Midiendo la Productividad del Desarrollo de Aplicaciones

Allan J. Albrecht
Proc. Joint SHARE/GUIDE/IBM Application Development Symposium (October, 1979),83-92
IBM Corporation, White Plains, New York
Traducido por Fatto Consultoría y Sistemas
(www.fattocs.com)

Traducción original disponible en:
http://fattocs.com/files/pt/artigos/MeasuringAplicationDevelopmentProductivity.pdf

En este documento me gustaría compartir con ustedes algunas experiencias en mediciones de productividad en proyectos de desarrollo de aplicaciones en la unidad DP Services de IBM. Tengo varios objetivos para este documento:

- Describir nuestra medida de productividad, la cual ha sido efectiva midiendo productividad en todas las fases de un proyecto incluyendo la fase de diseño, y nos ha permitido comparar los resultados de proyectos que usan diferentes lenguajes de programación y tecnologías.
- Mostrar como usamos esa medida para determinar la tendencia de productividad en nuestra organización.
- Identificar algunos factores que afectan la productividad y mostrar cómo pudimos determinar su importancia relativa.

La unidad DP Services se compone de cerca de 450 personas que trabajan en el desarrollo de aplicaciones para los clientes de IBM bajo contrato. Tanto los clientes como empleados están ubicados a lo largo de los Estados Unidos. En cualquier momento dado hay de 150 a 170 contratos en curso. Los proyectos cubren todas las industrias. Ellos abordan el espectro de requerimientos funcionales de procesamiento de datos: Entrada y control de pedidos, procesamiento de reclamaciones de seguros, los sistemas de información de pacientes del hospital, sistemas de control de comunicación de datos, etc. El tamaño promedio de los contratos es de 2 o 3 personas. Un número de proyectos cada año requiere de 15 a 20 personas, y otros varios requieren de hasta 35 o 40 empleados y clientes, trabajando bajo la supervisión de nuestro director de proyectos, para desarrollar la aplicación.

Cada proyecto se realiza bajo el Proceso de Desarrollo de Aplicaciones de la DP Services (figura 1). No quiero dar la impresión de que en cada uno de estos cientos de contratos realizamos proyectos completos. En la mayor parte de los contratos realizamos sólo las tareas específicas que cubren parte de la función proporcionada por el proyecto. Nuestro

enfoque es un enfoque por fases para el desarrollo de aplicaciones. El enfoque por fases consiste en un diseño de sistema seguido de la fase de implementación del sistema. El diseño del sistema es completado y aprobado antes de que de lugar el inicio de su implementación. El gráfico representa la distribución del esfuerzo de desarrollo en esas fases. Se basa en la experiencia de DP Services en proyectos en los últimos 3 o 4 años.

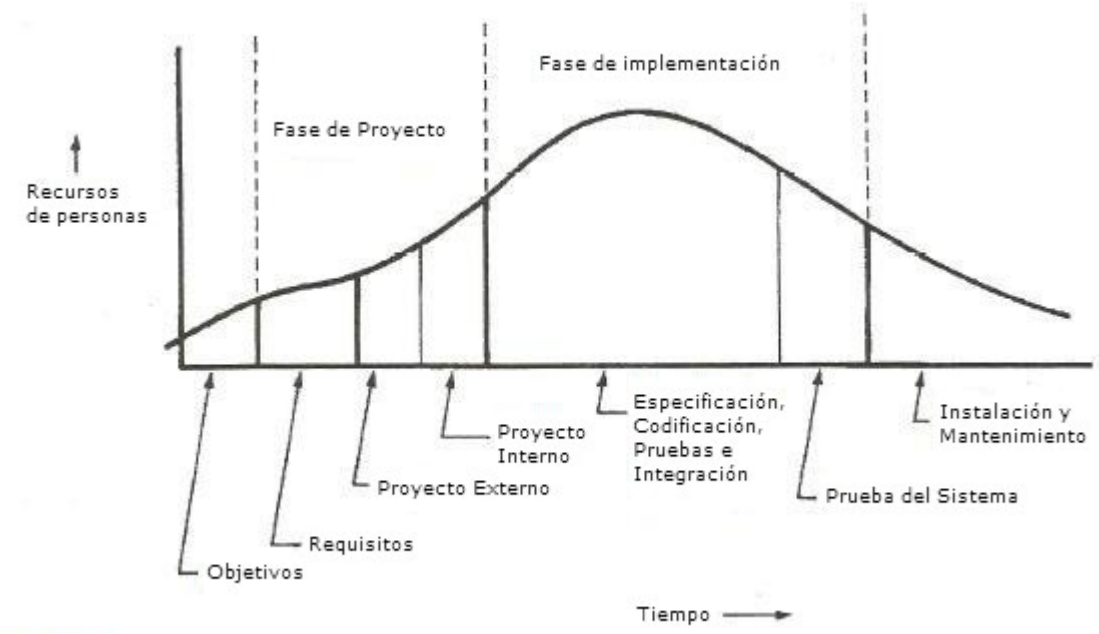

Figura 1

Nuestra experiencia muestra que la fase de diseño toma alrededor de 20% de las horas de trabajo y la fase de implementación dura aproximadamente 80% de las horas de trabajo en un proyecto terminado. Esta distribución ha sido bastante consistente. Nuestros últimos proyectos en comparación con nuestros primeros muestran la misma distribución de esfuerzo de trabajo.

Más tarde les mostraré un modelo del porcentaje de esfuerzo para cada tarea en este gráfico. También voy a demostrar que ha habido un aumento significativo en la productividad del proyecto durante ese tiempo. Como la forma de la distribución no ha cambiado, la conclusión es que hemos ido aumentando la productividad de la fase de diseño del sistema al mismo ritmo que la fase de implementación.

Las siguientes técnicas disciplinadas de gestión han ayudado a lograr esa mejora en la productividad.

Antes de que el proyecto de diseño comience nos aseguramos de que los objetivos del proyecto están completamente descritos y aprobados, incluyendo las funciones que deben proporcionarse, una declaración de trabajo para llevar a cabo, y la planeación y el costo estimado del proyecto.

La fase de diseño del sistema comienza con la definición de requisitos, donde los requerimientos del negocio se definen para la aprobación del cliente. Durante el diseño externo e interno del sistema nosotros proveemos al cliente con un diseño del sistema que obtendrán si lo aprueban. Ofrecemos un plan y una propuesta para un mayor desarrollo e implementación. La implementación del sistema sigue con el desarrollo del programa que termina con una prueba del sistema aprobado por el cliente y demostración. Ese es el enfoque por fases. Cada proyecto pasa por esas fases básicas.

Para llevar a cabo cada proyecto utilizamos un sistema de gestión de proyectos (figura 2). La figura 2 muestra los procesos de nuestro sistema de gestión de proyectos.

