

Medir el proceso de control de configuración, ¿una utopía para la Industria Nacional de Software?

MsC. Lic. Ailyn Febles Estrada

Instituto Superior Politécnico “José Antonio Echeverría”

La Habana, Cuba

afebles@ceis.ispjae.edu.cu

Ing. Isabel Pérez Estévez

Instituto Superior Politécnico “José Antonio Echeverría”

La Habana, Cuba

Resumen

Los métodos de producción de software están evolucionando desde formas artesanales a la producción industrial en gran escala. La industria de software cubana no está ajena a esos cambios. Para incidir positivamente en su desarrollo y lograr establecer en ella parámetros de excelencia es imprescindible implantar modelos de procesos tomando en consideración las mejores prácticas internacionales y adaptándolas creativamente a las condiciones concretas del país. Para esto, entre otros aspectos, es necesario **medir**.

El área de las mediciones de software, a pesar de ser una de las áreas en la ingeniería de software donde se ha investigado desde hace más de 30 años, todavía no ha sido bien comprendida, ni ampliamente aplicada en la industria, por tanto, la implementación de las mediciones en una organización requiere un cambio tecnológico, educacional y cultural importante. Las autoras, proponen un conjunto de métricas para aplicar en el proceso de Control de Configuración. Estas métricas puedan ser obtenidas a partir del modelo de procesos de control configuración definido en trabajos anteriores por el grupo de Ingeniería de Software del CEIS. También se enuncian, algunas reglas básicas a tener en cuenta cuando se aplican estas métricas en una empresa.

Palabras Claves: Gestión, Configuración, métricas, defectos, cambio.

1. Introducción

El software se ha convertido en un tema crítico en la sociedad moderna mundial. Todos parecen necesitar mejores software en menos tiempo y a menor costo. Los métodos intuitivos de desarrollo de software que se usan actualmente son, básicamente, aquellos que los propios individuos artesanalmente siguen, los cuales sólo servirán mientras la sociedad pueda tolerar la falta de predicción que ellos acarrearán.

Siguiendo la idea de que cuando un modelo de procesos estándar ha sido especificado, el desarrollo de software puede ser un proceso definido, repetible, en lugar de una actividad ad hoc reinventada para cada nuevo proyecto [10]; en trabajos anteriores, se propuso el Proceso de Control de configuración, el cual incluye el Proceso de Control de Cambios y el Proceso de control de versiones, y para la industria de software cubana

Con la aplicación de estos procesos es posible, además de disciplinar a los involucrados, almacenar y disponer de los datos históricos necesarios para lograr un trabajo más predecible y eficiente. No obstante, hasta ese punto la aplicación del proceso solo garantiza la recolección de los datos. Para que los proyectos que sigan el modelo definido puedan ser planificados, supervisados (monitoreados), y controlados, será necesario definir y usar métricas, las cuales se pueden definir en términos de los componentes del propio modelo de procesos.

El objetivo de este trabajo es presentar una selección y definición de métricas para la Gestión de Configuración que permitan, en un proyecto de software, tomar acciones correctivas a tiempo y mejorando sucesivamente los procedimientos definidos.

En este artículo se presenta una selección de métricas para el proceso de Control de Configuración de Software adaptados a procedimientos y metas estándares de la Industria Nacional de Software en Cuba que pudieran ser utilizados por pequeñas o medianas empresas, así como un grupo de sugerencias para la aplicación de estas métricas.

2. La calidad de software

La administración de la calidad total (TQM) es un término que se originó en 1985, que describe un estilo de administración japonés para mejorar la calidad. La TQM toma varios significados en dependencia de quien lo interprete y de cómo lo apliquen. En general representa un estilo de administración dirigido a lograr éxitos a largo plazo enlazando la calidad con la satisfacción del

cliente. Según Deming, Juran y Crosby, en la TQM, las métricas y el análisis de estas, son los elementos fundamentales para lograr un mejoramiento continuo de la calidad.

Varios ambientes de trabajo han sido propuestos para mejorar la calidad que pueden ser usados para substantiar la filosofía de TQM. Algunos ejemplos son [7]:

- Plan-Do-Check-Act
- Quality Improvement Paradigm (QIP) / Experience Factory Organization
- Capability Maturity Model (CMM) del Software Engineering Institute (SEI)
- Lean Enterprise Management

2.1 La Gestión de configuración de software

Según definición de la IEEE una configuración está conformada por requisitos, artefactos de diseño y de implementación que definen una versión particular de un sistema o de un componente de un sistema en un momento dado. [15]

La Administración de Configuración de Software es la disciplina de la Ingeniería de Software que comprende las herramientas y técnicas (procesos o metodologías) que una organización utiliza para administrar las configuraciones de los componentes de software. Tiene como objetivo mantener la integridad de los componentes del producto de software, evaluar y controlar los cambios sobre ellos así como facilitar la visibilidad del producto a todo el equipo de proyecto.[2][3][5]

La definición más utilizada de Gestión de Configuración de Software (GCS) es la de la IEEE que establece: [15]

