¿Quality Control = Project Control?
Indicadores Objetivos para Control de Proyectos de Desarrollo de Software

Lic. Juan Pablo Pussacq Laborde
Jefe de la Oficina de Proyectos, RMyA

Introducción

El Problema

Es común que los proyectos de desarrollo de software terminen con importantes desvíos en costo y calendario o, en algunos casos, que no terminen. Existen muchas causas a este problema. Una de ellas es no contar con buenas **herramientas de control**, a pesar de que generalmente tenemos buenas herramientas de planificación. El objetivo de este trabajo es presentar algunas ideas que ayudan a tratar esta causa.
Como líderes de proyecto, nos hacemos muchas veces las siguientes preguntas:

¿Cuál es el grado de avance? *¿En qué fecha terminaremos?* *¿Cuánto gastaremos realmente?*

Normalmente no encontramos las respuestas o las encontramos y son muy **poco confiables**. Eso hace que no tengamos buenos elementos para tomar decisiones.

Problemas clásicos

Uno de los problemas típicos es el síndrome del 90% (figura 1), síndrome por el cual el proyecto avanza sin problemas hasta que llega al 90% y se estanca.

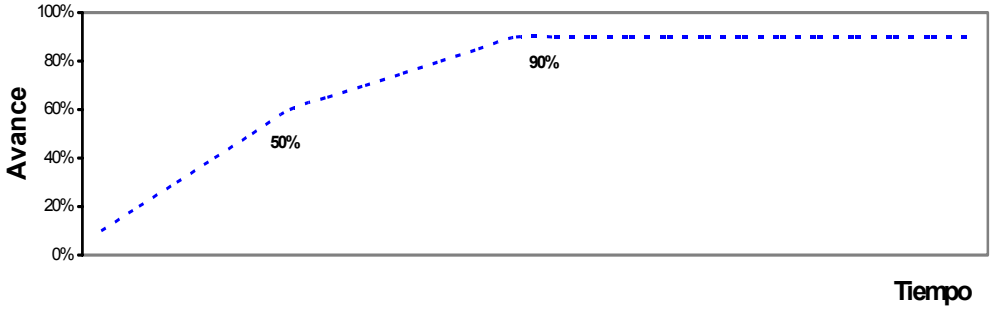

Figura 1 – Síndrome del 90%

Normalmente esto sucede porque utilizamos información **subjetiva** para registrar avance y cuando nos enfrentamos a la realidad, nos enteramos que el avance no era real. La realidad puede estar dada por una etapa de **prueba** o por un **usuario** enfrentándose por primera vez al producto. Si hubiésemos contado con información objetiva en el momento adecuado, tal vez hubiéramos sabido que el proyecto estaba al 20% cuando todos creíamos que estaba al 50%. Esto nos hubiera permitido tomar las decisiones correctas en el momento correcto.

Primer paso para una solución

Lo primero que debemos entender como líderes de proyecto es que no podemos basarnos en información subjetiva para tomar decisiones. Esto parece trivial, pero es muy común el "**pensamiento mágico**" en proyectos de desarrollo de software. Como líderes, no debemos concluir que hay avance sólo porque alguien dijo "avanzamos", debemos reunir evidencia física.

Control de Proyectos basado en Control de Calidad

Una posible evidencia física que sirve como base para la registración de avance puede ser una porción de software construida. En un proyecto en donde existe un equipo de desarrollo y un equipo de prueba que trabajan en paralelo, podemos registrar avance periódicamente basado en evidencia física. **¿Cuál es la evidencia?**

*El producto **desarrollado**...* *pero también **probado**...* *y **estabilizado** (sin defectos críticos).*

Originally published as a part of 2004 PMI Global Congress Proceedings – Buenos Aires, Argentina

De esta manera estamos utilizando el resultado de Control de Calidad como información de entrada para el control. Adicionalmente podremos medir no sólo la velocidad de construcción, sino también la **velocidad de corrección**, obteniendo información sobre cuánto demorará la estabilización del producto.

¿Cuál es el grado de avance?

