December 2006

Mejorando la Administración de los Procesos Mediante el Uso de Métricas de Software: Un caso de Estudio

Hilda Pineda
Banco Central de Costa Rica

Marcelo Jenkins
Universidad de Costa Rica

Mejorando la Administración de los Procesos Mediante el Uso de Métricas de Software: Un caso de Estudio

Hilda Pineda
SINPE
Banco Central de Costa Rica
San José, Costa Rica
hildapr@hotmail.com

Marcelo Jenkins
Escuela de Computación e Informática
Universidad de Costa Rica
San Pedro, Costa Rica
mjenkins@ecci.ucr.ac.cr

RESUMEN

Este artículo describe un caso de estudio sobre la experiencia en diseñar e implementar un sistema de métricas de software para medir y mejorar los procesos de una organización de desarrollo de sistemas. Explicamos cómo diseñamos e implementamos nuestro sistema de métricas incluyendo el uso de una herramienta para la recolección y análisis de las métricas en todos los niveles de la organización, y resumimos los beneficios que hemos obtenido de su uso desde el punto de vista del mejoramiento continuo del proceso. Los temas tratados en este artículo pueden interesar a las organizaciones que desean mejorar sus procesos de software mediante el uso sistemático de métricas de calidad y productividad para desarrollo y mantenimiento de software.

Palabras clave

Administración de Sistemas de Información, métricas de software, CMM.

Enhancing Process Management Using Software Metrics: A Case Study

ABSTRACT

This paper describes a case study about the experience in designing and implementing a software metrics program to measure and improve the process in a systems development organization. We explain how the metrics system was designed and implemented, including the use of a software tool we developed to collect and analyze the metrics results at all levels of the organization, and summarize the benefits obtained from their use from the process improvement standpoint. The issues discussed in this paper may interest organizations that want to improve their processes through the systematic use of quality and productivity metrics for software development and maintenance.

Keywords

Management Information Systems, software metrics, CMM.

INTRODUCCIÓN

Las organizaciones de desarrollo de software se ven actualmente en la necesidad de mejorar sus procesos de desarrollo y mantenimiento de los sistemas de información. Las entidades bancarias no están fuera de este ámbito; por el contrario, debido a lo delicado y crítico de sus operaciones, se han preocupado por mejorar el software y optimizar los servicios de TI que prestan reduciendo costos de producción y mejorando la productividad y la calidad.

A raíz de esto, han surgido numerosas normas, modelos y estándares para mejorar la calidad de las organizaciones de software. Uno de ellos es el Modelo de Capacidad/Madurez (CMM) Paulk et al. (1993) por sus siglas en inglés creado por el SEI (Software Engineering Institute). En el año 2002, el CMM evolucionó a otra versión, denominada CMMI. Este modelo

de calidad promueve el uso sistemático de métricas de software como requisito indispensable para administrar eficazmente y mejorar los procesos de software. La medición es un aspecto crítico en la efectividad de todo proceso de desarrollo de sistemas.

En este artículo describimos un caso de estudio de una institución financiera que ha implementado un programa de métricas de software como parte de su proceso de mejoramiento de la calidad de su proceso de software. Los temas tratados en este artículo pueden interesar a las organizaciones que desean mejorar sus procesos de software mediante el uso sistemático de métricas de calidad y productividad para desarrollo y mantenimiento de software.

MARCO TEÓRICO

Métricas de software

Existen múltiples metodologías y procesos que se han definido para determinar el conjunto apropiados de métricas para una organización de software. Una de ellas es Meta/Interrogante/Métrica (GQM en inglés), que es un paradigma desarrollado por Basili et al. (1984) y consiste en identificar las metas que quieren ser logradas y asociarlas con un conjunto de preguntas relacionadas con cada meta. Las respuestas a estas preguntas deben hacer posible la identificación las medidas cuantitativas que son necesarias para proveer las respuestas, y así lograr las metas. Tal y como se muestra en la Figura 1, el resultado de la aplicación del modelo GQM es la especificación de un sistema de métricas orientado a un conjunto particular de aspectos y un conjunto de reglas de interpretación de los datos obtenidos en las mediciones en tres niveles diferentes: conceptual(Meta), operacional(Interrogante) y cuantitativo(Métrica). Esta metodología se ha usado con mucho éxito en múltiples organizaciones Basili et al. (1988), Daskalantonakis (1992), Fenton et al. (1997), Kan (2003), Jenkins et al. (2002), Jenkins et al. (2004), Jenkins et al. (2005).

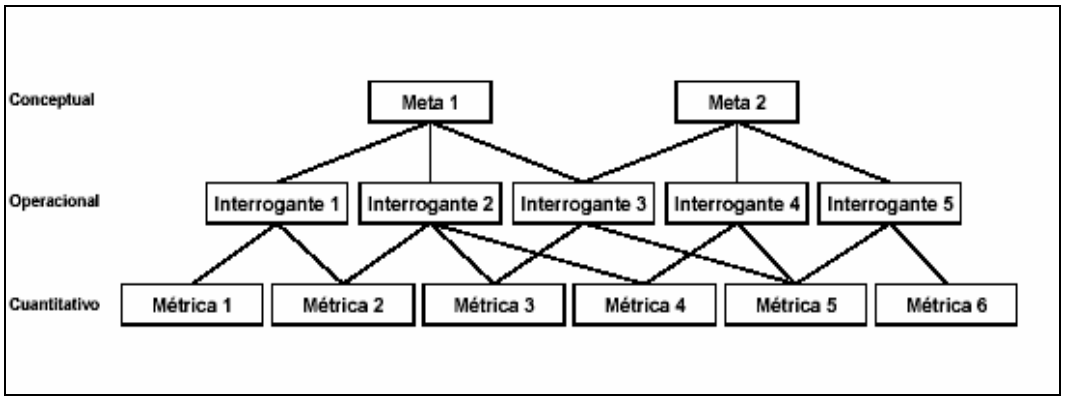

Figura 1. Niveles de definición del Modelo GQM.