“Gestión de configuración es la disciplina que abarca todo el ciclo de vida de la producción de software y productos asociados. Específicamente, requiere de la identificación de los componentes a controlar y la estructura del producto, controla todos los cambios sobre los elementos y garantiza mecanismos para auditar todas las acciones”

3. El modelo de madurez de las capacidades del SEI (CMM)

“Los procesos son como los hábitos: difíciles de establecer e incluso mucho más difíciles de romper” [5]. Los procesos para desarrollar software a gran escala, pueden ser muy grandes y complejos. Pueden ser difíciles de definir e incluso mucho más difíciles de introducir. Por esta razón fue que el Software Engineer Institute (SEI) de la universidad Carnegie-Mellon desarrolló

un ambiente de trabajo (framework) de madurez de procesos de software [4]. Este framework es una forma ordenada para las organizaciones de determinar las capacidades de sus procesos actuales y establecer prioridades en su mejora. Se hace a través del establecimiento de 5 niveles de prioridad progresivos logrando procesos de capacidades más maduros [11]. Por cada nivel se han definido los principios elementales o KPAs (Key Process Areas) que proveen las metas y ejemplifican las prácticas a llevar a cabo (figura 1).

CMM ha sido revisado y refinado por varios especialistas y representan el mejor juicio actual y el método más efectivo para conseguir los objetivos de cada nivel de madurez.

	Características	KPAs	Resultado
5 Optimizado	Bases cualitativas para una inversión capital continua en la automatización y mejoramiento de la retroalimentación de los procesos.	-Prevención de defectos -Administración de la tecnología de cambio -Administración del proceso de cambio	Productividad y calidad
4 Administrado	(cuantitativo) -Procesos medidos -Control estadístico razonable sobre la calidad del producto	-Medición y análisis de los procesos -Administración de la calidad	
3 Definido	(cuantitativo) -Costo y planificación confiables -Procesos definidos e institucionalizados -Rendimiento de la calidad mejorado pero impredecible	-Definición y mejoramiento de los -procesos organizacionales -Programas de entrenamiento -Administración integrada de software -Coordinación intergrupar -Inspecciones entre compañeros -Ingeniería del producto de software	
2 Repetible	(intuitivo) -Procesos dependientes de los individuos - Costo y calidad altamente variables - Métodos y procedimientos informales y “ad hocs”	-Administración de los requerimientos -Planificación y vigilancia de los proyectos de software -Administración de la subcontratación de software -Aseguramiento de la calidad del software -Administración de la configuración de software	
1 Inicial	-Ad hoc -Caótico -costos, planificación y rendimiento de la calidad impredecibles		

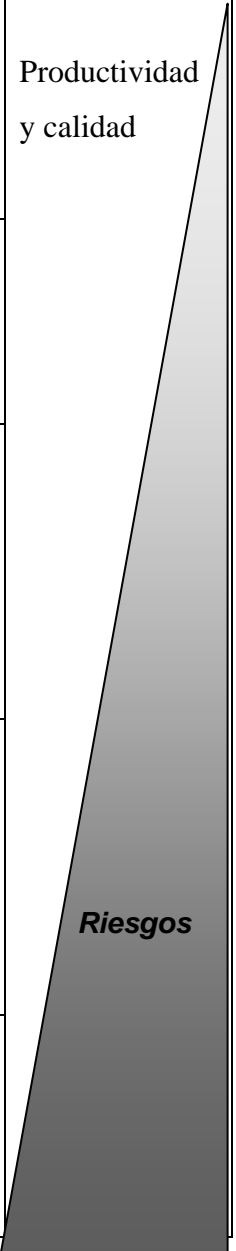


Figura 1 Representación esquemática de CMM

3.1 PSP: Proceso personal de software

El proceso personal de software (PSP, Personal Software Process) es un proceso de auto mejoramiento diseñado para ayudar a controlar, administrar y mejorar la forma en que se trabaja individualmente. Está estructurado por formularios, guías y procedimientos para desarrollar

software. Si es usado apropiadamente, brinda los datos históricos necesarios para trabajar mejor y lograr que los elementos rutinarios del trabajo sean más predecibles y eficientes [5].

Usando PSP, se pueden construir programas de más de 10 000 líneas de código (LOC), sin embargo, hay dos problemas típicos en los grandes programas. Primero, mientras se crece en tamaño, también lo hace el tiempo y el esfuerzo requerido. Esto puede ser un problema particular si sólo existe un ingeniero en el proyecto. Segundo, la mayoría de los ingenieros tienen problemas en la visualización de todas las facetas importantes de un programa, incluso cuando su tamaño es moderado. Existen muchos detalles e interrelaciones que deben tenerse en cuenta, muchas dependencias lógicas, interacciones en el tiempo o condiciones excepcionales. Una de las formas más poderosas de resolver estos problemas es el proceso de software del equipo (Team Software Process, TSP).

3.2 TSP: Proceso de software del equipo

Una definición acertada de un equipo es la dada por Dyer [1] [6], donde un equipo consiste en al menos dos personas que trabajan para lograr una meta/ objetivo/ misión común, donde cada persona tiene asignado un rol específico o funciones específicas que desarrollar, y donde el completamiento de la misión requiere alguna forma de dependencia entre los miembros del equipo.