El Problema

Existen **errores clásicos** a la hora de medir avance en proyectos de desarrollo de software:
a) **Avance por calendario**: se mide avance sólo por paso del tiempo. Aunque parezca trivial, es algo muy común. Planificamos una tarea para que se complete en 20 días. Al décimo día informamos que estamos al 50% sin verificar que efectivamente hayamos realizado el 50% del trabajo.
b) **Avance por código completo**: se mide avance cuando una porción del código está completa, pero sin saber el grado de estabilidad que tiene. Decimos que un determinado módulo de la aplicación está al 100% porque el desarrollo ha terminado, pero ¿Tiene defectos? ¿Cumple con los requerimientos de los usuarios?
Ambos errores llevan al síndrome del 90% que nombramos anteriormente.

El Indicador de Funcionalidad Completa

El indicador de Funcionalidad Completa mide avance cuando una funcionalidad está completa, pero...
No hay avance si la funcionalidad no está completa
No está completa la funcionalidad si no está desarrollada, probada y estabilizada.

Paso 1 - Determinar las Funcionalidades

El primer paso para armar este indicador es **encontrar las funcionalidades** (Anderson, 2004, p. 184). Para ello dividimos el producto a construir en partes. ¿Cuántas funcionalidades? Debemos buscar una cantidad óptima que balancee el detalle necesario para medir avance con la carga administrativa para mantener el indicador.

Paso 2 - Asignar un Peso a cada funcionalidad

Hay funcionalidades más costosas que otras. Es indispensable conocer el **costo de las funcionalidades**, ya que en caso contrario, registraríamos el mismo avance por una funcionalidad simple que por una compleja. Dado que no es sencillo conocer el costo exacto de cada funcionalidad se propone para este indicador asignar un **peso** a cada funcionalidad. Cuando más detallado sea el peso, más exacta será la información de avance, pero más difícil será calcularlo. En la práctica suele alcanzar con clasificar a las funcionalidades en **Simples**, **Medianas** o **Complejas**.
Existen variantes para asignar pesos. La elección correcta depende del tamaño del proyecto y de la información que tengamos al momento de calcular el peso. Pero no debemos olvidar que tenemos que **balancear exactitud con carga administrativa**. Veamos algunos ejemplos de convenciones de peso:
a) **Caso 1**: Clasificar a las funcionalidades en Simples (1), Medianas (2) o Complejas (3)
b) **Caso 2**: Variante del anterior, pero utilizando 5 estados (Anderson, 2004, p. 187)
c) **Caso 3**: Utilizar como peso la cantidad de esfuerzo que se ha estimado para construir la funcionalidad. En este caso tendremos más exactitud y puede ser utilizable cuando ya disponemos de esta información en el momento de armar el indicador.

Paso 3 - Estimar la fecha en que la funcionalidad estará completa

Paso 4 - A medida que avanza el proyecto registrar las fechas reales de funcionalidad completa

Con toda la información registrada (tabla 1) obtenemos el Indicador de Funcionalidad Completa (figura 2).

Funcionalidad	Peso	Fecha estimada	Fecha Real
Parte A	2	12/10/2004	
Parte B	3	15/10/2004	
Parte C	1	17/10/2004	18/10/2004
...			

Tabla 1 – Datos para el Indicador de Funcionalidad Completa

Originally published as a part of 2004 PMI Global Congress Proceedings – Buenos Aires, Argentina

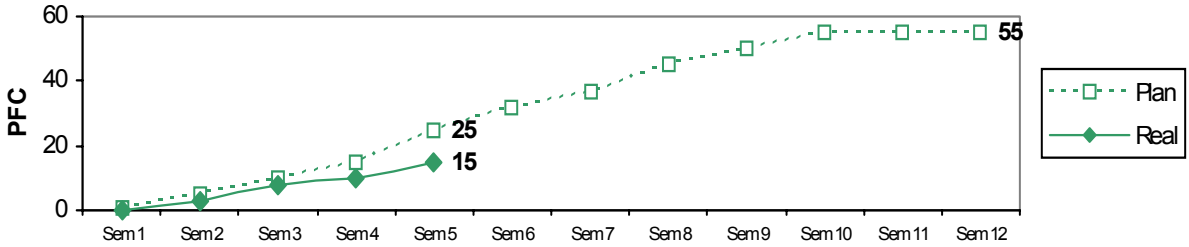

Figura 2–Indicador de Funcionalidad Completa

Obsérvese en el ejemplo que el eje vertical acumula los puntos de funcionalidad completa (PFC), sumando los pesos en cada fecha. En este caso la convención es: una funcionalidad simple equivale a 1 PFC, una funcionalidad media a 2 PFC y una funcionalidad compleja a 3 PFC.

