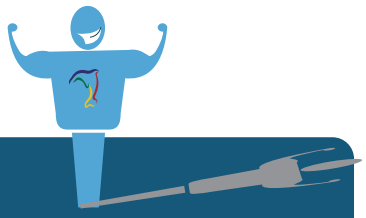
autentia

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...

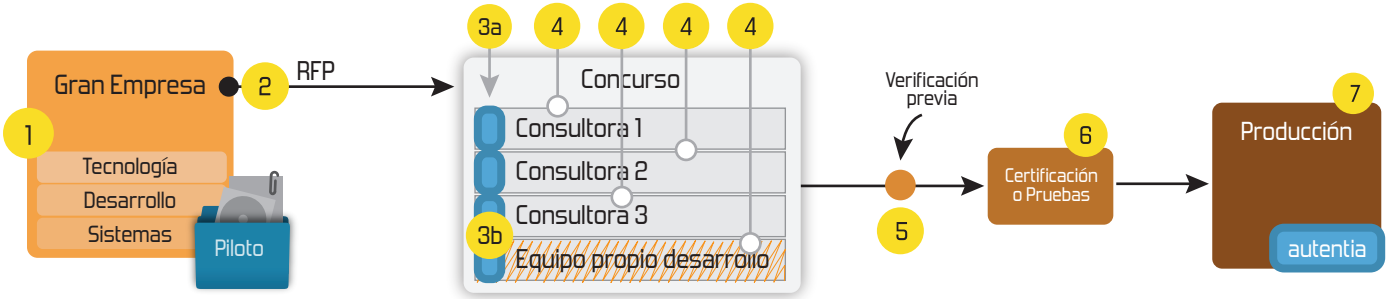
1. Desarrollo de componentes y proyectos a medida



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

Borja Lázaro de Rafael

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero Técnico en Informática de Sistemas

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

Somos expertos en Java/J2EE

Seguir a @borjaldr

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2015-03-03

Tutorial visitado 20 veces Descargar en PDF

¡Colega! ¿dónde están mis métricas?: Sonarqube 4.5+ LTS

0. Índice de contenidos.

1. Introducción

Desde septiembre de 2014, Sonarqube está liberando las últimas versiones LTS (*Long Term Support, 4.5.X*); por lo que ya hay que ir pensando en actualizarse. Desde el 26 de febrero de 2015 está disponible la versión 4.5.4 (LTS), aunque este tutorial está escrito con la versión anterior 4.5.2 (LTS) liberada el 7 de enero de 2015, no implica cambios respecto a lo que aquí se explica.

Si esta es la primera versión de Sonarqube que utilizas, te vamos a contar qué significan algunos de los conceptos que utiliza y cómo hay que configurar algunos parámetros para que los análisis de calidad realmente tengan sentido, ya que con la configuración *out of the box* del producto considero que éstos informes pueden llevar a una falsa sensación se tranquilidad y calidad que realmente no es real, pudiendo la calidad técnica de tus proyectos mucho peor de lo que piensas.

Si eres de los que te actualizas desde la versión LTS anterior (3.7.4), también tienes que estar preparado mentalmente porque cuando hagas tu primera análisis de un proyecto en la nueva versión, vas a abrir su *dashboard* y tú primera impresión sera .... *"¡Colega! ¿dónde están mis métricas?"* ya que han cambiado, y algunas de las actuales representan conceptos distintos.

Además verás que en esta versión, la cobertura de tu código no aparece. Todo esto te puede echar para atrás a la hora de actualizarte, pero vamos a ver que no es así. Ya sabes que no te conviene quedarte en una versión que ya se puede considerar obsoleta debido a todos lo cambios introducidos. Y si lo que te preocupa es la calidad técnica de los proyectos creo que con esta versión de Sonarqube puedes tener más argumentos para transmitir esa preocupación a todos los responsables. Por que no olvidemos que todo aquello que queramos mejorar o invertir viene condicionado por la cantidad de recursos que tenemos disponibles.

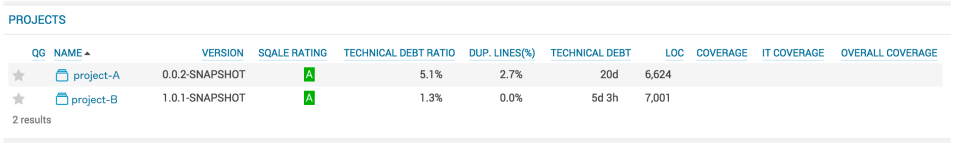
Para intentar ayudar a enteder estas métricas voy a usar dos proyectos de ejemplo, que nos van a servir para comprobar las implicaciones que los cambios de configuración necesarios Sonarqube 4.5.2 LTS

2. ¡Esto no es lo que era!