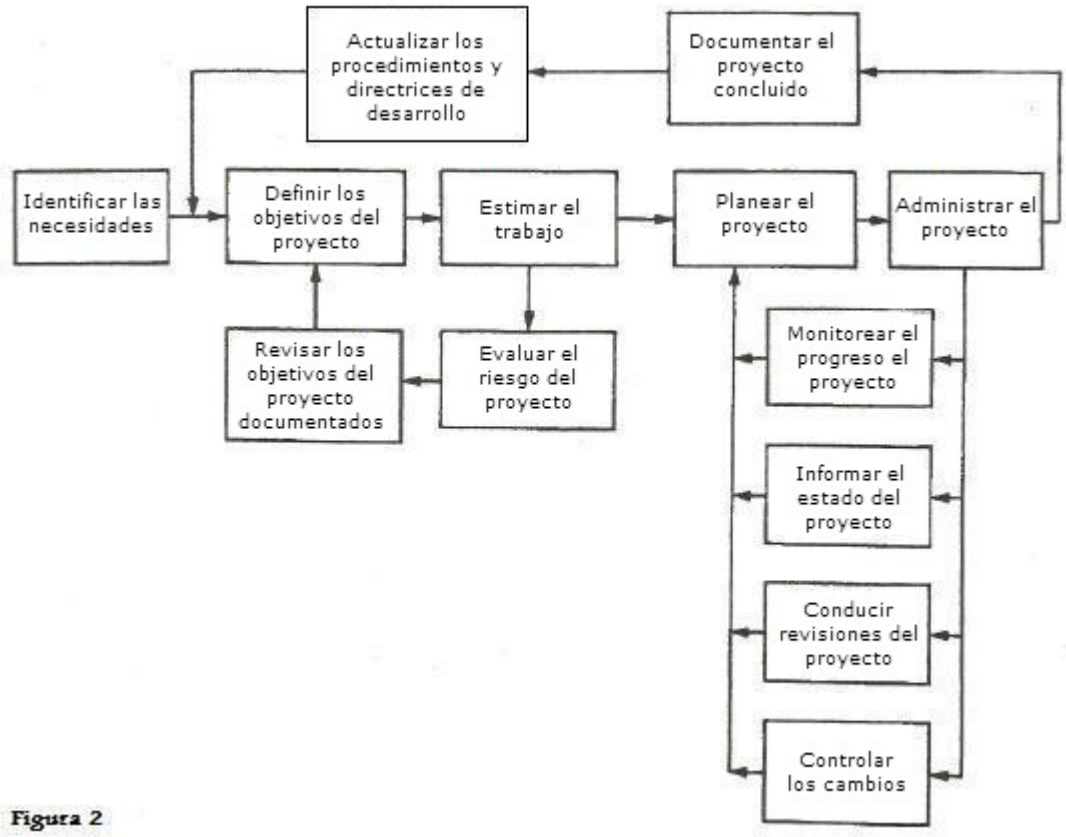

Figura 2

Después de identificar la necesidad con el cliente, los objetivos del proyecto están completamente documentados. Usando retroalimentación de productividad de trabajos anteriores, y una planificación completa de las tareas del proyecto a realizar, estimamos el proyecto. Luego pasamos por el primero de los ciclos de retroalimentación donde evaluamos el riesgo del proyecto con un cuestionario estructurado de 28 preguntas. Tenemos a un grupo independiente de garantía de los sistemas revisando todo el documento de objetivos del proyecto para asegurarse de que está correcto, de que no contiene información errónea, y que contiene toda la información acordada. Después de que nosotros y el cliente estemos de acuerdo con sus objetivos, y que tenemos la intención de entregar estos objetivos, empezamos el proyecto.

El proyecto entonces es planeado con detalles. Pedimos que el proyecto tenga un tamaño promedio de tareas de 20 a 40 horas de trabajo. Hacemos un seguimiento de los resultados con el plan una vez por semana, y reportamos el estado del proyecto a la gestión de IBM y al cliente. Continuamos la revisión de proyectos teniendo al grupo de garantía de sistemas revisando el proyecto cada seis meses. Por encima de todo controlamos cambios, no para prevenirlos, pero para asegurarnos de que cada parte entienda el valor y el costo, y apruebe el cambio, antes de que sea implementado.

Un elemento clave del sistema de gestión de proyecto es, a medida que completamos proyectos más grandes de 1000 horas de trabajo, documentamos el proyecto terminado. ¿Cuántas horas se gastaron en las diversas tareas? ¿Qué fue entregado? ¿Cuáles fueron las acciones útiles o eventos perjudiciales? ¿Bajo qué sistema y ambiente el desarrollo tuvo lugar? Este documento es analizado a continuación, junto con otros informes de terminación de proyecto para actualizar las normas que utilizamos para guiar futuros proyectos. Medimos la productividad de desarrollo de aplicaciones a través del ciclo de retroalimentación en la parte superior de la figura 2, el resultado del proyecto completo es documentado.

¿Por qué medir la productividad? Para responder algunas preguntas para la gestión como para nuestros clientes. ¿Estamos haciendo lo mejor que podemos? ¿Somos competitivos? ¿Estamos mejorando? La medición de la productividad está destinada a proporcionar algunas respuestas. La medición de la productividad ayuda a identificar y promover específicamente aquellas cosas que mejoran la productividad y evitar las cosas que perjudican a la productividad. Tenemos que seguir identificando y seleccionando los sistemas de desarrollo y tecnologías que ofrezcan más funciones de aplicación con menos esfuerzo y a menor costo.

Hemos aprendido que hay cosas a tener en cuenta en la medición de la productividad. Para empezar, se debe medir todo el proceso, incluyendo la fase de diseño. Los costos pueden ser incurridos en la fase de diseño del sistema también. A continuación, hay tareas y funciones tales como la gestión de proyectos y la arquitectura del sistema que deben ser incluidos ya que contribuyen significativamente a los resultados de productividad. Ignorarlos da una imagen falsa de los costos reales. Por último, todas las medidas son relativas. Se puede medir proyectos contemporáneos uno contra el otro y se puede medir tendencias de tiempo dentro de su organización, pero las comparaciones entre las organizaciones deben manejarse con cuidado a menos que utilicen las mismas definiciones.

La medición de la productividad puede hacer daño. Para evitar esto, mantenga los principales objetivos del proyecto en perspectiva – a tiempo, dentro del presupuesto, un cliente satisfecho. No permita que las mediciones de productividad le desvíen su atención de los principales objetivos del proyecto. La medición de la productividad evita una dependencia en medidas como líneas de código que pueden tener valores muy diferentes en función de la tecnología utilizada. Hacemos un seguimiento de líneas de código, pero sólo como una medida secundaria para comparar proyectos usando el mismo lenguaje.

Los proyectos que fueron analizados fueron 22 proyectos de desarrollo de aplicaciones completos de DP Services. 16 fueron escritos en COBOL, 4 fueron escritos utilizando PL/1, y 2 usaron D MS/VS.

(Aunque todavía usamos una cantidad incidental de código en ensamblador (ALC) para funciones especializadas en algunos de nuestros proyectos, ya no hacemos proyectos completos utilizando ALC, porque no es lo suficientemente productivo. En consecuencia, no tenemos proyectos de ALC en nuestras mediciones. Para aquellos que pudieran tener los datos, el método descrito debería funcionar bien en la medición de la productividad relativa en proyectos ALC.)

Fechas de terminación de proyectos iban desde mediados de 1974 hasta 1979. Los proyectos variaron en tamaño de 500 horas de trabajo a 105.000 horas de trabajo.

Veintidós proyectos pueden parecer ser un número pequeño, teniendo en cuenta que DP Services probablemente completaron cerca de 1500 contratos durante ese tiempo. Pero, éstos 22 eran todos los proyectos que pasaron los criterios de selección en ese periodo de tiempo.

Estas fueron las reglas básicas de selección:
1. Sólo proyectos completos que hayan procedido a través de todas las fases desde definición de requerimientos hasta la prueba final del sistema y la demostración, y que hayan entregado un producto a nuestro cliente, serán analizados.
2. Todo el proyecto tuvo que ser hecho bajo nuestra gestión de proyectos con definiciones de tareas consistentes y procedimientos de gestión.
3. Todas las horas de trabajo dedicadas por nuestro personal y el personal de los clientes tuvieron que ser conocidas y contabilizadas cuidadosamente.
4. Los factores funcionales tuvieron que ser conocidos.

Hemos encontrado que aproximadamente de 3 a 7 proyectos que satisfacen los criterios se completan por año.

Para medir la productividad, tuvimos que definir y medir un producto y un costo. El producto que se analizó fue el valor entregado por las funcionalidades. El número de entradas, consultas, salidas y archivos maestros entregados fueron contados, ponderados, sumados, y ajustados por la complejidad de cada proyecto. El objetivo fue desarrollar una medida relativa de valor de la función entregado al usuario de que era independiente de la tecnología particular o método utilizado.

La base de este método fue desarrollado en los últimos 5 años a partir de la experiencia de estimación de proyectos de DP Services. Como parte de esa estimación validamos cada estimación con una serie de preguntas ponderadas sobre la función de la aplicación y el entorno de desarrollo. Se encontró que el valor básico de la función de aplicación fue sistemáticamente proporcional a un recuento ponderado del número de entradas de usuario externo, salidas, consultas y archivos maestros.

El enfoque general es contar el número de entradas externas de usuarios, consultas, salidas y los archivos principales entregados por el proyecto de desarrollo. Estos factores son las manifestaciones externas de cualquier aplicación. Cubren todas las funciones en una aplicación.

