¿Cómo analiza Total Quality la calidad de nuestros proyectos?.

Como dijimos en el punto anterior, Total Quality aplica diferentes fórmulas a diferentes métricas de Sonar. Las fórmulas aplican un peso (que es totalmente configurable) a las distintas métricas y con el resultado se obtiene la calidad total.

Todo parte de una fórmula:

Total Quality = 0.25*ARCH + 0.25*DES + 0.25*CODE + 0.25*TS, donde:

- ARCH: Calidad de la arquitectura de nuestro proyecto. Por defecto es el 25% de la calidad total de nuestro proyecto.
- DES: Calidad del diseño de nuestro proyecto. Por defecto es el 25% de la calidad total de nuestro proyecto.
- CODE: Calidad del código de nuestro proyecto. Por defecto es el 25% de la calidad total de nuestro proyecto.
- TS: Calidad de los tests de nuestro proyecto. Por defecto es el 25% de la calidad total de nuestro proyecto.



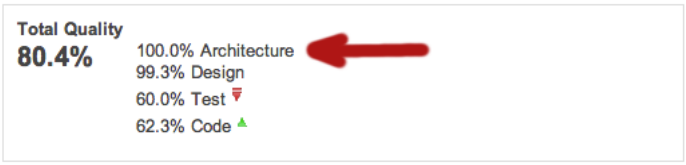
A continuación, explicaremos cómo se calculan cada uno de estos cuatro factores.

6. Calculando la calidad de la arquitectura.

Para Total Quality el indicador de calidad de la arquitectura (ARCH) tiene un peso de un 25% (por defecto) de la calidad total. Para obtener este indicador utiliza la siguiente fórmula:

ARCH = 100 - TI, donde:

- TI: Acoplamiento entre clases de distintos paquetes. Es el factor que indicará la calidad de la arquitectura.

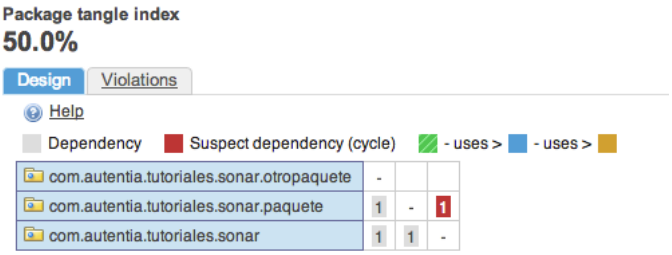


6.1 Package Tangle Index (TI).

Este valor es calculado en base al acoplamiento (dependencias) entre clases de un paquete con otro paquete. Mide los ciclos de dependencias, esto es, una o varias clases de un paquete A importan clases de un paquete B. A su vez, una o varias clases del paquete B importan clases del paquete A.

La métrica de Sonar que utiliza Total Quality es: *package_tangle_index* del proyecto.

Un valor de un 0% significa que no existen ciclos entre paquetes de un proyecto un valor del 100% significa que hay un acoplamiento máximo entre paquetes (o lo que es lo mismo, que el diseño de la arquitectura es un desastre :-S).



7. Calculando la calidad del diseño.

Para Total Quality el indicador de calidad del diseño (DES) tiene un peso de un 25% (por defecto) de la calidad total. Para obtener este indicador utiliza la siguiente fórmula:

DES = 0.15*NOM + 0.15*LCOM + 0.25*RFC 0.25*CBO + 0.20*DIT, donde:

- NOM: Complejidad ciclomática. 15% de la calidad total del diseño.
- LCOM: Cohesión entre métodos y atributos. 15% de la calidad total del diseño.
- RFC: Métodos que pueden ser potencialmente ejecutados. 25% de la calidad total del diseño.
- CBO: Dependencia entre clases. 25% de la calidad total del diseño.
- DIT: Profundidad el árbol de herencia. 20% de la calidad total del diseño.