Tanto si eres nuevo en esta versión de Sonarqube, como si vienes de la versión anterior, cuando hagas tú primer análisis de un proyecto te vas a encontrar con unas métricas que hay que interpretar. Además si eres de los que vienen de las versión 3.7.4 LTS anterior, verás que algunas métricas son familiares y que hay otras nuevas que hay que aprender a interpretar. Pero eso no significa que hayamos perdido el objetivo principal: asegurar la calidad de los proyectos y mantenerla en unos niveles que sean aceptables y podamos asumir en función de nuestro contexto.

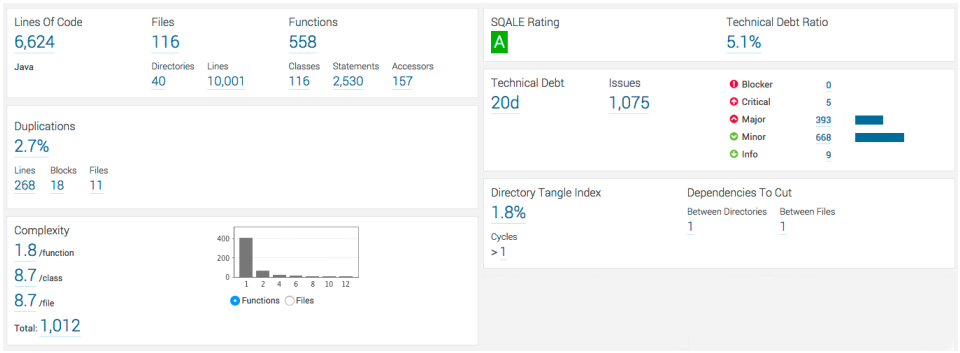
Lo primero que hacemos es eliminar el widget de bienvenida de la página principal, y así tendremos una vista general de los proyectos. También he añadido algunas columnas para que aparezcan en el Widget de proyectos. No voy a contar como configurar estas vistas, ya que considero que con poco que entres a "trastear" en la configuración de las pantallas puedes verlo fácilmente. Además de que en la documentación oficial de Sonarqube puedes encontrar esta información y recientemente se ha creado la Comunidad Hispana de usuarios de Sonarqube donde puedes encontrar información y hacer todo tipo de preguntas.

Bueno, vamos a ver qué nos encontramos tras el primer análisis de los 2 proyectos:

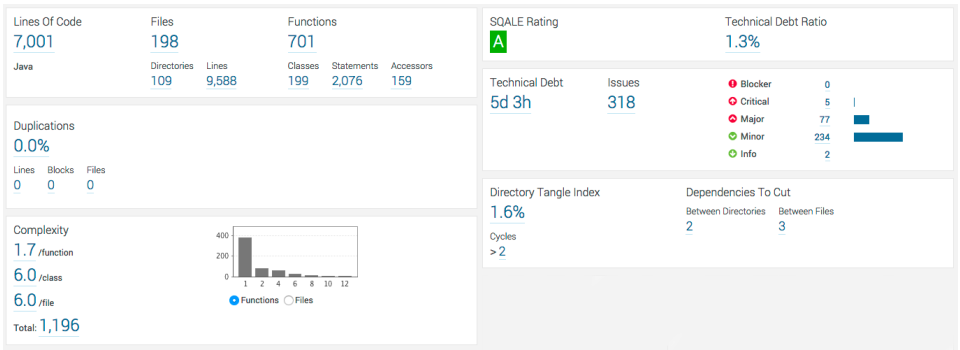


Ahora vamos a entrar a ver el dashboard de cada uno de los proyectos:

Proyecto A:



Proyecto B:



A primera vista podemos ver que son dos proyectos muy parecidos, con unas métricas de tamaño equivalentes, etc. Ahora tenemos que interpretar el estado de salud de los proyectos para saber si nos encontramos con unos parámetros cómodos o no. En la parte izquierda del *dashboard*, nos encontramos con métricas que ya conocemos de la versión anterior o que son fácilmente interpretables, son las relacionadas con las *"métricas de tamaño"* como líneas de código, número de ficheros, etc. también las que nos muestran el porcentaje de código duplicado que tenemos en el proyecto o las relacionadas con la complejidad de clase y función. Una primera aproximación de cómo interpretar estas métricas sería la siguiente:

- **Métricas de tamaño**: Inicialmente no nos van a indicar nada respecto a la calidad del proyecto, más bien se refieren a la magnitud del mismo. Y esto tiene una varianza muy grande dependiendo de los tipos de proyecto y de cada proyecto en particular.
- **Porcentaje de código duplicado**: La intuición ya nos indica que cuanto menos código duplicado tengamos en un proyecto, en términos de calidad es mejor. Seguramente tengamos un mejor diseño y desde luego reducimos los tiempos de mantemiento frente cambios y el riesgo de dejar una parte duplicada sin actualizar un cambio.
- **Complejidad**: Aquí también es intuitivo que una complejidad menor es mejor que una más alta. Independientemente del tipo que sea.

