Ministerio de Planificación Federal
Inversión Pública y Servicios
Tribunal de Tasaciones de la Nación

NORMA TTN 24.0 **6 de diciembre de 2007**

TASACION DE SOFTWARE

1. INTRODUCCIÓN:

El software es un sistema creado por un lenguaje de programación que imprime valor agregado y debe considerarse como un bien de uso integrado al bien que lo contiene.
El software es a su vez, es un bien intrínsecamente mueble, por su posibilidad de instalarse de un bien que lo contiene a otro.
Existen dos tipos bien diferenciados de software, el comercial que viene asociado a una licencia de uso, que legalmente no puede ser alterado o modificado por el usuario, y el de aplicación.
El segundo, es creado por una necesidad productiva a requerimiento del usuario, que también suele poseer una licencia de uso y que generalmente puede ser alterado, mejorado o modificado a requerimiento del usuario.
El valor del software comercial se determina por las ofertas del mismo, aplicando el método comparativo y la Norma TTN 10.x.
El valor del software denominado generalmente de aplicación, será determinado según el procedimiento establecido en la presente norma.
Sintéticamente, se estimará el esfuerzo que se ha incurrido para desarrollar el sistema, calculando el costo mensual en salario del técnico más los gastos, afectado por un coeficiente persona - mes.
Los costos que intervienen en un producto software son variados y complejos, además de la tendencia a la disminución del valor del hardware que provoca que el costo más importante sea el de recursos humanos utilizados. Esto se refleja en la cantidad de horas de análisis, diseño, programación y prueba que se deben invertir para obtenerlo.
Los sistemas poseen diferentes características y también diferentes problemas que se describen a continuación:

- El personal que ha trabajado en el proyecto no continúa después en el mismo.
- La documentación si existe es escasa o esta desactualizada.
- No es posible determinar si el equipo que ha trabajado en el proyecto ha seguido una metodología de construcción de software determinada.
- Es muy común que suceda que el código fuente de los sistemas no se

encuentre disponible o este incompleto.
- Las especificaciones funcionales que se han previsto al comienzo, no se condicen con la realidad porque ha habido cambios no documentados o porque no es posible comprobar la recepción efectiva de los sistemas.

Por lo tanto, se deben utilizar técnicas que permitan determinar el valor de productos de software terminados, siendo necesario acceder al sistema en sí y a la documentación del mismo.
Las técnicas son las siguientes: Métricas del Software y la Técnica de Puntos de Función; para poder determinar el tamaño y el esfuerzo que se ha incurrido y el uso de modelos de estimación, en particular COCOMO II.

2. METRICAS DE SOFTWARE

Medir el software es complejo porque es el resultado de una actividad intelectual y comprenden dos conceptos: cuando se habla de un producto, un código, un diseño, una especificación o cuando se hace referencia a una etapa de construcción del software.
Según la definición de métrica de software de la IEEE Standard Glossary of Software Engineering Terms :

Una métrica de software es una medida cuantitativa del grado en el que un sistema, un componente o un proceso poseen un determinado atributo.

En general, las diferentes métricas intentan estimar dos grandes magnitudes generales: el tamaño del producto y la productividad del proceso de construcción del producto.

2.1 METRICAS DEL PRODUCTO Y DEL PROCESO

1) Las métricas del producto, que miden diferentes aspectos del software obtenido, a menudo a partir de código fuente expresado en un lenguaje de programación determinado.
2) Las métricas del proceso, que intentan medir determinados atributos que hacen referencia al entorno de desarrollo del software y tienen en cuenta la manera de construirlo.
Estas se pueden utilizar solas o combinadas, como indicadores que permiten proporcionar una visión resumida del producto de software o del proceso de su construcción.

2.2 METRICAS DE PRODUCTIVIDAD

Las métricas de productividad recogen la eficiencia del proceso de producción y

relacionan el software que se ha construido con el esfuerzo que ha costado elaborarlo.
En el caso de la productividad, existen medidas directas, como las líneas de código, el costo, el esfuerzo, el número de errores, la velocidad de ejecución, la memoria ocupada por un programa, los puntos de función, etc., así como las diferentes maneras de determinar el tamaño del producto obtenido relacionándolo con el tiempo y/o el esfuerzo que ha costado obtenerlo.
En la terminología del estándar de métricas de productividad de software del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE Standard for Software Productivity Metrics), se habla de Primitiva para hacer referencia al nivel más bajo en el que se recogen los datos. Existen primitivas de dos tipos:

- De salida, que recogen las características finales del software.
- De entrada, que miden lo que ha sido necesario utilizar para construir el software.

Una relación de productividad es la que se suele establecer para tener en cuenta la productividad entre las primitivas de salida, es decir, las unidades que miden la salida del proceso de construcción de software y las primitivas de entrada, que miden las entradas en el proceso de construcción de software.

2.3 EL ESFUERZO Y LA MEDIDA DE PRODUCTIVIDAD

La medida habitual del costo es la cantidad de dinero que cuesta producir un software determinado; mientras que las medidas habituales del esfuerzo se reducen al trabajo que lleva a cabo un profesional en una determinada unidad de tiempo: un día (persona-día), un mes (persona-mes) o un año (persona-año).

2.4 METRICAS ORIENTADAS AL TAMAÑO Y A LA FUNCION.

El tamaño del software, construido a partir de las líneas de código es la medida tradicional para evaluar el volumen de trabajo.
Además existen otras unidades de medida que tienen en cuenta la dificultad intrínseca de construir un software, como ser las funcionalidades que éste incorpora.
La medida de líneas de código y/o de puntos de función es la más habitual para determinar el volumen de un producto software, son las principales primitivas de salida que se utilizan para calcular relaciones de productividad.