A continuación, explicaremos cada uno de los factores que influyen en la calidad del diseño. Nótese que en en todas las fórmulas utilizadas para calcular los factores de calidad del diseño existe una variable **acel** (por defecto es 2) que es un acelerador que condicionará el valor del factor en función de si la métrica tiene un valor que mejora o empeora el valor que se considera "aceptable". Lo veremos mejor en los siguientes puntos.

7.1 Complejidad ciclomática (NOM).

Este valor es calculado en base a la complejidad ciclomática de nuestras clases y métodos. La complejidad ciclomática mide el **número de caminos independientes dentro de un programa o fragmento de código**.

Concretamente, Sonar mide la complejidad ciclomática incrementando en 1 el valor por cada: **if, for, while, case, catch, throw, return** (que no sea la última sentencia del método), **&&, ||** y **?** que encuentre. También se incrementa en 1 por cada método declarado.

Veamos un ejemplo:

```
1  // Complejidad 5
2  public int obtenerMayor(int a, int b) { // +1 (método)
3      if (a > b) { // +1 (if)
4          return a; // +1 (return que no es la última sentencia)
5      } else if (a < b) { // +1 (if)
6          return b; // +1 (return que no es la última sentencia)
7      }
8      return a; // +0 (return que SI es la última sentencia)
9  }
```

Para calcular el indicador de complejidad NOM, Total Quality utiliza la siguiente fórmula (por defecto):

NOM = (1 - ((class_complexity - 12) / (acel * 12))) * 50 + (1 - ((method_complexity - 2.5) / (acel * 2.5))) * 50

Como puede comprobarse, cuanto menor sea la compejidad de nuestras clases y métodos mayor será el indicador de complejidad y, por tanto, influirá positivamente en el diseño. Los valores "aceptables" que considera Total Quality son **12** para la complejidad por clase y **2.5** para la complejidad por método. Por supuesto, estos valores son configurables.

Las métricas de Sonar que utiliza son: *class_complexity y function_complexity* del proyecto (o recurso a analizar).

7.2 Lack of Cohesion In Methods (LCOM).

El índice de cohesión de métodos y atributos de una clase comprueba el **uso que hacen los métodos de los atributos**

en una clase.

El índice LCOM4 puede tomar los siguientes valores:

- **LCOM4 = 0**: La clase no tiene métodos.
- **LCOM4 = 1**: Es el valor ideal. Todos los atributos y métodos de una clase están relacionados tanto de forma directa (un método hace uso de un atributo) como indirecta (un método invoca a otro método que hace uso de un atributo).
- **LCOM4 > 1**: Clase que probablemente pueda ser descompuesta en dos o más clases (Principio de Responsabilidad Única).

Para calcular el indicador LCOM, el plugin Total Quality utiliza la siguiente fórmula (por defecto):

LCOM = (1 - ((LCOM4 - 1) / (acel * 1))) * 100

El valor obtenido será mayor (y por tanto mayor la calidad de nuestro diseño) cuando LCOM4 esté más próximo a 1 que, como vimos anteriormente, es el valor ideal.

La métrica de Sonar que usa Total Quality para realizar este cálculo es: *lcom4*.

Sin embargo, debemos tener cuidado con cómo interpretamos esta métrica ya que existen determinadas clases (que podríamos considerar como excepciones) donde este valor puede ser muy alto como puede ser el caso de las clases de utilidades o JavaBeans.

7.3 Response for Class (RFC).

Esta métrica indica el número total de métodos que pueden ser ejecutados en respuesta a un mensaje recibido por un objeto de una clase. Dicho en otras palabras, es el número de métodos que son invocados por los métodos de una clase. Estos métodos invocados pueden ser tanto métodos de esa misma clase como de otras.

Cada llamada a un mismo método se cuenta una única vez de tal forma que, si se invoca tres veces al un método "toString" el valor de RFC únicamente se incrementará en 1.