Estos conteos se ponderan por los números diseñados para reflejar el valor de la función para el cliente. Las ponderaciones utilizadas fueron determinadas mediante debate y juicio. Estos pesos nos han dado buenos resultados:

- Número de entradas x 4
- Número de salidas x 5
- Número de consultas x 4
- Número de ficheros maestros x 10

Luego ajustamos el resultado para efecto de otros factores.
Si las entradas, salidas, o archivos son extra complicados, añadimos 5%. Procesamiento interno complejo puede añadir otro 5%. Funciones y rendimiento en línea se abordan en otras preguntas. El ajuste máximo posible es del 50%, expresado como +/- 25% de modo que la suma ponderada es la complejidad media.

Esto nos da un número adimensional definido en puntos de función los cuales hemos encontrado que son una medida relativa efectiva de valor de la función entregada a nuestro cliente. Estas definiciones están presentadas en la hoja de trabajo Valor de Función (figura 3).

IBM DP Services
PLANILLA DEL ÍNDICE DE VALOR DE FUNCIÓN

Fecha: _____
ID del Proyecto: _____

Nombre del Proyecto: _____

Elaborado por: _____ Fecha: _____ . Revisado por: _____ Fecha: _____ .

Resumen del Proyecto	Fecha de Inicio	Fecha Final	Horas Trabajadas	Puntos de Función Entregados o Diseñados
				(calculados)

Calculo de Puntos de Función (Entregados o Diseñados):

Nota: Las definiciones están en la parte de atrás.

	Asignación estimada por el gerente de proyecto: Entregado por nuevo codigo	Entregado al modificar código existente	Entregado al instalar y probar un paquete	Entregado al usar un generador de código	Total (Identificar lenguaje predominante)	
Lenguaje						
Entradas						X 4
Salidas						X 5
Archivos						X 10
Consultas						X 4
Horas de Trabajo						Total
Proyecto						Puntos de Función
Implementación						no ajustados

Ajuste de complejidad: (Grado de influencia estimado para cada factor)

- ___ Copia de seguridad confiable, recuperación y/o la disponibilidad del sistema son proporcionados por el diseño de la aplicación o la implementación. Las funciones pueden ser proporcionadas por código de aplicación diseñado específicamente o mediante el uso de las funciones proporcionadas por software estándar. Por ejemplo, las funciones de copia de seguridad y recuperación estándar IMS.
- ___ Las comunicaciones de datos se proporcionan en la aplicación.
- ___ Funciones de procesamiento distribuido se proporcionan en la aplicación.
- ___ El rendimiento debe ser considerado en el diseño o implementación.
- ___ Además de considerar el rendimiento, se debe sumar la complejidad de una configuración operacional muy utilizada. El cliente quiere ejecutar la aplicación en hardware existente o comprometido, en consecuencia, hay una sobrecarga.
- ___ La entrada de datos en línea se proporciona en la aplicación.
- ___ La entrada de datos en línea es proporcionada por la aplicación, y además de eso la entrada de datos es un requisito de conversación para que una transacción de entrada sea construida a lo largo de múltiples operaciones.
- ___ Los archivos maestros se actualizan en línea
- ___ Entradas, salidas, archivos o consultas son complejas en esta aplicación.
- ___ El procesamiento interno es complejo en esta aplicación.

Grado de influencia sobre la función:

0 - Ninguno	3 - Promedio
1 - Incidental	4 - Significativo
2 - Moderado	5 - Esencial

_____ Grado de influencia total (N)

_____ Ajuste de complejidad igual a (0,75 + 0,01 (N))

Total no ajustado x Ajuste de complejidad = Puntos de Función entregados o diseñados

_____ _____ _____

Figura 3

Definiciones:

Instrucciones generales:

Contar todas las entradas, salidas, archivos maestros, consultas, y funciones que se ponen a disposición del cliente a través de los esfuerzos de diseño, programación o de prueba del proyecto. Por ejemplo, contar las funciones proveídas por un IUP, FDP, o producto de programa si el paquete fue modificado, integrado, probado y en consecuencia proveído al cliente a través de los esfuerzos del proyecto.

Horas de Trabajo:

Las horas de trabajo registradas deben ser las horas de IBM y de los clientes que se utilizaron en las tareas estándares de DP Services aplicables a la fase del proyecto. Las horas de los clientes deben ajustarse a las horas equivalentes de IBM considerando experiencia, capacitación y la eficacia en el trabajo.

Conteo de Entradas:

Cuente cada entrada de sistema que proporciona función empresarial de comunicación de los usuarios al sistema informático. Por ejemplo:
- Formularios de datos
- Pantallas de los terminales
- Formas o tarjetas de escáner
- Transacciones con clave

No contar dos veces las entradas. Por ejemplo, considere una operación manual que toma datos de un formulario de entrada, para formar dos pantallas de entrada, utilizando un teclado para formar cada pantalla antes de pulsar la tecla entrar. Esto se debe contar como (2) entradas no cinco (5).
Cuente todas las entradas únicas. Una transacción de entrada debe ser contada como única si se requiere una lógica diferente de procesamiento que otras entradas. Por ejemplo, las operaciones tales como agregar, eliminar o modificar puede tener exactamente el mismo formato de la pantalla, pero tienen que ser contadas como entradas únicas si requieren procesamiento diferente.
No cuente pantallas de terminales de entrada o salida que son necesarios solamente por el sistema debido a la aplicación técnica específica de la unción. Por ejemplo, las pantallas de DMS / VS, que sólo se proporcionan para llegar a la siguiente pantalla y no proporcionan una función de negocio para el usuario, no deben ser contados.

No cuente con la entrada y salida de cinta y conjuntos de datos de archivo. Estos se incluyen en el recuento de archivos.

No cuente transacciones de consulta. Estos se tratan en una pregunta posterior.

Conteo de Salida:

Cuente cada salida del sistema que proporciona comunicación función de negocio del sistema informático a los usuarios. Por ejemplo:
- Informes impresos
- Pantallas de los terminales
- Salida impresa de terminal
- Mensaje del operador

Contar todas las entradas externas únicas. Una entrada es considerada única si tiene un formato que difiere de otras entradas y salidas externas, o, si se requiere lógica de procesamiento única para proporcionar o calcular los datos de salida.

No incluya pantallas de los terminales de salida que proporcionan sólo un mensaje de error simple o acuse de recibo de la transacción de entrada, a menos que se requiere lógica de procesamiento única significativa además de la edición asociada con la entrada, que fue contado.

No incluya salidas de transacciones de consulta en línea donde la respuesta se produce de inmediato. Estos se incluyen en una pregunta posterior.

Conteo de Archivos:

Cuente cada archivo lógico legible por la máquina, o agrupación lógica de datos desde el punto de vista del usuario, que se genera, utiliza, o es mantenido por el sistema. Por ejemplo:
- Archivos de tarjeta de entradas
- Archivos de disco
- Archivos de cinta

Contar los principales grupos de datos de usuario dentro de una base de datos. Contar archivos lógicos, no conjuntos de datos físicos. Por ejemplo, un archivo de cliente que requiere un archivo de índice por separado debido al método de acceso se cuenta como un archivo lógico no dos. Sin embargo, un archivo de índice alfabético para ayudar en el establecimiento de la identidad del cliente sería contado.
Cuente todas las interfaces leídas por máquina a otros sistemas como archivos.

Conteo de Consultas:

Cuente cada pareja de entrada/respuesta donde una entrada en línea genera y causa directamente una salida en línea. Datos no son ingresados excepto para propósitos de control y, por tanto, sólo los registros de transacciones se alteran.

Cuente cada consulta con formato de forma única o procesados de forma única que se traduce en una búsqueda de archivos de información o resúmenes específicos que se presentan como respuesta a esa pregunta.

No contar también consultas como entradas o salidas.

Figura 3(Continuación)

Un artículo reciente de Trevor Crossman de Mayo de 1979 describe un enfoque funcional similar para la medición de la productividad del programador. Su enfoque definió las funcionalidades basadas en la estructura del programa. Nuestro enfoque define función con base en atributos externos. Se concentró en tareas de programador de diseño, código, y prueba de unidad. Como puede ver, contemplamos todo el ciclo de desarrollo de aplicaciones desde el diseño del sistema hasta las pruebas. Ambos puntos de vista sobre la necesidad de un enfoque funcional parecen estar de acuerdo.