Tú dirás que esto está muy bién y que ya lo sabías, pero que lo que quieres saber es qué valores concretos de porcentaje de código duplicado y de complejidad consideramos como aceptables. ¡Amigo! ¿quién sabe eso? realmente no hay una medida justa, y como diría *Ramón de Campoamor (ley de campoamor): "todo depende del cristal con el que se mira"*. Efectivamente, en nuestro mundo este "cristal" lo conocemos comúnmente como "gorras", es decir, si me pongo la "gorra de control de calidad del software" puedo llegar a pedir valores que cercanos a la perfección, mientras que con una "gorra de responsable de negocio" no me interesa tanto la calidad técnica como otros criterios y puedo ir relegándola hasta el infinito.Pues "ni tanto, ni tan calvo", además estos valores también pueden depender de tecnologías concretas, si es un proyecto nuevo o no, etc. ya que no se puede prentender que un proyecto que lleva 10, 5, X años sin preocuparse de estos valores pase de un día para otro a ciertos valores de calidad, esta corrección también lleva un esfuerzo de tiempo y dinero que hay que valorar. Si insistes en que quieres saber valores concretos, yo suelo recomendar unos valores por debajo un 1% o 2% de código duplicado, complejidad de método por debajo de 5 y de clase por debajo de 10. Pero ya os digo que no son valores que puedan valer en todos los casos y que deben revisarse en función de la cultura y contexto de la empresa/proyecto.

Ahora voy a la parte derecha del *dashboard*, veo que aquí sí que hay conceptos nuevos y que ya no están los que venía manejando hasta ahora. Es decir, ya no tengo la métrica "Cumplimiento de reglas" y tampoco el concepto de "Calidad Total". Estas métricas Sonarqube las ha deshechado para incorporar otras dos que podríamos decir que son las equivalentes "Deuda Técnica" (*Technical Debt*) y *"SQALE rating"*. Así que para saber el estado de salud de mis proyectos, primero tengo que saber qué siginifican y qué implican estas métricas. Vamos a ver qué son y como interpretarlas en el siguiente punto.

3. Deuda Técnica y *SQALE Rating*

Empiezo por el concepto de "Deuda Técnica" (*Technical Debt*). No quisiera extenderme mucho más de lo necesario y entrar en debates de qué se puede considerar o no "Deuda Técnica". El concepto de "Deuda Técnica" se basa en una metáfora expuesta por *Ward Cunningham* con la que se transmite en términos financieros la necesidad refactorizar para mantenerse en unos niveles de calidad técnica aceptables. Se ha incurrido previamente en una deuda por la que actualmente se están pagando intereses; es decir, la refactorización vendría a ser un modo de amortización de la deuda. Un buen resumen para entender el concepto de "Deuda Técnica" lo tenemos en este artículo de *Martin Fowler*; y si seguimos un punto de vista estricto se puede considerar que un mál código no estaría considerado como "Deuda Técnica", es simplemente un mal código, este razonamiento lo podemos encontrar en este post del blog de *Robert C. Martin*. En el siguiente vídeo tenemos al propio autor de la metáfora, *Ward Cunningham*, hablando sobre este tema:

Últimos Tutoriales del Autor

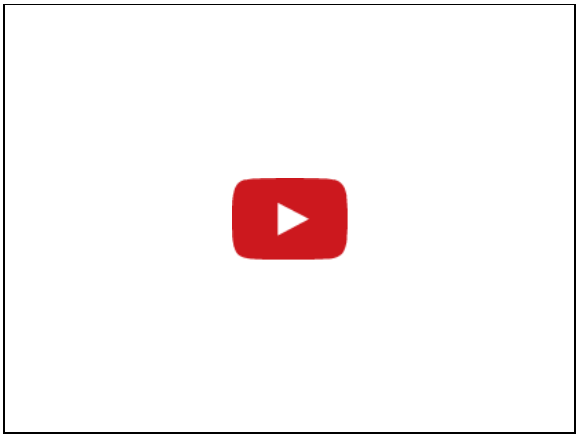
» Test de servicios REST con Spring MVC y Spring Test

» Maven, Jenkins, Sonar y tests de integracion

» Prototipado de pantallas con Pencil

» Token con caducidad en Spring Security

» Gestión de eventos en el cliente con el soporte Ajax de PrimeFaces



Y en este artículo hay una reflexión, comentarios, debate, y más analogías sobre qué es la "Deuda Técnica".