El proceso del equipo de software (TSP, Team Software Process) es un proceso que al igual que el PSP, está basado en el modelo CMM, TSP está diseñado para ayudar a controlar, administrar y mejorar la forma en que trabaja un equipo de software. Al igual que PSP, está estructurado por formularios, guías y procedimientos para desarrollar software. [6].

4. Justificación de la selección del modelo CMM

La propuesta de métricas se basa fundamentalmente en el modelo CMM con las especificaciones que introduce TSP y PSP, porque a criterio de la autora, es el más completo de los estándares analizados. Entre otras, fueron valoradas las siguientes razones: [3][5][15][16] [21]

1. Es un modelo más detallado y dirigido específicamente a los productos de software.
2. Exige la calidad en el cumplimiento de los procedimientos y las instrucciones.
3. Guía la empresa a convertirse en empresa madura

4. Se logra un incremento de la productividad, mejor comunicación entre los profesionales y con los clientes, así como una mayor satisfacción de estos.
5. Identifica los procesos de calidad requeridos para la producción de software, y define las actividades a desarrollar.
6. Los modelos de evaluación (assessment) basados en CMM ayudan a la organización a realizar las auditorías internas y medir la efectividad del proceso.
7. CMM provee una base para definir objetivos medibles de calidad (en términos de los niveles de CMM) y planear las actividades de mejoramiento.

Con la aplicación del modelo CMM se conduce a las empresas por tres líneas fundamentales[21]:

- Cómo garantizar el control de sus procesos para desarrollar y mantener software.
- Cómo evolucionar hacia una cultura de Ingeniería de Software y de administración de excelencia.
- Cómo desarrollar las empresas hasta que se consideren empresas maduras.

5. Mediciones

Para lograr una buena calidad del producto es necesario identificar las mediciones y los criterios que serán utilizados para identificar el nivel deseado de la calidad y determinar si se está alcanzando. Las mediciones describen el método para capturar los datos que serán utilizados para evaluar la calidad, mientras que los criterios definen el nivel o el punto en el cual el producto logra la calidad aceptable (o inaceptable). [12][18][19][20]

Es una afirmación indiscutible que las mediciones son cruciales en el progreso de todas las ciencias. El progreso científico se logra a través de observaciones y generalizaciones basadas en datos y mediciones y la derivación de teorías como resultado de estas [7]. Sin la verificación a través de los datos y las mediciones, las teorías y las proposiciones permanecerían en un nivel abstracto.

Los principios fundamentales que deben seguir las métricas son [12]:

- deben ser simples, objetivas, fáciles de coleccionar, fáciles de interpretar y difíciles de malinterpretar,
- su recolección debe ser automática y no intrusiva, o sea, no interferir en las actividades de los desarrolladores,

- deben contribuir a la evaluación de la calidad temprana en el ciclo de vida, cuando los esfuerzos por mejorar la calidad del software son efectivos,
- los valores absolutos y las tendencias de las métricas, deben ser usados activamente por el personal administrativo y el personal ingenieril, para comunicar progreso y calidad en un formato coherente,
- la selección de un mínimo o más extensivo conjunto de métricas, dependerá de las características y contexto del proyecto: Si es muy grande o si tiene restricciones de seguridad o de confiabilidad de los requerimientos; y si el equipo de desarrollo y de valoración (evaluación) es conocedor de las métricas, lo cual hará muy útil coleccionar y analizar las métricas técnicas.

5.1 Métricas analizadas

Las métricas de calidad de software son un subconjunto de las que se enfocan al producto, al proceso y al proyecto. Estas métricas pudieran ser divididas en; las que miden la calidad del producto final y las métricas del proceso en si.

Las métricas analizadas fueron seleccionadas según los criterios de Kan [7] y estuvieron alrededor de los siguientes aspectos:

- Densidad de defectos
- Problemas de los usuarios
- Satisfacción del cliente
- Densidad de defectos detectados en las pruebas
- Defectos eliminados por fase
- Efectividad en la eliminación de defectos

La revisión de estas métricas, teniendo en cuenta las metas de las organizaciones de la Industria Nacional de Software (INS) y el proceso en concreto que estamos analizando (Control de Configuración) arrojó un conjunto de criterios para seleccionar que métricas introducir en esta primera etapa y como introducirlas en la industria.

5.2 Criterios de selección

Los criterios de selección giraron en torno a la información existente en la INS hoy y a los instrumentos de control que se proponen introducir con el proceso de control de Configuración y

que servirán para almacenar datos en la empresa. También se agruparon y clasificaron según un subconjunto de las 683 métricas para las áreas de proceso de los niveles 2,3,4,5 de CMM que publicó el SEI en el año 2000 [13]. Dentro de las más importantes para el control de configuración se encuentran las siguientes:

1. Esfuerzo real realizado para las tareas de control de configuración.
2. Tareas terminadas en el proceso de control de configuración
3. Resumen y estado de las solicitudes de cambio.
4. Estimado del trabajo actual completado en las actividades de control de configuración.
5. Esfuerzo estimado en las actividades de control de configuración.
6. Número de solicitudes de cambio liberadas por unidad de tiempo.
7. Resumen de las solicitudes de cambio
8. Resumen de los defectos, incluidos los resueltos

Se incluyeron en esta propuesta un conjunto de métricas asociadas con las peticiones de cambio, este término es manejado por las organizaciones y además se almacena en las empresas alguna información de estas. En los procedimientos que se proponen [2] para el proceso de Control de Configuración se incluye este elemento como un instrumento de control fundamental.