Creo que el enfoque de DP Services, basado en atributos externos, será más efectivo en la determinación o prueba de las ventajas de productividad en otros lenguajes de mayor nivel y tecnologías de desarrollo.

El costo utilizado fue el de horas de trabajo aportados por IBM y de los clientes trabajando en el proyecto, a través de todas las fases (diseño e implementación). (figura 4). Estos son los elementos de coste incluidos en el análisis. Las distribuciones porcentuales en el modelo actual se han desarrollado a partir de la experiencia en proyectos de DP Services en los últimos 3 o 4 años. La intención es establecer el costo de desarrollo en términos de horas de trabajo que se utilizaron para diseñar, programar y probar el proyecto de aplicación.

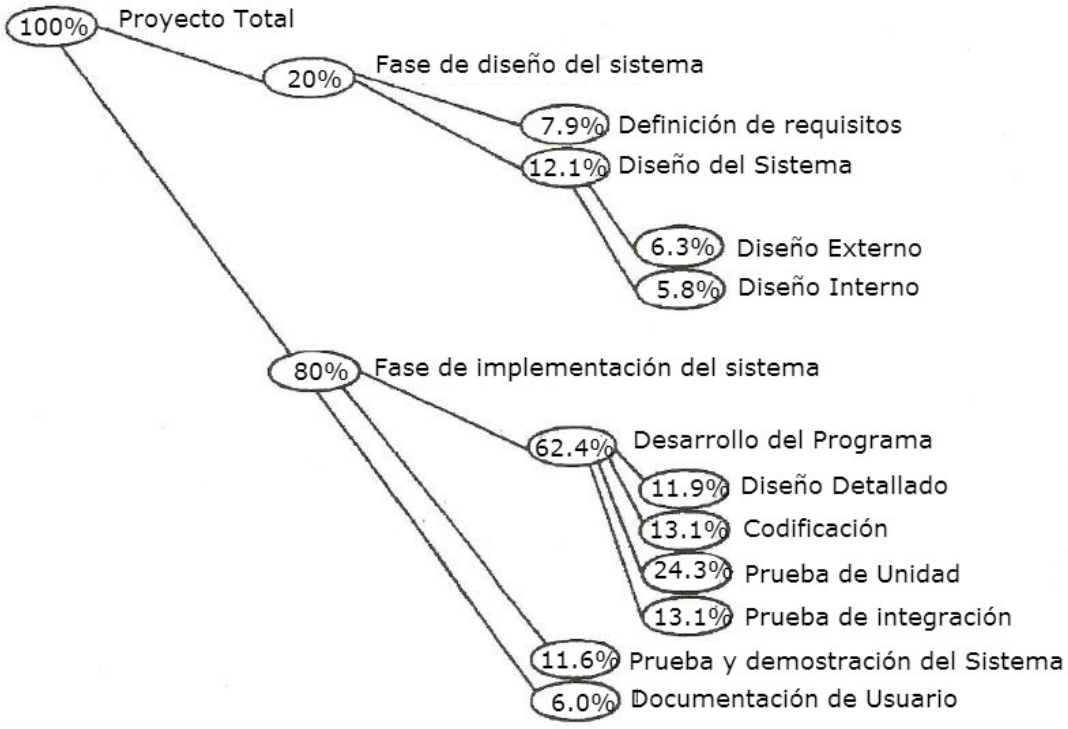

Figura 4

La figura 5 es nuestra tendencia temporal de productividad de 1974 hasta 1978. Hemos trazado las horas trabajadas por cada punto de función entregado. Esto significa que cuanto más bajo mejor, bajo significa que un menor número de horas se gastaron generando

cada punto de función. Hay tres tipos de proyectos representados aquí. Cada punto rojo representa un proyecto de COBOL, cada punto azul representa un proyecto de PL/1 y cada punto blanco representa un proyecto DMS/VS. La línea es un ajuste lineal de los cuadrados mínimos para todos los puntos.

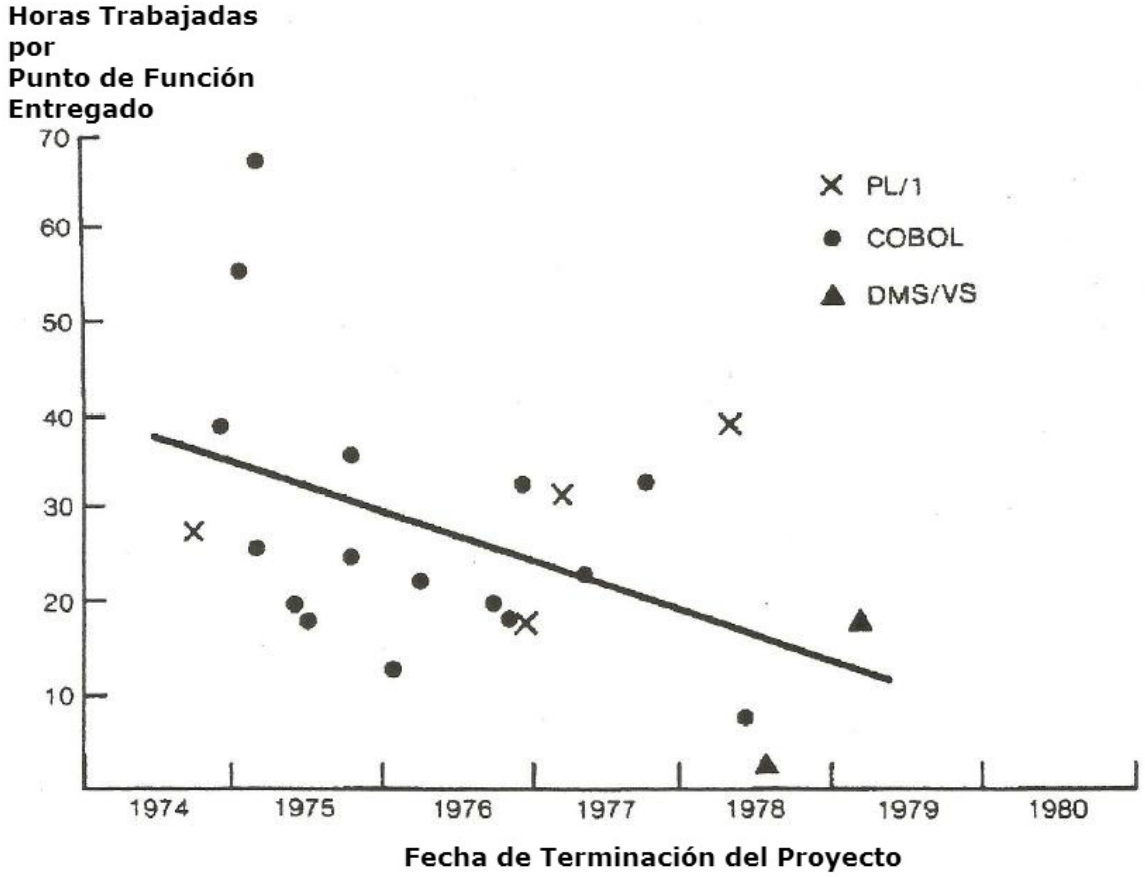

Figura 5

La mejora en la productividad mostrada entre 1974 y 1978 es de aproximadamente 3 a 1. Proyectos de PL/1, DMS/VS y COBOL se combinan para mostrar una tendencia significativa hacia la mejora en la productividad. Debido a que tres idiomas de programación están representados, creemos que tenemos una medida que puede analizar la productividad relativa de los diferentes lenguajes y diferentes tecnologías. Los proyectos de DMS/VS, en particular, han seguido la creciente tendencia de la productividad, aunque aún estamos en el inicio de la curva de aprendizaje con DMS.

La figura 6 muestra la relación entre el tamaño y la productividad del proyecto. Se deriva de los mismos proyectos. Esto demuestra que se requieren más horas de trabajo para producir cada punto de función mediante los proyectos se hacen más grandes. Este efecto ha sido conocido durante mucho tiempo. Esta gráfica simplemente cuantifica el resultado de los 22 proyectos que analizamos. Esto demuestra que el tamaño del proyecto debe ser considerado en cualquier comparación de la productividad. Además, las dos curvas muestran

que para toda la gama de tamaños de proyecto, en promedio, PL/1 es probablemente alrededor de 25% más productivo que COBOL. La indicación inicial de DMS/VS es que es probablemente más productivo que PL/1 o COBOL. Si se confirma esta tendencia, DMS/VS podría llegar a ser un 30% más productivo que COBOL.