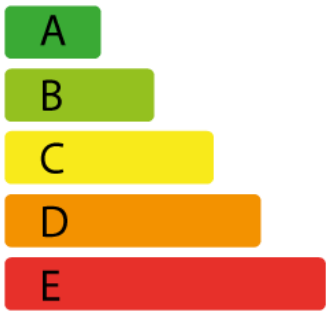
Esto esta muy bien, pero ¿Cómo Sonarqube distingue lo que sería "Deuda Técnica" de lo que simplemente es mal código? Pues efectivamente, no puede. Si nos fijamos, los valores de la métrica de "Deuda Técnica" están expresados en unidades de tiempo. Así que la interpretación de lo que Sonarqube considera como "Deuda Técnica" es el tiempo que se tendría que invertir para corregir esa carencia, ya sea porque es una mal código o porque es una deuda técnica asumida previamente.

A efectos prácticos, nos ayuda a entender que cuanta menor "Deuda Técnica" tenga un proyecto, podemos considerar que está en un mejor estado de salud. Y al tener asignado un peso temporal, puede servir para hacerse una idea del por qué se tarda tanto en incluir ciertos cambios y si es necesario asignar recursos para amortizar la deuda, refactorizar, o incluso si podemos considerar la "banca rota" del proyecto, no seguir invirtiendo en nuevas funcionalidades y como responsables pedir un presupuesto para que éstas sirvan como embrión de otro proyecto en el que la inversión de su desarrollo no implique seguir acumulando deuda.

Ahora voy a ver qué es eso de *SQALE Rating*. Bien, pues lo primero es ¿de dónde viene el nombre *"SQALE"*? su origen es *Software Quality Assessment based on Lifecycle Expectations* y que viene de un método con el mismo nombre que tiene por objetivo asegurar la cálidad del código fuente. Podemos ver en qué consiste con más detalle en www.sqale.org. Si vemos un poco el detalle de cómo se calcula, y cómo Sonarqube nos lo muestra, vemos que está intimamente relacionado con la "Deuda Técnica", más concretamente con el ratio de la "Deuda Técnica" (documentación en Sonarqube). Buscándole un sentido práctico, sirve para transmitir de forma rápida, breve y visual la calidad del proyecto al responsable dándole una categoría entre la A y la E siguiendo la siguiente escala:

- **A: Ratio de "Deuda Técnica" menor al 10%**. Podemos considerar que el proyecto está sano y no habría que hacer nada especial.
- **B: Ratio de "Deuda Técnica" entre el 10% y el 20%**. Se puede considerar que está en unos parámetros aceptables, pero hay que seguir de checkeo periódico para vigilar que no empeora. Se puede empezar a plantearse tomar medidas preventivas
- **C: Ratio de "Deuda Técnica" entre el 21% y el 50%**. Entramos en unos valores en los que la salud ya no es buena, hay que empezar a tomar medidas correctivas. Si bien el caracter de urgencia puede venir marcado por la desviación de los datos respecto a los valores aceptables.
- **D: Ratio de "Deuda Técnica" entre el 51% y el 100%**. Es urgente tomar medidas que lleven de vuelta el proyecto a zonas más saludables.
- **E: Ratio de "Deuda Técnica" superior al 100%**. Se puede decir que hemos entrado en la zona de rescate, hay que tomar una decisión entre apostar por un rescate, con todos los esfuerzos que esto significa de tiempo y dinero, o bien dejar que el proyecto entre en banca rota para buscar otro camino fuera del mismo.

Esta categorización del *SQALE Rating* por letras y colores, es muy similar a la que tenemos en la certificación energética tan de moda hoy en día. Donde el color aporta la información de lo bien construido que está el proyecto y el tamaño de la barra el esfuerzo que habría que hacer para llevarlo a una zona más confortable.



Así que ya sabes que gran parte del peso de los criterios de calidad de un proyecto para saber si está en un buen estado de salud o no recaen en el **"Ratio de Deuda Técnica"**. Pero ¿cómo se calcula realmente esta métrica? Bueno no es algo sencillo explicar, aunque se puede resumir en varios cálculos que comparan el esfuerzo de corregir las carencias detectadas frente a volver a construirlo desde cero. Para más información sobre cómo se calculan estos valores se pueden consultar estos enlaces: SQALE - Technical Debt Ratio, Tenchincal Debt.

4. ¡El tamaño NO es lo más importante!

Hasta aquí todo normal con estas métricas ¿verdad que aparentemente no hay nada raro? En principio basta con interiorizar estos nuevos conceptos y usarlas del mismo modo que usábamos los conceptos de cumplimiento de reglas anteriores. El caso es que me encontraba haciendo la auditoría de varios proyectos y tenía delante mío un código que ... digamos que ampliamente mejorable y dificilmente entendible. Mi experiencia previa en revisiones de calidad técnica me decía que ese proyecto no estaba bien, mi intuición me decía que cada vez que hubiese que introducir cualquier cambio era algo que se tardaba en hacer, con una alta probabilidad de que al mismo tiempo se colasen *bugs*, cosa que los clientes de dichos proyectos me confirmaron. El caso es que la calificación del "SQALE Rating" de dichos proyectos estaba en una aún verde y bonita "B".