El esfuerzo es analizado en la solución a las peticiones de cambios sobre el producto como una de las tareas del control de configuración.

Los defectos fueron incluidos por ser generadores de peticiones de cambio y de ellos fueron tenidos en cuenta elementos como la eficiencia en su eliminación, resúmenes según diferentes criterios, frecuencia de defectos, etc.

De manera general las métricas deben ser sencillas, claras de interpretar y asociadas con el esfuerzo y el tiempo que se dedica a resolver un pedido de cambio y/o un defecto según R. Preeman en la última edición de su libro “Ingeniería de Software, un enfoque práctico” [17]

6. Métricas definidas

6.1 Pedidos de cambio

6.1.1 Estado de los pedidos de cambio

Con este resumen se puede llegar a la conclusión clara de en que estado están estancado los pedidos de cambio y por tanto darle una advertencia al jefe del proyecto de donde tiene que insistir e incluso si es necesaria una reunión de la Junta de Control de Cambios (Figura 2).

En el proceso están definidos 5 estados de las peticiones de cambio:

- Aprobada: Peticiones de cambio que ya han sido aprobadas por la junta de control de cambios pero no se han comenzado a desarrollar por un especialista
- Cerrada: Petición de cambio que ya fue resuelta
- En cola: Petición de cambio por analizar por la junta de control de cambios.
- En desarrollo: Petición de cambio aprobada por la junta que está siendo resuelta por un desarrollador
- Posible rechazada: Solicitudes de cambio que la junta tiene pendiente por tener incompleto los datos.

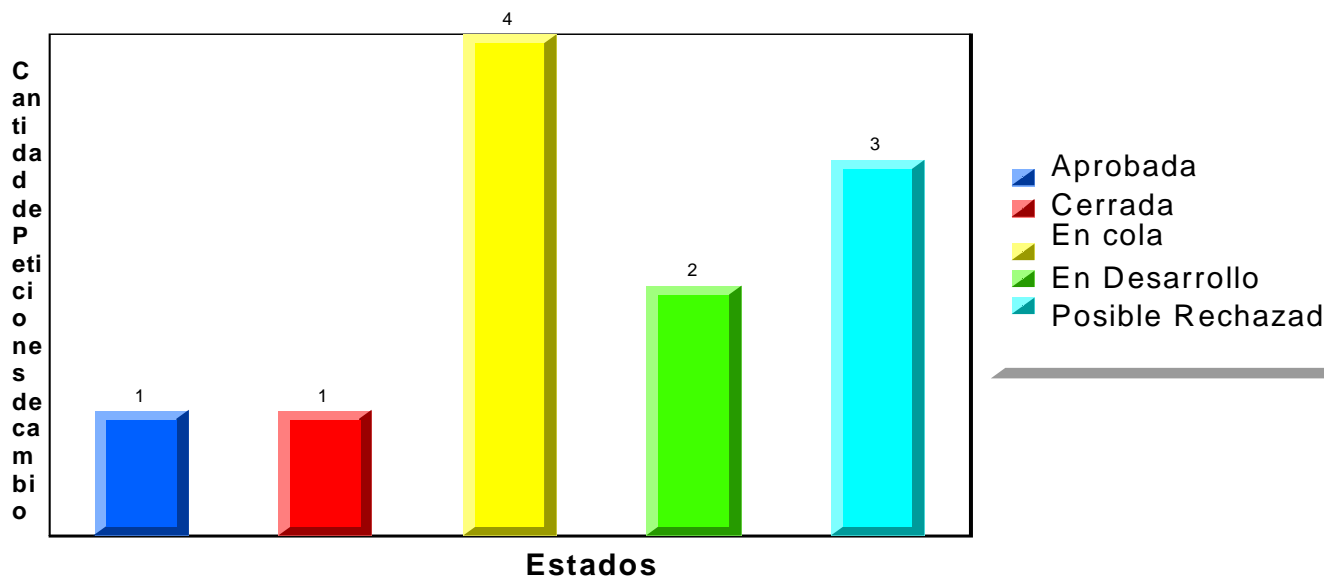


Figura 2 (Falta el título de la figura)

6.1.2 Algunos resúmenes

Resulta interesante para los directivos tener acceso, además de a la información general, a listados de las peticiones de cambio por categoría, por categoría en una unidad de tiempo, asignadas a un desarrollador en particular, etc.

Estos listados deben ser adicionados en los sistemas que se propongan para estos procesos en tablas estándares para los jefes de proyectos y los directivos.