La IEEE ha definido dos estándares relacionados con métricas de software:

1. El *IEEE Standard 1061-1998 for a Software Quality Metrics Methodology* IEEE (1999a) provee una metodología para establecer requerimientos de calidad de software, además de identificar, implementar, analizar y validar métricas de calidad de procesos y productos de software. Para cada atributo de calidad identificado se determinan los factores y sub-factores que se desean medir, y seguidamente se definen las métricas específicas que miden esos factores.
2. El *IEEE Standard 1045-1992 for Software Productivity Metrics* IEEE (1999b) propone un proceso para la recolección y cálculo para la medición de la productividad en proyectos de software. Estas métricas de productividad se definen usando la razón de la cantidad de salida producida entre el esfuerzo invertido para producirla.

El *Personal Software Process* (PSP) Humphrey (1995) y su complemento el *Team Software Process* (TSP) Humphrey (2000) definen conjuntamente un proceso de software que incorpora un conjunto básico de mediciones de calidad y desempeño para evaluar la calidad del trabajo de desarrollo de software.

La metodología *Practical Software Measurement* (PSM) McGarry et al. (2002) y el marco de trabajo propuesto en Florac et al. (1999) proponen modelos de medición del proceso de software como un proceso sistemático. Estas metodologías incluyen una fase de planificación de la medición donde se determinan las métricas apropiadas a cada proyecto con base en las necesidades de información de la organización.

Finalmente, el Estándar internacional *ISO/IEC 15939 Software Engineering—Software Measurement Process* ISO/IEC (2002) propone un proceso de medición que es aplicable a todas las actividades de ingeniería de software. El proceso define las actividades para la definición de las necesidades de información, la aplicación de las mediciones y su análisis, y cómo determinar si los resultados del análisis de las métricas es válido.

El modelo CMM/CMMI

El modelo de capacidad/madurez (CMM) Paulk et al. (1993) es un modelo que permite evaluar la madurez del proceso de desarrollo de software de una organización. Fue desarrollado por el Instituto de Ingeniería de Software (SEI) y describe un conjunto de mejores prácticas para el proceso de software en términos evolutivos de acuerdo a cinco niveles, desde un proceso caótico (nivel 1) hacia un proceso disciplinado (nivel 5). Como parte integral del modelo existe un tema que se repite en cada una de las características del modelo: la medición.

En el año 2002, el CMM evolucionó al Modelo de Capacidad Madurez Integrado (CMMI) Chrissis et al. (2004). El CMMI-SE/SW/IPPD/SS versión 1.1 incorpora 4 disciplinas: la ingeniería de sistemas (SE), la ingeniería de software (SW), del desarrollo integrado de producto y proceso (IPPD) y la contratación externa (SS).

Las áreas de proceso (PA´s) conforman los bloques de prácticas que conforman el modelo y que sirven para determinar el grado de madurez de una organización de sistemas. Cada PA define un conjunto de de metas específicas que pueden ser alcanzadas implementando una serie de prácticas relacionadas. El CMMI-SE/SW/IPPD/SS contiene un total de 25 PA´s, las que conjuntamente agrupan a más de 500 prácticas específicas, organizadas en dos representaciones alternas: la representación escalonada que agrupa las PA´s en 5 niveles de madurez, y la representación continua que agrupa los PA´s en 4 categorías: administración de proyectos, administración de procesos, ingeniería, y soporte.

METODOLOGÍA

Para realizar este trabajo, utilizamos un análisis de caso como estrategia de investigación. Queríamos probar cómo podríamos utilizar técnicas de análisis estadístico para mejorar los procesos de desarrollo y mantenimiento de sistemas de información. Existen múltiples experiencias de este tipo de proyecto de mejoramiento en la literatura (ver Florac et al. (1999), Basili et al. (1988), Daskalantonakis (1992), Fenton et al. (1997), Kan (2003), Jenkins et al. (2002), Jenkins et al. (2004), Jenkins et al. (2005), y Pressman (2001)). Nuestro proyecto se nutrió de estas experiencias y utiliza algunas de las mismas técnicas para diseñar la solución.

La necesidad del BCCR de utilizar métricas de software surge de la necesidad de mejorar el proceso de desarrollo de sus servicios (i.e., sistemas de información) brindando mayor calidad, agilidad y confianza a sus clientes con miras a obtener una certificación CMM nivel 3. Actualmente, el SINPE utiliza varias herramientas de software dentro de su proceso de desarrollo y mantenimiento. De todas estas herramientas se pueden obtener mediciones, pero éstas no se encuentran centralizadas en un único repositorio que permita consultarlas y brindar la información a las personas interesadas de manera oportuna.

Por otra parte, según un estudio efectuado en abril del 2003, con respecto al modelo CMM, algunos KPA's (Key Process Area o Áreas Clave de Proceso) de niveles 2 y 3 que se encontraban implementados en cierto porcentaje mientras que existían otros cuyas prácticas no están del todo implementadas. Mucho de este problema radicaba en la ausencia de métricas de software, de ahí la necesidad de definir e implementar las métricas de calidad y la importancia de este proyecto para el mejoramiento del proceso del SINPE.

LA ORGANIZACIÓN

El Sistema Interbancario de Negociación y Pagos Electrónicos (SINPE) del Banco Central de Costa Rica (BCCR) ha iniciado la definición de su proceso de calidad utilizando el CMM. El BCCR es el órgano central de la economía costarricense desde 1950, y por sus funciones se ha caracterizado por la utilización de tecnología de punta en el desarrollo de sus sistemas. Uno de ellos es el SINPE. Este sistema de información es un software mediante el cual se conecta el sistema financiero nacional de todo el país. Consta de alrededor de 1,3 millones de líneas de código en Visual Basic .NET y conecta más de 65 instituciones financieras permitiendo realizar más de 100.000 transacciones diarias por un monto de $500 millones Alvarado (2002), Microsoft Corp. (2002). Debido a todas las características y funciones citadas anteriormente, se considera al SINPE como un sistema de misión crítica. Por esto, es deseable que su desarrollo y mantenimiento se encuentre bajo estrictas normas de aseguramiento de la calidad.