2.5 ANALISIS Y DETERMINACION DE PUNTOS DE FUNCION

Los Puntos Función son la medida para determinar el tamaño y el esfuerzo involucrado en el proceso de desarrollo del software.

El Análisis de Puntos Función es una técnica estructurada para la clasificación de las componentes de un Sistema, un método para descomponer los Sistemas en componentes más pequeños que puedan ser entendidos y analizados con facilidad.

En el método del Análisis de Puntos Función se cuantifican los datos que usa la aplicación y las transacciones de entrada-salida como base para estimar el esfuerzo de construcción de esa aplicación.

El recuento de los puntos de función se elabora a partir de determinadas características funcionales, que pueden ser de datos o de transacción:

1) Las características funcionales de datos:

a) Archivos lógicos internos (LIF): grupo de datos interrelacionados lógicamente y que pueden ser identificados claramente y siempre se mantienen dentro de los límites de la aplicación, son internos y propios de la aplicación. Son los Archivos de la aplicación.

b) Archivos de interfaces externas (EIF): grupo de datos interrelacionados lógicamente o a veces información de control, que proceden de fuera de los límites de la aplicación y se mantienen al margen de ésta. Son los Archivos o las tablas que la aplicación utiliza pero que no crea ni mantiene**.**

2) Las características funcionales de transacción son las siguientes:

a) Entradas externas (EI): hacen referencia a los tratamientos que procesan datos o información de control introducidos en la aplicación desde fuera de sus límites. Son las entradas a la aplicación.

b) Salidas externas (EO): cualquier proceso elemental que genere datos o información de control que salga de los límites de la aplicación. Equivale a las salidas que genera la aplicación.

c) Consultas externas (EQ): proceso elemental formado por una combinación de entrada y salida que permite la recuperación de datos. La salida no debe contener datos derivados y se pueden utilizar los Archivos lógicos internos.

Son las consultas internas de la aplicación, que a menudo se resuelven con consultas en los Archivos o tablas propias.

En la tabla siguiente se muestra la ponderación de cada característica funcional y la disposición de los cálculos para obtener lo que se denomina total de puntos de función sin ajustar (TUFP):

Puntos de Función sin ajustar (TUFP)					
Tipo	Descripción	Complejidad			Total
		Simple	Media	Compleja	
EI	Entradas Externas	,,,,x3	,,,,x4	,,,,x6	……
EO	Salidas Externas	,,,,x4	,,,,x5	,,,,x7	……
LIF	Ficheros lógicos internos	,,,,x7	,,,,x10	,,,,x15	……
EIF	Interfaces lógicas externas	,,,,x5	,,,,x7	,,,,x10	……
EQ	Consultas externas	,,,,x3	,,,,x4	,,,,x6	……
Total de puntos de función sin ajustar (TUFP)					……

TABLA I – Tabla de total de función sin ajustar (TUFP)

2.6 RELACION ENTRE PUNTOS DE FUNCION Y LINEAS DE CODIGO

LOC por punto de función			
Lenguaje	LOC/FP	Lenguaje	LOC/FP
Ensamblador	320	Basic ANSI/Quick/Turbo	64
Macroensamblador	213	Java	53
C	150	Visual C++	34
Fortran	106	Foxpro 2,5	34
Cobol	106	Visual Basic	32
Pascal	91	Delphi	29
Cobol ANSI 85	91	C++	29
Basic	91	Visual Cobol	20
RPG	80	Clipper	19
PL/I	80	Power Builder	16
Ada	71	Hoja de Calculo	6

TABLA II – LOC por punto de función.

3. MODELOS DE ESTIMACION

Los modelos de estimación proporcionan sistemas y métodos generales para proceder a realizar la estimación de costos en la construcción de software.

Los diferentes modelos de estimación de costos y/o esfuerzos en la construcción de software se pueden dividir en cuatro grupos:

1) Los modelos de base histórica son los más antiguos y primitivos. A menudo se basan en la analogía con otros proyectos parecidos y se fundamentan casi exclusivamente en la experiencia profesional de los que efectúan la estimación.

2) Los modelos de base estadística superan la experiencia histórica de los profesionales que intervinieron en el proyecto y a partir del estudio estadístico de los datos reales obtienen fórmulas que relacionan las diferentes unidades de medida del software, a menudo las líneas de código (LOC) y el esfuerzo (generalmente medido en hombre-mes).

3) Los modelos de base teórica parten de una serie de ideas generales sobre el proceso de construcción de software y elaboran fórmulas que relacionan diferentes métricas de software.

4) Los modelos compuestos consideran los dos sistemas anteriores: estadísticos y teóricos.

3.1 MODELOS COMPUESTOS: EL MODELO Cocomo II

El modelo COCOMO II incluye tres modelos que corresponden a diferentes fases y modalidades del ciclo de vida:

1) Modelo de composición de aplicaciones: incluye el uso de prototipos para disminuir los riesgos potenciales que surgen con las interfaces gráficas de usuario típicas de herramientas RAD y otras herramientas actuales de productividad y de la orientación a objetos. En este modelo se definen unos puntos objeto que vendrían a ser una adaptación y modernización de los puntos de función de Albrecht.

2) Modelo de diseño primerizo: intenta obtener una primera aproximación en las fases iniciales del ciclo de vida, cuando todavía se conocen pocas de las características y datos definitivos del proyecto. Utiliza como primitivas de salida tanto las líneas de código como los clásicos puntos de función de Albrecht sin ajustar (TUFP).

3) Modelo de post-arquitectura: versión más completa, corresponde a la modernización del COCOMO tradicional de 1981. Se aplica cuando se considera que el proyecto dispone ya de requerimientos estables y para productos software terminados. También utiliza como primitivas de salida las líneas de código y los puntos de función de Albrecht sin ajustar (TUFP) y tiene en cuenta indicadores de la reutilización de software, cinco factores de escala y hasta diecisiete factores específicos diferentes.