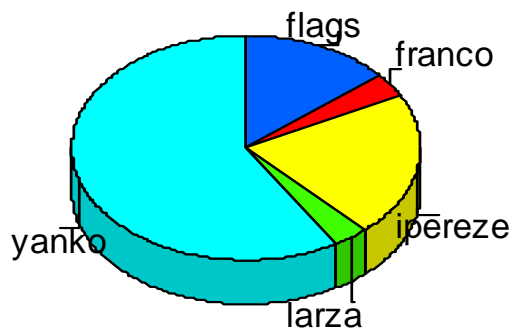
6.1.3 Esfuerzo del personal en la solución de los cambios

Para que el jefe del equipo tenga más elementos a la hora de asignar, reasignar carga de trabajo y analizar el esfuerzo del personal bajo su mando se definieron las siguientes mediciones del esfuerzo del personal:

- carga laboral real de cada uno de los trabajadores del proyecto, en el caso particular de los procesos que son definidos en trabajos anteriores puede ser controlado muy fácilmente utilizando las Órdenes de Trabajo asignadas a cada trabajador. [2][3]
- horas planificadas y horas laboradas en un gráfico de barras (personal y del equipo completo), también en estos trabajos se definen Hojas de tiempo asociadas a las Ordenes de Trabajo, estas horas pueden ser calculadas con la suma de todas las Hojas de Tiempo asociadas (horas laboradas) contra las planificadas en las Ordenes de Trabajo
- horas mensuales de personal, planificadas y reales, estas pueden ser obtenidas con los mismos instrumentos de control descritos anteriormente.

Con un análisis de estos gráficos, como los que se muestran en la figura 3, se puede lograr una reflexión más profunda en la asignación de tareas a los desarrolladores. Los gráficos de esfuerzo del personal están basados en propuestas de métricas del PSM y son calculados utilizando los datos de las Ordenes y de las Hojas de Tiempo [8][2].

Para el equipo de proyecto



Para un desarrollador: Yanko

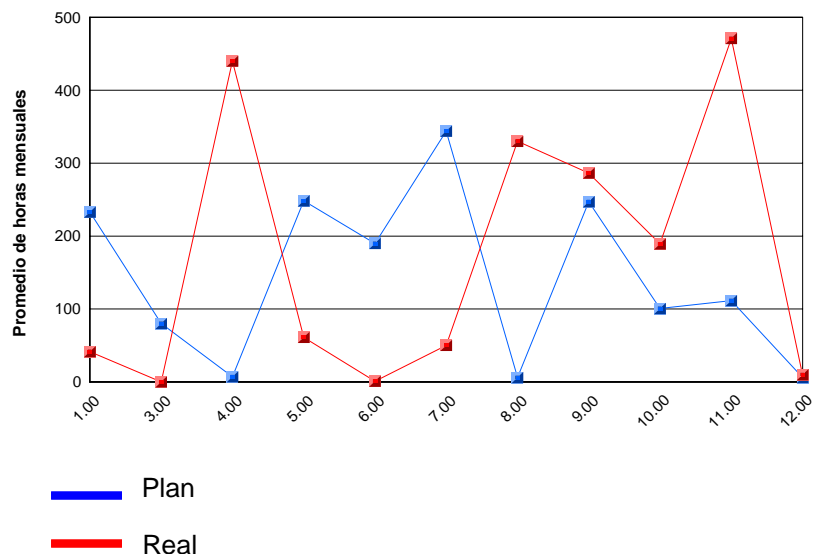


Figura 3: Algunos gráficos de esfuerzo del personal

Como se destaca en el ejemplo que representa la figura, en el quipo de proyecto el desarrollador Yanko está sobrecargado en comparación con el resto de los desarrolladores. En la figura individual se detecta adicionalmente que existe una irregularidad el cumplimiento de las horas planificadas que pudieran ser por dos razones: o el desarrollador no se está esforzando para el cumplimiento de sus tareas o la planificación no fue buena.

6.2 De los defectos

Durante el desarrollo del proceso de control de configuración, muchos de los cambios que son solicitados es producto de defectos en el software o el prototipo entregado al usuario final. Por esta razón es importante controlar los defectos que aparecen de esta manera en el producto.

6.2.1 Resumen de defectos

Una forma de analizar el estado de los defectos es a través del Resumen de defectos, donde se analizan la densidad de defectos por etapas, y los escapes netos. Se considera escapes netos a todos los defectos que se insertaron antes o durante una etapa determinada, pero que no fueron encontrados en esa etapa, sino en otra etapa posterior.

Un ejemplo de un Resumen de defectos, se muestra en la tabla 1.