El Departamento del BCCR encargado de construir y dar mantenimiento a este importante sistema de información se denomina Tecnologías del Sistema de Pagos. Éste se encuentra formado por ocho secciones o áreas, cada una de ellas

dirigida por un coordinador y con funciones propias y específicas, tal y como muestra la figura 2. Las ocho áreas se describen a continuación:

1. Área de Análisis y Diseño: Realiza la definición de los requerimientos, el análisis y la generación de diagramas de casos de uso.
2. Área de Soporte: Encargada de darle solución a los reportes de problemas de usuarios o de remitirla a alguna de las otras áreas. También realiza las liberaciones de nuevas versiones, instaladores y de las pruebas de integración.
3. Área de Base de Datos: Es la encargada de crear y dar mantenimiento a las estructuras de la base de datos del SINPE.
4. Área de Infraestructura: Realiza la programación de los componentes que forman parte de la lógica de negocio del sistema.
5. Área de Interfaz Cliente: Desarrolla las pantallas y procesos a nivel de interfaz con los usuarios. Está muy asociada con el área de Infraestructura en el desarrollo del SINPE.
6. Área de Telecomunicaciones: Es la encargada del mantenimiento de la red de los diferentes ambientes que existen en el departamento de Tecnologías del Sistema de Pagos.
7. Área de Hardware: Su función es el mantenimiento y nomenclatura de los diferentes equipos que se utilizan en el SINPE, tanto de servidores como de estaciones de trabajo.
8. Área de Seguridad: Es la encargada de la implementación y mantenimiento de los diferentes esquemas de seguridad física y lógica que tiene el SINPE.

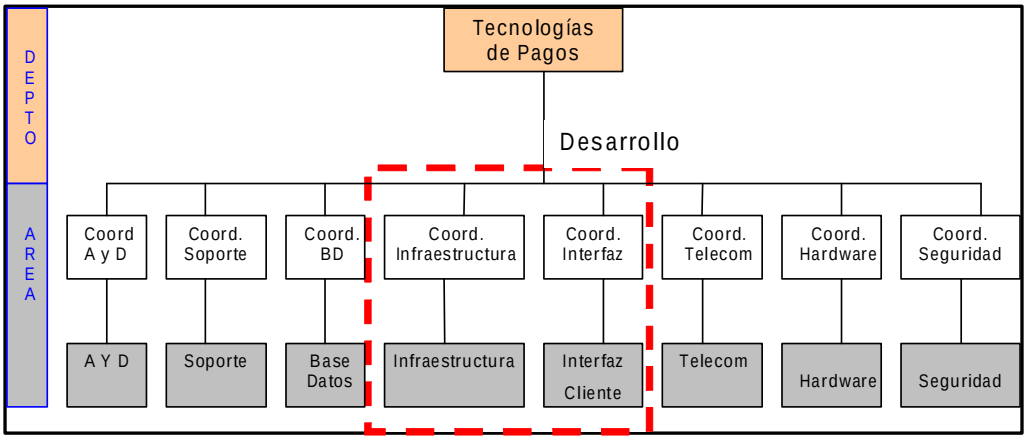

Figura 2. Estructura del departamento de Tecnologías del Sistema de Pagos

ANÁLISIS DEL CASO

Como una primera fase de implementación de las métricas, se seleccionaron tres de las ocho áreas del SINPE para llevar a cabo el diseño y la implementación del sistema de métricas. Las tres áreas del SINPE elegidas fueron: Análisis y Diseño, Desarrollo (que incluye Interfaz e Infraestructura), y Soporte (ver Figura 3). Se seleccionaron estas tres áreas debido a que se consideran las más importantes con relación al ciclo de vida del software.

La Solución Propuesta

Para definición de las métricas del SINPE decidimos utilizar la metodología GQM básicamente porque es simple, ha sido ampliamente probada, y porque teníamos experiencia previa utilizándola en otras organizaciones de sistemas.

En la figura 3 se presenta el diagrama GQM para una de las áreas: Análisis y Diseño. Primero, se identificaron cuatro metas (descritas en el segundo nivel del diagrama) relacionados con 3 KPA´s del CMM (mostrados en el primer nivel del diagrama), a saber: Requirements Management (RM), Project Tracking and Oversight (PTO), y Software Product Engineering (SPE).

A partir de las tres metas definidas por la Gerencia del SINPE se derivaron ocho interrogantes (mostrados en el tercer nivel). Finalmente, se derivaron ocho métricas que se muestran en el cuarto nivel del diagrama, y en el quinto nivel se especifican

los valores meta propuestos para cada una de estas métricas. Para algunas métricas, los valores meta no se han definido todavía.

De la misma forma se definió un conjunto de 9 métricas a ser implementadas para el área de Desarrollo y otro conjunto de 6 métricas para el área de Soporte. De esta forma, se obtuvieron un total de 23 métricas de software para estas tres áreas del SINPE, las que se especifican en detalle en la tabla 1.