Información adicional

a) **Curva de código completo**: muestra cuándo el equipo de desarrollo entrega código nuevo al equipo de control de calidad. Ese código aún no está probado ni estabilizado. Se debe tener especial cuidado con las conclusiones que se obtengan con esta curva ya que no está basada en evidencia física validada y suele ocultar el síndrome del 90%.
b) **Curva de funcionalidad aprobada por usuario**: marca la funcionalidad que además de estar completa ha sido validada por el usuario final. Indica el **avance más seguro** ya que posee una **aprobación de usuario**. A diferencia de la funcionalidad completa, es imposible lograr avance semanal con esta curva, pero debería aumentar escalonadamente cada vez que el proyecto realice una entrega al usuario. Es muy útil graficarla, especialmente si trabajamos con un esquema de entregas incrementales al usuario (Pussacq Laborde, 2003, p. 12).
c) **Productividad**: se obtiene dividiendo los PFC reales sobre la unidad de tiempo. En la figura anterior, la productividad es 3 PFC x Sem (15 PFC reales / 5 semanas reales). Esto nos permitiría hacer la siguiente extrapolación lineal: necesitaré 13,33 semanas para obtener los 40 PFC restantes (40/3). Puede servir para predecir el futuro (figura 3).

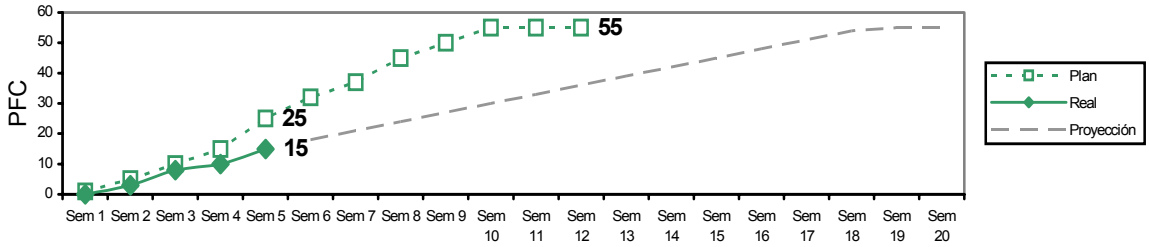

Figura 3–Indicador de Funcionalidad Completa con proyección

Consideraciones sobre este indicador

a) **Validez**: este indicador sólo es valido en etapa de construcción. No se utiliza en el período final de estabilización ya que durante ese período la funcionalidad real es similar a la planificada.
b) **Proceso**: este indicador no generará avance si el equipo no se focaliza en cerrar temas. El indicador es binario, la funcionalidad está completa o no está.
c) **Síndrome del 0%**: si se considera completa a una funcionalidad cuándo no tiene ningún defecto, evitaremos el síndrome del **90%,** pero caeremos en el síndrome del **0%,** es decir, no registraremos avance porque siempre habrá algún defecto pendiente. Es por eso que una funcionalidad está completa cuando no posee defectos críticos, pero sí tiene defectos pendientes.
d) **Funcionalidad = código**: es su forma más pura, este indicador sólo considera funcionalidad al producto final para el usuario. Una especificación no es producto final. Sin embargo, en algunos proyectos en donde hay una marcada etapa de análisis al principio y un equipo de control de calidad que verifica el análisis, puede considerarse funcionalidad a la especificación para poder utilizar este indicador como una herramienta de control.

