CRITERIOS ESTÁNDAR DE CALIDAD

MEDIDA	META	COMENTARIO
Porcentaje Libre de Defectos		
Compilación	> 10%	
Pruebas de Unidad	> 50%	
Pruebas de Integración	> 70 %	
Pruebas del Sistema	> 90 %	
Defectos por cada mil líneas de código (KLOC)		
Total de defectos inyectados	100-200	
Compilación	< 10	Todos los defectos indicados por el compilador
Pruebas de Unidad	< 5	Sólo los defectos mayores
Pruebas de Integración	< 0.5	Sólo los defectos mayores
Pruebas del Sistema	< 0.2	Sólo los defectos mayores
Proporciones para los defectos		
Defectos DLD/defectos pruebas unidad	> 2.0	Sólo los defectos mayores
Defectos código revisado/defectos compilación	> 2.0	Sólo los defectos mayores
Proporciones para el Tiempo de Desarrollo		
Inspección requerimientos/tiempo requerimientos	> 0.25	Incluye tiempo de recolección
Inspección HLD/Tiempo HLD	> 0.5	Sólo trabajo de diseño, no los estudios
DLD/tiempo codificación	> 1.00	
Revisión DLD/Tiempo DLD	> 0.5	
Revisión código/tiempo codificación	> 0.5	
Ritmo de las revisiones e inspecciones		
Páginas de requerimientos/hora	< 2	Páginas con texto con espaciado sencillo
Páginas HLD/hora	< 5	
Líneas de texto DLD/hora	< 100	1 de seudocódigo equivale ~3 líneas de código
Líneas de código/hora	< 200	Líneas de código lógico
Ritmo de inyección de defectos		
Defectos requerimientos/hora	0.25	Sólo los defectos mayores
Defectos HLD/hora	0.25	Sólo los defectos mayores
Defectos DLD/hora	2.0	Sólo defectos de diseño
Defectos código/hora	4.0	Sólo los defectos mayores
Defectos Compilación/hora	0.3	Sólo los defectos indicados por el compilador
Defectos pruebas de unidad/hora	0.2	Sólo los defectos mayores
Ritmo de remoción de defectos		
Defectos inspección requerimientos/hora	0.5	Sólo los defectos mayores
Defectos inspección HLD/hora	0.5	Sólo los defectos mayores
Defectos revisión DLD/hora	2.0	Sólo los defectos de diseño
Defectos inspección DLD/hora	0.5	Sólo los defectos de diseño
Defectos revisión código/hora	6.0	Sólo los defectos mayores
Defectos inspección código/hora	1.0	Sólo los defectos mayores
Resultados por fase		
Inspecciones a los requerimientos	~70%	Sin contar comentarios editoriales
Revisiones e inspecciones al diseño	~70%	Usando análisis de estados
Revisiones e inspecciones al código	~70%	Usando listas de chequeo personales
Compilación	~50%	90+ % de defectos de sintaxis
Pruebas de unidad con 5 o menos defectos/KLOC	~90%	Para más defectos/KLOC: 50-75%
Integración, pruebas del sistema < 1.0 defectos/KLOC	~80%	Para más defectos/KLOC: 30-65%
Resultados por proceso		
Antes de compilación	> 75%	Asumiendo buenos métodos de diseño
Antes de pruebas de unidad	> 85%	Asumiendo pruebas lógicas en las revisiones
Antes de integración	> 97.5%	Para pequeños productos, máximo 1 defecto
Antes de pruebas del sistema	> 99%	Para pequeños productos, máximo 1 defecto

EXPLICACIÓN DE TÉRMINOS

***Porcentaje Libre de Defectos**:* mide el porcentaje en el cual un componente de un producto no tiene defectos en una fase dada. Por ejemplo, si un producto tiene cinco componentes y cuatro de ellos tienen defectos de compilación en la fase 1, el producto tendrá un 20% libre de defectos. Esto es cierto sin importar el número de defectos de compilación de cada parte. De manera similar, si solo dos de las cinco partes tienen defectos en la fase de pruebas de integración, el producto tendrá un 60% libre de defectos en las pruebas de integración. Hay que tener presente que si un producto solo consta de una parte, cualquier defecto encontrado resultará en un producto con el 0% libre de defectos para la fase siendo considerada. La tendencia ideal de este indicador de calidad es que se vaya incrementando paulatinamente, de lo contrario indicará que existen problemas de calidad.