4. VIDA UTIL Y DEPRECIACION:

La vida útil de un software debe considerarse según dos aspectos:

- El software comercial, tiene una vida útil general de 3 a 5 años, debido principalmente a cambios tecnológicos, comerciales o de necesidades del mercado.
- El software de aplicación, tratado en esta norma, tiene una vida útil estimada de 10 a 20 años, principalmente por sus costos de desarrollo, que haría antieconómico productos con menor tiempo de uso. A su vez, un período mayor no sería conveniente considerarlo ya que los avances tecnológicos harían obsoleta la aplicación.
- En ambos casos la depreciación es lineal.

RECONOCIMIENTO:

La presente Norma fue desarrollada por la Dirección de Informática del Organismo, a los cuales el Tribunal de Tasaciones de la Nación les brinda su debido reconocimiento.

Ministerio de Planificación Federal
Inversión Pública y Servicios
Tribunal de Tasaciones de la Nación

ANEXO A

Aplicación de la Técnica de Puntos de Función

1. DESARROLLO DE LA TÉCNICA DE PUNTOS DE FUNCIÓN

Los puntos de función miden el software cualificando la funcionalidad que proporciona externamente, basándose en el diseño lógico del sistema. Por lo tanto, en el caso de subsistemas diseñados independientemente los puntos de función se calcularán para cada uno de ellos y luego se sumarán.
Los objetivos de los puntos de función son:

- Medir independientemente de la tecnología utilizada en la implantación del sistema.
- Proporcionar una métrica de tamaño que dé soporte al análisis de la calidad y a productividad.
- Proporcionar un medio para la estimación del software.

El análisis de los puntos de función se desarrolla considerando cinco parámetros básicos externos del sistema:

1. Entrada (EI, del inglés *External Input).*
2. Salida (E0, del inglés *External Output).*
3. Consultas (EQ, del inglés *External Query).*
4. Grupos de datos lógicos internos (ILF, del inglés *Internal Logic File).*
5. Grupos de datos lógicos externos (EIF, del inglés *External Interface File).*

Con estos parámetros, se determinan los puntos de función sin ajustar.
La aplicación de la técnica de puntos de función comprende los siguientes pasos:

- Definición de parámetros.
- Valoración de la complejidad.

2. DEFINICIÓN DE PARÁMETROS

Para poder determinar la existencia de los componentes que contribuirán al total final hay que definirlos previamente.
Estos componentes pueden ser clasificados como tipos de funciones y son de dos clases: datos o transacciones.

2.1. TIPOS DE FUNCIÓN DATOS

Representan la funcionalidad proporcionada a los usuarios para cumplir con sus requisitos de datos internos y externos.
Son de dos tipos: Archivos lógicos internos y Archivos de interfase externos.

2.1.1. Archivos lógicos internos

Un Archivo lógico interno es un grupo de datos lógicamente relacionados, identificables por los usuarios o información de control, mantenidos y utilizados dentro de los límites de la aplicación.
Ejemplos de ILF son:

- Archivos maestros.
- Aplicaciones de seguridad de datos.
- Datos de auditoría.
- Mensajes *help.*
- Mensajes de error.
- Archivos internos lógicos mantenidos por más de una aplicación.

2.1.2. Archivos interfase externos

Representan un grupo de datos relacionados lógicamente identificables por el usuario o información de control utilizada por la aplicación, pero mantenida por otra aplicación.

2.2. TIPOS DE FUNCION TRANSACCION

Comprende tres tipos de función:

- Entradas externas (EI): mantiene datos almacenados internamente.
- Salidas externas (E0): datos de salida de la aplicación.
- Consultas externas (EQ): combinación de una entrada y de una salida.

2.2.1. Entradas externas

Las entradas externas son datos o información de control que sé introducen en la aplicación desde fuera de sus límites.
Estos datos mantienen un Archivo lógico interno. La información de control está constituida por datos utilizados por un proceso dentro de los límites de la aplicación y puede mantener directamente un Archivo lógico interno.

Una entrada externa debería ser considerada única si tiene un formato distinto de las demás o el diseño lógico requiere una lógica de procesamiento distinta de otra entrada externa del mismo formato. Una entrada externa se considera única si los datos en un Archivo lógico interno (ILF) y el formato de entrada es único o la lógica del proceso es única.

Estos tipos de función son:

• Las transacciones: datos introducidos para mantener Archivos lógicos internos.

• Las pantallas de entrada: hay que añadir una unidad a entradas externas por cada función que mantiene un Archivo lógico interno. Por ejemplo, si los datos introducidos en esa pantalla pueden añadir, cambiar y borrar información en un Archivo lógico interno, se contarían tres entradas externas.

• Las entradas por lotes: por cada proceso único que mantiene un Archivo interno lógico se debe añadir una entrada externa por cada adición, modificación o borrado de datos.

• Un Archivo físico de entrada, cuando se analiza lógicamente, corresponde a una entrada externa o varias, dependiendo de los tipos de registros contenidos y del proceso requerido para su tratamiento.

• Asimismo dos o más Archivos físicos de entrada pueden corresponder a una entrada externa si el proceso lógico y el formato son idénticos para cada uno de los Archivos.

• Las entradas externas duplicadas: si distintos procesos de entrada solicitados expresamente por el usuario duplican una entrada externa, son contados independientemente cada uno.

2.2.2. Salidas externas

Las salidas externas son datos o información de control que sale de los límites de la aplicación. Esta salida debe ser considerada única si tiene un formato único o si el diseño lógico requiere un proceso distinto de otras salidas del mismo formato.
Son salidas externas:

• La transferencia de datos a otras aplicaciones: datos que residen en un Archivo lógico interno que son formateados y procesados para ser utilizados por otra aplicación.