Originally published as a part of 2004 PMI Global Congress Proceedings – Buenos Aires, Argentina

El Indicador de Nivel de Calidad

Como vimos hasta ahora, el indicador de funcionalidad completa divide al producto en dos estados: funcionalidad **completa o no completa**. Algunas veces necesitamos utilizar **estados intermedios**. Estos estados muestran el ciclo de vida por el que va pasando el producto desde el punto de vista de la calidad (figura 4) y permiten tener un indicador más detallado que denominamos **Nivel de Calidad** (tabla 2, figura 5).

Figura 4–Estados utilizados por el Indicador de Nivel de Calidad

Funcionalidad	Peso	Fecha estimada	Fecha Real	Estado
Parte A	2	12/10/2004		**Con defectos críticos**
Parte B	3	15/10/2004		**En elaboración**
Parte C	1	17/10/2004	18/10/2004	**Aprobado por Usuario**

Tabla 2 – Datos del Indicador de Nivel de Calidad

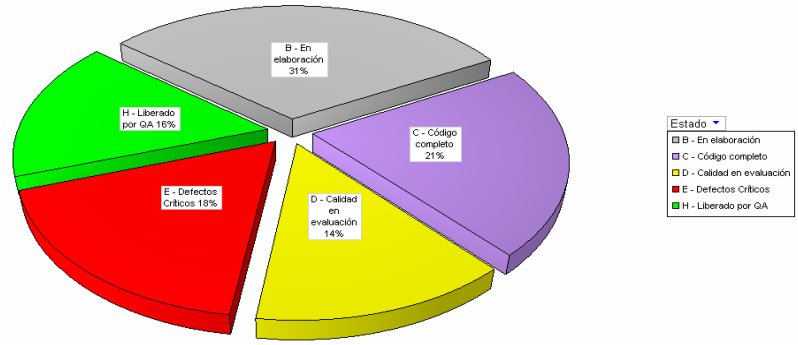

Figura 5 – Indicador de Nivel de Calidad

Análisis de Avance

Si clasificamos cada funcionalidad de diferentes maneras, podremos hacer distintos **análisis de avance** con mayor o menor nivel de detalle. Por ejemplo por: Tarea, Producto (figura 6), Sub nivel de Producto, Equipo, WBS, Ciclo de Negocio, etc.

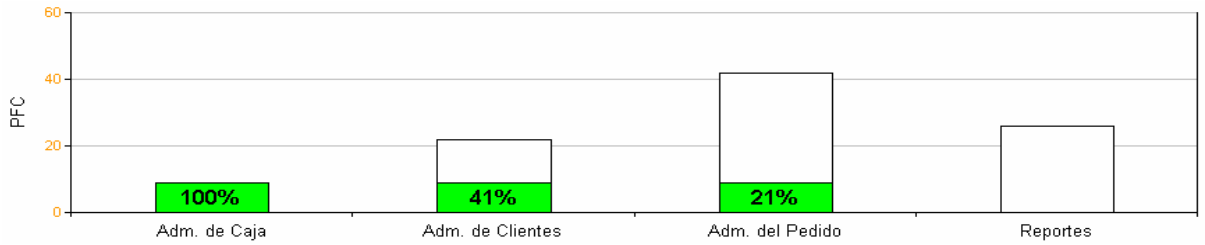

Figura 6 – Funcionalidad Completa dentro de cada Producto en un Proyecto

Si bien la funcionalidad completa es binaria (está completa o no), podemos obtener un porcentaje de avance si agrupamos en un nivel superior. El producto Administración de Clientes (figura 6) puede contener 100 funcionalidades de las cuales hay 41 que ya están completas.

Originally published as a part of 2004 PMI Global Congress Proceedings – Buenos Aires, Argentina

¿En qué fecha terminaremos?

El Problema

Como dijimos, es muy común que los proyectos de software se desvíen. Con el **indicador de funcionalidad completa** podemos obtener una **tendencia que nos permite predecir el futuro** y en consecuencia la fecha de fin. Pero siempre es una incertidumbre conocer cuánto costará en tiempo y recursos la **estabilización** (Microsoft, 2002, p. 34-40) de un producto. La etapa de estabilización comienza cuando el indicador de funcionalidad completa deja de tener validez. En este punto ya está todo construido, sólo resta corregir los defectos remanentes y encontrar los que no encontramos aún. La pregunta entonces es **¿Cuándo liberaré?**

El Indicador de Evolución de la Prueba

El indicador de Evolución de la Prueba se basa en una idea simple: **medir los defectos**, cuántos aparecen y cuántos se cierran por día (McConnell, 1996, p. 384):

Paso 1 - Registrar los defectos nuevos a medida que aparecen
Paso 2 - Registrar los cambios de estado de los mismos hasta que se cierran definitivamente

Cumpliendo esos dos pasos, podemos obtener el Indicador de Evolución de la Prueba (figura 7).