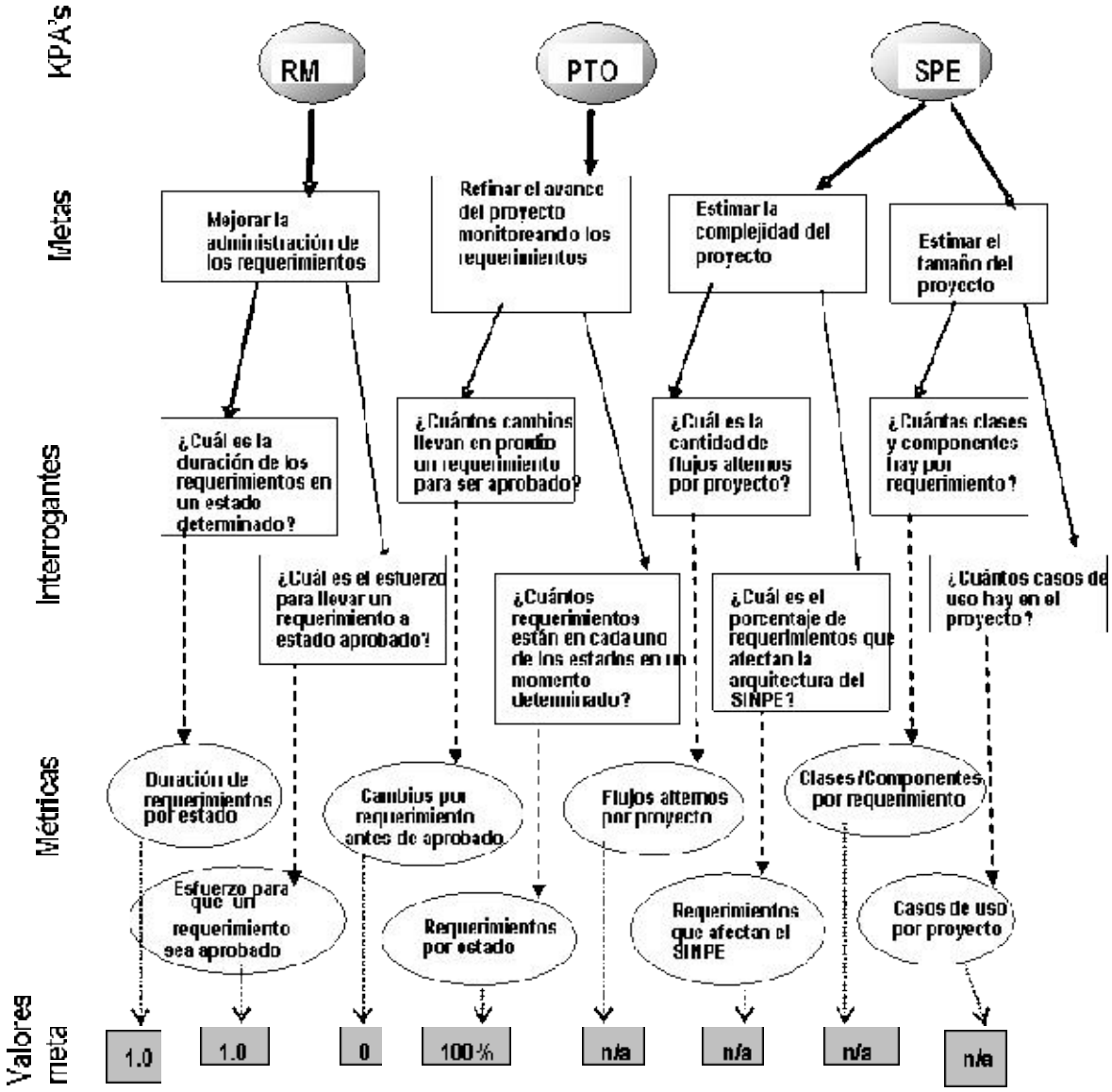

Figura 3. Derivación de las métricas para el área de Análisis y Diseño.

Implementación de las métricas

La Gerencia del SINPE seleccionó algunas métricas para su implementación. La decisión para escoger estas métricas se basó en dos parámetros: las entrevistas con los coordinadores de las áreas, y las métricas que permitían automatización para realizar el cálculo. Las 10 métricas elegidas para su implementación fueron:

- Área de Análisis y Diseño: Casos de Uso por Proyecto, Flujos Alternos por Proyecto, Cantidad de Requerimientos por Estado por Proyecto.
- Área de Desarrollo: CPI, SPI, SV y CV para las tareas de Desarrollo.
- Área de Soporte: Duración Real en resolución de casos, Tiempo Muerto, Relación de Tiempo Desarrollo-Soporte.