• Las salidas son identificadas según los procesos requeridos para el tratamiento de los datos. Un Archivo físico de salida, cuando se analiza lógicamente, puede corresponder a varias salidas. De una manera similar, dos o más Archivos físicos de salida pueden corresponder a una salida externa si el proceso lógico y los formatos son idénticos para cada uno de ellos. Un método para

identificar múltiples salidas externas es ver cuántos tipos de registros distintos hay en el Archivo.

• Los informes: cada informe producido por la aplicación se cuenta como una salida externa.

• Dos informes que tengan el mismo formato pero los datos de origen sean distintos, se contarán como dos salidas.

• Dos informes idénticos, producidos en diferentes soportes debido a requisitos específicos de los usuarios, se cuentan como salidas distintas.

• Los informes *on-line* que no corresponden a la salida de una consulta, se contarán también como una salida.

• Los mensajes de error/configuración: no se contarán si están asociados a una consulta.

• Los gráficos**:** cada gráfico distinto, solicitado por el usuario, debería ser contado como una salida. Así, si unos datos estadísticos se presentan en formato de tabla, diagrama de barras o torta, se contarán como tres salidas.

• Los generadores de informes: una salida desarrollada por el usuario con un generador de informes debería ser contada como una salida para cada tipo de informe especificado.

Si el usuario solicita una facilidad de generación de informes como parte de la aplicación para ser confeccionados por él mismo, la cuenta será la siguiente:

• Debería contarse una entrada externa por cada parámetro para la definición de informes o comando (ej. *select, compare, sort merge, calculase, format,* etcétera) utilizado por el usuario para controlar la generación del informe.

• Debería contarse una salida por el informe total.

• Debería contarse un Archivo lógico si se crea un nuevo Archivo y éste se graba.

No se deben contar como salidas:

• Las ayudas.

• Las distintas formas de invocar la misma salida lógica.

• Los mensajes de error/confirmación asociados con tipos de función distintos de entradas externas.

Por ejemplo, no se contabilizarán como salida los mensajes de error / confirmación asociados a una consulta externa.

• Los informes múltiples/valores únicos de datos: informes idénticos con el mismo formato y la misma lógica de proceso, pero que existen debido a distintos valores de datos, no se cuentan como salidas distintas. Dos informes idénticos en formato y construcción, el primero de los cuales contiene nombres comenzando desde la A a la L y el segundo desde la M a la Z se cuenta como una única salida.

• Las totalizaciones: los informes de totales no constituyen una salida.

• Los informes *ad hoc:* cuando el usuario dirige y es responsable de la creación mediante la utilización de lenguajes como SQL de un número indefinido de informes, no se cuentan como salidas.

2.2.3. Consultas externas

Las consultas representan los requisitos de información a la aplicación en una combinación única de entrada/salida que se obtiene de una búsqueda de datos, no actualiza un Archivo lógico interno y no contiene datos derivados. Una consulta se considera única si tiene un formato distinto de otras consultas, ya sea en entrada o salida, o si el diseño lógico requiere ediciones distintas a las de otras consultas.
Se entiende por datos derivados aquellos que requieren un proceso distinto a la búsqueda directa, edición o clasificación de información de Archivos lógicos internos o Archivos interfases externos.

Consultas son:

• La búsqueda inmediata de datos.
•Las consultas no explícitas: las pantallas de modificación/borrado que proporcionan capacidad de búsqueda de datos antes de la funcionalidad de cambio/borrado se consideran como consultas, si la entrada y salida de la consulta son idénticas en las funciones de modificación y borrado, se contará una sola consulta.
• Los menús con consultas implícitas: las pantallas de menú que proporcionan una selección de pantallas y entradas para la búsqueda de datos para la pantalla llamada, se cuenta como una consulta.
• Pantallas de conexión: las pantallas de logon que proporcionarían seguridad se cuentan como una consulta.
• Las ayudas: son una consulta donde la entrada y la salida (texto) son únicas. Un texto que puede ser accedido o mostrado en pantalla mediante distintas peticiones o diferentes áreas de una aplicación, se cuenta como una consulta.
• Las salidas duplicadas: consultas iguales que producen una salida en diferentes soportes, como consecuencia de especificaciones del usuario, se cuentan como consultas distintas.
• Las salidas gráficas: cada salida gráfica diferente solicitada por el usuario, debería contarse como una consulta.
• Tutoriales: los sistemas de software relativos a la formación de usuarios deberían contarse como un sistema distinto.

No se consideran como consultas:

• Los mensajes de error/confirmación.
• La utilización de distintos métodos de llamada a la misma consulta.

3. VALORACIÓN DE LA COMPLEJIDAD

Para cada uno de los parámetros externos se ha de indicar su complejidad como baja, media o alta. Para las entradas, salidas y consultas, se puede evaluar su complejidad en función del número de campos que contengan y del número de Archivos a los que hagan referencia. Para los Archivos, por el contrario, su complejidad vendrá dada en función del número de registros y de campos que tengan.

Parámetro	Complejidad	X	Peso	Total
Entrada	Alta	x	6	=
	Media	x	4	=
	Baja	x	3	=
Salida	Alta	x	7	=
	Media	x	5	=
	Baja	x	4	=
Archivo lógico Interno	Alta	x	15	=
	Media	x	10	=
	Baja	x	7	=
Archivo lógico Externo	Alta	x	10	=
	Media	x	7	=
	Baja	x	5	=
Consultas	Alta	x	6	=
	Media	x	4	=
	Baja	x	3	=
Total				

TABLA I – La suma total da los puntos de función sin ajustar del sistema

Ministerio de Planificación Federal
Inversión Pública y Servicios
Tribunal de Tasaciones de la Nación

Anexo B

Aplicación de la Técnica - COCOMO II- Modelo de Post-Arquitectura

La fórmula básica es : $\mathbf{PM_{nominal} = A * (Size)^B}$

1. COMPONENTES DE LOS MODELOS

Esta ecuación calcula el esfuerzo nominal para un proyecto expresado en Meses-persona (**PM**).