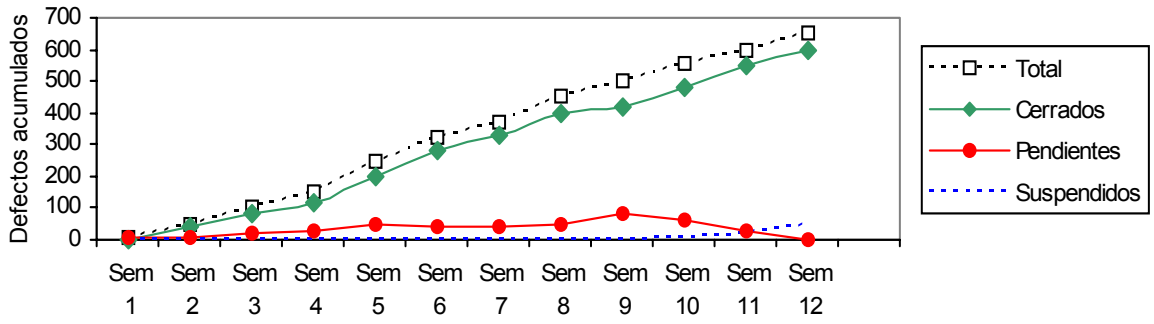

Figura 7 – Indicador de Evolución de la Prueba

Analizando el comportamiento de los defectos a lo largo del proyecto este indicador nos permite obtener estadísticas que determinan la **velocidad de corrección** y por lo tanto el **tiempo estimado** para estabilizar el producto. Podemos saber cuántos defectos se encuentran y cuántos se cierran en un determinado período de tiempo, por ejemplo las últimas dos semanas. Si la muestra es representativa y se elige el período correcto de análisis, se pueden obtener algunas **conclusiones**:
- Buenas conclusiones: el proyecto está bajo control, el proyecto finaliza en tres semanas.
- Malas conclusiones: aparecen más defectos por día de los que cerramos, no terminaremos nunca.

Cuando aplicamos un proceso de prueba en paralelo, el volumen de defectos es alto, lo que nos permite que las muestras analizadas en general sean representativas. Teniendo en cuenta esto y tomando con cautela las proyecciones lineales, el indicador **nos puede ayudar a tomar algunas decisiones** como las siguientes: agregar más recursos para corregir defectos, recortar funcionalidad, priorizar el orden de corrección, no corregir todos los defectos (suspender), etc.

Consideraciones sobre este indicador

a) **Validez**: este indicador es válido durante todo el proyecto cuando hay prueba en paralelo al desarrollo. Si sólo hay prueba al final, el indicador es muy útil durante la etapa de prueba final y estabilización.
b) **Proceso**: el indicador es muy útil para detectar el síndrome del 90 %, por ejemplo cuando avanza el código completo, pero la funcionalidad posee gran cantidad de defectos pendientes. Asimismo ayuda a detectar si se está aplicando en forma correcta el proceso de desarrollo y prueba en paralelo. En el ejemplo (figura 7) el proceso se está aplicando adecuadamente.

Originally published as a part of 2004 PMI Global Congress Proceedings – Buenos Aires, Argentina

El Indicador de Cobertura de la Prueba

Si se complementa la Evolución de la prueba con la Cobertura de la prueba, podemos obtener información más exacta. La Cobertura de la Prueba muestra cuánto habría que probar, cuánto se pudo probar y cuánto funciona bien. Está medición la realiza a partir de los estados (Kit, 1995, p. 135) de los casos de prueba (figura 8):
a) **Planificados**: cantidad de casos a ejecutar.
b) **Disponibles**: lo que realmente puedo ejecutar teniendo en cuenta lo que el equipo de desarrollo entregó al equipo de prueba.
c) **Ejecutados**: lo que el equipo de prueba pudo ejecutar.
d) **Ejecutados OK**: casos ejecutados sin errores. Es otra forma de ver **avance del proyecto**.