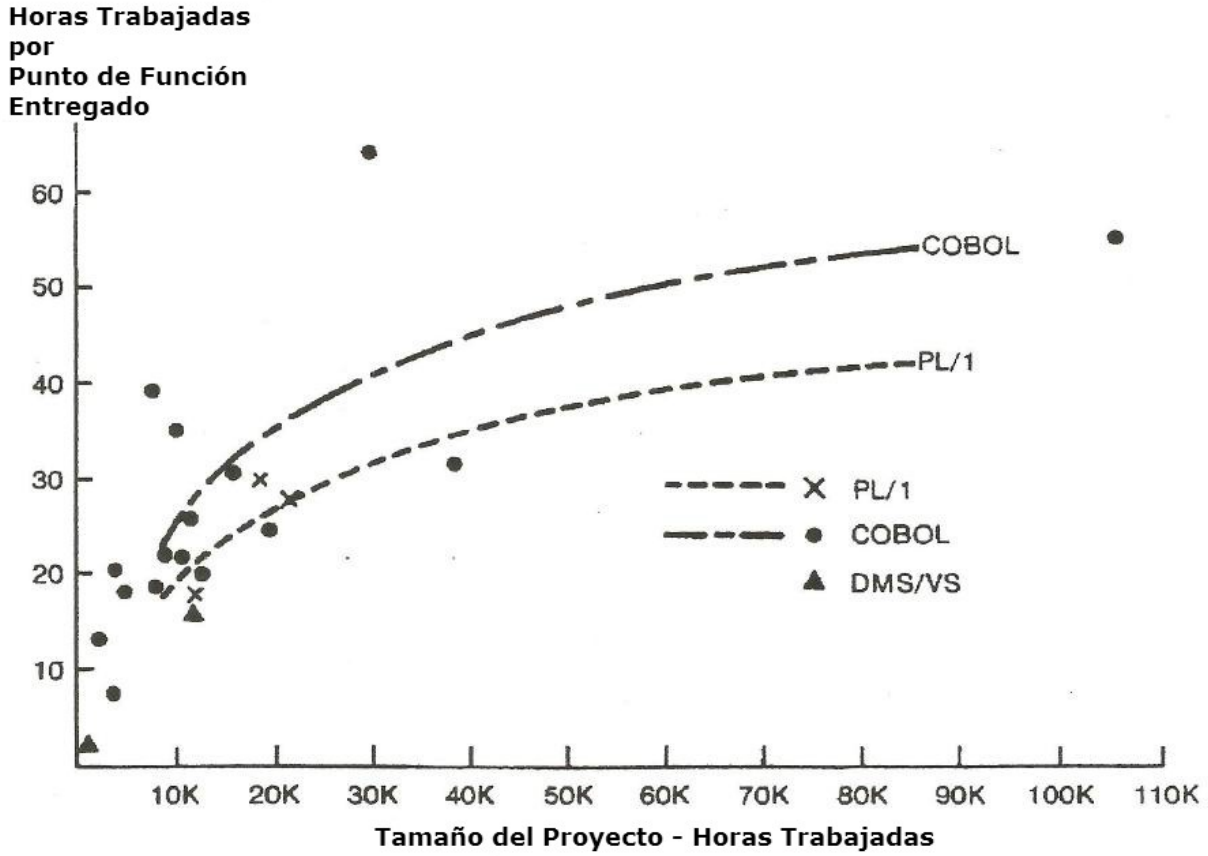

Figura 6

La promesa de ser capaz de analizar la productividad relativa de técnicas tan diferentes como COBOL y DMS/VS es el mayor potencial de este enfoque funcional. Puede proporcionar los medios para probar la eficacia de los lenguajes de alto nivel y enfoques tales como DMS/VS y ADF. La figura 7 utiliza los mismos datos, pero elimina el efecto del tamaño del proyecto en nuestra línea de tendencia de la productividad.

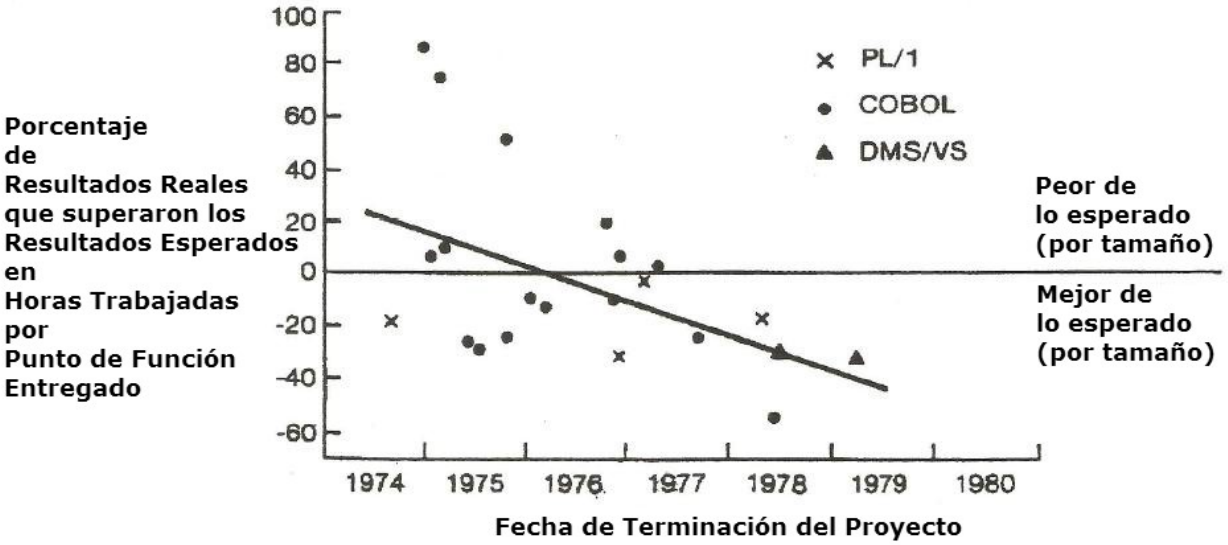

Figura 7

Recuerden que "bajo" todavía significa que un menor número de horas se gastaron para producir cada punto de función. Entre más bajo mejor. Con el efecto del tamaño del proyecto removido, todavía hay aún una tendencia de mejora significativa de 1974 a 1978.

La figura 7 también nos permite dividir nuestros proyectos en dos grupos: Aquellos por debajo de la línea cero que superaron sus expectativas de productividad, y aquellos por encima de la línea que no alcanzaron sus expectativas de productividad. Entonces podemos sacar algunas conclusiones.

- El proyecto se completó después de 1976, el tiempo en que el proceso disciplinado de desarrollo de aplicación, instalado en 1974, estaba empezando a tener sus efectos a través de todos nuestros proyectos.
- Los lenguajes y tecnologías de programación PL/1, DMS/VS, y el desarrollo en línea se utilizaron.
- Las tecnologías de programación mejoradas -codificación estructurada, la implementación de arriba hacia abajo, biblioteca de desarrollo y documentación HIPO - se utilizaron.

Fueron aprobados y ahora se utilizan en todos nuestros contratos (naturalmente tales decisiones como lenguaje de programación y el uso de DMS/VS siguen siendo la prerrogativa de nuestros clientes).

Un último gráfico nos puede dar algunas pistas sobre dónde viene esta productividad. ¿Cuántas líneas de código diseñadas, escritas y probadas se deben producir para ofrecer un punto de valor de función? (figura 8). Con respecto a los proyectos, DMS/VS promedió 25 líneas de código por punto de función entregado, PL/1 promedió 65 y COBOL promedió 110 líneas de código por un punto de valor de la función. Se puede ver el porqué DMS/VS y PL/1 están demostrando ser más productivos que COBOL.