Etapas	Insertados	Eliminados	Acumulativo Insertados	Acumulativo Eliminados	Escapes Netos	Rendimiento
Planificación	1	0	1	0	1	
Diseño	5	0	6	0	6	
Inspección Diseño	0	3	6	3	3	50 %
Codificación	15	1	21	4	17	
Inspección Código	0	8	21	12	9	47.1%
Prueba	0	6	21	18	3	
Control de cambios	0	3	21	21	0	
Total	21	21				57.1%

Tabla 1 Resumen de defectos por etapas

Nótese que con la tabla de resumen de defectos, no se observa el avance del rendimiento, solamente su valor final. Sin embargo el método de calcularlo es mucho más fácil, el rendimiento por cada etapa se podrá calcular por:

$$\text{Rendimiento} = \frac{\text{Defectos eliminados} * 100}{(\text{Defectos eliminados} + \text{Escapes Netos})}$$

En rendimiento total del proceso se calcula como:

$$\text{Rendimiento Total Proceso} = \frac{\text{Defectos Eliminados antes prueba} * 100}{(\text{Defectos Eliminados antes prueba} + \text{Escapes antes de prueba})}$$

Nótese que estos valores son equivalentes a:

$$\text{Rendimiento Total Proceso} = \frac{\text{Acumul Defectos Eliminados inspec. código} * 100}{(\text{Acumulado Defectos Eliminados inspec. Código} + \text{Escapes inspec. Código})}$$

Debe quedar claro que un valor de rendimiento alto es mucho mejor que un valor bajo, el cual es un resultado pobre. El objetivo o meta es un rendimiento de 100%, o sea que se traten de eliminar los defectos tan pronto como sea posible.

Estos cálculos también pueden ser realizados utilizando las Ordenes de Trabajos asociadas a defectos.

Para analizar el avance del rendimiento de un equipo de proyectos, la métrica más eficiente es graficar el rendimiento total por cada proyecto, un ejemplo se muestra en la figura 4.

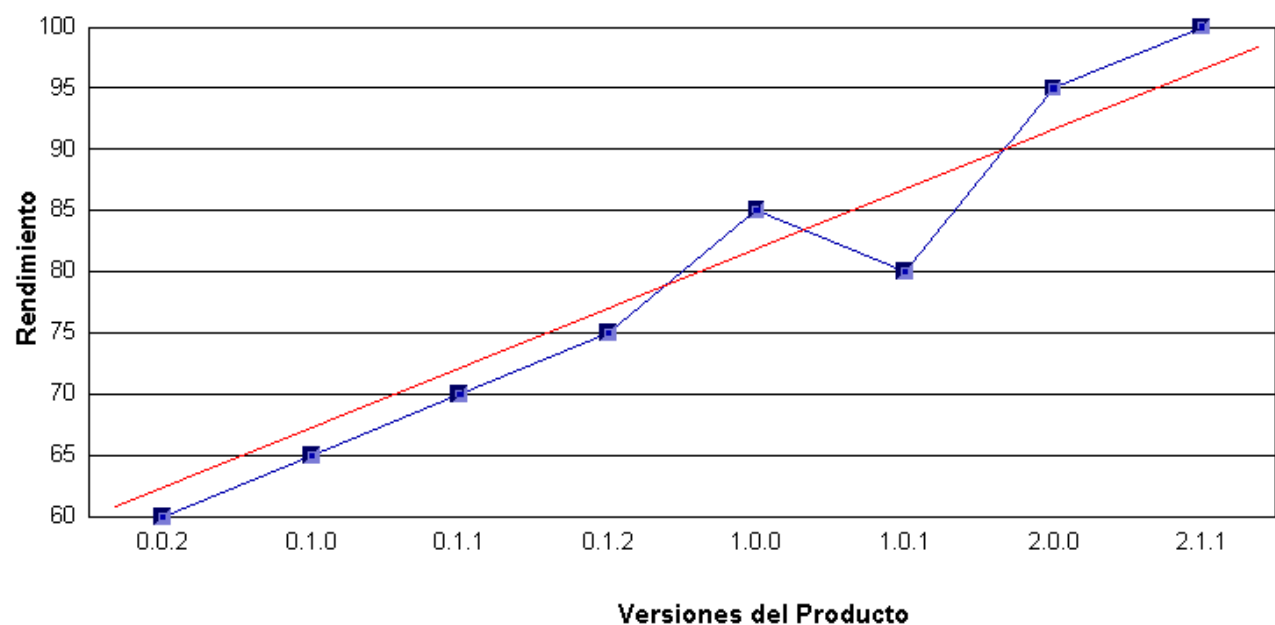


Figura 4: Gráfico de avance del rendimiento.

Una buena tendencia es la señalada por la línea roja continua, o sea en la medida que el equipo de proyecto gana en experiencia, el valor del rendimiento debe ir aumentando. En el caso de la figura la línea con cuadritos representa el avance del proyecto.

6.2.2 Defectos por hora

Los defectos encontrados por hora en una fase específica indican la efectividad del tiempo dedicado a las inspecciones. En la medida que el rendimiento incrementa, es natural una disminución de los defectos por unidad de tiempo.

Los defectos eliminados por unidad de tiempo en una etapa específica se pueden calcular como:

$$\text{Defectos encontrados por hora} = \frac{60 * \text{Defectos eliminados en la etapa}}{\text{Minutos dedicados a esa etapa}} \quad \text{(Cualquier otra conversión de unidad de tiempo es válida)}$$

6.2.3 Eficiencia en la eliminación de defectos

La eficiencia en la eliminación de defectos provee una métrica útil de la efectividad relativa de varios métodos de eliminación de defectos. Da la razón de defectos eliminados por unidad de tiempo de 2 etapas cualesquiera que sean. Típicamente se hace comparando con la etapa de prueba.

$$\text{Eficiencia} = \frac{\text{Defectos eliminados en la etapa}}{\text{Minutos dedicados a esa etapa}} \quad \text{(Cualquier otra unidad de tiempo es válida)}$$

La eficiencia debe ir aumentando progresivamente en la medida que se apliquen correctamente los procesos de desarrollo.