***Defectos por cada mil líneas de código**:* primero, un defecto es cualquier elemento del sistema que no concuerda con lo especificado en los requerimientos, un elemento de diseño o la implementación y que, si no se corrige, hará que la discordancia persista. Un defecto mayor es cualquier problema que cuando se arregla obliga a cambiar el ejecutable. En consecuencia, los defectos en comentarios o apariencia del código escrito generalmente son considerados menores.

***Proporciones para los defectos**:* provee una indicación de la calidad del diseño y las revisiones de código. Por ejemplo, la experiencia muestra que cuando los ingenieros encuentran dos veces más de defectos cuando hacen la revisión de código comparados con los hallados durante la compilación, es porque generalmente han hecho una buena revisión de código. Este caso produciría una proporción de defectos código revisado/compilación de 2.0. Algo similar ocurre con la proporción entre los defectos hallados en la revisión del diseño y el número hallado en las pruebas de unidad. Si, por ejemplo, en un momento dado la proporción es de 0.96 y lo planeado fuera de 2.0, esto significaría que el equipo se debería concentrar en mejorar el tipo de revisión que presente este comportamiento.

***Proporciones para el tiempo de desarrollo**:* es otra manera de evaluar un proceso. Cuando los ingenieros invierten más tiempo en el diseño detallado (DLD) que en la codificación, por ejemplo, la experiencia muestra que generalmente producen un diseño de calidad. Esta afirmación no está garantizada, pero es altamente probable. De manera similar, cuando invierten más del 50% tiempo del diseño detallado en una revisión de ese diseño detallado, la experiencia muestra que esa revisión fue razonablemente completa. Una vez más, esto no está garantizado, pero es altamente probable. Para los requerimientos, una regla razonable es invertir alrededor del 25% o más del tiempo dedicado a los requerimientos a la inspección de los mismos. Para el diseño de alto nivel, un objetivo razonable es invertir 50% o más del tiempo de diseño en revisiones e inspecciones.

***Ritmo de las revisiones e inspecciones**:* para lograr una alta calidad, hay que planear invertir el suficiente tiempo en revisiones e inspecciones. Recuerde que el tiempo total de la inspección es la suma de todos los tiempos utilizados en la preparación mas el tiempo de la reunión de revisión multiplicado por el número de ingenieros en la reunión. Aunque es importante un ritmo lento para las revisiones, el solamente gastar tiempo en una revisión no garantiza que encuentre defectos. Es necesario utilizar métodos efectivos de revisión. Para las revisiones de código e inspecciones, utilice una lista de chequeo personal y actualícela frecuentemente con base en los

defectos hallados en la compilación y pruebas de los programas que haya revisado e inspeccionado.

***Ritmo de inyección de defectos**:* una de las características más comunes de los programadores es que inyectan una gran cantidad de defectos. Cuando se tienen datos acerca del ritmo con que se inyectan los defectos, se tiene una base para la estimación de cuántos defectos se inyectarán durante cada fase del trabajo de programación. Luego, solo necesita multiplicar el ritmo esperado de inyección de defectos por el tiempo esperado a ser invertido en cada fase. Este cálculo le dará un estimado razonable de los defectos que se pueden esperar inyectar.

***Ritmo de remoción de defectos**:* es posible calcular la remoción de defectos para el trabajo de programación. Se ha encontrado que el ritmo de remoción en la revisión del código va desde cero hasta más de 20 defectos por hora. El ritmo más probable es alrededor de seis defectos/hora. Para las revisiones del diseño, los resultados típicos son igualmente diversos: dos o más defectos removidos por hora.

***Resultados por fase**:* se refiere al porcentaje de defectos en un programa que fueron removidos durante una fase dada. En consecuencia, si tiene 19 defectos en un programa al momento de iniciar la revisión del código, se inyecta un defecto durante la revisión del código, y se encuentran 15 de estos defectos en la revisión, se tendría un 75% de resultados para la revisión del código. Es decir: resultado = 100*(defectos hallados)/(defectos en el producto) = 100*15/(19+1) = 75%. Esta misma fórmula se puede aplicar a cualquier fase del proceso.