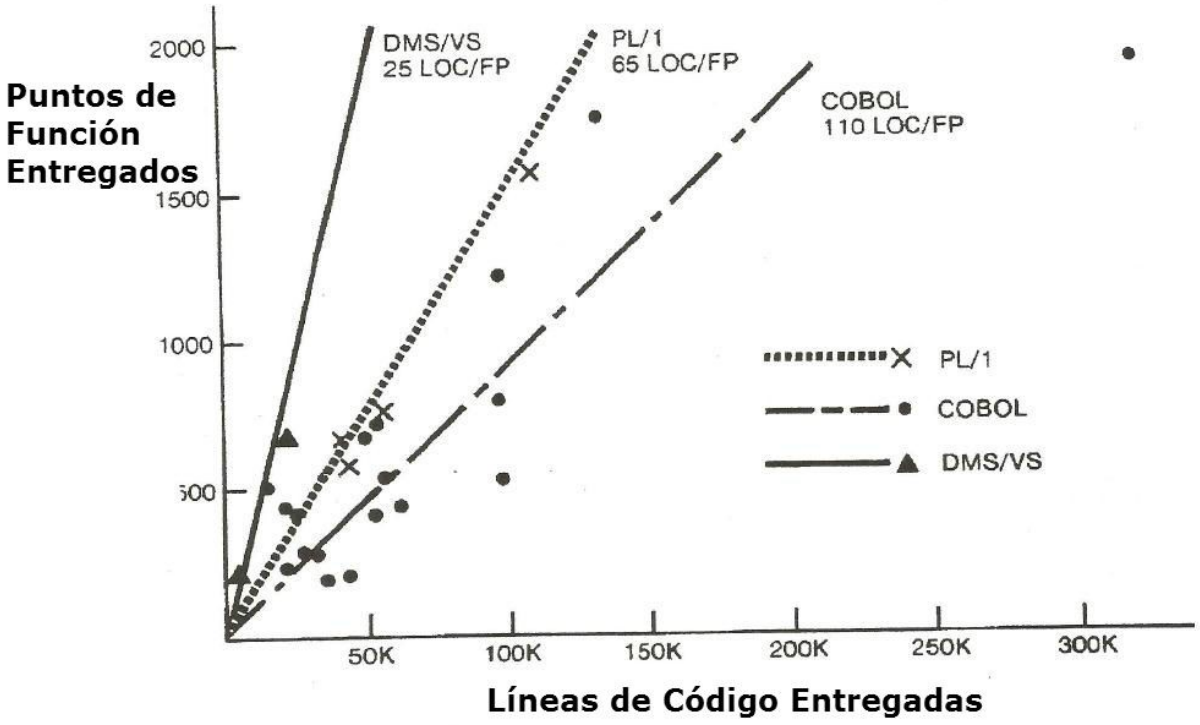

Figura 8

En resumen, medido funcionalmente en los últimos 5 años, la productividad de desarrollo de aplicación de DP Services se ha incrementado cerca de 3 a 1. Este es un aumento de la productividad compuesto de aproximadamente 25% por año.

Hemos utilizado la medida de valor de la función para determinar la productividad relativa de los diferentes lenguajes, tecnologías y tamaños de proyectos. Tenemos la intención de seguir utilizando esta medida funcional para seleccionar y promover tecnologías que puedan ayudar a mantenernos competitivos.

Encontramos que estas tecnologías de programación mejoradas definitivamente contribuyen a una alta productividad. Son una parte importante de cada uno de nuestros contratos.

Hemos encontrado una fuerte evidencia de que el proceso de desarrollo disciplinado de la aplicación promueve significativamente el éxito del proyecto. No sólo dio constantemente proyectos exitosos por las medidas de "a tiempo" (*on time*), dentro del presupuesto, y cliente satisfecho, pero también está fuertemente asociado con una alta productividad. Además, proporciona retroalimentación fiable que necesitamos para hacer un análisis significativo.

La eliminación gradual de proyectos, que es una parte importante del proceso de desarrollo de aplicaciones, ha ayudado a definir nuestros proyectos en trozos más pequeños. A su vez estos pequeños proyectos han contribuido a una mayor productividad.

En cuanto a la medición de la productividad del proyecto, encontramos que teníamos que incluir todo el proceso, incluyendo la fase de diseño, en la medición de la productividad para sacar conclusiones significativas.

Hemos encontrado que un proceso disciplinado es un ingrediente esencial para el significado de las medidas de productividad. La disciplina proporciona definiciones acordadas de producto y el costo consistente a través de los proyectos que se miden.

La medición basada en la funciones ha demostrado ser una forma eficaz para comparar la productividad entre proyectos. Antes de que se estableció, sólo podíamos comparar proyectos que fueran parecidos en lenguaje y tecnología, o tendríamos que afrontar el difícil problema de comparar estimaciones de proyectos hipotéticos contra resultados reales. Tenemos la intención de seguir utilizando e evolucionando la medición por valor de función.

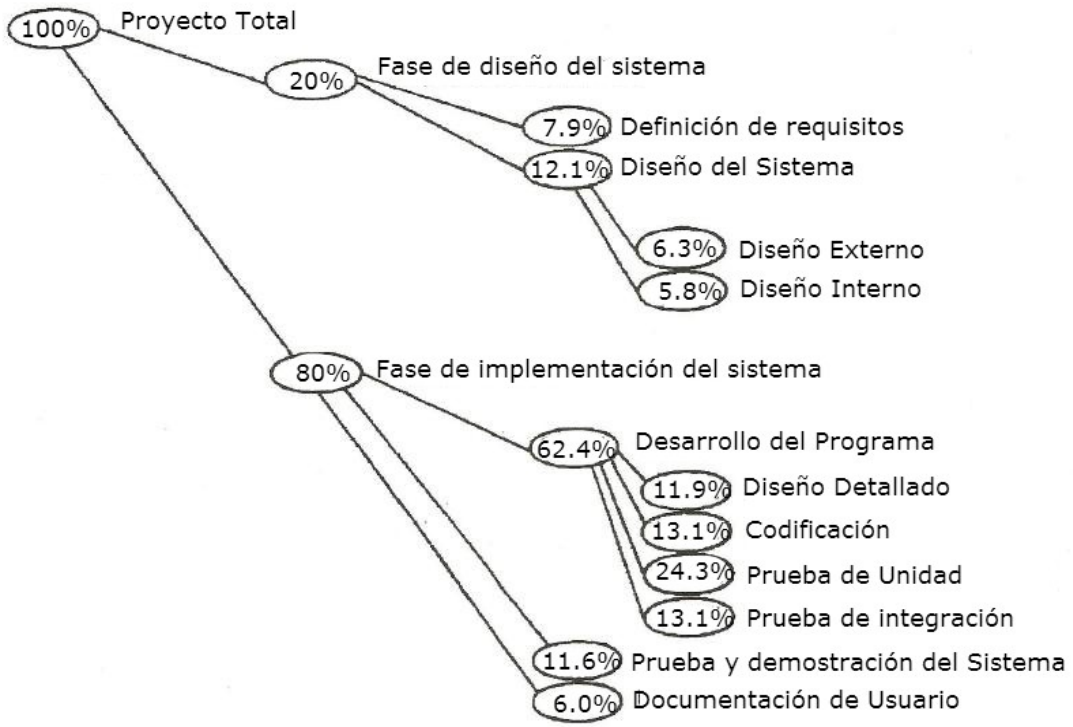

Figura 4

La figura 5 es nuestra tendencia temporal de productividad de 1974 hasta 1978. Hemos trazado las horas trabajadas por cada punto de función entregado. Esto significa que cuanto más bajo mejor, bajo significa que un menor número de horas se gastaron generando

cada punto de función. Hay tres tipos de proyectos representados aquí. Cada punto rojo representa un proyecto de COBOL, cada punto azul representa un proyecto de PL/1 y cada punto blanco representa un proyecto DMS/VS. La línea es un ajuste lineal de los cuadrados mínimos para todos los puntos.