Algo me decía que aquí había algo que no me habían contado respecto al "Ratio de Deuda Técnica" o "SQALE Rating". Me puse a ver que más había detrás del cálculo de esos valores y finalmente descubrí el *quid de la cuestión*: La métrica de "Ratio de Deuda Técnica" se basa en número de líneas de código ... *"¿PERO QUE C....?"* perdón mejor en inglés que suena más educado aunque seá lo mismo *"WTF!!!"*. ¿Otra vez primando el tamaño en Ingeniería del Software? Y me vino a mi mente una frase de mi profesor de universidad cuando nos explicaba la métrica del *punto función*: *¡El tamaño NO es lo más importante!*. Tenía que haber algo para cambiar ese criterio por defecto y efectivamente, se puede cambiar en *"Settings -> Technical Debt -*

> *Size Metric"* y eligiendo como valor la complejidad (*Complexity*).

Así que cambié estos valores y ví como aquellos proyectos pasaban de una verde y bonita "B" a otros valores no tan verdes y no tan bonitos.

Sabiendo esto, cambio el criterio para usar la complejidad y vuelvo a pasar el análisis a los dos proyectos de ejemplo. Viendo el panel de control que queda de esta forma:

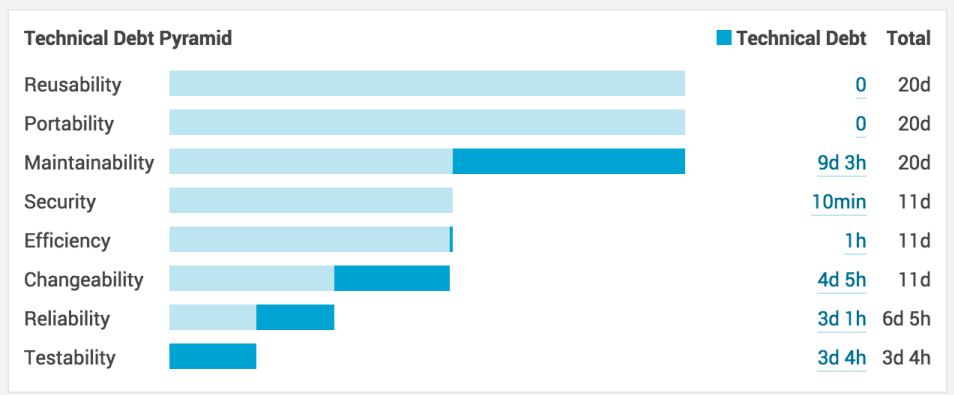
PROJECTS

QG	NAME	VERSION	SQALE RATING	TECHNICAL DEBT RATIO	DUP. LINES(%)	TECHNICAL DEBT	LOC	COVERAGE	IT COVERAGE	OVERALL COVERAGE
	project-A	0.0.2-SNAPSHOT	C	33.2%	2.7%	20d	6,624			
	project-B	1.0.1-SNAPSHOT	A	7.3%	0.0%	5d 3h	7,001			

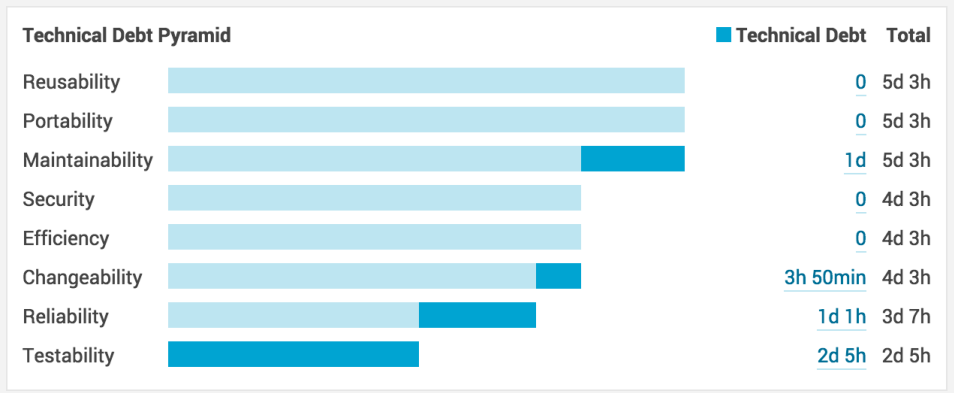
2 results

Vaya esto ya no se parece tanto. Así que voy a ver el detalle de la deuda técnica de cada uno de los proyectos. Para ver esto de un modo rápido, activo el *widget* "Technical Debt Pyramid" donde aparece un desglose por disintas categorías del esfuerzo que suma la deuda técnica de los proyectos.