Por tanto, el valor de RFC será mayor cuanto **mayor sea el número de métodos de nuestra clase y mayor sea el número de métodos distintos invocados de otras clases**.

Para calcular el RFC, Total Quality utiliza la siguiente fórmula (por defecto):

RFC = (1 - ((response_for_class - 50) / (acel * 50))) * 100

Observamos que el valor "aceptable" que considera Total Quality es **50** (aunque podemos configurar el valor que queramos).

La métrica de Sonar que usa Total Quality para realizar este cálculo es: *rfc*.

7.4 Coupling Between Object Clases (CBO).

Esta métrica indica el número de clases de las que otra clase depende.

Lo vemos en un ejemplo:

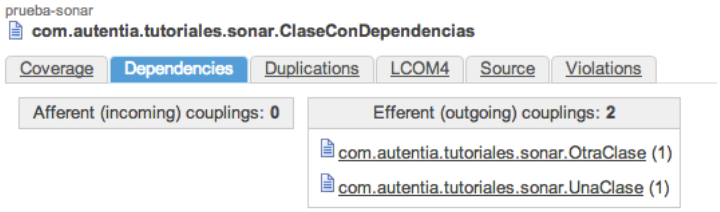
```
1  public class ClaseConDependencias {
2
3      private UnaClase unaClase; // +1
4
5      private OtraClase otraClase; // +1
6
7  }
```

Para calcular el CBO, Total Quality utiliza la siguiente fórmula (por defecto):

CBO = (1 - ((efferent_coupling - 5) / (acel * 5))) * 100

Observamos que el valor "aceptable" que considera Total Quality es **5**.

La métrica de Sonar que usa Total Quality para realizar este cálculo es: *ce* (efferent coupling).



7.5 Depth of Inheritance Tree (DIT).

Mide el nivel de profundidad de nuestras clases en el árbol de herencia. Esta métrica se tiene en cuenta para intentar que no se abuse de la herencia que, a su vez, hará el código muy difícil de mantener.

Para calcular el DIT, Total Quality utiliza la siguiente fórmula (por defecto):

DIT = (1 - ((depth_of_inheritance_tree - 5) / (acel * 5))) * 100

Observamos que el valor "aceptable" que considera Total Quality es **5**.

La métrica de Sonar que usa Total Quality para realizar este cálculo es: *dit*.

8. Calculando la calidad del código.

Para Total Quality el indicador de calidad del código (CODE) tiene un peso de un 25% (por defecto) de la calidad total. Para obtener este indicador utiliza la siguiente fórmula:

CODE = 0.15*DOC + 0.45*RULES + 0.40*DRYNESS, donde:

- DOC: Javadoc en API´s publicas. 15% de la calidad total del código.
- RULES: Cumplimiento de reglas. 45% de la calidad total del código.
- DRYNESS: Código duplicado. 40% de la calidad total del código.

8.1 Documented API Density (DOC).

Este valor indica el porcentaje de clases públicas, interfaces, constructores y métodos públicos que tienen comentarios (Javadoc). Existen ciertas excepciones como constructores vacíos o getters/setters.

La métrica de Sonar que usa Total Quality para realizar este cálculo es: *public_documented_api_density* y le da un peso de un 15% sobre el total de la calidad del código.

```
/**
 * Clase que suma dos números
 */
public class ClaseConJavadoc {

    private int a;

    private int b;

    /**
     * Crea una instancia con los dos números
     * @param a primer numero
     * @param b segundo número
     */
    public ClaseConJavadoc(int a, int b) {
        this.a = a;
        this.b = b;
    }

    /**
     * Suma los dos números
     * @return suma
     */
    public int sumaNumeros() {
        return a + b;
    }

}
```

Personalmente, no me considero muy amigo de esta métrica como indicador de la calidad del código, probablemente sea preferible aplicar un poco de Clean Code ya que, los comentarios hay que mantenerlos y pueden decir la verdad o no, sin embargo el código no mentirá nunca ;).