Área	Descripción de la Métrica	Fórmula de cálculo
Análisis y Diseño	1. CPI para requerimientos aprobados: Se utilizará para medir la eficacia de la estimación de los requerimientos aprobados. El cálculo se realiza dividiendo el costo estimado (BCWP) entre el costo real (ACWP).	$CPI = \frac{BCWP}{ACWP}$
	2. SPI para requerimientos por estado: Se utilizará para medir la eficacia en la estimación de la duración en las tareas de mantenimiento de requerimientos. Se calcula dividiendo la duración estimada para realizar las tareas de mantenimiento (BCWP) entre la duración real en las mismas actividades (BCWS).	$SPI = \frac{BCWP}{BCWS}$
	3. Cambios por requerimiento antes de la aprobación: Se utilizará para medir el número de cambios por los que pasa un requerimiento antes de ser aprobado. Se calcula contando la cantidad de cambios realizados desde que el requerimiento es recolectado hasta que es aprobado.	CRAA=#cambios realizados desde que el req. es recolectado hasta aprobado
	4. Flujos alternos por proyecto: Se utilizará para medir el número total de flujos alternos que tienen los casos de uso de un proyecto. Se calcula contando la cantidad de flujos alternos que se identifiquen por proyecto.	FAP = # flujos alternos por proyecto
	5. Cantidad de requerimientos en un estado: Se utilizará para medir el número de requerimientos en cada estado en un momento determinado. Se calcula identificando cada uno de los posibles estados de un requerimiento, y contando cuántos requerimientos están en ese estado en un momento determinado.	RPE = # requerimientos en cada estado
	6. Requerimientos que afectan la Arquitectura del SINPE: Mide la complejidad del proyecto en términos de los requerimientos que afectan la arquitectura ya existente del SINPE. Se divide la cantidad de requerimientos que afectan la arquitectura del SINPE, entre el total de requerimientos del proyecto.	RAAS=% requerimientos que afectan la arquitectura del SINPE, por proyecto
	7. Clases/Componentes por requerimiento: Se utilizará para medir el número de clases y/o componentes por cada requerimiento en cada proyecto. Se obtiene contando la cantidad de clases y/o componentes que se derivan de cada requerimiento.	CLAR=#clases o componentes por requerimiento
	8. Casos de uso por proyecto: Se obtiene contando la cantidad total de casos de uso que hay por proyecto.	CUS=# CU para un proyecto
Desarrollo	9. Razón de errores encontrados en revisión entre colegas. Se utilizará para medir la eficiencia de los peer revisiones entre colegas realizadas. Se calcula dividiendo el número de errores encontrados en la revisión entre el número de horas invertidas en la revisión.	$RDPR = \frac{\#\ errores\ en\ PR}{\#\ horas\ invertidas}$
	10. Retrabajo: Se utilizará para medir el esfuerzo involucrado en actividades de retrabajo. Su cálculo se realiza dividiendo el número de horas persona, dedicadas al retrabajo, multiplicada por 100 y dividiendo entre las horas persona totales invertidas en el proyecto.	$Retr = \frac{\#horas\ retrabado\ X\ 100\%}{\#\ horas\ invertidas}$
	11. CPI para tareas de desarrollo: Se utilizará para medir la eficacia de la estimación de esfuerzo de desarrollo de cada proyecto. Se obtiene dividiendo el esfuerzo estimado en realizar la tarea entre el esfuerzo real.	$CPI = \frac{BCWP}{ACWP}$
	12. Duración real de atención de solicitudes de cambio: Su cálculo se hace restando la fecha en que se resuelve la solicitud de cambio (FR), menos la fecha en que fue ingresada (FI).	DSC = FR-FI
	13. SV en tareas de desarrollo: Se utilizará para medir la diferencia de la estimación de duración de cada proyecto. Se calcula restando el costo estimado menos el costo presupuestado en las actividades del proyecto.	SV = BCWP – BCWS
	14. Cobertura del código en pruebas unitarias: Se utilizará para medir la cobertura del código durante las pruebas unitarias. Este porcentaje es calculado por la herramienta Rational Pure Coverage.	CCPU= % cobertura del código generado por Pure Coverage
	15. Defectos por liberación: Se utilizará para medir el número de reportes de defectos de software enviados por las entidades para una liberación. Se calcula contando la cantidad de defectos reportados por liberación.	DL = # defectos reportados por liberación
	16. SPI para tareas de desarrollo: Se utilizará para medir la diferencia de la estimación de duración de cada proyecto. Se obtiene dividiendo el costo estimado entre el costo real a la fecha de cálculo.	$SPI = \frac{BCWP}{BCWS}$
	17. CV para las tareas de desarrollo: Se utilizará para medir la diferencia de la estimación de esfuerzo de cada proyecto. Se calcula restando el esfuerzo estimado menos el esfuerzo real invertido en esas tareas.	CV = BCWP – ACWP
Soporte	18. Duración real del área de Soporte en resolución de casos: Se utilizará para medir la eficacia de la resolución de los casos. Se obtiene restando la fecha en que el caso fue resuelto, menos la fecha en que fue inicio las pruebas.	ESRC = FPAS-FRAD
	19. SPI para tareas de puesta en marcha: Controlar la duración de tareas de puesta en marcha, a partir de que el proyecto es liberado por el área de Desarrollo, para un proyecto del SINPE en días hábiles. Se calcula dividiendo el costo estimado entre el presupuestado para realizar las tareas de puesta en marcha.	$SPI = \frac{BCWP}{BCWS}$
	20. Tiempo Muerto: Se utilizará para medir el tiempo muerto de los casos, en días calendario. Se obtiene restando la fecha de inicio de las pruebas, menos la fecha de resolución de Desarrollo por cada caso.	TM = FIP-FSD
	21. Fallas o errores en mantenimiento: Se utilizará para medir el número de reportes de fallas o errores encontrados en mantenimiento. Se calcula contando la cantidad de fallas por liberación detectadas en mantenimiento.	FM = # fallas reportadas por liberación
	22. Relación Desarrollo-Soporte: Se utilizará para medir la relación de tiempo en días calendario, que invierten ambas áreas en la resolución de casos. Se calcula comparando el tiempo invertido por Desarrollo en resolver un caso, con el tiempo invertido por el área de Soporte.	$RDS = \frac{TD}{TS}$
	23. Cantidad de defectos en producción: Se utilizará para medir el número de reportes de defectos enviados por las entidades para una liberación. Se calcula contando la cantidad de defectos por liberación, detectados cuando el software ya está en producción.	DP=# defectos reportados por liberación

Tabla 1. Descripción de las métricas definidas para el SINPE.

Para obtener los datos necesarios para hacer el cálculo de las métricas, se utilizó la herramienta de extracción Data Transformation Server (DTS) que ofrece SQL Server utilizando las siguientes tres fuentes de datos existentes: Rational Requisite Pro, Rational Clear Quest, y Microsoft Project. La figura 4 muestra el diagrama del proceso de extracción de datos. El paquete de consultas SQL generado por el DTS permite esta consolidación de fuentes diferentes en una única base de datos que es finalmente consultada desde Excel para elaborar los gráficos de las métricas.

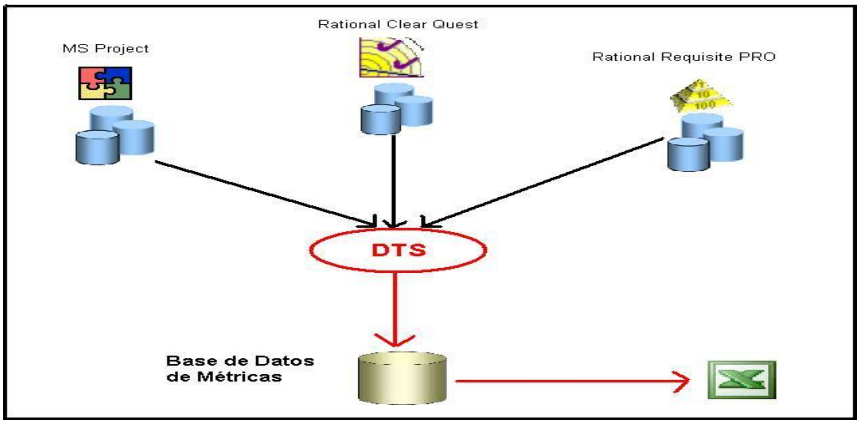

Figura 4. Diseño de un DTS para consultas de métricas.

RESULTADOS OBTENIDOS

A continuación se presentan algunos ejemplos de los gráficos obtenidos y su interpretación para cada una de las 3 áreas del SINPE.

Área de Análisis y Diseño

Una de las métricas de Análisis y Diseño, cantidad de requerimientos por proyecto en un estado determinado, se presenta en la figura 5 mediante un diagrama de barras en 3 dimensiones, esto debido a la complejidad del mismo, ya que tiene cuatro series de datos: estado, tipo de requerimiento, proyecto y cantidad. Con esta métrica el administrador del proyecto puede mejorar la administración de los requerimientos dado que conoce en cualquier momento el estado de cada uno de los requerimientos de su proyecto, determinando así el progreso del proyecto.