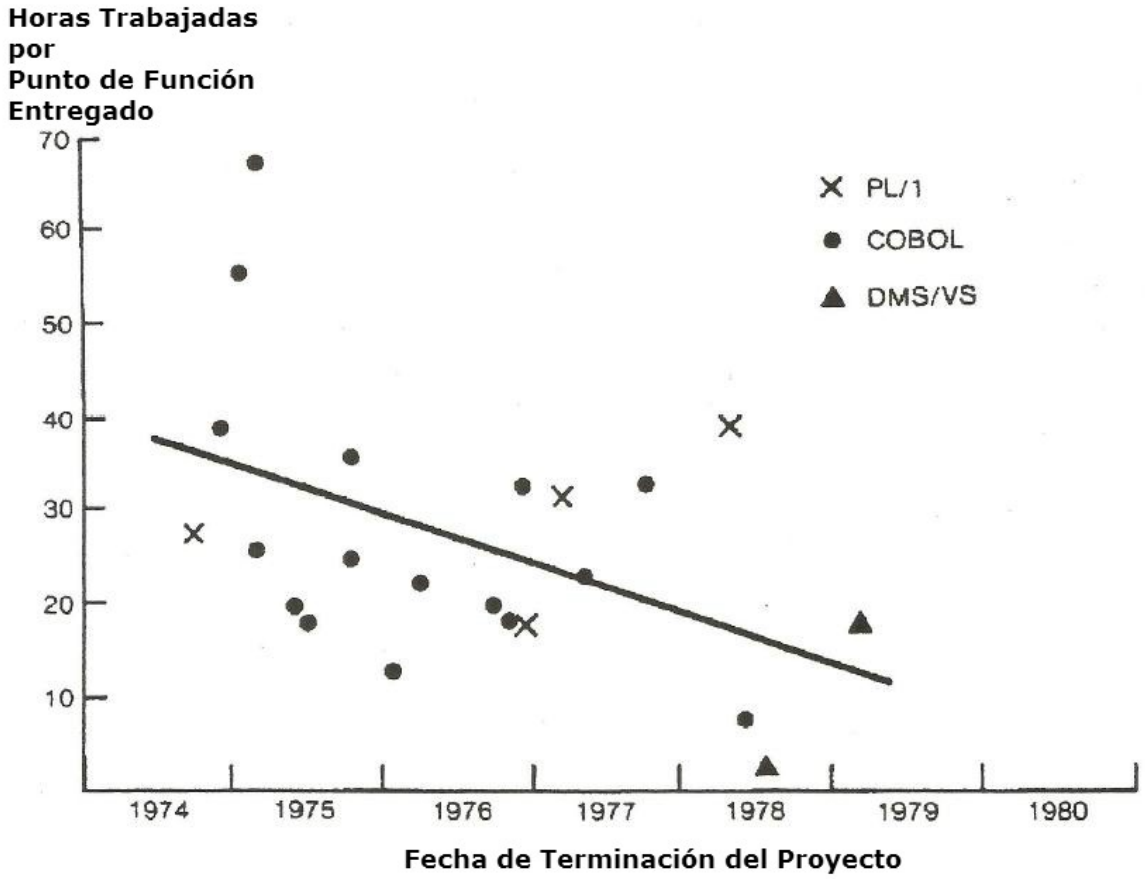

Figura 5

La mejora en la productividad mostrada entre 1974 y 1978 es de aproximadamente 3 a 1. Proyectos de PL/1, DMS/VS y COBOL se combinan para mostrar una tendencia significativa hacia la mejora en la productividad. Debido a que tres idiomas de programación están representados, creemos que tenemos una medida que puede analizar la productividad relativa de los diferentes lenguajes y diferentes tecnologías. Los proyectos de DMS/VS, en particular, han seguido la creciente tendencia de la productividad, aunque aún estamos en el inicio de la curva de aprendizaje con DMS.

La figura 6 muestra la relación entre el tamaño y la productividad del proyecto. Se deriva de los mismos proyectos. Esto demuestra que se requieren más horas de trabajo para producir cada punto de función mediante los proyectos se hacen más grandes. Este efecto ha sido conocido durante mucho tiempo. Esta gráfica simplemente cuantifica el resultado de los 22 proyectos que analizamos. Esto demuestra que el tamaño del proyecto debe ser considerado en cualquier comparación de la productividad. Además, las dos curvas muestran que para toda la gama de tamaños de proyecto, en promedio, PL/1 es probablemente alrededor

de 25% más productivo que COBOL. La indicación inicial de DMS/VS es que es probablemente más productivo que PL/1 o COBOL. Si se confirma esta tendencia, DMS/VS podría llegar a ser un 30% más productivo que COBOL.

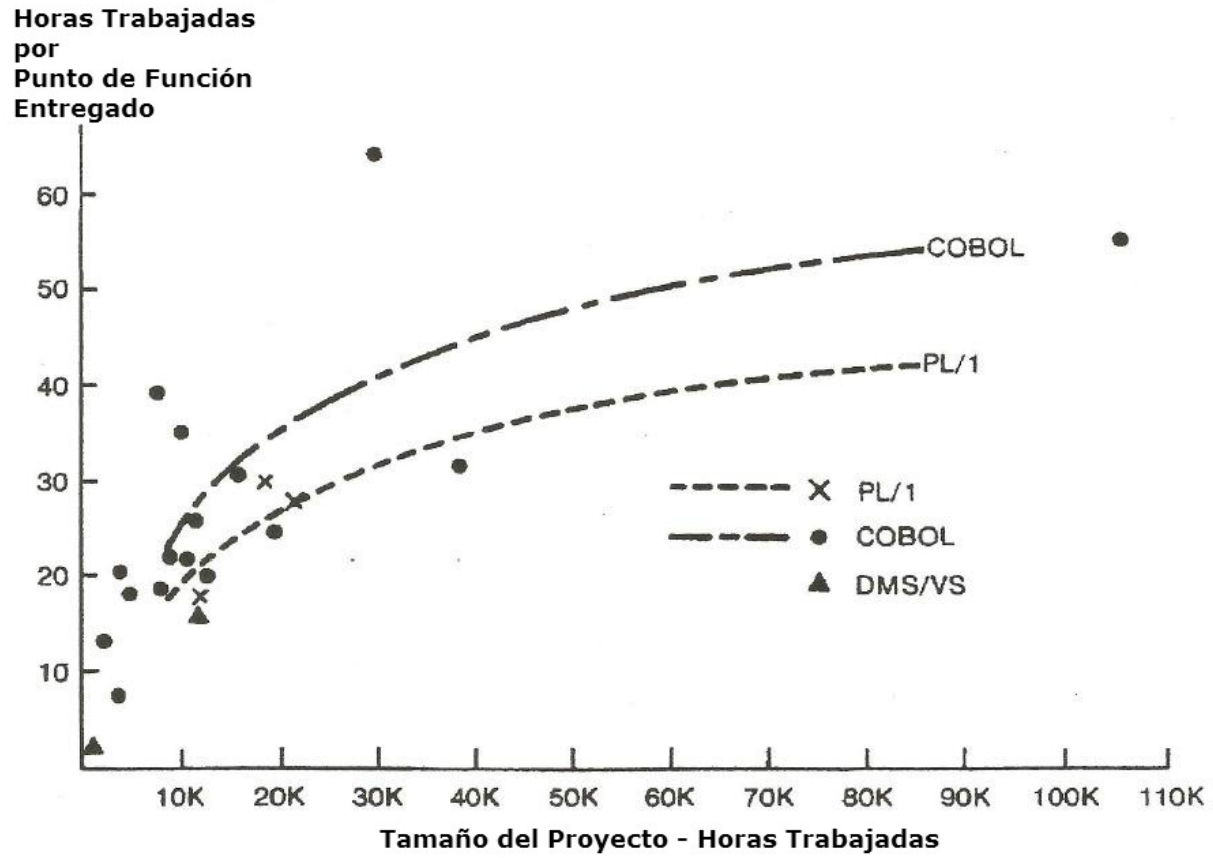

Figura 6

La promesa de ser capaz de analizar la productividad relativa de técnicas tan diferentes como COBOL y DMS/VS es el mayor potencial de este enfoque funcional. Puede proporcionar los medios para probar la eficacia de los lenguajes de alto nivel y enfoques tales como DMS/VS y ADF. La figura 7 utiliza los mismos datos, pero elimina el efecto del tamaño del proyecto en nuestra línea de tendencia de la productividad.