El factor de escala **B** explica el ahorro o gasto relativo de escala.

La constante **A** se le ha asignado un valor de **2,45**.

Variable **Size :** PFSA * SLOC/UFP (devuelve SLOC)

El tamaño de una aplicación se mide en unidades de líneas de código fuente (KSLOC). En Cocomo II puede estimarse también a partir de Puntos de Función sin ajustar convirtiendo a SLOC y luego dividiendo por 1000.

Si se opta por utilizar directamente el valor del número de líneas de código, el objetivo es medir la cantidad de trabajo intelectual que se emplea en el desarrollo del programa.

Si se opta por utilizar los Puntos de Función sin Ajustar (PFSA) para determinar el tamaño del proyecto, éstos deben convertirse en líneas de código fuente en el lenguaje de programación utilizado usando la tabla siguiente:

LOC por punto de función			
Lenguaje	**LOC/FP**	**Lenguaje**	**LOC/FP**
Ensamblador	320	Basic ANSI/Quick/Turbo	64
Macroensamblador	213	Java	53
C	150	Visual C++	34
Fortran	106	Foxpro 2,5	34
Cobol	106	Visual Basic	32
Pascal	91	Delphi	29
Cobol ANSI 85	91	C++	29
Basic	91	Visual Cobol	20
RPG	80	Clipper	19
PL/I	80	Power Builder	16
Ada	71	Hoja de Calculo	6

Sólo quedaría por hacer la conversión a KSLOC mediante la operación SLOC/1000.

Variable B (Ahorro y Gasto Software de escala)

$$B = 0{,}91 + 0{,}01 * \sum_{J=1}^{5} FE_J$$

El exponente **B** se obtiene mediante los denominados drivers de escala. Cada driver de escala tiene un rango de niveles de valores desde Muy Bajo hasta Extra Alto. Cada nivel de valores tiene un peso FE cuyo valor específico se llama factor de escala. Un factor de escala de un proyecto, FE_J se calcula sumando todos los factores y se usa para determinar el exponente B de escala.

Su valor, obtenido en la ecuación anterior se interpreta según la siguiente tabla:

Factores de Escala (FE_i)	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
PREC	Completamente sin precedentes	Prácticamente sin precedentes	Casi sin precedentes	Algo familiar	Muy familiar	Completa-mente familiar
FLEX	Riguroso	Relajación ocasional	Algo de relajación	Conformidad general	Algo de conformidad	Metas generales
RESL	Poco (20%)	Algo (40%)	A menudo (60%)	Generalmente (75%)	En su mayor parte (90%)	Por completo (100%)
TIEAM	Interacciones muy difíciles	Algo de dificultad en interacciones	Interacciones básicamente cooperativas	Bastante cooperativo	Altamente cooperativo	Completas interacciones
PMAT	Peso medio de respuestas "Sí" para el cuestionario de Madurez CMM					

Factores de Escala (W_i)	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
PREC	6,20	4,96	3,72	2,48	1,24	0,00
FLEX	5,07	4,05	3,04	2,03	1,01	0,00
RESL	7,07	5,65	4,24	2,83	1,41	0,00
TEAM	5,48	4,38	3,29	2,19	1,10	0,00
PMAT	7,80	6,24	4,68	3,12	1,56	0,00

(PREC) Precedencia

Variable de precedencia u orden secuencial.

Características en Precedencia	Muy Bajo	Nominal/Alto	Extra Alto
Comprensión organizacional	General	Considerable	Profundo
Experiencia en trabajo con sistemas Sw relacionados	Moderado	Considerable	Extenso
Desarrollo concurrente de nuevo Hw asociado y procedimientos operacionales	Extenso	Moderado	Algo
Necesidad de arquitecturas de proceso de datos innovadoras, algoritmos	Considerable	Algo	Mínimo

(FLEX) Flexibilidad de desarrollo.

Variable de flexibilidad de desarrollo.

Características de Flexibilidad de Desarrollo	Muy Bajo	Nominal/Alto	Extra Alto
Necesidad de conformidad del Sw con requisitos preestablecidos	Completo	Considerable	Básico
Necesidad de conformidad del Sw con especificaciones de interfaz externas	Completo	Considerable	Básico
Prioridad en finalización anticipada	Alto	Medio	Bajo

(RESL) Arquitectura/Resolución de Riesgos

Este factor combina dos factores de medida: "Minuciosidad del diseño por revisión del diseño del producto (PDR)" y "Eliminación de riesgos por PDR".

Características	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
El plan de gestión de riesgos identifica todos los ítems de riesgos críticos, establece hitos para resolverlos mediante PDR	Ninguno	Poco	Algo	Generalmente	A menudo	Comple-tamente
Horario, presupuesto, hitos internos con PDR compatible con el Plan de gestión de riesgos	Ninguno	Poco	Algo	General-mente	A menudo	Comple-tamente
Tanto por ciento de horario desarrollado dedicado a establecer la arquitectura dados los objetivos generales del producto	5	10	17	25	33	40
Porcentaje de arquitectos Sw de alto nivel requeridos, disponibles para el proyecto	20	40	60	80	100	120

para el proyecto						
Herramientas de soporte disponibles para resolver ítems de riesgo, desarrollar y verificar garantías de la arquitectura	Ninguno	Poco	Algo	Bueno	Fuerte	Completo
Nivel de incertidumbre en drivers de arquitectura. Claves: misión interfaz de usuario, COTS, Hw, tecnología, ejecución	Extremo	Significativo	Conside-rable	Algo	Poco	Muy Poco
Número y criticidad de ítems de riesgo	>10, critico	5-10, critico	2-4, critico	1, critico	>5, no critico	<5, no critico

(TEAM). Cohesión del Equipo

Explica la turbulencia o entropía del proyecto debido a dificultades en la sincronización de los implicados en el proyecto, usuarios, clientes, desarrolladores, los que lo mantienen.