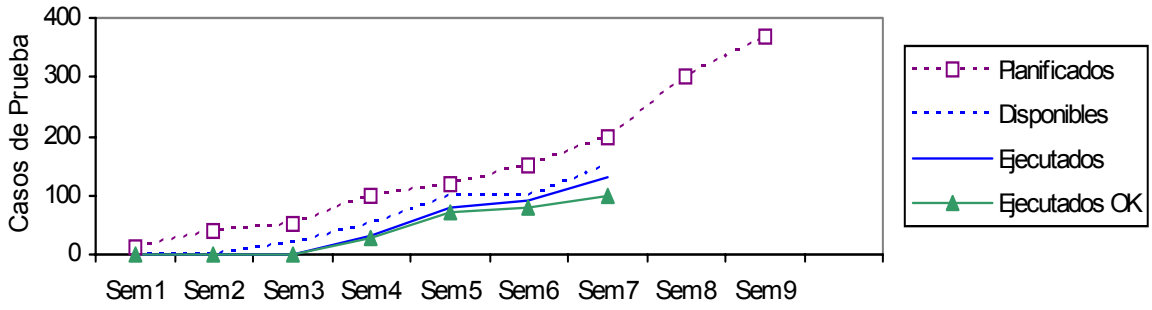

Figura 8 – Indicador de Cobertura de la Prueba

Este indicador puede ser utilizado para tener otra forma de validar **avance del proyecto**. El avance está dado por los casos de prueba que han sido ejecutados sin problemas.

Por otro lado, utilizando el indicador de cobertura para saber cuánto falta probar y tomando la información de defectos del indicador de evolución de la prueba podemos calcular cuántos defectos más aparecerán. Con este dato y la fecha de fin estimada podríamos controlar cuántos defectos deberíamos estar encontrando y cerrando por día. El objetivo: mantener **bajo control la etapa de estabilización**.

¿Cuánto gastaremos realmente?

El Problema

Ya conocemos el grado de avance y tenemos menos incertidumbre sobre la fecha de fin, pero... **¿Cuánto costará realmente?** Consideramos útil para responder esta pregunta utilizar el indicador de **Earned Value**. Una de los problemas más importantes de este indicador es la dificultad para calcular el valor ganado en proyectos de desarrollo de software. La propuesta es utilizar el avance calculado por funcionalidad completa para calcular el valor ganado. Esto tiene algunas ventajas y algunas desventajas que analizaremos.

El Indicador de Earned Value

No es propósito de este trabajo explicar Earned Value (Durrenberger, 2003), sino mostrar un indicador de Earned Value creado a partir de un indicador de Funcionalidad Completa (tablas 3 y 4):

Valor	¿Cómo calcularlo basado en Funcionalidad Completa?	Ejemplo
BAC	Tomar el costo del proyecto que podamos relacionar directamente con la construcción de software	**BAC = $ 55.000** (hipótesis)
Plan	Se obtiene a partir del valor unitario de un PFC (VU PFC): • Plan = PFC planeados x VU PFC • VU PFC = BAC / PFC totales	**VU PFC = $1.000** = $55.000 / 55 PFC totales **Plan = $ 25.000** = 25 PFC planeados x $ 1.000
Actual	Se obtendría a partir del costo real, por ejemplo el valor de las horas insumidas hasta el momento.	**Actual**: **$ 30.000** (hipótesis)
Earned Value	El Valor ganado son las funcionalidades completas. • Earned Value = PFC reales x VU PFC	**Earned Value = $ 15.000** = 15 PFC x $ 1.000

Tabla 3 – Earned Value basado en Funcionalidad Completa

BAC	Plan	Actual	Earned Value	CPI	SPI
$ 55.000	$ 25.000	$ 30.000	$ 15.000	0.5	0.6
55 PFC planeados totales	25 PFC planificados a la fecha		15 PFC reales		