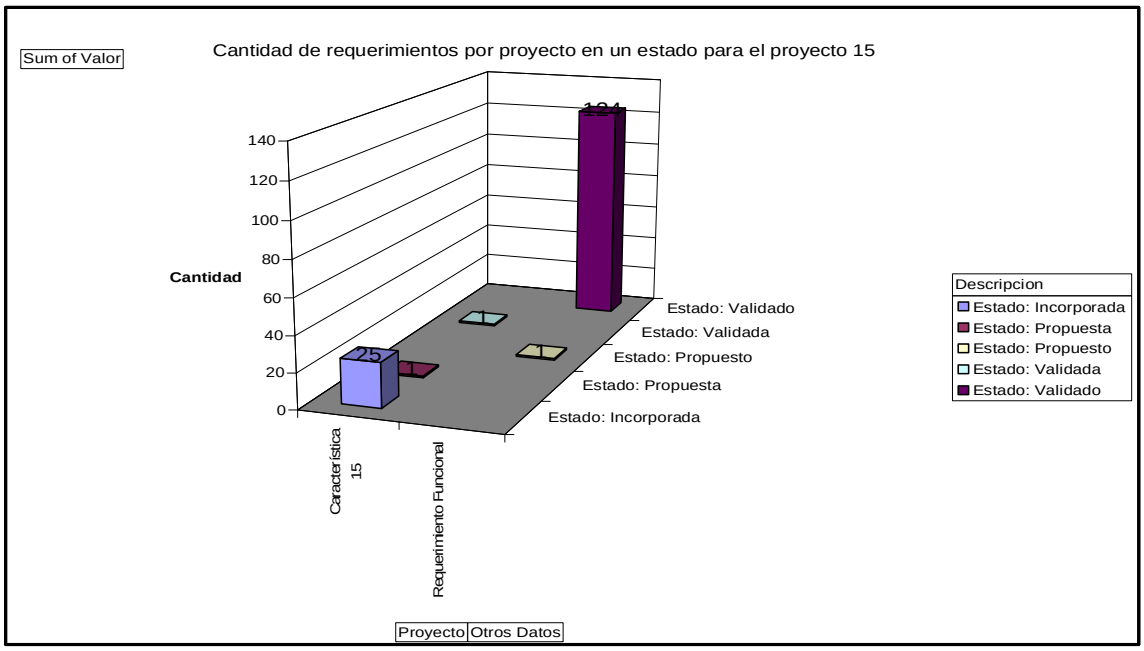

Figura 5. Requerimientos por estado para el proyecto 15

Otra métrica importante para el área de Análisis y Diseño es la cantidad de casos de uso por proyecto, lo que permite controlar el tamaño de los mismos. Un ejemplo de esta métrica se muestra en la figura 6, donde se muestra el tamaño de 8 proyectos diferentes (medido en número de casos de uso) que se encuentran en etapa de desarrollo.

Otra métrica para el área de Análisis y Diseño es el de la complejidad de los proyectos, tomando en cuenta los flujos alternos. La figura 7 muestra el número total de flujos alternos para los casos de uso que definen los requerimientos de 3 proyectos diferentes. Los flujos alternos en los casos de uso se consideran un factor de complejidad que permiten calcular de una manera más acertada los tiempos de implementación y pruebas del proyecto, así como el porcentaje de cobertura que se debe lograr durante las pruebas.

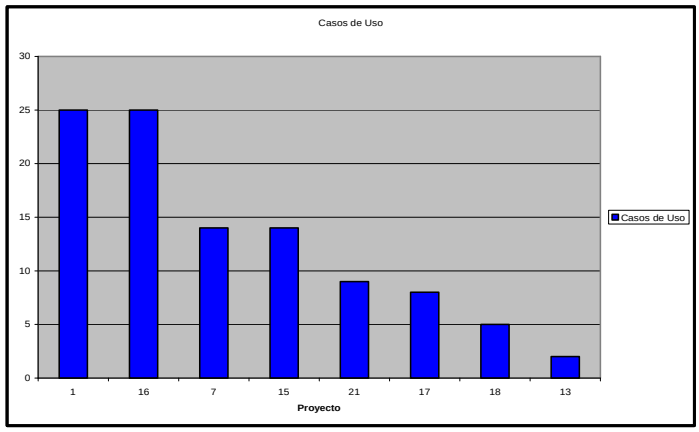

Figura 6- Número de casos de uso por proyecto.

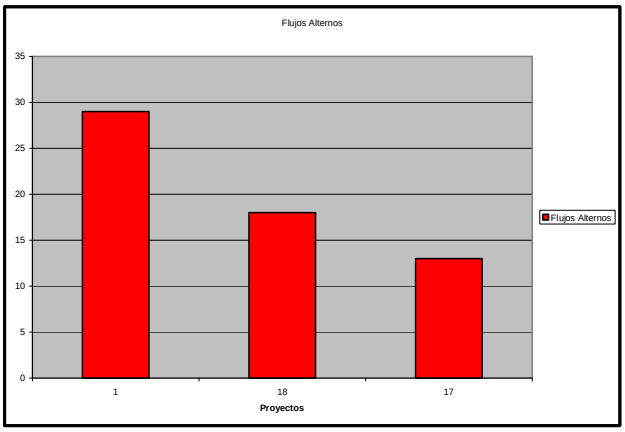

Figura 7- Número de flujos alternos por proyecto.

Área de Desarrollo

El Cost Performance Index (CPI) mide la relación del costo estimado de un proyecto en horas persona de esfuerzo para la realización al costo real en horas persona del proyecto a lo largo del tiempo. La figura 8 muestra el CPI para un proyecto particular. El valor meta del la métrica CPI es de 1, por lo que durante los primeros 8 meses del proyecto se cumplió con el valor meta propuesto. Por otro lado, el Schedule Performance Index (SPI) de este mismo proyecto se muestra en la figura 9. El SPI lo que mide es la eficacia de la estimación del tiempo del proyecto. En el último tercio el SPI estuvo por debajo de uno, lo que denota que el proyecto se atrasó.