Características	Muy Bajo	Bajo	Nominal	Alto	Muy Alto	Extra Alto
Consistencia de objetivos y culturas	Poco	Algo	Básico	Considerable	Fuerte	Completo
Habilidad y servicialidad para acomodar otros objetivos	Poco	Algo	Básico	Considerable	Fuerte	Completo
Experiencia en operar como un equipo	Nada	Poco	Poco	Básico	Considerable	Extensa
Lograr visión compartida y compromisos	Nada	Poco	Poco	Básico	Considerable	Extensa

(PMAT) Madurez del proceso

El procedimiento para determinar PMAT se obtiene a través del Modelo de Madurez de Capacidad del Instituto de Ingeniería del Software (CMM).

Nivel de Madurez Global

Nivel 1 CMM (Mitad inferior)

Nivel 1 CMM (Mitad superior)

Nivel 2 CMM

Nivel 3 CMM

Nivel 4 CMM

Nivel 5 CMM

Áreas de Proceso Principales

La segunda está organizada en base a 18 áreas de proceso principales (KPA's) en el modelo de Madurez de Capacidad SET. El procedimiento para determinar PMAT es decidir el porcentaje de conformidad para cada uno de los KPA's. Se usan los niveles de conformidad para las metas de los KPA's para poner el nivel de conformidad.

El nivel basado en meta de conformidad se determina haciendo una media basada en juicio de las metas de cada área de proceso principal y llenando en la tabla siguiente para las 18 áreas de proceso principales el porcentaje de conformidad:

Áreas de Proceso

- Gestión de requisitos
- Planificación de proyectos Software
- Seguimiento de proyectos Software
- Gestión de subcontrato Software
- Aseguramiento de la calidad Software
- Gestión de la configuración Software
- Focos de proceso de organización
- Definición de proceso de organización
- Programa de formación
- Gestión del Software integrado
- Ingeniería de producto Software
- Coordinación Intergrupos
- Informes detallados
- Gestión de proceso cuantitativo
- Gestión de calidad Software
- Prevención de defectos
- Gestión de cambio de tecnología
- Gestión de cambio de proceso

Se establece un tratamiento para cada uno de los 7 niveles considerados: 1. Casi siempre (>90%); 2. Frecuentemente (60-90%); 3. Intermedio (40-60%); 4. Ocasional (10-40%); 5. Pocas veces (<10%); 6. No se Aplica; 7. No se Conoce).

Verificar casi siempre, cuando las metas se logran de forma consistente y se establecen procedimientos que operan bajo la norma.

Verificar frecuentemente cuando las metas se logran relativamente a menudo pero alguna vez se pasan por alto bajo circunstancias.

Verificar intermedio cuando las metas se logran alrededor de la mitad del tiempo.

Verificar ocasionalmente cuando las metas se logran a veces.

No verificar casi nunca, si las metas no se alcanzan casi nunca.

No aplicar verificación, cuando se tiene conocimiento requerido para el proyecto u organización y el KPA, pero siente que el KPA no se adapta a tus circunstancias.

Después de que el nivel de conformidad del KPA se determina, se pesa cada nivel de conformidad y calcula un factor PMAT. Inicialmente todos los KPA's tendrán el mismo peso.

$$\mathbf{PMAT = 5 - \sum_{}^{18}[(KPA\% / 100) * (5 / 18)]}$$

2. AJUSTE POR DRIVERS DE COSTO

Los drivers de costo se usan para capturar características del desarrollo del software que afectan al esfuerzo para completar el proyecto. Los drivers de costo tienen un nivel de medida que expresa el impacto del driver en el esfuerzo de desarrollo.

El peso se llama multiplicador de esfuerzo (ME). La medida asignada a un driver de costo es 1,0 y el nivel de medida asociado con ese peso se llama nominal.

Si un nivel de medida produce más esfuerzo de desarrollo de software, entonces su correspondiente ME está por encima de 1,0 y sino a la inversa.

Los ME se usan para ajustar el esfuerzo Meses-persona nominal

$$\mathbf{PM_{ajustado} = A \times (Size)^{B} \times \sum ME_{I}}$$

Drivers de costo del Modelo Post-Arquitectura
RELY, DATA, CPLX, DOCU
RUSE
TIME, STOR, PVOL
ACAP, PCAP, PCON
AEXP, PEXP, LTEX
TOOL, SITE
SCED

2.1. Drivers de Costo para el Modelo de Post-Arquitectura

Factores de producto

(RELY). Fiabilidad Requerida de Software

Esta es la medida de hasta qué punto el software debe realizar su función esperada durante un periodo de tiempo.

	Muy Bajo	***Bajo***	***Nominal***	***Alto***	***Muy Alto***	***Extra Alto***
RELY	Inconveniente ligero	Bajo, pérdidas fácilmente recuperables	Moderado, pérdidas fácilmente recuperables	Alta pérdida financiera	Riesgo de vidas humanas	

(DATA). Medida del Volumen de Datos

Esta medida intenta capturar lo que afecta en el desarrollo del producto los requerimientos de datos grandes. La medida se determina calculando D/P.