8.2 Rules Compilance Index (RULES).

Pues este indicador es bien sencillo. El porcentaje de cumplimiento de reglas que cumpla nuestro proyecto.

La métrica de Sonar que usa Total Quality para realizar este cálculo es: *violations_density* y le da un peso de un 45% sobre el total de la calidad del código.

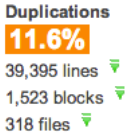
8.3 Duplicated Lines Density (DRYNESS).

Indica el porcentaje de líneas de código duplicadas en nuestro proyecto. Recordemos que, probablemente, el mayor enemigo del código limpio es el código duplicado.

La fórmula que utiliza Total Quality para sacar este valor es:

DRYNESS = 100 - Duplicated lines density

La métrica de Sonar que usa Total Quality para realizar este cálculo es: *duplicated_lines* y le da un peso de un 40% sobre el total de la calidad del código.

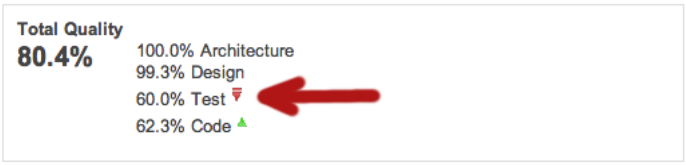


9. Calculando la calidad de los tests.

Para Total Quality el indicador de calidad de los tests (TS) tiene un peso de un 25% (por defecto) de la calidad total. Para obtener este indicador utiliza la siguiente fórmula:

TS = 0.80*COV + 0.20*SUC, donde:

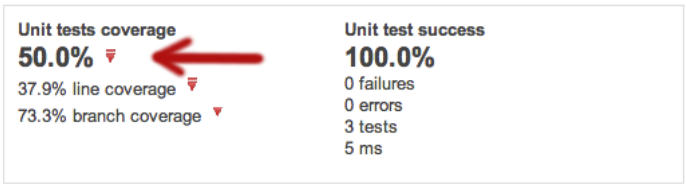
- COV: cobertura. 80% de la calidad total de los tests.
- SUC: tests OK. 20% de la calidad total de los tests.



9.1 Cobertura (COV).

El porcentaje de líneas de código cubiertas por nuestros tests.

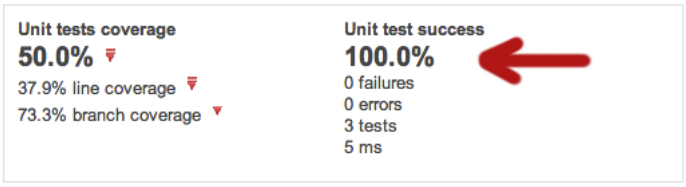
La métrica de Sonar que usa Total Quality para realizar este cálculo es: *coverage* y le da un peso de un 80% sobre el total de la calidad de los tests.



9.2 Unit test success (SUC).

El porcentaje de tests que terminaron con el resultado esperado.

La métrica de Sonar que usa Total Quality para realizar este cálculo es: *test_success_density* y le da un peso de un 20% sobre el total de la calidad de los tests.



10. Referencias.

- Total Quality
- Métricas de Sonar
- Analizando la calidad del código Java con Sonar

11. Conclusiones.

En este tutorial hemos presentado el plugin Total Quality de Sonar cuyo objetivo es medir la calidad todal de nuestro proyecto en base a la arquitectura, diseño, código y tests del mismo. Además, la forma de calcular esta métrica de calidad es totalmente configurable de forma que pueda adaptarse a las necesidades o requerimientos que consideremos más oportunos.

Dedicado a Borja, Irene y Nahia. ¡Enhorabuena familia!.

Espero que este tutorial os haya sido de ayuda. Un saludo.

Miguel Arlandy

marlandy@autentia.com

Twitter: @m_arlandy

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» Registrate y accede a esta y otras ventajas **«**