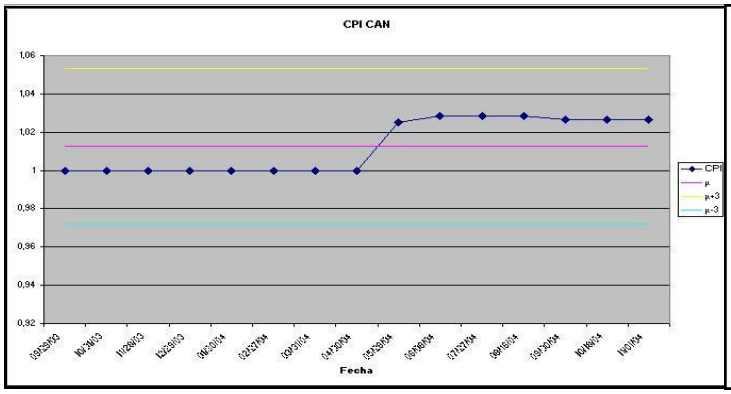

Figura 8- Ejemplo del CPI de un proyecto.

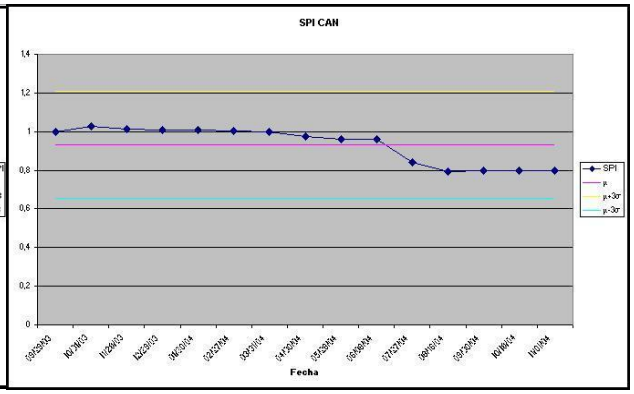

Figura 9- Ejemplo del SPI de un proyecto.

Área de Soporte

La primera métrica a graficar se relaciona con el tiempo muerto, es decir, el período que pasa un caso "ocioso" luego de qu es resuelto por el área de Desarrollo y hasta que es atendido por el área de Soporte (pasa al estado En Pruebas), medida en días calendario (incluyendo feriados y fines de semana). Esta métrica permite identificar retrasos en el proceso de resolución de casos, con el fin de mejorar la atención al cliente. Cada punto en los gráficos indica un caso. La muestra de casos se obtuvo de la base de datos de Rational ClearQuest. Se consideraron todos los casos que estaban registrados al momento de tomar la muestra para un total de 538, clasificados en las diferentes categorías.

La figura 10 muestra el comportamiento del tiempo muerto para los casos de tipo crítico. Para los casos críticos se calculó una media de 1.9 días y un rango de variación de 15.7 días. Se identifica claramente un número de caso que se sale del rango establecido, con un valor de 36 días. Con esta información, se generó un diagrama de causa-efecto que ilustra el motivo de los casos sobresalientes y que se muestra en la figura 11. El Coordinador del área de Soporte puede ahora utilizar esta información para mejorar el proceso de atención de casos tomando medidas que resuelvan estas causas.

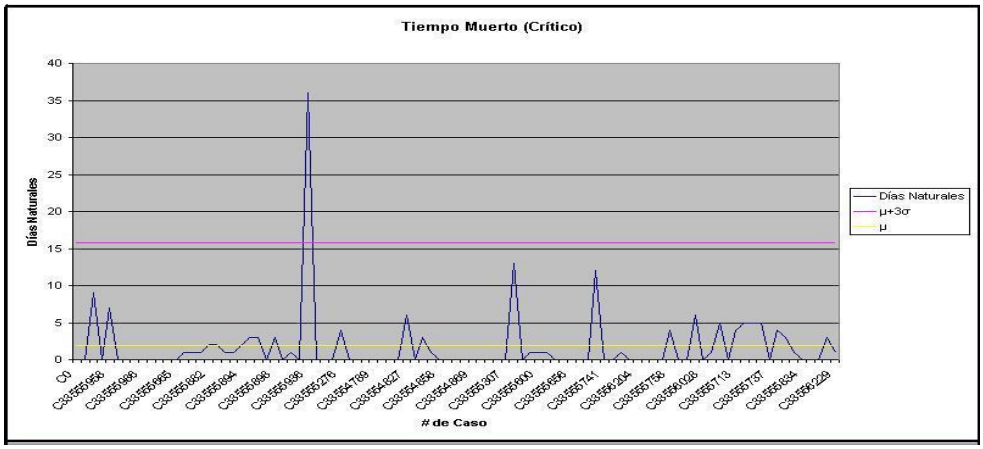

Figura 10- Tiempo muerto para los casos de mantenimiento de prioridad crítica.

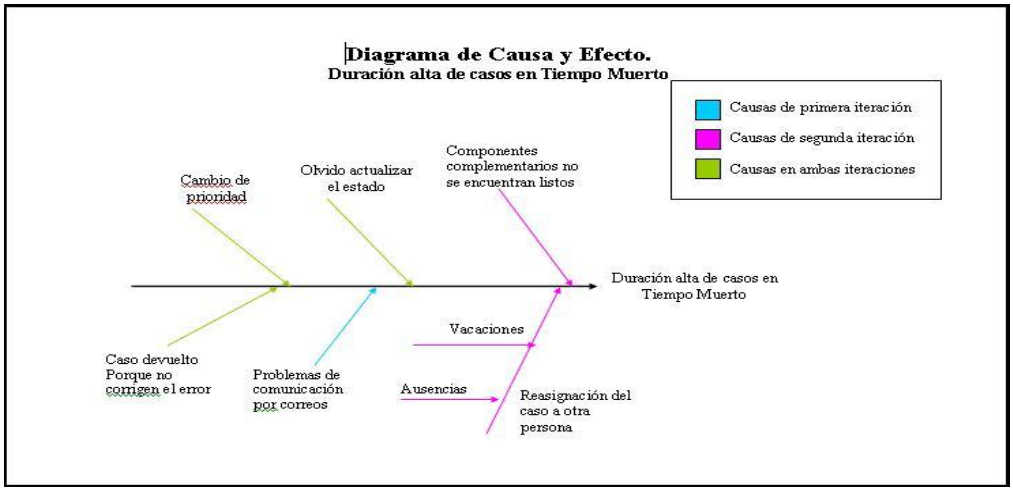

Figura 11 Diagrama de causa y efecto para las causas asignables del tiempo muerto.