D/P = DataBaseSize (Bytes) /ProgramSize (SLOC)

DATA se valora como Bajo si D/P es menor que 10 y Muy Alto si es mayor que 1000.

	Muy Bajo	***Bajo***	***Nominal***	***Alto***	***Muy Alto***	***Extra Alto***
DATA		D/P < 10	10 ≤ D/P < 100	100≤D/P<1000	D/P ≤1000	

(CPLX). Complejidad del Producto

Las Medidas de complejidad del módulo versus Tipo de módulo proporciona la escala de valores CPLX de Cocomo II. La complejidad se decide en 5 áreas: Funcionamiento de control, Funcionamiento computacional, Funcionamiento de Dispositivos dependientes, Funcionamiento del sector de datos y Funcionamiento del Gestor de Interfaz de Usuario.

Se seleccionará el área o combinación de áreas que caracterizan al producto o a un subsistema del producto. La medida de complejidad es la media subjetiva de estas áreas.

Funcionamiento de control

Muy Bajo: código con unos pocos operadores de programación no anidados.

Bajo: anidamiento sencillo de operadores de programación estructurados.

Nominal: Mayoría de anidamiento simple.

Alto: Operadores de programación estructurados altamente anidados compuestos de muchos predicados.

Muy Alto: Codificación recursiva. Manejo de interrupciones con prioridades fijadas. Sincronización de tareas, retorno de llamadas complejas, proceso distribuido, etc.

Extra Alto: Múltiples recursos de planificación con cambio dinámico de prioridades. Control a nivel de microcódigo. Control distribuido en tiempo real.

Funcionamiento computacional

Muy Bajo: Evaluación de expresiones simples, por ejemplo A = B+C*(D-E)

Bajo: Evaluación de expresiones de nivel moderado: p.ej D = SQRT (B**2 – 4*A*C)

Nominal: Uso de rutinas matemáticas y estadísticas estándar.

Alto: Análisis numérico básico: ecuaciones diferenciales ordinarias.

Muy Alto: Análisis numérico difícil pero estructurado: ecuaciones de matrices, ecuaciones diferenciales parciales.

Extra Alto: Análisis numérico difícil y sin estructurar.

Funcionamiento dispositivos dependientes

Muy Bajo: Declaraciones simples de lectura-escritura con formatos simples.

Bajo: No necesidad de conocimiento de características del procesador particular o del dispositivo de I/O.

Nominal: El proceso de I/O incluye selección de dispositivo, estado de validación y proceso de errores.

Alto: Operación de I/O a nivel físico (traducción de direcciones físicas de almacenamiento: búsquedas, lecturas, etc.).

Muy Alto: Rutinas para diagnóstico de interrupciones, servicio de enmascaramiento. Manejo de línea de comunicación.

Extra Alto: Dispositivo dependiente temporalmente de la codificación. Operaciones microprogramazas.

Funcionamiento del sector de datos

Muy Bajo: Arrays simples en memoria principal. Actualizaciones, Queries simples

Bajo: Archivos únicos sin cambio en la estructura de datos, sin ediciones, sin archivos intermedios. Actualizaciones, Queries moderadamente complejos

Nominal: Entradas de múltiples archivos y salida de un único archivo. Cambios estructurales simples, ediciones simples. Actualizaciones, Queries complejos.

Alto: Encadenamientos simples activados por contenidos de hileras de datos.

Muy Alto: Coordinación de bases de datos distribuidas. Encadenamientos complejos. Optimización de la búsqueda

Extra Alto: Acoplamiento fuerte, estructuras de objetos y relacionales dinámicas.

Funcionamiento de gestor de Interfaces de usuario

Muy Bajo: Forma de entrada simple, generadores de informes

Bajo: Uso de constructores simples de interfaces gráficos de usuario

Nominal: Uso simple de conjunto Widget

Alto: Extensión y desarrollo de conjunto Widget. Voz simple de I/O multimedia

Muy Alto: Multimedia complejo 2D/3D, gráficos dinámicos multimedia

Extra Alto: Multimedia complejo, realidad virtual.

(RUSE). Reutilización Requerida

Explica el esfuerzo adicional necesario para construir componentes pensados para se reutilizados en otros proyectos.

	Muy Bajo	***Bajo***	***Nominal***	***Alto***	***Muy Alto***	***Extra Alto***
RUSE		Nada	A lo largo del proyecto	A lo largo del programa	A lo largo de la línea de producto	A lo largo del las líneas de producto múltiples

(DOCU). Documentación Asociada a las Necesidades del Ciclo de Vida

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
DOCU	Muchas necesidades del ciclo de vida sin cubrir	Alguna necesidad del ciclo de vida sin cubrir	Necesidades correctas al ciclo de vida	necesidades excesivas para el ciclo de vida	Necesidades muy elevadas para el ciclo de vida	

Factores de plataforma

(TIME). Restricción del Tiempo de Ejecución

Esta es una medida de la restricción del tiempo de ejecución.