6.2.4 Frecuencia de defectos

Es importante darle un seguimiento a los defectos que se detectan, la forma más cómoda y visual de lograr este seguimiento es a través de los gráficos de frecuencia de defectos. Un ejemplo pudiera ser el que se muestra en la figura 6. En este gráfico se puede llegar a la conclusión clara de que tipo de defectos son los que más se suelen insertar al producto, los que más fácil se eliminan, etc.

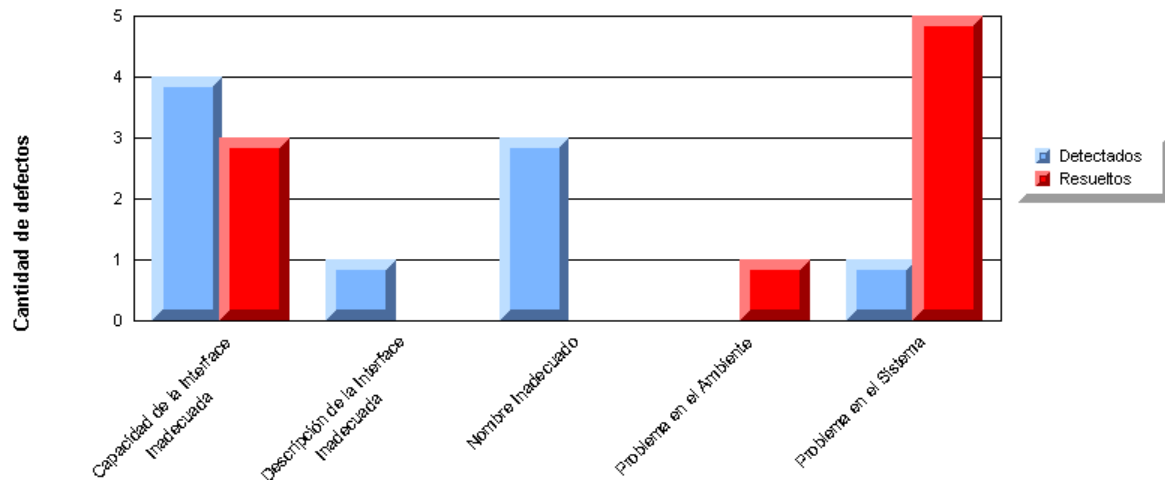


Figura 5: Cantidad de defectos por tipo en una etapa.

Algo similar se puede analizar de un gráfico que en lugar de los tipos de defectos muestre el estado en que están, dígame: detectado, resuelto, en desarrollo, etc.

7. Algunas reflexiones sobre la aplicación de las métricas

Ninguna discusión de la selección y el diseño de métricas de software estaría completa sin abordar el tema de cómo las métricas afectan a las personas y como las personas afectan a las métricas. Aunque sea indiscutible la utilidad de las métricas para la organización, siempre dependerán de las actitudes de las personas involucradas.

Suele haber preocupación de que las mediciones señalarán problemas en un proyecto o en una organización que no eran visibles antes de que el proceso de medición fuera implementado. Estas preocupaciones son reales, y sobreponerse a ellas, requiere un entendimiento de las mediciones, así como saber como usar los resultados de las mediciones apropiadamente en todos los niveles de la organización

La mejor forma de evitar el problema del factor humano en el trabajo con las métricas es seguir algunas reglas básicas tales como las que se enuncian a continuación [18][19][20]:

- **No mida a los individuos:** El ejemplo clásico de este error es medir la productividad de los individuos. Si se mide la productividad, en líneas de código (LOC) por horas, puede ocurrir que las personas se concentren en su propio trabajo en detrimento del equipo y del proyecto. Hay que enfocarse en los procesos y en los productos, no en las personas.

El cumplimiento de esta regla en las experiencias prácticas realizadas ha sido muy importante para el éxito obtenido en la implantación de los procesos.

- **Nunca usar las métricas como un “garrote”:** La primera vez que se usen las métricas en contra de los individuos o los equipos será la última vez que se obtendrán datos válidos.

Esta regla está muy relacionada con la anterior, es importante que los miembros del equipo sientan que es importante dar la información verídica sobre la ejecución del proyecto, que esto no lo perjudica individualmente y ayuda considerablemente al equipo de desarrollo

- **No ignorar los datos:** Una forma segura de matar un programa de métricas es ignorar los datos cuando se toman decisiones.

Muchas veces se abandonan los programas de mejora en las organizaciones por que se engavetan los datos resultantes (generalmente por que son alarmantes). Si esto se hace se podrá saber nunca cual es el resultado final y los miembros del equipo no entregaran los datos al ver que no son utilizados.