Proyecto A:



Proyecto B:



¡Vaya! veo que un buen pedazo de "Deuda Técnica" se lo lleva la parte de las fiabilidad, *Reliability*, así que en siguiente punto voy a analizar qué pasa cuando incorporo pruebas a los proyectos de ejemplo.

5. Con tests ... me siento seguro

Hasta ahora he pasado los análisis de los proyectos A y B deshabilitando los tests en ambos proyectos para que no influyesen en la compración. Ahora voy a activarlos y espero obtener una mejora siendo la calidad de mis proyectos un poco mejor que el actual análisis sin tests. Así que me pongo con ello, activo los tests y vuelvo a ejecutar el análisis en ambos proyectos.

PROJECTS

QG	NAME	VERSION	SQALE RATING	TECHNICAL DEBT RATIO	DUP. LINES(%)	TECHNICAL DEBT	LOC	COVERAGE	IT COVERAGE	OVERALL COVERAGE
	project-A	0.0.2-SNAPSHOT	C	33.2%	2.7%	20d	6,624			
	project-B	1.0.1-SNAPSHOT	A	7.3%	0.0%	5d 3h	7,001		13.9%	

2 results

¡Sorpresa! No ha cambiado prácticamente nada en las métricas de los proyectos, excepto que me muestra el número de tests ejecutados, éxitos, fallos, etc. Pero no hay cambios respecto a la "Deuda Técnica" ni a su ratio. Y ¿qué pasa con la cobertura? en el "proyecto B" muestra información sobre la cobertura de los tests de integración pero no hay información de sobre la cobertura de los tests unitarios ni de la cobertura global.

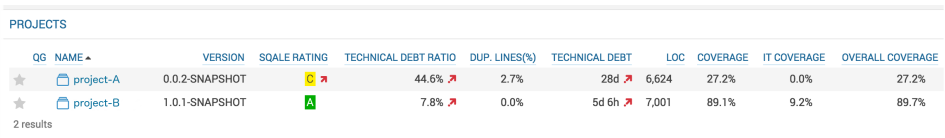
Hasta ahora Sonarqube analizaba estos datos, y es cierto que para que contabilizase todos los tests había que añadir alguna configuración especial, puedes consultar el tutorial de Maven, Jenkins, Sonar y tests de integracion si te interesa ver cómo se hacía en la versión anterior de Sonarqube, pero resulta que ahora sin cambiar nada en la configuración de los proyectos se han perdido estas métricas de cobertura. Así que me pongo a mirar que ha cambiado en Sonarqube hasta que doy con esta nota, Unit Test Execution in SonarQube, donde explica que a partir de la versiçon 4.2+, Sonarqube no va a calcular la cobertura junto con el análisis. Si quiero que en los proyectos se muestre la métrica de cobertura, previamente al análisis de Sonarqube hay que sacar un informe de cobertura de forma independiente, y pasar los ficheros de los informes de cobertura de forma explícita al análisis de Sonarqube.

Así que eso hago, cambio la configuración del fichero de Maven de los proyectos de ejemplo para obtener los informes de cobertura mediante el plugin de JaCoCo. Así que en el "pom.xml" tengo la configuración del plugin de JaCoCo tanto para tests unitarios como para tests de integración, y la configuración de generación de informes también para estos tipos de tests. El "pom.xml" quedaría de la siguiente forma:

```
...
```

```
                ...
                ${project.build.directory}/jacoco-it.exec
                ${project.build.directory}/jacoco.exec
        ...


                        ...
                                org.jacoco
                                jacoco-maven-plugin
                                0.7.2.201409121644


                                                pre-unit-test

                                                        prepare-agent


                                                        surefireArgLine
                                                        ${sonar.jacoco.utReportPath}
                                                        true



                                                pre-integration-test

                                                        prepare-agent-integration


                                                        failsafeArgLine
                                                        ${sonar.jacoco.itReportPath}
                                                        true



                                org.apache.maven.plugins
                                maven-surefire-plugin
                                2.15

                                        ${surefireArgLine}

                                        ${skip.unit.tests}

                                                **/*IT.java



                                org.apache.maven.plugins
                                maven-failsafe-plugin
                                2.15

                                                org.apache.maven.surefire
                                                surefire-junit47
                                                2.14.1



                                                integration-tests

                                                        integration-test
                                                        verify


                                                        ${failsafeArgLine}

                                                        ${skip.integration.tests}
                                                        true
                                                        ${project.build.directory}/surefire-reports


    ...
```

Con esta confiruación, ejecuto los análisis de Sonarqube que recibe los ficheros de los informes de cobertura y este es el resultado:



PROJECTS

QG	NAME	VERSION	SQALE RATING	TECHNICAL DEBT RATIO	DUP. LINES(%)	TECHNICAL DEBT	LOC	COVERAGE	IT COVERAGE	OVERALL COVERAGE
	project-A	0.0.2-SNAPSHOT	C	44.6%	2.7%	28d	6,624	27.2%	0.0%	27.2%
	project-B	1.0.1-SNAPSHOT	A	7.8%	0.0%	5d 6h	7,001	89.1%	9.2%	89.7%

2 results

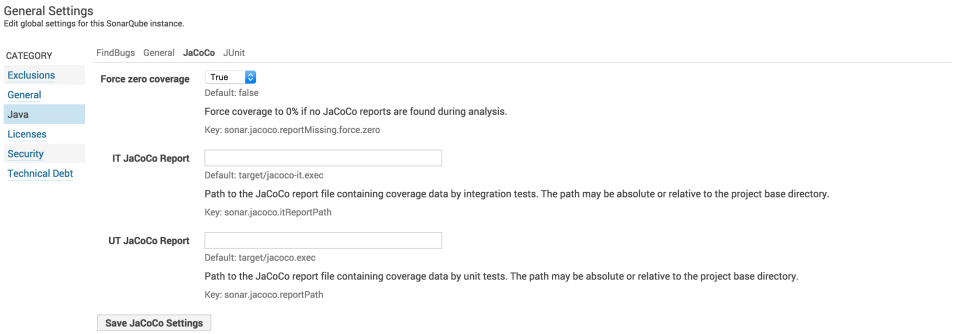
Y viendo este resultado va otro .... ***WTF!!!*** ¿Cómo es posible que por el mero hecho de incluir tests empeore la calidad de mi proyecto? ¿No debería ser al contrario? La lógica me dice que debería ser así; si tengo tests debería tener más fiabilidad en lo ya implementado y entre otras muchas cosas, al realizar un cambio se detectarían los posibles *bugs* introducidos antes de que

fuese demasiado tarde. Pero si te fijas en las métricas de "Deuda Técnica" y su ratio, resulta que no sólo ha disminuido si no que además ha aumentado.

Vuelvo a tener una sensación de que aquí falta algo que no me han contado. Hasta que al final descubro que el cálculo de deuda técnica tiene 2 formas de realizarse:

- Si no existen los informes de cobertura, ésta no se tiene en cuenta y no computa a la hora de sumar deuda.
- Si existen los informes de cobertura, ésta se tiene en cuenta y hay que sumar una deuda técnica del esfuerzo necesario para alcanzar una cobertura de un 80%. (ver http://docs.codehaus.org/display/SONAR/Technical+Debt+Calculation)

Desde mi punto de vista, no tener tests es parte de la deuda técnica, por lo que me interesa que siempre se tenga en cuenta la cobertura que éstos proporcionan. Al final encuentro el parámetro que me permite configurar Sonarqube para que siempre tenga en cuenta la cobertura en *"Settings -> Java -> JaCoCo -> Force zero coverage"* poniendo el valor a *"true"*.



Con esto lo que conseguimos es que si no se proporcionan informes de cobertura, ésta se ponga a un 0%, por lo que se sumará a la "Deuda Técnica" el esfuerzo necesario para llevarla desde el 0% hasta el 80%.

Si vuelvo a ejecutar el análisis de los 2 proyectos de ejemplo ignorando de nuevo los tests, pero habiendo configurado el *"Force zero coverage"*, puedo comprobar como la "Deuda Técnica" es mayor a la obtenida anteriormente.

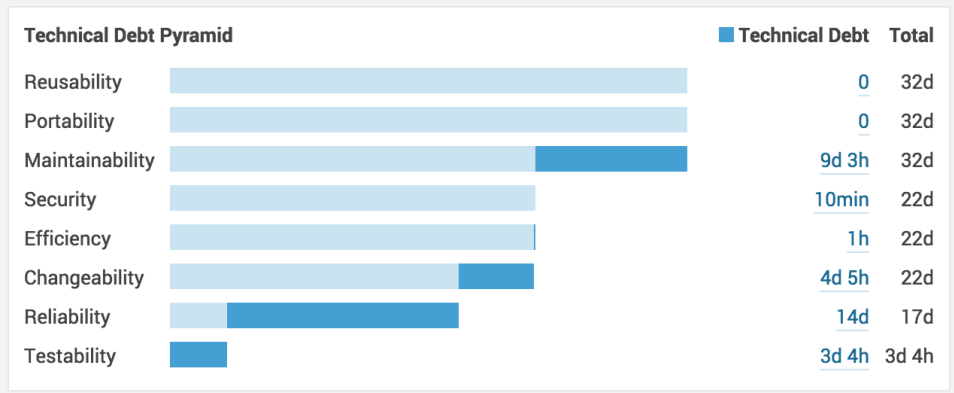
PROJECTS

QG	NAME	VERSION	SQALE RATING	TECHNICAL DEBT RATIO	DUP. LINES(%)	TECHNICAL DEBT	LOC	COVERAGE	IT COVERAGE	OVERALL COVERAGE
	project-A	0.0.2-SNAPSHOT	D	50.9%	2.7%	32d	6,624	0.0%	0.0%	
	project-B	1.0.1-SNAPSHOT	C	20.3%	0.0%	15d	7,001	0.0%	0.0%	