***Resultados por proceso**:* mide el porcentaje de defectos removidos antes de una fase dada. Por lo tanto, el resultado del proceso antes de la compilación mide el porcentaje de los defectos inyectados antes de la compilación que fueron removidos antes de la compilación.

Un ejemplo completo de cómo calcular los resultados por fase y por proceso. Los resultados, por ejemplo, de la revisión se refieren al porcentaje de defectos en el diseño o el código al momento de la revisión, que fueron encontrados por esa revisión. Esta medida no puede ser calculada de una manera precisa hasta que el programa revisado no haya sido probado completamente y haya sido usado de manera intensa. Sin embargo, se pueden hacer ciertas aproximaciones a estos resultados de la revisión. Una manera fácil de hacerlo es construir una tabla con la estructura que se muestra a continuación. Las columnas defectos inyectados y defectos removidos se llenan contando el número de defectos registrado en su histórico de defectos (esto significa que hay que ir llevando cuenta de la aparición de los defectos) que fueron inyectados y removidos en cada fase. Las columnas acumulado de inyectados y acumulado de removidos son las sumas acumuladas de cada columna a nivel de cada fase. Los defectos escapados para cada fase se pueden calcular restando, para cada fila, la columna acumulado de removidos de la columna acumulado de inyectados. Los resultados para cada fase y por proceso se pueden calcular utilizando las siguientes fórmulas:

$$\text{Resultados\_por\_fase} = 100 * \frac{defectos\_removidos\_en\_la\_fase}{defectos\_removidos\_en\_la\_fase + defectos\_escapados}$$

$$\text{Resultados\_por\_proceso} = 100 * \frac{defectos\_removidos\_antes\_de\_la\_fase}{defectos\_removidos\_antes\_de\_la\_fase + defectos\_escapados}$$

Ejemplo de un Resumen de Defectos					
Fase	**Defectos Inyectados**	**Defectos Removidos**	**Acumulado Inyectados**	**Acumulado Removidos**	**Defectos Escapados**
Planeación	1	0	1	0	1
Diseño detallado (DLD)	5	0	6	0	6
Revisión del DLD	0	3	6	3	3
Codificación	15	1	21	4	17
Revisión codificación	0	8	21	12	9
Compilación	0	6	21	18	3
Pruebas	0	3	21	21	0
Total	21	21			

Haciendo uso de la fórmula de resultados por fase tendríamos:

$$\text{Re}\,sultado\_revisión\_del\_diseño = 100 * \frac{3}{3+3} = 50\%$$

$$\text{Re}\,sultado\_revisión\_del\_código = 100 * \frac{8}{8+9} = 47.1\%$$

$$\text{Re}\,sultado\_de\_la\_compilación = 100 * \frac{6}{6+3} = 66.7\%$$

De manera similar, haciendo uso de la fórmula de resultados por proceso tendríamos:

$$\text{Re}\,sultado\_antes\_de\_la\_codificación = 100 * \frac{3}{3+3} = 50\%$$

$$\text{Re}\,sultado\_antes\_de\_la\_compilación = 100 * \frac{3+1+8}{3+1+8+9} = 100 * \frac{12}{21} = 57.1\%$$

Cómo darle tratamiento a una baja calidad. Cuando los datos de calidad sugieran que existen problemas, mire los datos de esa parte para encontrar la causa. La clave es mirar aquellas partes que tienen unos números muy altos de defectos en diferentes fases. Para encontrar la fuente de los problemas, primero mire los datos de los módulos y observe cuáles de ellos tienen datos pobres de defectos, proporciones, ritmos, y resultados. Luego mire tanto el diseño del programa y su respectivo código fuente para entender por qué el programa está generando problemas. Si el diseño parece bien, el problema probablemente proviene de prácticas deficientes de programación, revisiones de código inefectivas, o inspecciones pobremente realizadas.

Tomado de:

Introduction to the Team Software Process. Watts S. Humphrey. Addison-Wesley. 2000. Pag. 97-107.
A Discipline for Software Engineering. Watts S. Humphrey. Addison-Wesley. 1995 (8th printing September 1999). Pag. 248-252.