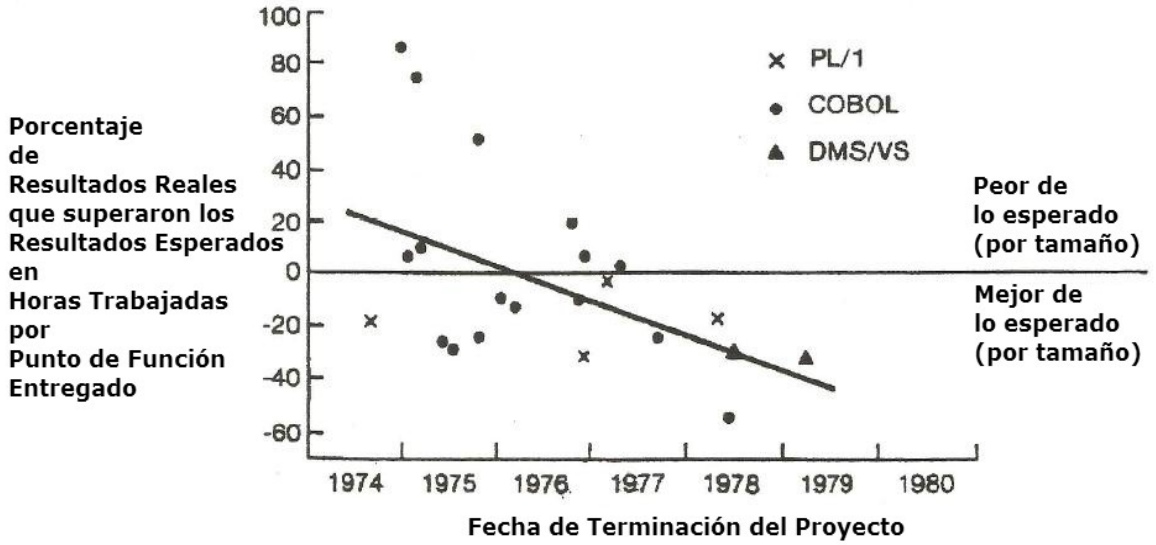

Figura 7

Recuerden que "bajo" todavía significa que un menor número de horas se gastaron para producir cada punto de función. Entre más bajo mejor. Con el efecto del tamaño del proyecto removido, todavía hay aún una tendencia de mejora significativa de 1974 a 1978.

La figura 7 también nos permite dividir nuestros proyectos en dos grupos: Aquellos por debajo de la línea cero que superaron sus expectativas de productividad, y aquellos por encima de la línea que no alcanzaron sus expectativas de productividad. Entonces podemos sacar algunas conclusiones.

- El proyecto se completó después de 1976, el tiempo en que el proceso disciplinado de desarrollo de aplicación, instalado en 1974, estaba empezando a tener sus efectos a través de todos nuestros proyectos.
- Los idiomas y tecnologías de programación PL/1, DMS/VS, y el desarrollo en línea se utilizaron.
- Las tecnologías de programación mejoradas -codificación estructurada, la implementación de arriba hacia abajo, biblioteca de desarrollo y documentación HIPO - se utilizaron.

Fueron aprobados y ahora se utilizan en todos nuestros contratos (naturalmente tales decisiones como lenguaje de programación y el uso de DMS/VS siguen siendo la prerrogativa de nuestros clientes).

Un último gráfico nos puede dar algunas pistas sobre dónde esta productividad viene. ¿Cuántas líneas de código diseñadas, escritas y probadas se deben producir para ofrecer un punto de valor de función? (figura 8). Con respecto a los proyectos, DMS/VS promedió 25 líneas de código por punto de función entregado, PL/1 promedió 65 y COBOL promedió 110 líneas de código por un punto de valor de la función. Se puede ver el porqué DMS/VS y PL/1 están demostrando ser más productivos que COBOL.

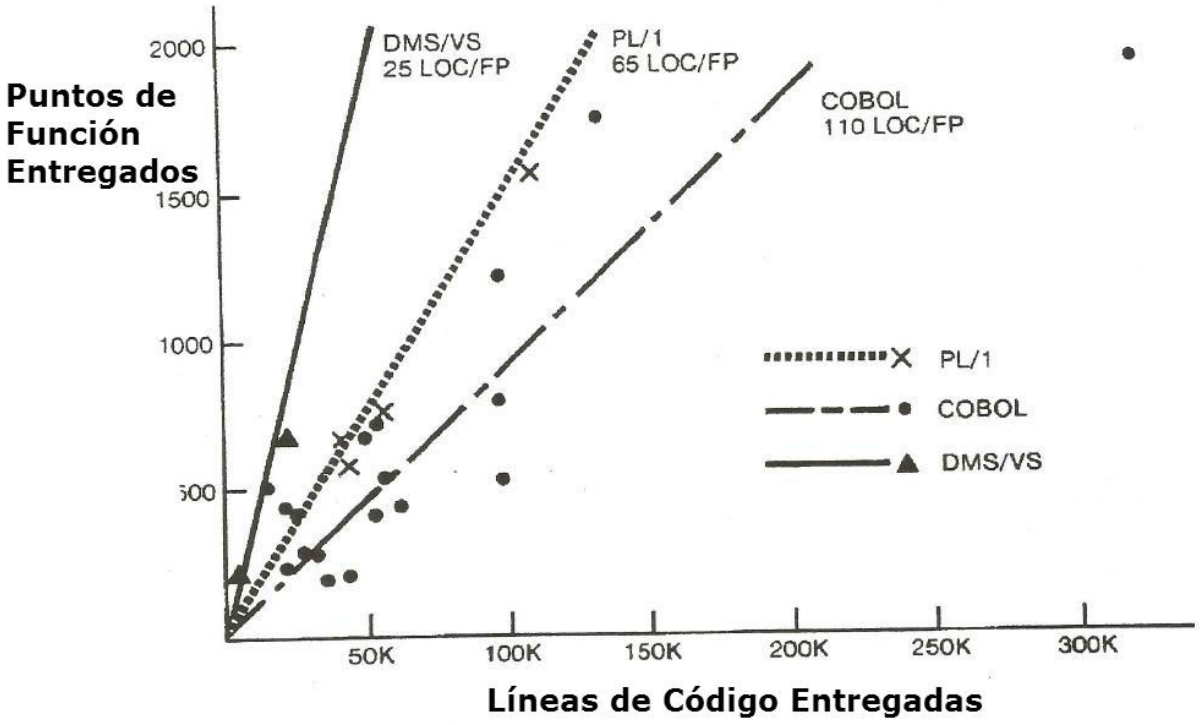

Figura 8

En resumen, medido funcionalmente en los últimos 5 años, la productividad de desarrollo de aplicación de DP Services se ha incrementado cerca de 3 a 1. Este es un aumento de la productividad compuesto de aproximadamente 25% por año.

Hemos utilizado la medida de valor de la función para determinar la productividad relativa de los diferentes lenguajes, tecnologías y tamaños de proyectos. Tenemos la intención de seguir utilizando esta medida funcional para seleccionar y promover tecnologías que puedan ayudar a mantenernos competitivos.

Encontramos que estas tecnologías de programación mejoradas definitivamente contribuyen a una alta productividad. Son una parte importante de cada uno de nuestros contratos.

Hemos encontrado una fuerte evidencia de que el proceso de desarrollo disciplinado de la aplicación promueve significativamente el éxito del proyecto. No sólo dio constantemente proyectos exitosos por las medidas de “a tiempo” (on time), dentro del presupuesto, y cliente satisfecho, pero también está fuertemente asociado con una alta productividad. Además, proporciona retroalimentación fiable que necesitamos para hacer un análisis significativo.

La eliminación gradual de proyectos, que es una parte importante del proceso de desarrollo de aplicaciones, ha ayudado a definir nuestros proyectos en trozos más pequeños. A su vez estos pequeños proyectos han contribuido a una mayor productividad.

En cuanto a la medición de la productividad del proyecto, encontramos que teníamos que incluir todo el proceso, incluyendo la fase de diseño, en la medición de la productividad para sacar conclusiones significativas.

Hemos encontrado que un proceso disciplinado era un ingrediente esencial para la medición de la productividad significativa. La disciplina proporciona definiciones acordadas de producto y el costo consistente a través de los proyectos que se miden.

La medición basada en la función ha demostrado ser una forma eficaz para comparar la productividad entre proyectos. Antes de que se estableció, sólo podíamos comparar proyectos que fueran parecidos en lenguaje y tecnología, o tendríamos que afrontar el difícil problema de comparar estimaciones de proyectos hipotéticos contra resultados reales. Tenemos la intención de seguir utilizando e improvisando la medida de valor de función.