Resumen de Resultados

En la tabla 2 se muestra un resumen con las métricas implementadas en este proyecto, el valor meta que debían cumplir, y un indicador que señala si la meta se cumplió o no. En este caso, únicamente se cumplieron a cabalidad las metas de indicador de variación de costo del proyecto para el área de Desarrollo y la métrica de estimación de esfuerzo.

Métrica	Valor Meta	Cumplimiento
Flujos alternos por proyecto	No definido	N/A
Requerimientos por estado por proyecto	100% estado incorporado	No
Casos de uso por proyecto	No definido	N/A
Diferencia entre el tiempo estimado y real del proyecto.	0	No
Diferencia entre el costo estimado y real del proyecto	0	Si
Esfuerzo en las tareas de desarrollo por proyecto	1	Si
Duración en las tareas de desarrollo por proyecto	1	No
Tiempo muerto entre fin del desarrollo hasta el inicio de las pruebas en días calendario	0	No
Relación esfuerzo de Desarrollo-pruebas	1	No
Tiempo (duración en días calendario) promedio en resolución de casos por complejidad	0	No

Tabla 2. Cumplimiento actual de los valores meta para las métricas implementadas.

CONCLUSIONES

La definición e implementación del sistema de métricas descrito aquí ha sido clave para el mejoramiento de los procesos de desarrollo y mantenimiento de sistemas de información que componen el SINPE. Esta era una de las brechas más importantes que debían cerrarse para lograr un nivel de madurez más alto según el CMM.

Nuestra experiencia muestra que la medición es clave para mejorar la administración de los sistemas de información. No es posible mejorar algo que no se puede medir. Para esto, se definieron e implementaron métricas de software que no existían en el SINPE antes de este proyecto. La parte más complicada de este proyecto fue la implementación del conjunto de 10 métricas que se seleccionó pues esto requirió implementar las herramientas de recolección y análisis de datos, así como instaurar la disciplina de medición en una organización que no la tenía antes de este proyecto.

Los beneficios obtenidos hasta el momento son significativos. Ahora los coordinadores de 3 de las áreas del SINPE cuentan con información importante sobre los proyectos que supervisan, lo que les permite hacer una administración mucho más eficaz y oportuna del desarrollo y mantenimiento del sistema de información del SINPE. Esta cultura de medición que está ahora perneando al resto de la organización. Además, con la ejecución de este proyecto hemos podido solucionar algunos de las brechas más importantes que tenía el proceso del SINPE con miras a una evaluación CMM nivel 3. El siguiente paso en el proyecto de mejoramiento del SINPE es implementar el segundo conjunto de 13 métricas que está definido pero aún no implementado, medir los resultados y revisar el conjunto de métricas total para determinar su utilidad.

RECONOCIMIENTO

Se agradece a la Gerencia del SINPE por permitirnos realizar este trabajo en su organización.

REFERENCIAS

1. Alvarado E. (2002). Avance en Sistema Financiero, Periódico *La Nación*, Costa Rica. 17/09/2002, www.nacion.co.cr.
2. Basili, V.R., Weiss D.M. (1984) A Methodology for Collecting Valid Software Engineering Data, *IEEE Transaccitons on Software Engineering, Vol 10, 1984, pp. 728-738.*
3. Basili, V.R., Rombach, H.D. (1988) The Tame Project: Towards improvement-oriented software environments, *IEEE Transaccitons on Software Engineering, Vol 14, 1988, pp. 758-773.*
4. Chrissis M.B. et al (2004) *CMMI guidelines for process integration and product improvement*. Addison-Wesley.
5. Daskalantonakis, M.K. (1992) A Practical View pf Software Measurement and Implementation Experiences within Motorola, *IEEE Transaccitons on Software Engineering, Vol 18, 1992, pp. 998-1010.*
6. Fenton N.E., Pfleeger S.L. (1997) *"Software Metrics"*, 2nd edition, PWS Publishing Company.
7. Florac W., Carleton A. (1999) *Measuring the Software Process*, Addison Wesley.
8. Grady R.B., Caswell D.L. (1987)*"Software Metrics: Establishing a Company-wide Program"*, Prentice-Hall.
9. Humphrey, W. (1995) *Discipline for Software Engineering*, Addison-Wesley.
10. Humphrey, W. (2000) *Introduction to the Team Software Process*, Addison-Wesley.
11. IEEE (1999a) *IEEE Standard 1061-1998, IEEE Standard for a Software Quality Metrics Methodology"*. IEEE Inc.
12. IEEE (1999b) *IEEE Standard 1045-1992, IEEE Standard for Software Productivity Metrics"*. IEEE Inc.
13. ISO/IEC (2002) *International Standard ISO/IEC 15939 Software Engineering—Software Measurement Process*, ISO/IEC.
14. Jenkins M., López R. (2002) Diseño e Implementación de un Sistema de Métricas de Software para mejorar los Procesos de una Organización de Desarrollo de Sistemas, *II Workshop de Ingeniería de Software, Jornadas Chilenas de Computación*, Copiapó, Chile.
15. Jenkins M., Jiménez G. (2004) Métricas para CMM Nivel 3, *Software Engineering Process Group Latin America (SEPGLA 2004)*, Guadalajara, México.
16. Jenkins M., Varela M., García C. (2005) Administración Cuantitativa de los Procesos en una Organización de software Pequeña, *Software Engineering Process Group Latin America (SEPGLA 2005)*, Guadalajara, México.
17. Kan S. (2003) *Metrics and models in software Quality Engineering*, 2nd Edition. Addison-Wesley.
18. McGarry, J. et al. (2002) *Practical Software Measurement: Objective Information for Decisión Makers*, Addison-Wesley.
19. Microsoft Corp. (2002) *Interbank Electronic Payment System Fast, Stable with Visual Basic .NET and SQL Server 2000*, http://download.microsoft.com/documents/customerevidence/6125_BCCR_CaseStudy Final.doc. Octubre 2002.
20. Paulk M, Curtis B, Crisis M, Weber C. (1993) *Capability Maturity Model for Software, Version 1.1*, Addison-Wesley.
21. Pressman, R. (2001) *Software Engineering, A practiotioner's approach*, 5ta edición, McGraw Hill.