2 results

Y en la pirámide de "Deuda Técnica" tengo un valor significativo en la categoría de fiabilidad (muestro sólo la del proyecto A):

Proyecto A:



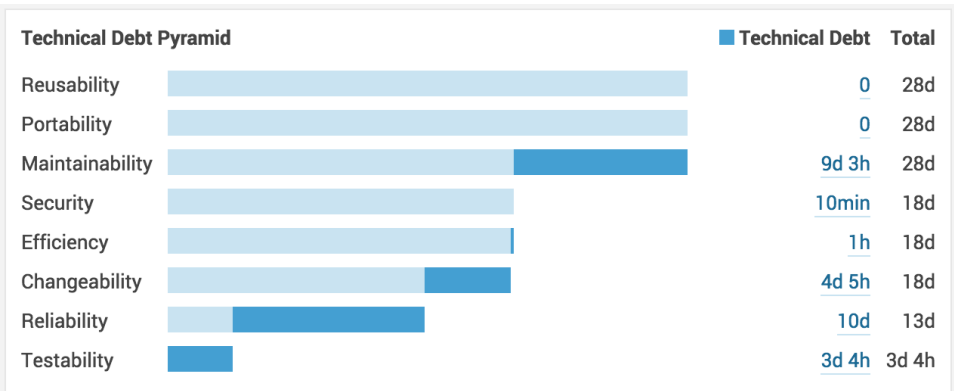
Vuelvo a ejecutar el análisis de los proyectos, con los tests otra vez activados, para tener la visión real de éstos una vez he realizado todos los cambios en la configuración de Sonarqube.

PROJECTS

QG	NAME	VERSION	SQALE RATING	TECHNICAL DEBT RATIO	DUP. LINES(%)	TECHNICAL DEBT	LOC	COVERAGE	IT COVERAGE	OVERALL COVERAGE
	project-A	0.0.2-SNAPSHOT	C	44.6%	2.7%	28d	6,624	27.2%	0.0%	27.2%
	project-B	1.0.1-SNAPSHOT	A	7.8%	0.0%	5d 6h	7,001	89.1%	9.2%	89.7%

2 results

Y también compruebo cómo el valor correspondiente a la categoría de fiabilidad de la pirámide de "Deuda Técnica" se ha reducido:



6. Conclusiones

¡Fíjate que resultado! dos proyectos que inicialmente estaban muy parejos ahora resulta que divergen al cambiar la configuración *out of the box* de Sonarqube. Por estas cosas son por las que al inicio te contaba que se podía llegar a tener una falsa sensación de tranquilidad. Y lo que es peor, se podría estar dando por buena la calidad de un proyecto cuando realmente habría que estar tomando medidas para mejorar dicha calidad.

Además, si se trabaja con la configuración por defecto, sin haber cambiado nada, uno no entendería porque en dos proyectos con una calidad técnica aparentemente similar, en uno apenas hay problemas y el impacto y esfuerzo real de introducir un cambio es relativamente bajo, mientras que en el otro proyecto puede ser muy variable, incluso alto. Pues bien, como has visto, esto se debe a que la calidad técnica de los 2 proyectos no es tan similar.

Con esto lo que quiero decir, es que por norma general las herramientas nos facilitan gran parte de nuestro trabajo, pero no debemos fiarnos sin más de los datos que nos presentan. Debemos ser capaces de saber si esos datos son coherentes, y en caso de no serlo buscar una explicación, como que no estamos usando bien la herramienta o no la tenemos bien configurada.

Espero que esta información te sea de utilidad y haberte transmitido un poco la preocupación por la calidad técnica del código y las consecuencias que puede tener no cuidarla o no hacer un correcto seguimiento de la misma.

7. Referencias

- http://martinfowler.com/bliki/TechnicalDebt.html
- https://sites.google.com/site/unclebobconsultingllc/a-mess-is-not-a-technical-debt
- https://www.youtube.com/watch?v=pqeJFYwnkjE
- https://medium.com/@joaomilho/festina-lente-e29070811b84
- http://www.sqale.org/details
- http://www.sonarqube.org/sqale-the-ultimate-quality-model-to-assess-technical-debt/
- http://www.sonarqube.org/evaluate-your-technical-debt-with-sonar/
- http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=Maven_Jenkins_Sonar_y_tests_de%20integracion
- http://www.sonarqube.org/unit-test-execution-in-sonarqube/
- http://www.eclemma.org/jacoco/trunk/doc/maven.html