- **Nunca usar una sola métrica:** Los software son complejos y multifacéticos. Un programa de métricas debe reflejar esa complejidad. Debe mantenerse un balance entre los atributos del costo, la calidad y los cronogramas de forma que se satisfagan todas las necesidades de los usuarios. Enfocarse en una única métrica puede causar que el atributo que es medido mejore a costa de otros atributos.
- **Proveer retroalimentación:** Proporcionando una retroalimentación regular al equipo sobre los datos que ellos ayudan a coleccionar tiene varios beneficios, por ejemplo; ayuda a mantener el foco en la necesidad de coleccionar los datos. Si los miembros del equipo se mantienen informados sobre los detalles específicos de cómo los datos son usados, ellos tendrán menos posibilidades de empezar a sospechar sobre su uso, y ayuda a educar a los miembros del equipo en la responsabilidad de coleccionar los datos.
- **Lograr “pertenencia”:** Para lograr un sentido de pertenencia tanto en las metas como en las métricas en un programa de medición, se tiene que lograr la participación en la definición de las métricas.
- **Respetar la privacidad de los datos y de las métricas,** clasificando cada elemento de dato que se colecciona en alguno de los tres niveles que propone Wiegers [14]:

1. Individual

2. Equipo de proyecto
3. Organización

Que cada cual conozca lo que le corresponde y que de cada quien se publique lo necesario. Definir estas fronteras es muy saludable para lograr resultados en la implantación de las métricas y confianza en el equipo de desarrollo.

8. Conclusiones

- o En el artículo fueron descritas métricas para el proceso de Control de Configuración de Software adaptados a procedimientos y metas estándares de la Industria Nacional de Software en Cuba que pudieran ser utilizados por pequeñas o medianas empresas productoras de software en el mundo.
- o Fueron definidas un total de 12 métricas que abarcan los aspectos fundamentales del control de la calidad de los procesos de desarrollo de software, el seguimiento y monitoreo de los procesos, y en alguna medida de la planificación de los proyectos, desde el punto de vista del control de cambios.
- o La selección y definición de métricas para la Gestión de Configuración así como las sugerencias para su implantación, presentadas en el trabajo, ha permitido en los proyectos de software donde se han implantado tomar acciones correctivas a tiempo y mejorando sucesivamente los procedimientos definidos.

Referencias bibliográficas

- [1]. Jean L. Dyer. "Team Research and team training: a-state-of-the-art review". Human Factors Reviews, The Human Factors Society, Inc, 1984.
- [2]. Ailyn Febles. "Case Corporativo para el proceso de control de cambios". Tesis presentada en opción al título de Master en Informática Aplicada. Ciudad de la Habana, 2001.
- [3]. Yanko Hernández, Idael Banderas. "Case para la planificación y control de configuración de software". Trabajo de Diploma para optar por el título de Ingeniero Informático. Instituto Superior Politécnico "José Antonio Echeverría". Ciudad de la Habana, 2001.
- [4]. Watts S. Humphrey. "Managing the Software Process". Addison- Wesley, 1989
- [5]. Watts S. Humphrey. "A discipline for software engineering". Addison- Wesley, 1995

- [6]. Watts S. Humphrey. "Introduction to the Team Software Process (sm)". Addison-Wesley, 2000.
- [7]. Stephen H. Kan. "Metrics and Models in Software Quality Engineering": Addison- Wesley. 2000
- [8]. John McGarry, Elizabeth Bailey, David Card, Joseph Dean, Fred Hall, Cheryl Jones, Beth Layman y George Stark; "Practical Software Measurement: A Foundation for Objective Project Management Version 3.1a". 1998
- [9]. Bill Munro. "Measuring IT Project Performance". <http://www.tsepm.com/tsepm/spring01/article3.html> (31/01/2002)
- [10]. Ronald E. Nusenoff, y Dennis C. Bunde." A Guide Book and Spedsheet Tool for a Corporate Metrics Program". Holanda, 1993
- [11]. Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, Carnegie Mellon University and Software Engineering Institute. "Capability Maturity Model, The: Guidelines for Improving the Software Process". Addison-Wesley, 1995.
- [12]. Rational Unified Process. Rational Software Corporation. "Rational Unified Process". Version 2001A.04.00, Copyright 1987-2001.
- [13] SEI Level 2, 3, 4, & 51, Metrics (683), 2001
- [14] Linda L. Westfall. "Software metrics that meet your information needs", <http://www.asq-software.org/metrics/aqc95.html>. 1995. (29/05/2002)
- [15] IEEE, IEEE Standard for Software Configuration Management Plans, American National Standards Institute, 1990, Std. 828-1990
- [16] Álvarez S., CMM en el contexto de ISO-9000:2000, 2002
- [17] Preesman R., Ingeniería de Software un enfoque práctico, 2000, ed. 4ta.
- [18] Pfleeger S., Oman P., Applying Software Metrics IEEE Computer Society Press, 1997
- [19] Pfleeger S., Software Engineering: Theory and Practice, second edition, Prentice Hall, 2001
- [20] Fenton N. , Hall T., Implementing effective software metrics programs. IEEE Software, 14(2):55--64, 1997.
- [21] SEI, Capability Maturity Model® Integration (CMMISM), 2000, disponible en: <http://www.sei.cmu.edu/cmm/cmms/cmms.integration.html>