	Muy Bajo	**Bajo**	**Nominal**	**Alto**	**Muy Alto**	**Extra Alto**
TIME			≤ 50%	70%	85%	95%

(STOR). Restricción de Almacenamiento Principal

Esta medida representa el grado de restricción de almacenamiento principal.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
STOR			≤ 50%	70%	85%	95%

(PVOL). Volatilidad de la Plataforma

Plataforma: significa la complejidad del Hardware y Software (OS, DBMS, etc.) que el producto software necesita para realizar sus tareas.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
PVOL		Cambios cada: año (mayor); mes (menor)	6 meses (mayor); (menor) 2 semanas	2 meses (mayor) semana (menor)	0,5 mes (mayor) 2 días (menor)	

Factores de personal

(ACAP). Habilidad del Analista

Lo que se debe considerar en esta medida son la habilidad de análisis y diseño, la eficiencia y la habilidad para comunicar y cooperar que posee el Analista, no se debe considerar aquí el nivel de experiencia ya que se mide con AEXP.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
ACAP	15º percentil	35º percentil	55º percentil	75º percentil	90º percentil	

(PCAP). Habilidad del Programador

Lo que se debe considerar en esta medida son la habilidad de análisis y diseño, la eficiencia y la habilidad para comunicar y cooperar que posee el Programador, no se debe considerar aquí el nivel de experiencia ya que se mide con AEXP.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
PCAP	15º percentil	35º percentil	55º percentil	75º percentil	90º percentil	

(AEXP). Experiencia en las Aplicaciones

Esto mide el nivel de experiencia en aplicaciones del equipo de proyecto en el desarrollo de sistemas.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
AEXP	≤ 2 meses	6 meses	1 año	3 años	6 años	

(PEXP). Experiencia en la Plataforma

Aquí se reconoce la importancia de entender el uso de plataformas más poderosas, incluyendo más interfaces gráficas.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
PEXP	≤ 2 meses	6 meses	1 año	3 años	6 años	

(LTEX). Experiencia en la Herramienta y en el Lenguaje

Esta es una medida del nivel de experiencia en el lenguaje de programación.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
LTEX	≤ 2 meses	6 meses	1 año	3 años	6 años	

(PCON). Continuidad del Personal

La escala de valores mide el movimiento de personal del proyecto anualmente.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
PCON	48% /año	24% /año	12% /año	6% /año	3% /año	

Factores del proyecto

(TOOL) Uso de Herramientas Software

Los valores para la herramienta van desde edición y código simple, Muy Bajo, hasta herramientas integradas de gestión del ciclo de vida, Muy Alto.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
TOOL	Editar, código, depuración	Simple, frontend, backend, CASE, integración baja	Herramientas de ciclo de vida básicas	Fuerte, herramientas de ciclo de vida maduras, moderada-mente integrada	Fuerte, maduro, herramientas de ciclo de vida proactivas, bien integrado con los procesos, métodos, reutilización	

(SITE). Desarrollo Multilugar

Determinar la medida del driver de costo incluye el cálculo y la medida de 2 factores: Localización del lugar (desde totalmente localizado hasta distribución internacional) y soporte de comunicación (desde correo de superficie y algún acceso telefónico hasta multimedia totalmente interactivo).

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
SITE localización	Internacional	Multiciudad y multicompañía	Multiciudad o Multicompañía	Misma ciudad o área metrop.	Mismo edificio o complejo	Totalmente localizado
SITE comunica-ciones	Algo teléfono, mail	Teléfono individual, FAX	Banda estrecha, email	Comunicación electrónica de banda ancha	Comunicación electrónica de banda ancha, Videoconferencia ocasional	Multimedia interactiva

(SCED). Calendario de Desarrollo Requerido

Este valor mide las restricciones de horario impuestas al equipo de proyecto que desarrolla el software. Los valores se definen en términos de porcentaje de aceleración o alargamiento sobre el calendario respecto de un calendario nominal para un proyecto que requiere una cantidad de esfuerzo dada.

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
SCED	75% del nominal	85%	100%	130%	160%	

Tabla que refleja los coeficientes necesarios para el cálculo:

	Muy Bajo	*Bajo*	*Nominal*	*Alto*	*Muy Alto*	*Extra Alto*
RELY	0.82	0.92	1.00	1.10	1.26	
DATA	--	0.90	1.00	1.14	1.28	
CPLX	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	--	0.95	1.00	1.07	1.15	1.24
DOCU	0.81	0.91	1.00	1.11	1.23	
TIME	--	--	1.00	1.11	1.29	1.63
STOR	--	--	1.00	1.05	1.17	1.46
PVOL	--	0.87	1.00	1.15	1.30	
ACAP	1.42	1.19	1.00	0.85	0.71	
PCAP	1.34	1.15	1.00	0.88	0.76	
PCON	1.29	1.12	1.00	0.90	0.81	
AEXP	1.22	1.10	1.00	0.88	0.81	
PEXP	1.19	1.09	1.00	0.91	0.85	
LTEX	1.20	1.09	1.00	0.91	0.84	
TOOL	1.17	1.09	1.00	0.90	0.78	
SITE	1.22	1.09	1.00	0.93	0.86	0.80
SCED	1.43	1.14	1.00	1.00	1.00	

Anexo C

Procedimiento Final para el Calculo del Valor Total de Software

Si no se posee el Código Fuente :

Como paso previo, se deben calcular los Puntos de Función sin Ajustar (PFSA) para determinar el tamaño del proyecto según la explicación de la Técnica desarrollada precedentemente y éstos deben convertirse en líneas de código fuente en el lenguaje de programación utilizado (SLOC).

Si se posee el Código Fuente :

Si se posee el código fuente o se debió realizar el paso anterior:

1: Se obtiene el esfuerzo nominal medido en Persona-Mes con la aplicación de la siguiente formula:

$$PM_{nominal} = A * (SLOC)^{B}$$

2: Se obtiene el esfuerzo ajustado también medido en Persona-Mes ajustado por los Multiplicadores de Esfuerzo detallados precedentemente, con la aplicación de la siguiente formula:

$$PM_{ajustado} = PM_{nominal} \times \sum ME_{I}$$

3: Se estima el costo mensual que se pagaría en salarios a una persona + todos los gastos que se incurriría y se multiplica por el valor PM ajustado, dando como resultado el valor total del Software en $, según la siguiente formula:

$$\text{Valor Total Software} = PM_{ajustado} \times \text{Costo Mensual Persona}$$