Tabla 4 – Indicador de Earned Value resumido

Consideraciones sobre este indicador

a) **Alcance**: es necesario separar los costos que no están asociados directamente a la construcción de software. Por ejemplo, la adquisición de un equipo, la planificación inicial, el despliegue, etc. Otros costos, como la administración del proyecto, podrían distribuirse dentro de los PFC.
b) **Validez**: al igual que el indicador de Funcionalidad Completa, sólo aplica a la construcción, no a la estabilización.
c) **Margen de error**: existe cierto margen de error aceptable con el objetivo de no aumentar la carga administrativa debido a:

- Los pesos generan información inexacta
- El Actual está insumido en funcionalidades que aún no están completas
- Los costos indirectos que se distribuyen en los PFC pueden generar margen de error.

Conclusiones

Hemos presentado una serie de indicadores cuyo objetivo es controlar un proyecto de desarrollo de software. Cuanto más exacto es un indicador más difícil es su aplicación real, por lo cual hemos intentando balancear las siguientes características:
a) **Objetividad**: los indicadores deben registrar información objetiva, basada en evidencia física y no en una opinión subjetiva del que la genera.
b) **Administración**: si bien los indicadores tienen costo, no deben poseer demasiada carga administrativa si queremos que la implementación sea eficiente.
c) **Foco en resultados**: un buen indicador no mide inversión, sino lo que obtengo de ella.
d) **Comprensión**: deben mostrar información comprensible por el auditorio.

Todos los indicadores presentados tienen como objetivo mostrar el **estado del proyecto**, por ejemplo a través de un Tablero de Control (figura 9) y mejorar la toma de decisiones. Pero es importante tener en cuenta que además son muy útiles para detectar **fallas en el proceso** como por ejemplo: equipo mal enfocado, acumulación de código, casos de prueba poco eficientes, necesidades de priorización de defectos, etc.

Avance planificado	50%
Avance real	27%

Fecha comprometida	Enero de 2005
Fecha estimada	Marzo de 2005

BAC	$ 55.000
Plan	$ 25.000
Actual	$ 30.000
Earned Value	$ 15.000
CPI	0.5
SPI	0.6
EAC	$ 110.000

	Real	Plan
Defectos Totales	1.000	2.000
Defectos Abiertos	200	-
Detectados por día	30	40
Cerrados por día	21	60
Cobertura de la Prueba	50%	-

No iniciado	30 %
En elaboración	10 %
Código completo	-
En evaluación	10 %
Defectos críticos	23 %
Defectos no críticos	-
Liberado por QA	27 %
Aceptado por Usuario	-

Figura 9 – Tablero de Control

Los indicadores presentados han sido utilizados en **proyectos reales** demostrando su valor a los responsable de la toma de decisiones. Se debe tener en cuenta que no todos los proyectos necesitan todos los indicadores. Esto puede ser variable dependiendo de las características iniciales del proyecto y de los cambios que el mismo proyecto sufra a lo largo de su historia. Sin embargo, en líneas generales podemos recomendar que como mínimo se utilice el indicador de **Funcionalidad Completa** y el indicador de **Evolución de la Prueba**.

Bibliografía

Anderson, D.J. (2004). *Agile Management for Software Engineering*. Upper Saddle River: Prentice Hall.

Durrenberger, M. (2003). *An Earned Value Tutorial*. Retrieved 20/Jul/2004, from http://www.oakinc.com/pdf/EVTutorial.pdf

Kit, E. (1995). *Software Testing in the real world*. USA: ACM Press.

McConnell, S. (1996). *Rapid Develoment*. Redmond. Washington: Microsoft Press.

Microsoft. (2002, Junio). *MSF Process Model v.3.1. Microsoft Solutions Framework.* Retrieved 5/Jul/2004, from http://www.microsoft.com/msf

Pussacq Laborde, J.P. (2003). *Daily Build & Smoke Test - Administración*. Retrieved 27/Oct/2004, from http://www.rmya.com.ar/Download/Daily%20Build%20and%20Smoke%20Test.pdf